# Dissertation of Practical Course: End to End Learning for Self-Driving Cars

Jiachen Lu

***Abstract*—Recent work on end-to-end learning using convolutional neural network(CNN) has achieved impressive results in the field of autonomous driving when compared to traditional solutions. The aim of this research is to introduce an end-to-end deep learning approach for autonomous driving. This involves replacing the traditional decentralized driving task processing with a single neural network. The paper will provide a detailed exploration of the adopted deep learning approach and model for automated driving. The performance of the model will be evaluated through the UDACITY simulator to validate its effectiveness. Finally, this paper provides an overview of the current work in progress to deploy an end-to-end autonomous driving model to a logistics robot based on a ROS system, in order to create an autonomous driving system on park.**

***Index Terms*—Autonomous Driving, Deep Learning, End-to-end Learning, Simulator**

## I. Introduction

CONVENTIONAL autonomous driving system comprises several independent modules, such as perception, prediction, planning, and control. The hardware and software systems of these modules are developed separately and then integrated into the on-board system. Autonomous vehicles (AVs) rely on various sensors, including radar, lidar, ultrasonic sensors, and cameras, to perceive their surroundings. These sensors provide crucial information about the driving environment, such as the position and velocity of the driving area and surrounding obstacles. [1] The perception module produces a local map centred on the vehicle using data from the sensors. Simultaneously, the perception module uses the generated map and landmarks for the self-localization of the AV itself, and this algorithm is known as SLAM (Simultaneous Localization and Mapping). [2][3]

To operate safely and efficiently on the road, AVs must not only understand the current state of nearby road users, but also proactively predict their future behaviour, especially in shared road scenarios with other vehicles, pedestrians, and cyclists. Prediction algorithms and models commonly used include Hidden Markov Models (HMM) [4], Finite State Machines (FSM) [5], and Dynamic Bayesian Networks (DBN) [6]. However, the interdependence of vehicle behaviours, the impact of traffic rules and dynamic driving environments, and the multimodality of vehicle behaviours present new challenges. To address these challenges, more advanced and flexible prediction models are required to adapt to complex and dynamic traffic scenarios. [7]

Another important task for AVs is planning. Planning is typically divided into three hierarchical levels: route planning, behavior planning, and motion planning. [1][8]

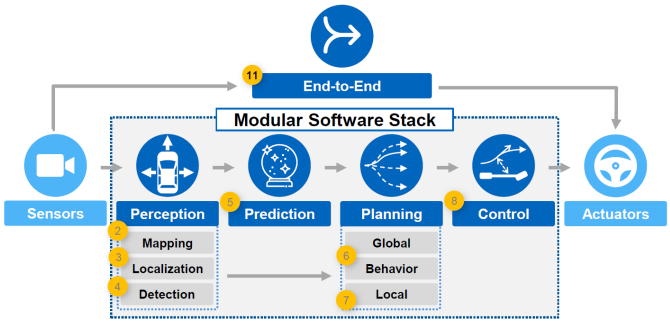



Fig. 1: The traditional autopilot pipeline consists of perception, prediction, planning, and control. An end-to-end model integrates all of these modules into one model.

### A. Route Planning

Route planning is often based on map information, where the decision system needs to select a path through the road network between the vehicle's current location and the destination. It is important to note that all evaluations presented are objective and free from bias. Path planning is commonly addressed using graph search algorithms, such as Dijkstra [9] or A* [10].

### B. Behaviour Planning

Once path planning is complete, behaviour planning is responsible for ensuring that the autonomous vehicle complies with traffic rules and interacts with other traffic participants safely, while progressing gradually along the planned path. As the available driving behaviours can often be modelled as a finite set, FSM [5] can automatically execute corresponding strategies. However, in real-world driving, particularly in urban environments, the intentions of other traffic participants introduce uncertainty. Therefore, the use of Markov Decision Processes (MDP) [11] and Partially Observable Markov Decision Processes (POMDP) [12] is more appropriate for behavior planning.

### C. Motion Planning

After behaviour planning determines the driving actions to be performed, the actual vehicle motion is determined by low-level motion planning. Motion planning aims to generate a trajectory that is both dynamically feasible and comfortable for passengers, while also avoiding collisions with obstacles detected by onboard sensors. Traditional approaches to motion planning include numerical optimization, graph search, and incremental tree methods.

