P-ISSN 2586-9000
E-ISSN 2586-9027
Homepage : https://tci-thaijo.org/index.php/SciTechAsia
Science & Technology Asia
Vol. 27 No.4 October - December 2022
Page: [343-362]
Original research article

Automatic Driving for Road Tracking and Traffic Sign Recognition

Shukai Ding, Jian Qu*

Faculty of Engineering and Technology, Panyapiwat Institute of Management, Nonthaburi 11000, Thailand

Received 1 October 2021; Received in revised form 31 May 2022;
Accepted 20 July 2022; Available online 31 December 2022

ABSTRACT

Automatic driving has become a very important research field that may change how humans travel. Most of the existing research on automatic driving has been on road tracking. However, there has been little research on automatic driving to simultaneously achieve road tracking and traffic sign recognition (relying on only one camera as input). In this paper, we achieve the simultaneous implementation of both functions and explore the effects of the model, speed, and batch size on the automatic driving of the smart car. We constructed a smart car based on a Jetson Nano and one camera. In addition, we designed three real simulation environments and three groups of experiments. Firstly, the smart car is trained and tested in Experiment Group 1 using Environment 1. ResNet18 and ResNet34 are compared with speed modifications, and it screens a base model for Experiment Groups 2 and 3. Secondly, the smart car performs road-tracking experiments in an untrained environment (Environment 2) with the model from Experiment Group 1. Thirdly, road tracking and traffic sign recognition are achieved simultaneously in Environment 3. We conducted a total of 53 experiments in different network setups and found out that S0.6 is the best speed, ResNet34_B32 is the best model, and road tracking can be achieved in both trained and untrained environments. The results of Experiment Group 3 show that the smart car can perform traffic sign recognition smoothly during road tracking.

Keywords: Automatic driving; CNN; Deep Learning; Jetson Nano; Neural network model

1. Introduction

The application of AI in automatic driving is powered by deep learning algorithms of three significant functions: environment perception, planning and decision-making, and controlling. Smart cars use methods such as deep learning, fuzzy logic, expert systems, and genetic algorithms [1-3] to have a certain level of intelligence in automatic driving through autonomous learning.

Deep learning has been applied to automatic driving research in the automotive sector, making automatic

***Corresponding author:** jianqu@pim.ac.th **doi:** 10.14456/scitechasia.2022.91

driving a hot area. Google and Baidu conduct research on automatic driving in real cars, which is undoubtedly the most realistic method of conducting the research. However, this approach is costly and dangerous and thus cannot be applied to a broad range of ordinary research. As a result, two different types of research are becoming more and more widely adopted: research conducted in a virtual online environment and research conducted in a realistic simulation.

Road tracking and traffic sign recognition are two critical functions of automatic driving for smart cars. ZTÜRK et al. [4] studied autonomous vehicles in a virtual environment. However, using virtual environments on the Web to simulate complex actual-world scenarios is difficult, and the test results are not convincing. Qi Zhang et al. [5] used reinforcement learning to train in a virtual environment and then migrated to a real-world environment for testing. Experimental results in a virtual environment are positive, but it is difficult to transfer such experimental results to a real-world environment. The real-world environment is dynamic and ever changing, owing to factors such as lighting, shadows, and other noise sources. Furthermore, none of these factors is considered in a virtual environment. Therefore, we use smart cars in an already constructed simulation environment for training and testing. Our approach can be closer to real-world automatic driving than the abovementioned methods.

Additionally, Xi Li et al. [6] evaluated their study about a related study on automatic driving tasks using a high-performance graphics card on a PC platform. Although the test performed well, this was attributed to powerful graphics cards and large servers. The difference is that we deployed deep learning on an embedded development platform, which enabled autonomous driving and achieved good results. We used the Jetson Nano as a development board to build a smart car for autonomous driving experiments. The Jetson Nano is similar to a microcomputer and it can be used as a controller, a processor, and a deep learning platform.

In addition, we have constructed three environments as driving simulation tracks for smart cars. Environment 1 is used for data collection and testing; Environment 2 discusses the effects of noise on smart cars; and Environment 3 demonstrates that smart cars are capable of road tracking and traffic sign recognition. In addition, we discuss the impact of model, batch size and speed on automatic driving.

2. Related Research

2.1 Hardware

Constructing a smart car platform takes into account the hardware configuration of the development board and sensors. Sumeth Yuyong and Jian Qu [7] chose Arduino as the development board and used reinforcement learning to implement path planning. Their smart car requires a Bluetooth connection between the Arduino and the PC, then transmits the decisions of PC to the Arduino, which controls the car. Therefore, it is not a stand-alone agent. Moreover, Arduino is a hardware platform based on a microcontroller, and it can only execute programs that have been pre-programmed into the chip. Arduino can only be used in a few simple scenarios because of its limited performance, and the implementation of certain functions requires the assistance of a large number of external sensors. For instance, Ayesha Iqbal et al. [8] chose Arduino for their research, but they required cameras and infrared sensors to accomplish road tracking. Numerous previous studies [9-12] have used Raspberry Pi rather than Arduino. Raspberry Pi is a hardware platform based on a microprocessor capable of handling complex application scenarios such as network communication and image processing. For instance, Xinxin Du et al.

[13] used cameras and radar to achieve road tracking for smart cars in their study. While many studies use the Raspberry Pi as a development board, we found that most of these studies used CNN and RNN deep neural networks, which have few layers and simple structures. Although the Raspberry Pi has improved performance compared to the Arduino, it lacks in computing power and execution speed compared to the Jetson Nano, as the Raspberry Pi has no video memory. Therefore, we recommend using the Jetson Nano as a development board for smart cars.

2.2 Deep neural networks

A Convolutional Neural Network (CNN/ConvNet) is a deep learning model or a multilayer perceptron [14] that is similar to an artificial neural network. The multilayer perceptron is like a black box, including one input layer, multiple hidden layers, and one output layer (Fig. 1).

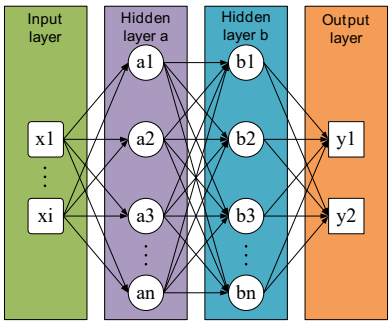


Fig. 1. Schematic diagram of deep learning.