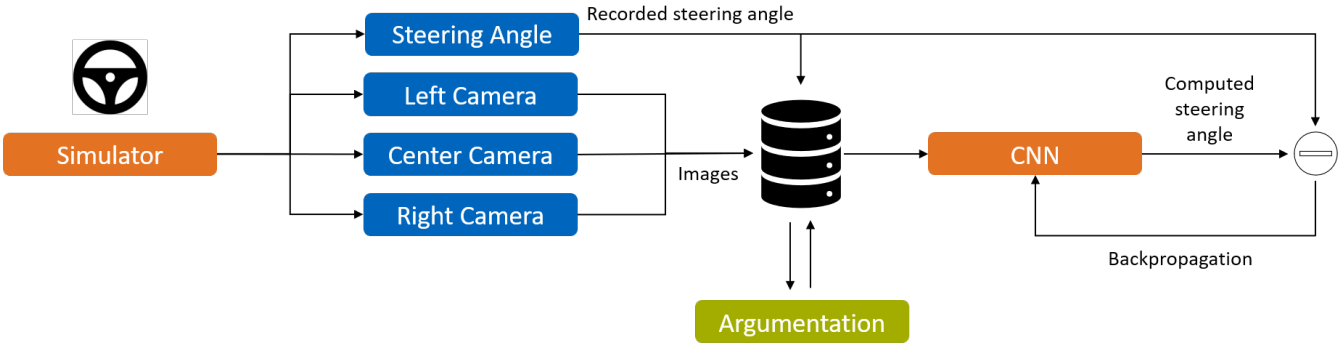



Fig. 2: Training pipeline. First, the training data, i.e. the images $I_{center,t}, I_{left,t}, I_{right,t}$ taken by the three cameras in the forward direction of the vehicle and the steering angle $S_t$ at the current moment, are obtained from the real car or the simulator. The data is then pre-processed and augmented. Finally, ResNet50 [13] is used as the detection backbone network to infer the steering angle and optimise the network using backpropagation.

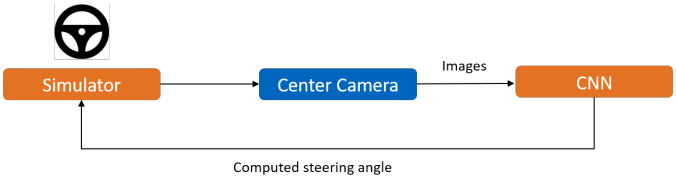



Fig. 3: During testing, the work pipeline involves inferring the steering angle $S_t$ directly from images $I_{c,t}$ captured by the vehicle's central camera using a trained deep neural network. Next, the vehicle in the simulator, or the real vehicle, performs a steering operation based on the steering angle $S_t$ at the current moment, and captures the image data of the next frame $I_{c,t+1}$ to predict the steering angle at the next moment $S_{t+1}$.

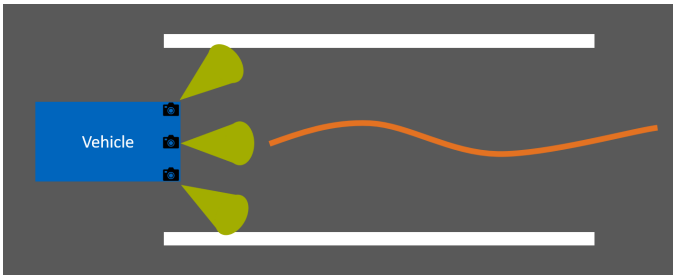



Fig. 4: All vehicles used in this work will be equipped with three front-facing cameras to capture image data for predicting the steering angle.

## II. Related Work

Autonomous driving technology has made significant progress in recent years and is an important research direction in the field of intelligent transportation. A large number of advanced models have greatly promoted the development of autonomous driving. Among them, E2E-AD-DenseTraffic [14] can perform end-to-end learning based on expert demonstration or self-supervised learning. LeTFuser [15], based on the Transformer [16] model, integrates perception and control modules. BEVGPT [17] and UniAD [18] are two large-scale generative pre-trained models that integrate multiple tasks for joint learning.

## III. Methodology

Based on Nvidia's work [19] on end-to-end learning for autonomous driving, we propose an end-to-end architecture based on ResNet50 [13] to enable autonomous driving in simulators and in the real world. The training pipeline is illustrated in Figure 2, while Figure 3 shows the testing pipeline.

### A. Problem Statement

The aim of this work is to train an end-to-end autopilot model to adjust the steering angle of the vehicle so that it can drive safely and autonomously on the track without collision. Figure 4 illustrates the installation of three cameras at the front of the vehicle to capture images of the center front, left front and right front of the vehicle. As illustrated in Figure 2, the model is fed with images from three cameras and sampled steering angles during the training period. However, during testing, only the images from the center camera are used to drive the car, as shown in Figure 3.

### B. Simulator and Real Vehicle

The main objective of this work is to train an end-to-end model using a simulator and a simulated vehicle. We have selected the UDACITY simulator, which is a Unity-based driving simulator that offers two driving modes: training and autonomous mode. In the training mode, we can manually drive the car and collect training data, including image data from three cameras and steering angle data. The autonomous mode allows us to test and evaluate the performance of the trained model. Additionally, the simulator offers two different tracks, as depicted in Figure 5.

Another objective of this work is to extend the concept of end-to-end learning models to a real vehicle, as shown in Figure 6. The vehicle is equipped with the same hardware configuration as the simulated vehicle, including three forward-facing cameras and steering angle sensors. The goal remains the same: to collect training data from the park and train an end-to-end autonomous driving model that will enable the vehicle to drive autonomously.


Fig. 5: The Udacity Simulator is a Unity-based driving simulator that offers two driving modes and two tracks for acquiring training data and evaluating autonomous driving models. In the training mode, the simulator provides a recording function, i.e. depending on the software configuration of the simulator, it can automatically record and save the image data and steering angle data of each moment to a local folder. Once the recording is complete, a CSV-format driving log is automatically generated for data preprocessing or model training.

Fig. 6: The real vehicle used in this study is ROS-based and has a hardware configuration similar to the simulated vehicle. It is also equipped with three forward-facing cameras at the front to record the surrounding environment. Additionally, the vehicle can be driven manually to collect training data or to execute and evaluate a trained autonomous driving model.

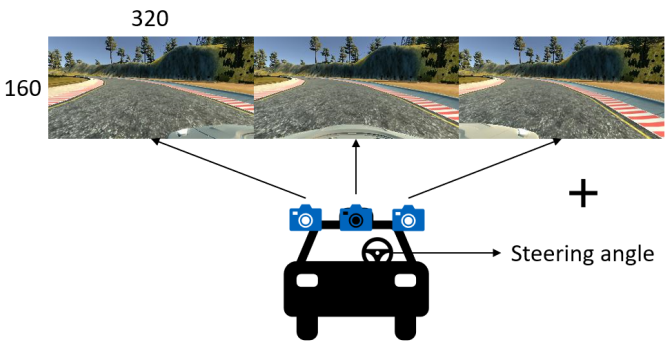



Fig. 7: The vehicle's three forward-facing cameras acquire images of the centre front $I_{c,t}$, left front $I_{l,t}$ and right front $I_{r,t}$ of the car. The steering angle $S_t$ is recorded by the steering angle sensor.