A CNN is usually used to analyze visual images and is one of the critical algorithms for deep learning. The CNN was first proposed by Yann LeCun in 1989 [15-16] and applied to handwritten font recognition (MINST). The role of CNN is to restore the image to a form that is easier to process without losing the features that are essential to obtain good predictions.

Compared to other classification algorithms, CNNs require much less preprocessing. In the original method, the filter was designed manually. Furthermore, with adequate training, CNNs can learn these functions.

Convolutional networks typically consist of convolutional layers, pooling layers, and fully connected layers. CNN simulates feature differentiation through convolution, reduces the order of magnitude of network parameters through weight sharing and pooling of convolution, and finally accomplishes tasks such as classification through a fully connected layer [17].

Convolution:

An example of a convolution operation shows in Fig. 2.

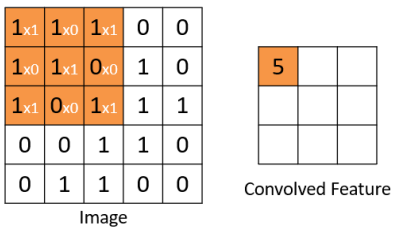


Fig. 2. Example diagram of the convolution operation.

These numbers are a 5×5 image with a 3×3 convolution kernel:

$$\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}$$

Use the convolution kernel to convolve the image. The convolution kernel traverses all the image pixels from left to right and top to bottom in the image. The convolution kernel and the corresponding image pixel values are multiplied and summed to obtain a 3×3 convolution result. We can understand it as using a filter (convolution kernel) to filter each small area of the image to get the feature value of these small areas. In the actual training process, the value of the convolution kernel is learned during the learning process.

After the convolution operation, the image size can be relatively reduced, but the image is still large, so pooling is required to reduce the data dimension of the image [18].

Pooling is down sampling. The pooling process is shown in Fig. 3.

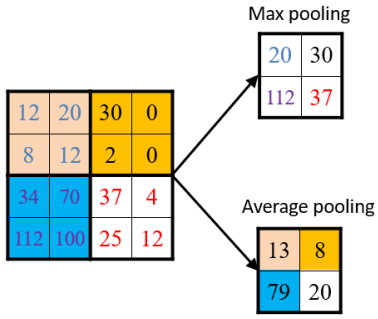


Fig. 3. Example diagram of the pooling operation.

The original image is 4×4, we downsample it and the sampling window is 2×2, and finally, it is downsampled into a 2×2 feature map. In practical applications, pooling is divided into max pooling and average pooling according to the down-sampling method.

The key to automatic driving is deep neural networks, but most existing research has chosen simple CNN [19-21] neural networks. For instance, Viktor Rausch et al. [19] used a 6-layer CNN neural network in their study. Truong-Dong Do et al. [20] used a CNN neural network with nine layers. However, shallow layer neural networks have the potential to cause two issues. First, feature extraction is insufficient; second, tens of thousands of image datasets are required. Pranav Gupta et al. [22] used VGG to address these issues in their study. While the layers of this network are deep, they are also large, creating a latency issue. These neural networks were initially designed for deployment on desktop computers equipped with powerful GPUs rather than mobile, edge-based devices. To address this issue, we chose Resnet18. Kaiming He et al. proposed Resnet18 [23] in 2016. They proposed adding a residual network (Fig. 4) to the neural network, which significantly increases the number of layers in the network and alleviates the network's degradation problem.

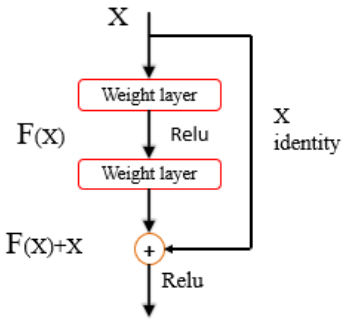


Fig. 4. Residual learning structure diagram.

3. Experimental Setup

In chapter 3, we focus on two factors: lane line detection (road tracking) and traffic sign detection (road tracking-based). Then, in Section 3.1, we introduce the environment construction; in Section 3.2, we describe the hardware; in Section 3.3, we screen the convolutional neural networks; and in Section 3.4, we demonstrate our experimental setup, which

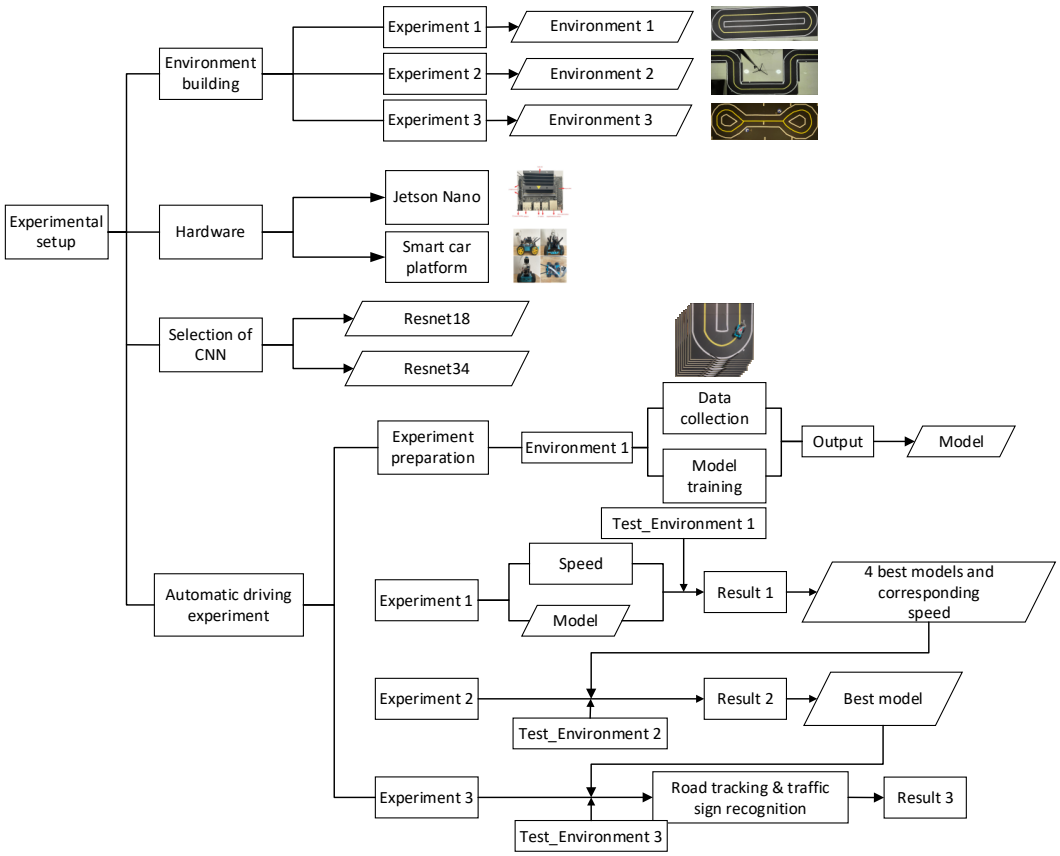


Fig. 5. Experimental framework diagram.

focuses on three experiments. The specific experimental framework diagram is shown in Fig. 5.