### C. Data Acquisition

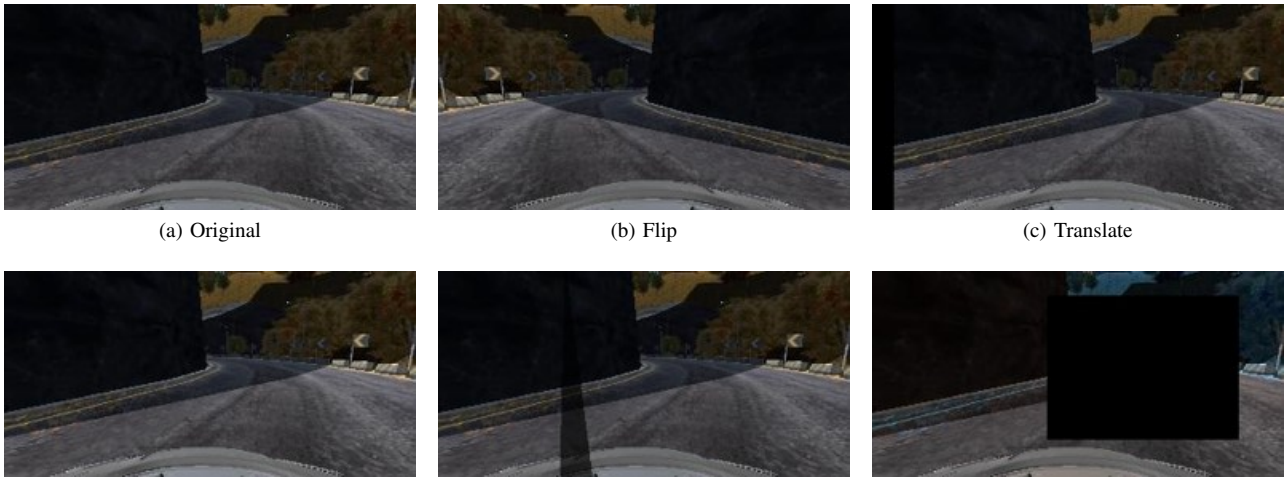
To train an end-to-end autonomous driving model, the first step is to collect training data. As shown in the work pipeline in Figure 2, we first need to collect image $I$ and steering angle $S$ data in the simulator. Figure 7 shows that the vehicle's three forward-facing cameras capture images of the center front $I_{c,t}$, left front $I_{l,t}$, and right front $I_{r,t}$ at each moment, with image dimensions of $320 * 160$. Additionally, the steering sensors provide steering angle data $S_t \in [-1, 1]$ for the current moment. Note that this is the steering angle data after normalization. The actual steering angle ranges from -25° to +25°. Furthermore, the simulator provides brake and throttle data at each moment. However, this information is not directly relevant to our work. Finally, the simulator will automatically generate a CSV file that contains all the data collected during the data acquisition process for model training.

However, the raw data cannot be used directly as training data. The simulator captures three pictures as image inputs, but only one steering angle at the same moment. Therefore, customised steering angles must be assigned to the left and right picture inputs. Based on experience and test results, we add 6.25° and subtract 6.25° to the steering angles corresponding to the left and right images respectively ($S_{l,t} = S_t + 0.25$ and $S_{r,t} = S_t - 0.25$), while the steering angle corresponding to the centre image is the sampled steering angle ($S_{c,t} = S_t$). This pre-processing resulted in pairs of image and steering angle that could be used as training data.

This work utilises both the official dataset from UDACITY and a manually collected dataset.

*1) Dataset from UDACITY:* The UDACITY official dataset comprises approximately 24,000 images and their corresponding steering angle data, as depicted in Figure 10a. The dataset is heavily skewed towards steering angles of 0, and there is a severe lack of data for cases where the steering angle is greater than 0.5. This imbalance can significantly impede training and hinder the model's ability to generalize, particularly when dealing with large steering angles. Therefore, we decided to discontinue the use of the UDACITY dataset and instead manually collect training data using the simulator's training mode.

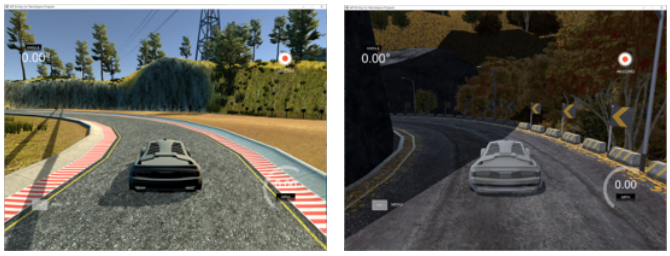
*2) Dataset from Manual Acquisition:* To address the above issues, we decided to manually collect training data using the training mode of the UDCAITY simulator. To operate the simulated vehicle, we use an Xbox controller instead of a keyboard. This decision was made to obtain fine or continuous steering angle data, as the data available from the keyboard is coarse or discrete. Figure 10b shows the data distribution of the manually collected dataset, which contains about 150,000 images. To improve the dataset's data distribution, we sampled more image data from the large steering angle region and reduced the frequency of image sampling from the straight ahead region. Although our dataset has improved compared to the UDACITY dataset in Figure 10a in terms of data distribution, there are still some problems in the actual training and testing process.