3.1 Environment construction

In Section 3.1, we introduced three experimental environments built for different experiments. We present them in two ways: schematic diagrams and real pictures of the environment.

3.1.1 Environment of Experiment 1 (Road tracking in the original environment)

Environment 1

Environment 1 is the scene of Experiment group 1, which has two main purposes: collecting data and verifying the road tracking of the smart car in the trained environment. Fig. 6 is the schematic diagram of Environment 1.

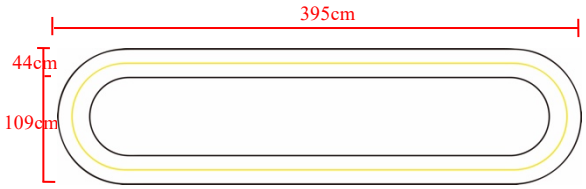


Fig. 6. Schematic diagram of Environment 1.

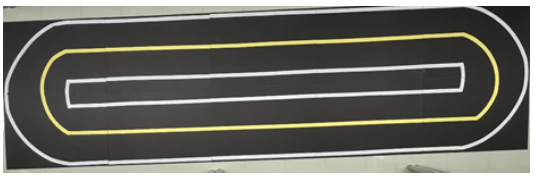


Fig. 7. Environment 1.

Fig. 7 is the picture of Environment 1. It is an oval track, and the yellow and white lines are the lane lines. We set up straight roads and curves to restore realistic roads with complex road conditions. We collected data in Environment 1. The data were collected for Experiment 1. (Remarks:

Collect training data in the form of taking photos.) We trained the model with the data collected in Experiment 1. In Experiment 1, we will test the effect of different models and different speeds on road tracking.

3.1.2 Environment of Experiment 2 (Road tracking in the new environment)

Environment 2

Environment 2 is the scene of Experiment group 2, which is used to verify the road tracking of the smart car in an environment without training. Fig. 8 is the schematic diagram of Environment 2. Fig. 9 shows the photos of Environment 2. They are stitched shapes based on Environment 1 and are used to test whether the car can achieve road tracking and adapt to unfamiliar environments.

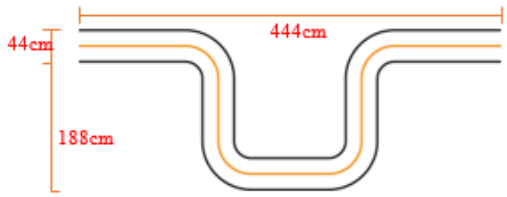


Fig. 8. Schematic diagram of Environment 2.

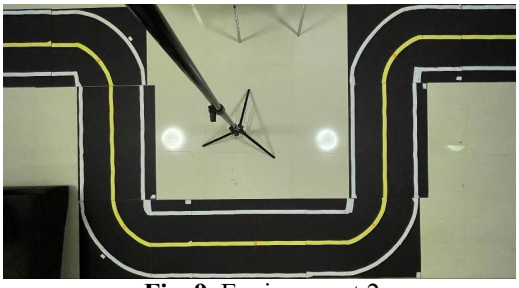


Fig. 9. Environment 2.

3.1.3 Environment of Experiment 3 (Road tracking & traffic sign recognition)

Environment 3

Environment 3 is the scene of Experiment group 3, which is used to verify that the smart car can achieve traffic sign recognition during road tracking. We used a circular track, which had two turn-offs, and we would randomly place left-turn or right-turn signs at the turn-offs. Fig. 10 shows the schematic diagram of Environment 3, and Fig. 11 shows the photos of Environment 3.

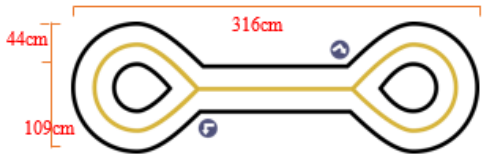


Fig. 10. Schematic diagram of Environment 3.

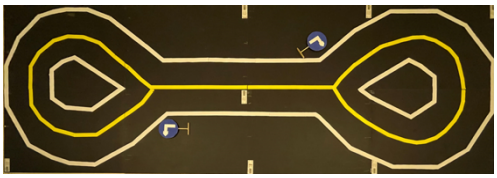


Fig. 11. Environment 3.

3.2 Hardware preparation

3.2.1 Introduction to Jetson Nano motherboard

The Jetson Nano is a product released by NVIDIA at the 2019 NVIDIA GPU Technology Conference. The specific information of the Jetson Nano is shown in Fig. 12.

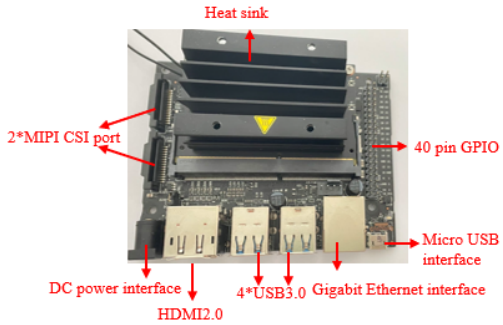


Fig. 12. Jetson Nano diagram.

The Jetson Nano has four high-speed USB 3.0 ports, MIPI CSI-2 camera connector, HDMI 2.0 and DisplayPort 1.3, Gigabit Ethernet, M.2 Key-E module, MicroSD card slot, and 40-pin GPIO connector. The specific information on the Jetson Nano is shown in Table 1.

Table 1. Information table of Jetson Nano.

Project	NVIDIA Jetson Nano
CPU	Quad-Core ARM Cortex-A57 64-bit @

	1.42Ghz
GPU	NVIDIA Maxwell w/128 CUDA cores @ 921 Mhz
Memory	4GB LPDDR4
Networking	Gigabit Ethernet/M.2 Key E(for Wifi support)
Video Encode	H.264/H.265(4Kp30)
Camera	MIPI CSI port
USB	4 * USB3.0, USB2.0 Micro-B
Display	HDMI2.0 and eDP1.4
Other	40-pin GPIO
Storage	Micro-SD

The Jetson Nano can run a wide range of advanced frameworks, including full native versions of popular machine learning frameworks, such as TensorFlow, PyTorch, Caffe/Caffe2, Keras, and MXNet. In addition, these frameworks can be used to build automatic machines and complex artificial intelligence systems by implementing robust image recognition, object detection, localization, pose estimation, and intelligent analysis.

This paper used the PyTorch framework for road tracking and traffic sign recognition, with the programming language Python. In addition, Torch, Torchvision, and OpenCV are also used for image processing and neural network deployment.

3.2.2 Smart car platform

We used a four-wheeled car as the platform; only a camera was used for the sensing part. This platform is equipped with a Jetson Nano development board, an aluminum alloy chassis, and four single-axis metal gear motors.

The specific image of the completed smart car is shown in Fig. 13.

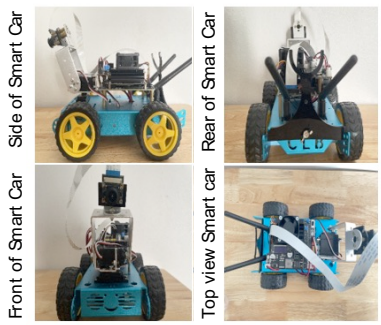


Fig. 13. Smart car platform.

3.3 Selection of convolutional neural networks

In 2012, Krizhevsky and Hinton founded AlexNet, which kicked off the craze for deep learning. The VGG, ResNet, and DenseNet were proposed immediately afterward, and they all contributed to the development of computer vision. As a technique for image processing, convolutional neural networks have become the basis of automatic driving. This paper aims to discuss the task of smart cars tracking and recognizing traffic signs. The selection of a high-quality CNN for experimentation is also crucial to the success of this paper. Therefore, we examined several different types of neural networks and produced the following conclusions through training, as illustrated in Table 2.

Nine neural networks were tested by loading them onto a Jetson Nano board, including the AlexNet, VGG, ResNet, and DenseNet series. According to the results, VGG, Resnet101, Resnet152, Resnext101-32x8d, Densenet121, and Densenet169 all exhibit delays. Complex neural networks require sufficient GPU memory to keep the model running smoothly. However, the Jetson Nano, an embedded development board, only has 4GB of memory, which is

Table 2. Comparison of convolutional neural networks.

Model	Validation loss	Remark
AlexNet	0.026504	Normal
VGG	0.018504	Delay
Resnet18	0.009344	Normal
Resnet34	0.007575	Normal
Resnet101	0.013330	Delay
Resnet152	0.013140	Delay
Resnext101_32x8d	0.013070	Delay
Densenet121	0.013070	Delay
Densenet169	0.010492	Delay

not enough to support smoothly running complex neural networks. Therefore, there are problems with delays during model testing. The delay is hazardous for smart cars, preventing them from predicting in real-time based on the road conditions. The delay increases the likelihood of accidents. Therefore, we chose Resnet18 and Resnet34 for model training and model testing. In addition, we avoided AlexNet despite its lack of delay issues but high validation loss value. It means that AlexNet does not perform well in driving prediction and does not enable automatic driving. Resnet18 and Resnet34 are the two top-performing neural networks, exhibiting negligible latency and very low validation loss values. Finally, for the automatic driving trials, we chose Resnet18 and Resnet34.

3.4 Experiments and methods

In Section 3.4, we introduce three elements. First, we describe the method and operational details of data collection. Second, we describe how model training is performed. These two sections were the preliminaries for the start of the three experiments. Finally, we introduce our three experimental components.

a. Experiment 1: Road tracking in a trained environment. In Experiment 1, we compared the two models, ResNet18 and ResNet34, and the different effects on the smart car when the batch size was equal to 16 and 32. Moreover, we also tested the effect of different speeds on it. Therefore, we are able to find out the appropriate batch size and speed for each model.

b. Experiment 2: Road tracking in a new environment without training. Experiment 2 used Environment 2 to test the smart car to verify whether the model trained from Experiment 1 can achieve road tracking in an unfamiliar environment. At the same time, we can also test the running effect of the four setups and pick the best model and the best speed.

c. Experiment 3: Road tracking & traffic sign recognition. In Experiment 3, we used the best model and the corresponding speed derived from Experiment 2 and applied them in Environment 3. We verified whether smart cars can achieve road tracking along with traffic sign recognition.

3.4.1 Data collection

We collected images using the smart car by taking photographs. Therefore, it is critical to avoid excessive daylight noise. We closed curtains and doors to prevent daylight from influencing the environment. In addition, the room lights were switched on to provide stable light conditions. Data were collected and tested in environments with such light conditions.

We used OpenCV to visualize and save images using tags. Libraries such as UUID and DATETIME are used for image naming; the IPython library is used for display and widgets, and the basic Python library for image tagging.

The neural network in this paper uses an image of 224x224 pixels as input. The image is set to this size to minimize the network. Then, we created two sliders (Fig. 14) to adjust the "x" and "y" values of the smart car. We moved the green dot and blue line of Fig. 14 by sliding the "x" and "y" sliders to get the position information of the smart car. As shown on the right of Fig. 14, this photo has an "x" value of 0.06 and a "y"