(a) Original (b) Flip (c) Translate

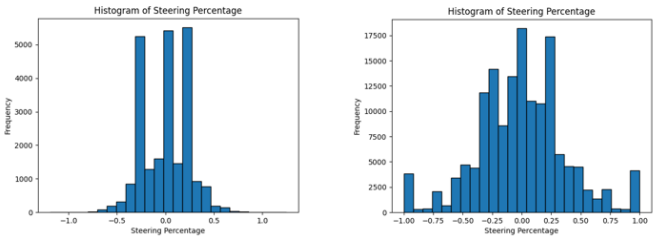
(d) Brightness (e) Random shadow (f) Random erasing

Fig. 8: Five different data augmentation methods were used to process the image data.

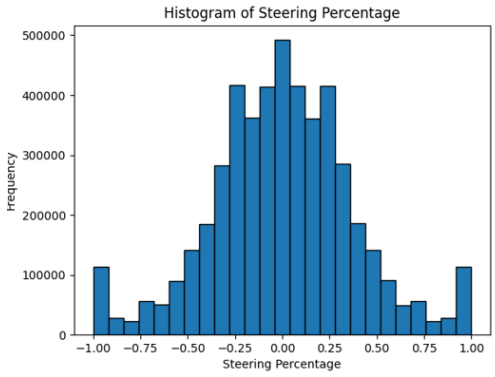


(a) Track1 (b) Track2

Fig. 9: Two different tracks in the UDACITY simulator.




(a) UDACITY Dataset (b) Manual Acquisition Dataset




(c) Dataset from Manual Acquisition after Augmentation

Fig. 10: Distribution of data within different datasets.

### *D. Data Augmentation*

Using the manually collected dataset, we trained and evaluated models in the UDACITY simulator. However, we found that these models had poor generalisation ability due to the significant difference between the two tracks in the simulator, as illustrated in Figure 9. Track 1 has clear route markings on both sides, while Track 2 has mostly unclear markings, some of which are covered with leaves. Furthermore, Track 1 is in a very bright environment, whereas Track 2 is in a dark environment. Finally, in Track 1 the road does not have very large turning areas, whereas on track 2, the opposite is true.

The above problem can be improved by driving for a longer period of time to collect more data in the simulator, but manual driving is very time consuming and tiring. For this reason, we decided to perform data augmentation on the manually collected training dataset to improve the model's performance.

Figure 8 shows that five different methods of data augmentation were used in this work. The first method is flipping, which was introduced to address the issue of data imbalance caused by collecting a large amount of data with positive or negative steering angles on the circular track of the UDACITY simulator during data collection. The second is translate. In real driving scenarios, simulated or actual vehicles may not always be able to drive in the centre of the track. Therefore, image translate is necessary to improve the model's generalisation ability and robustness. Additionally, random brightness adjustments are added to the image to adapt the model to different tracks, as there may be significant differences in brightness between tracks in the simulator or in the real world. Random shadow is added to the model to deal with the fact that some of the route markers on both sides of the road are covered by shadows, as shown in Figure 9. Additionally, the final method, random erasing, replaces parts of the original image with black areas. This approach aims to enable the model to infer steering angles based on partially valid image features. Although this means training the model is more difficult, the model can perform better.