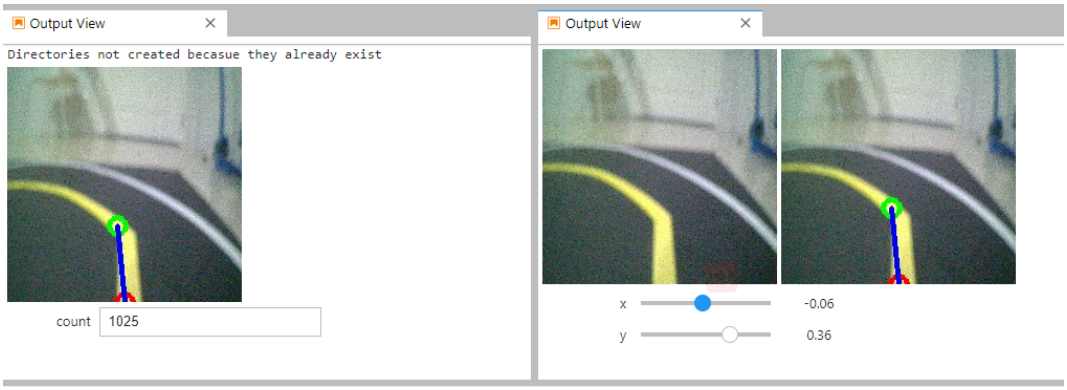


Fig. 14. Data collection diagram.

value of 0.36. At the same time, we created several small windows to observe the number of saved photos. As shown on the left of Fig. 14, we have collected a total of 1025 photos.

Instead of using video to collect images in the simulated environment, we collect images by taking pictures manually. Therefore, there is no frame rate for the model training data. However, during the testing of autonomous driving, the frame rate was 34 FPS, and we captured images in the field of view of the smart car. In Fig. 15, we compare the collected training image with the test screenshot, and they are clear and similar in resolution.

Fig. 15. Comparison of the collected training image and the screenshot of tests.

The specific steps of data collection are shown in Fig. 16.

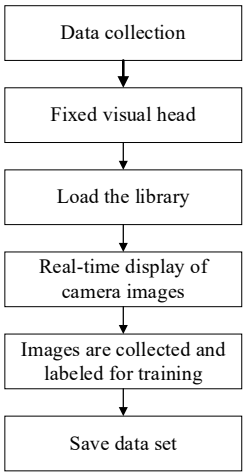


Fig. 16. Diagram of data collection steps.

3.4.2 Model training

Model training is the second and most important step in smart car research. First, we need to configure an environment that conforms to deep learning on the PC side and then use the dataset to train it. The specific steps are shown in Fig. 17.

We divided the collected dataset into two parts; one part was the training set, and the other part was the test set. The training set, which is 90%, is used to train the model, and the test set, which is 10%, is used to verify the accuracy of our trained model.

We performed a total of four setup training sessions. ResNet18 is the official

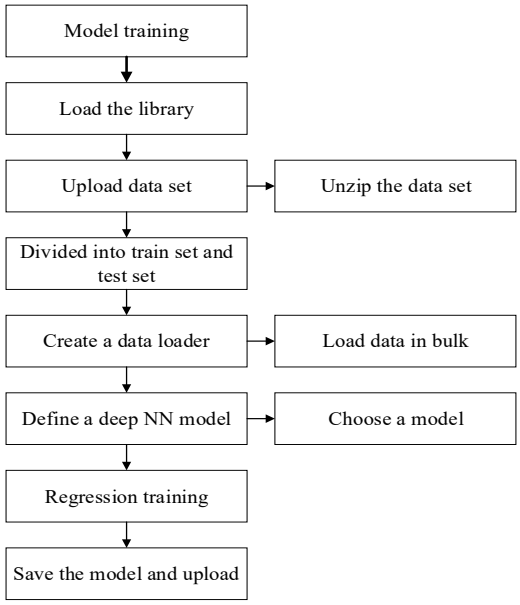


Fig. 17. Diagram of data training steps.

Table 3. Models used in Environment 1.

Setup	Remark
ResNet18_B16	Based on ResNet18, Batch_size is 16
ResNet18_B32	Based on ResNet18, Batch_size is 32
ResNet34_B16	Based on ResNet34, Batch_size is 16
ResNet34_B32	Based on ResNet34, Batch_size is 32

model provided by NVIDIA, while we chose ResNet34 to compare with it. To compare the effect of different Batch_size on the model, we chose Batch_size=16 and 32. Four setups were trained in this experiment (Table 3).

The size of Batch_size affects the degree of optimization of the model, and it directly affects the usage of GPU memory. We need to train the model as the setup shown in Table 3, and each round trains 70 epochs and saves the best model. We found that the epoch to get the best model every time is always between 50-70, more often between 60-70. We do not get a better model when the epoch is greater than 70.

3.4.3 Experiment

We now describe the three experiments in detail.

a. Experiment 1 (Road tracking in the original environment)

Experiment 1 is a road tracking experiment in the original environment (trained environment 1). We collected data through environment 1 and trained the best model. Using environment 1, we compared the effects of speed, model, and Batch_size on the experiment. It is verified that the smart car can achieve road tracking and get the best speed matching the four setups. We chose the best model from the four setups we tested in Environment 1.

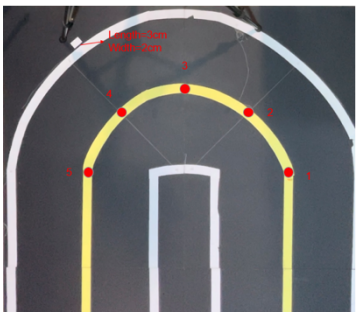


Fig. 18. Measurement points of environment 1.

In Experiment 1, we chose one curve of the track as the experimental object (because the smart car drove well on the straight lane), and the curve was preset with five points as the measurement points for measuring the distance (shown in Fig. 18). Angle (angle between the centerline of the smart car and the tangent line of the measurement point) and distance (distance between the center point of the smart car and the measurement point) are used to judge the degree of autonomy of a smart car. The smaller the angle and the smaller the distance, the smoother the smart car travels. The test data were collected by first taking videos; screenshots are taken and measured from such videos. We set up a tripod for the camera to collect the videos to ensure photo accuracy. We designed a standard rectangle (3cm in length and 2cm

in width), which is used as the scale length standard for the distance measurement later (shown in Fig. 18). Fig. 19 is a screenshot of the video we captured during Experiment 1 for distance and angle measurements.

Table 4. Experimental operation table of the curve.

Model	Speed										
Resnet18_B16	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70
Resnet18_B32	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70
Resnet34_B16	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70
Resnet34_B32	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70

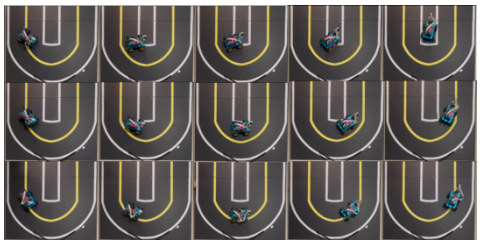


Fig. 19. Some photos of the collection of Experiment 1.

The specific demonstration steps are shown in Fig. 20.

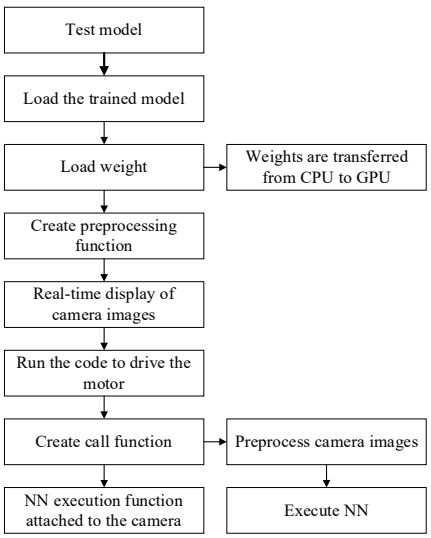


Fig. 20. Step-by-step diagram to demonstrate the model.

We have selected four setups, each of which measures 11 speeds so that 44 videos will be obtained (see Table 4 for details). The range of the smart car speed is between 0 and 1. As shown in Table 4, there is no speed of 0.1 due to ESC failing to start the motor with such speed; when the speed is over 0.70, the smart car is driven over the lines. Therefore, we chose a speed between 0.2 and 0.7.

We have two goals for Experiment 1: To verify that the smart car can execute road tracking in familiar environments, and to find out the four best setups and the corresponding best speed from the experiment results.

b. Experiment 2 (Road tracking in the new environment)

Experiment 2 is a road tracking experiment in the new environment (untrained Environment 2). We used Environment 2 to verify whether road tracking can still be performed in Environment 2. It is difficult for automatic vehicles to collect data on all road conditions and trains. Therefore, a good model should be able to handle unseen environments.

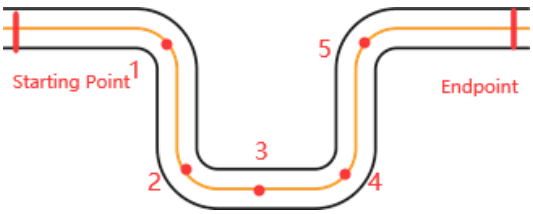


Fig. 21. Measurement points of Environment 2.

As shown in Fig. 21, we selected five locations as data measurement points. As described in the previous section, we also chose the angle between the midline tangent line of the smart car and the measurement

points and the distance between the center point of the smart car and the measurement points as the criteria for judging the goodness of smart cars for automatic driving. The smart car starts from the starting point, passes through 5 locations, and reaches the end. We applied the four setups with the optimal speed obtained in Experiment 1 to Experiment 2. Therefore, we performed four experiments to obtain 20 angles and 20 distances.

We have two goals for Experiment 2: To verify that smart cars can achieve road tracking in unfamiliar environments, and to filter the optimal settings and the corresponding optimal speed from the parameter settings.

c. Experiment 3 (Road tracking & traffic sign recognition)

The goal of Experiment 3 is to execute the recognition of traffic signs (turn left and right signs, as shown in Fig. 22) while carrying out road tracking (verified in Experiment 1 and Experiment 2). Therefore, we set up Environment 3 and added the traffic sign for turning left and right. We collected data from Environment 3 and conducted model training. In this experiment, we test whether the smart car can achieve the goal with the best model and corresponding speed obtained in Experiment 1.

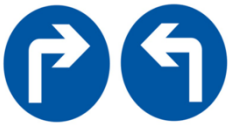


Fig. 22. Traffic signs for left turn and right turn.

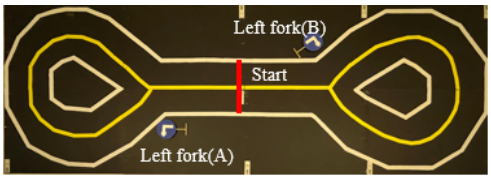


Fig. 23. Explanatory diagram of the experimental operation in Environment 3.

Specific experimental operation: We let the smart car drive in a loop in Environment 3, a cyclic track (as shown in Fig. 23). It will start from the starting position; first, it will arrive at the Left fork (A). We will place a turn sign (left or right) over there beforehand. Then it will arrive at the Left fork(B), where we will place a random turn sign in advance. Furthermore, it will return to the starting point. Thus, it is the process of the smart car cycles once. The smart car completes three cycles in each set of five tests. Moreover, we randomly placed turn signs at the forks six times, as shown in Table 5. Finally, we judged whether the smart car has achieved its goal by turning correctly according to the traffic signs.

Table 5. Operation of Experiment 3.

Group	Fork					
	Left fork(A)	**Left fork(B)**	**Left fork(A)**	**Left fork(B)**	**Left fork(A)**	**Left fork(B)**
Group1	Left	Left	Right	Left	Right	Right
Group2	Left	Right	Right	Right	Left	Right
Group3	Right	Left	Left	Right	Right	Left
Group4	Left	Left	Left	Left	Left	Left
Group5	Right	Right	Right	Right	Right	Right

4. Result

In this chapter, we described the results of our three experiments. Videos of the experiments can be viewed at https://github.com/Bryantding/Automatic-Driving-for-Road-Tracking-and-Traffic-Sign-Recognition.

4.1 Results and discussion of Experiment 1

In Experiment 1, to investigate the effects of models, Batch_size, and speed on smart car road tracking, we chose two ResNet models, two batch sizes, and 11 speeds while ensuring other conditions, such as the dataset were the same.

We chose Batch_size of 16 and 32 for ResNet18 and ResNet34, respectively, and performed three training sessions for each case. The training results are shown in Table 6; *Avg loss* is the average loss value of the three training sessions. Table 6 shows that the size of ResNet18 is 42.6MB, which is smaller than the size of ResNet34 (81.3MB). In addition, based solely on validation_loss values, ResNet18_B32 is a superior model to ResNet18_B16, and ResNet34_B32 is superior to ResNet34_B16. Although ResNet18_B32 obtained the best result (0.0090) in one of the training, ResNet34_B32 had the best result in terms of the average loss of the three training sessions. Losses are the result of the Colab training of the model, which can only be used as one factor in evaluating the effectiveness of the model. Finally, we deployed the trained models on the smart car and used its actual performance as an important criterion for evaluating the models.