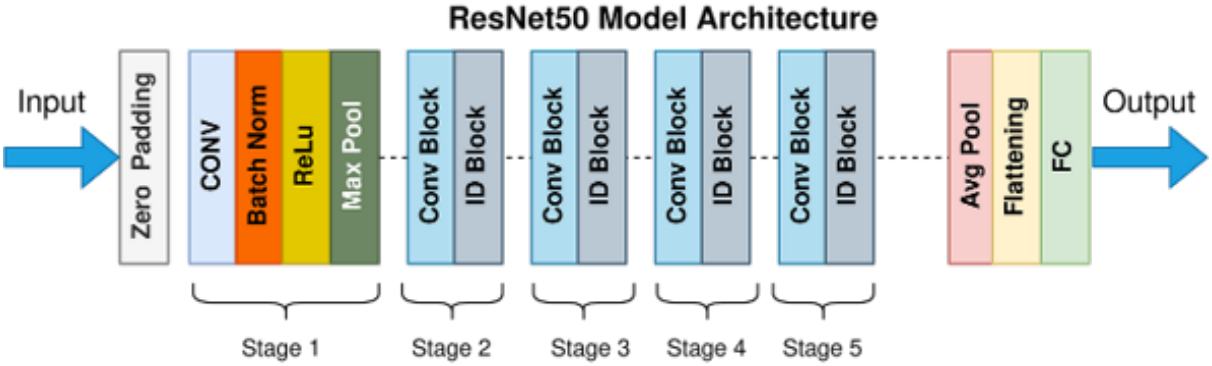



Fig. 11: ResNet50 Model Architecture

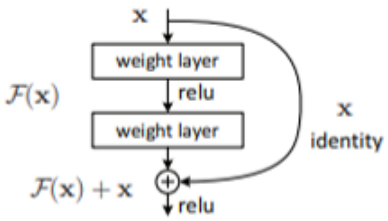



Fig. 12: Residual learning: a building block

As can be seen in Figure 10c, the size of the dataset after data augmentation is approximately 4.8 million images. And now the data distribution is more balanced compared to the previous dataset. Of course, there are some techniques to improve the quality of the dataset in data collection and data augmentation, such as randomly deleting the images with zero steering angle, and so on.

### E. ResNet Backbone

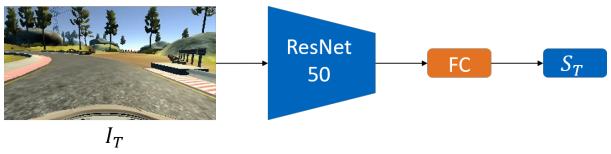
Having acquired sufficient high-quality data, our first task was to select the end-to-end model used to extract the image features. Based on Nvidia's work [19], we decided to use ResNet50 [13] as the detection backbone network. In this work, we construct two models based on ResNet50, i.e. pure ResNet50 and ResNet50+GRU [20] .

*1) ResNet50:* Convolutional modules are very capable of processing image data, and deep neural networks consisting of multiple layers of convolutional modules can extract features at different levels of the image to produce high-quality feature representations. However, deep neural networks are very difficult to train and the network suffers from degradation, which means that deep neural networks do not perform as well as normal models.

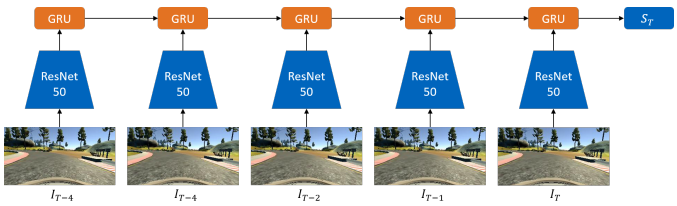
Based on the above problems, the residual learning module and the residual neural network were introduced. Figure 12 shows that the residual learning module, also known as identity mapping or skip connection, directly maps the unprocessed image features to the processed image features. This innovative approach preserves image features during convolutional processing, even if the network cannot extract useful information at this layer. As a result, deeper neural networks can be trained. The residual neural network is composed of multiple residual learning modules, as depicted in Figure 11. In this work, we selected ResNet50 as the detection backbone network to extract image features and predict the desired steering angle directly using the multilayer perceptron (MLP), as shown in Figure 13a.

*2) ResNet50+GRU:* In addition to using the pure ResNet50 model, in this work we have constructed another network model, both ResNet50+GRU. We believe that in real driving, the driver's actions at any moment (braking, accelerating, steering angle) should be based not only on the image of the current moment, but also on several moments before the current moment. Based on this idea, we decided to introduce the sequential data processing model, the GRU model.

Figure 13b shows that in this model, ResNet50 extracts image features at T moments and several moments prior to T moments from the input images $I_{T-4:T}$ to predict the steering angle at each moment $S_{T-4:T}$. The steering angle and image features of each moment are then combined as inputs to the GRU module. Finally, the GRU module for the current moment T infers the steering angle $S_T$ as the model output.




(a) ResNet50 Model




(b) ResNet50 + GRU Model

Fig. 13: Two different model architectures using ResNet as the detection backbone network. Figure 13a shows a pure ResNet50 model, where the steering angle is inferred directly from the input image. Figure 13b shows the ResNet50+GRU model, where the steering angle is inferred from multiple input images.

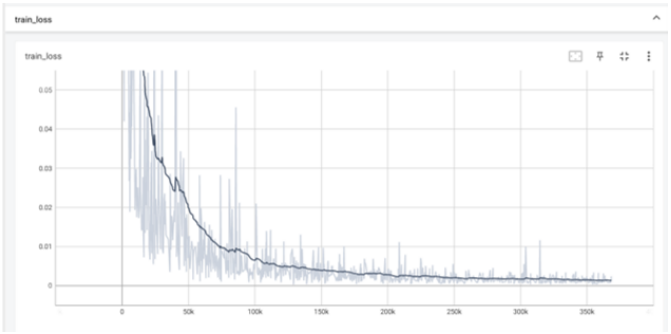



Fig. 14: After approximately 20 epochs, the training loss has only marginally decreased.