Table 6. Table of Validation_loss and size for each model.

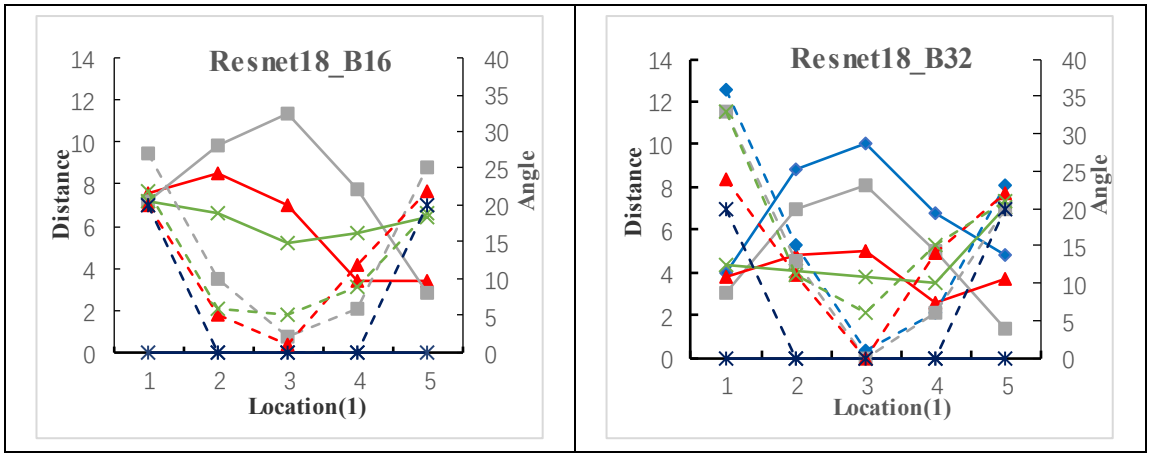
Model	Batch size	Validation_loss		Variance	Size
		Losses	Avg loss		
ResNet18	B16	0.0198 0.0167 0.0215	0.0193	3.67×10^{-6}	42.6MB
	B32	0.0090 0.0156 0.0165	0.0137	1.12×10^{-5}	42.6MB
ResNet34	B16	0.0157 0.0164 0.0134	0.0151	1.65×10^{-6}	81.3MB
	B32	0.0140 0.0120 0.0116	0.0125	1.10×10^{-6}	81.3MB

The results of Experiment 1 are shown in Table 7 and Fig. 24. What needs to be explained is that S0.20-S0.70 means speed from 0.20-0.70; L11-L15 means Location1-5 in Environment 1; "✓" means the smart car does not touch the line, driving normal; S0.60D means the distance when the speed is 0.60; S0.60A means the turning angle when the speed is 0.60; BestD1 is the best distance of the smart car driving through 5 locations in Environment 1; BestA1 is the best angle of the smart car driving through 5 locations in Environment 1. In addition, the underlined data indicates the best result for each group. For Resnet18_B16, because the smart car's speeds are S0.20-S0.50 and S0.70, it would touch the line; we call this situation unqualified road conditions. Therefore, we do not count the results for these speeds in Table 7. For Resnet18_B32, the road conditions would be unqualified when the speeds are S0.20-S0.45 and S0.70. Similarly, for Resnet34_B16, the road conditions would be unqualified when the speeds are S0.20-S0.55 and S0.70. In the same way, for Resnet34_B32, the road conditions would be unqualified when the speeds are S0.20-S0.40 and S0.70.

Table 7. Experimental results for each model of Experiment 1.

Speed	Model	Curve												Remark
		L_i1		L_i2		L_i3		L_i4		L_i5		Average		
		Distance(cm)	Angle	Distance(cm)	Angle	Distance(cm)	Angle	Distance(cm)	Angle	Distance(cm)	Angle	Distance(cm)	Angle	
S0.55	ResNet18_B16	7.22	27	9.87	10	11.39	2	7.72	6	2.86	25	7.81	14	✓
S0.60		7.53	20	8.47	5	6.96	1	3.4	12	3.42	22	5.96	12	✓
S0.65		7.14	22	6.66	6	5.21	5	5.69	9	6.39	19	6.22	12.2	✓
S0.50	ResNet18_B16	3.93	36	8.28	15	10	1	6.76	6	4.81	23	6.77	16.2	✓
S0.55		3.02	33	6.98	13	8.11	0	5.02	6	1.36	20	4.90	14.4	✓
S0.60		3.78	24	4.78	11	5.00	0	2.60	14	3.65	22	3.96	14.2	✓
S0.65		4.34	33	4.08	11	3.76	6	3.52	15	7.09	21	4.56	17.2	✓
S0.60	ResNet34_B16	8.54	31	11.85	8	11.16	7	4.78	20	3.13	30	7.89	19.2	✓
S0.45	ResNet34_B32	4.42	35	10.52	16	13.01	2	11.11	5	9.6	27	9.73	17	✓
S0.50		4.46	28	7.46	14	10.08	0	7.84	7	5.55	22	7.08	14.2	✓
S0.55		2.99	29	6.60	15	8.25	0	6.15	6	2.02	20	5.20	14	✓
S0.60		3.61	27	4.28	10	4.89	1	3.79	9	2.36	18	3.79	13	✓
S0.65		3.24	25	3.33	9	3.13	2	3.22	12	6.19	19	3.82	13.4	✓

First, speed has a significant influence on the quality of driving. Using the ResNet34_B32 model as an example: when the speeds were between S0.20-S0.40, the smart car always touched the line, so it did not drive well at those speeds. When the speeds were between S0.45-S0.65, the distances decreased from 9.73cm (S0.45) to 3.79cm (S0.60) and then increased to 0.82cm (S0.65). Similarly, the angles decreased from 17°(S0.45) to 13°(S0.60) and then increased to 13.4°(S0.65). Finally, when the speeds were over S0.70, it started to touch the line again. In summary, as the speed increased, the driving condition of the smart car improved from poor to good and then deteriorated from good to gradually