Fig. 15: Compared to the simulator, the driving environment of the real vehicle is more complex.

### F. Training Objective

For this project, we use L2 loss as our training objective function. As the input steering angle has been normalised to $\hat{S}_T \in [-1, 1]$, the L2 loss function is suitable for predicting the steering angle. Equation 1 demonstrates our training objective.

$$L_2 = \frac{1}{N}\sum_{i=1}^{N}(\hat{S}_{T,i} - S_{T,i})^2 \tag{1}$$

## IV. Experiments

### A. Experimental Settings

*1) Datasets:* Sections III-C and III-D provide a detailed description of how this work collects, preprocesses, and augments training data. During the model training process, we utilized the PyTorch Lightning dataset function. Specifically, we designated a portion of the training data as the validation set and shuffled the remaining data to create the training set.

*2) Evaluation Metrics:* In this work, we evaluated the performance of the model driving a simulated vehicle in a simulator using the collision rate, maximum speed and completion rate as evaluation metrics. All results of this work were based on qualitative analysis and the results of the evaluation can be seen in the video.

*3) Implementation Details:* Section IV-A1 describes how to create the required dataset. The models were trained using two RTX 3090 24GB GPUs and the training environment was set to Python 3.8, PyTorch 1.11.0 [21] and PyTorch Lightning 2.0.2 [22]. The Adam optimizer [23] was used with a learning rate of 1e-3 and a batch size of 64. Each model was trained for 20 epochs.

### B. Qualitative Analysis

As stated in section III-E, we constructed two ResNet-based models and evaluated their performance in the simulator following the practical details outlined in section IV-A3. Figure 14 shows that after 20 Epochs, the training loss of both models hardly decreases any further. Upon evaluation by the simulator, it was found that the ResNet50 model outperforms the ResNet50+GRU model. The latter is unable to complete collision-free autopilot on the track, while the former can achieve this at speeds of up to 20 MPH.

As ResNet50+GRU infers the steering angle based on multiple image inputs, the simulated vehicle driven based on this model will be slower and less flexible than the simulated vehicle driven by ResNet50 alone, which is detrimental to the realisation of high-speed autonomous driving.

### C. Ablation Study

*1) Does data augmentation help?:* Starting with the collection of training data, data augmentation is definitely helpful. Figures 10b and 10c show that it avoids repetitive collection of invalid data and creates a more even data distribution. Furthermore, by comparing the performance of models trained using different datasets and evaluating them on the simulator, data augmentation can effectively enhance the model's ability to handle various tracks and environments, thereby improving its generalisation ability.

*2) Does GRU help?:* The purpose of using a recurrent neural network, such as GRU, is to enable the model to infer the final steering angle based on multiple input images. However, during training and testing, we found that the ResNet50+GRU model performed worse. The main reasons for this are as follows: firstly, if you want to use a recurrent neural network, you need to have a smaller batch size or more GPUs because the model has to process multiple input images at the same time. Secondly, random data augmentation leads to more difficult training, which requires more time and effort to train the model. Additionally, the use of multiple input images can result in a dispersion of weights, meaning that the features of the input images at each moment can affect the steering angle at the final moment, potentially leading to a more sluggish and inflexible model.

### D. Real Car

This work includes training and evaluating models based on simulators and simulated vehicles, as well as training and evaluating an end-to-end autopilot model based on real world and real vehicles, as shown in Figures 6 and 15.

We still used the data augmentation approach described in Section III-D to process the input data and used ResNet50

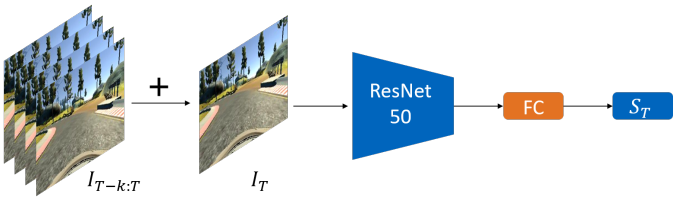



Fig. 16: Multiple input images are accumulated and processed by a ResNet50 to infer the steering angle.

as the detection backbone network to infer steering angles. However, the end-to-end trained autonomous driving model performs poorly on the actual vehicle and does not allow collision-free autonomous driving of the vehicle in the park. The reasons for this are as follows: firstly, the dimensionality of the image data captured by the real vehicle far exceeds that of the simulated vehicle, which means that we need more GPUs to process this data. As depicted in Figure 15, the real-world environment is much more complex than the simulated one. Road markings, weather conditions, and other road users can significantly impact the quality of the training data in real-world scenarios. Furthermore, end-to-end models that rely solely on ResNet50 lack the necessary generalisation capabilities when dealing with complex environments. Finally, realistic model testing is labour intensive, which is detrimental to our model evaluation. Based on the above issues, we still need to improve data acquisition, model architecture, training and evaluation methods, etc. to achieve end-to-end autonomous driving for real vehicles.

## V. Further Work

The proposed model performs well in the simulator. However, there are some issues that need to be addressed in future work.

*1) Processing Input Images as Sequence Data:* Chapter III-E2 proposes a model for processing multiple input images and predicting the steering angle using ResNet50+GRU. However, this model performs slightly worse compared to other models. To improve performance, we suggest accumulating multiple input images and processing them using a single ResNet50 instead of multiple ResNet50s. This will simplify the process and may improve the accuracy. Figure 16 illustrates this approach.

*2) Real-world Issues:* In Section IV-D, we found that our proposed end-to-end autonomous driving model performs poorly in the real world, and we believe that the main reason is that real-world environments are too complex, and models based only on ResNet50 do not have sufficient perception and generalisation capabilities to cope with complex environments. Therefore, to improve the generalisation ability of the model, we propose adding other model modules to the network architecture. These modules include IMU or GNSS modules to provide localisation capability, path planning and decision making modules, and perception fusion modules. By incorporating these modules, the performance of the autonomous driving model on real vehicles can be significantly enhanced.

## VI. Conclusion

This project proposes an end-to-end autonomous driving model based on Nvidia's work on end-to-end autonomous driving, using ResNet50 and ResNet50+GRU. The training data is collected through driving simulators and real-world vehicles. An approach for data augmentation is proposed to improve the quality of the dataset. Finally, the models are trained and evaluated in both the simulator and the real world.

## REFERENCES


[1] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[2] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[3] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," 2023.

[4] L. E. Baum and T. Petrie, "Statistical Inference for Probabilistic Functions of Finite State Markov Chains," *The Annals of Mathematical Statistics*, vol. 37, no. 6, pp. 1554 – 1563, 1966.

[5] C. E. Shannon and J. McCarthy, eds., *Automata Studies. (AM-34), Volume 34*. Princeton: Princeton University Press, 1956.

[6] K. P. Murphy, "Dynamic bayesian networks: representation, inference and learning," 2002.

[7] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis, "Deep learning-based vehicle behavior prediction for autonomous driving applications: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 33–47, 2022.

[8] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," 2016.

[9] E. DIJKSTRA, "A note on two problems in connexion with graphs.," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[10] N. J. Nilsson, *A mobile automaton: An application of artificial intelligence techniques*. Stanford research institute, 1969.

[11] R. Bellman, "A markovian decision process," *Indiana Univ. Math. J.*, vol. 6, pp. 679–684, 1957.

[12] Åström, Karl Johan, "Optimal Control of Markov Processes with Incomplete State Information I," vol. 10, pp. 174–205, 1965.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[14] J. Shrestha, S. Idoko, B. Sharma, and A. K. Singh, "End-to-end learning of behavioural inputs for autonomous driving in dense traffic," 2023.

[15] P. Agand, M. Mahdavian, M. Savva, and M. Chen, "Letfuser: Lightweight end-to-end transformer-based sensor fusion for autonomous driving with multi-task learning," 2023.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[17] P. Wang, M. Zhu, H. Lu, H. Zhong, X. Chen, S. Shen, X. Wang, and Y. Wang, "Bevgpt: Generative pre-trained large model for autonomous driving prediction, decision-making, and planning," 2023.

[18] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, and H. Li, "Planning-oriented autonomous driving," 2023.

[19] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016.

[20] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[22] W. Falcon and The PyTorch Lightning team, "PyTorch Lightning," Mar. 2019.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.