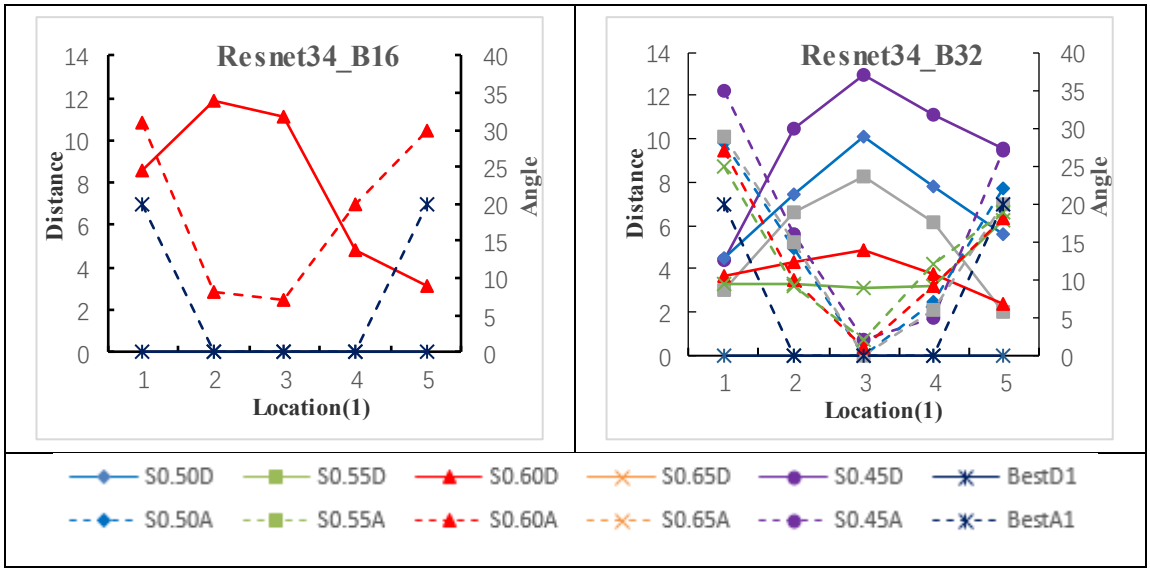


Fig. 24. Experimental results on velocity in environment 1.

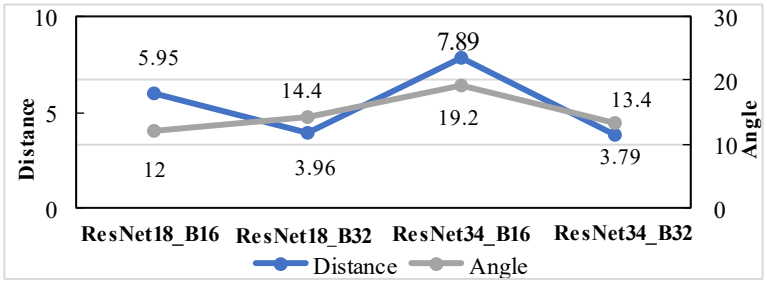


Fig. 25. Comparison of averages for the first part of the experiment in Experiment 1.

worse. The overall driving situation was the best when the speed was set at S0.60. Thus, S0.60 was the optimal speed for these models.

Second, for the same ResNet model, the greater the Batch_size, the better the smart car performed. Using the ResNet34 model as an example the distance of ResNet34_B16 compared with ResNet34_B32 was smaller (3.79 cm) than that of ResNet34_B16 (7.89 cm) at their optimal speed S0.60; the angle of ResNet34_B32 (13°) was smaller than that of the angle of ResNet34_B16 (19.2°).

Third, the model ResNet34 performed better than ResNet18. In Fig. 25, it depicted the distance and angle comparison plots for the best velocities of the four setups, the combined distance, and angle of ResNet34_B32 were better than the two models of ResNet18.

4.2 Results and discussion of Experiment 2

The results of Experiment 2 are shown in Fig. 26, Fig. 27, and Table 8. We applied the best velocity S0.60 of the four setups obtained from Experiment 1 to Environment 2, an unfamiliar and untrained environment. BestD2 was the best distance for the smart car driving through 5 locations in Environment 2; BestA2 was the best angle for the smart car driving through 5 locations in Environment 2. L21- L25 meant

Locations 1-5 in Environment 2, and the other markers were the same as in Experiment 1.

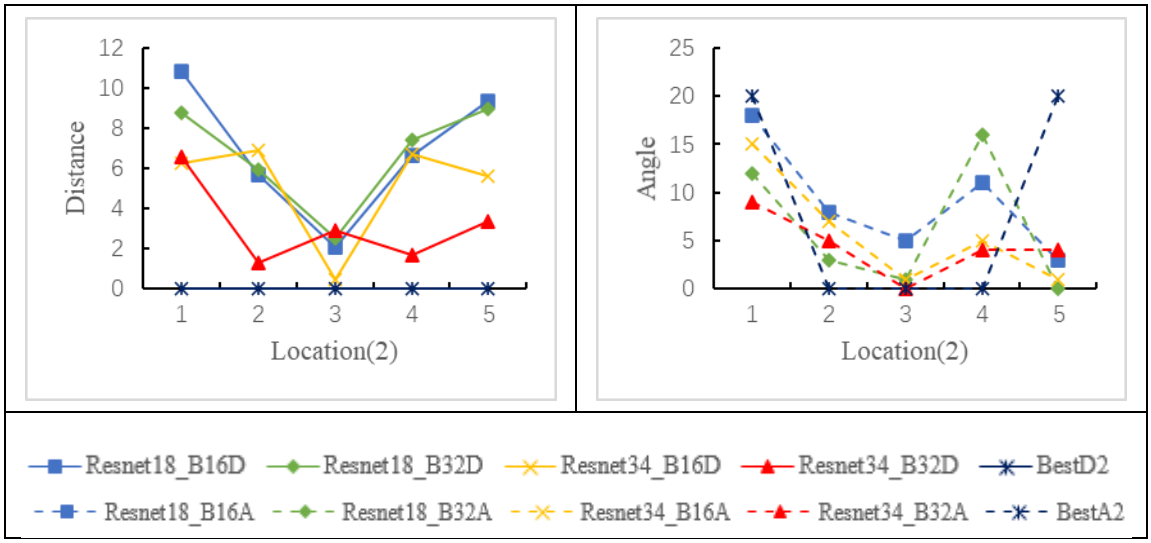


Fig. 26. Experimental results of the four setups in Environment 2.

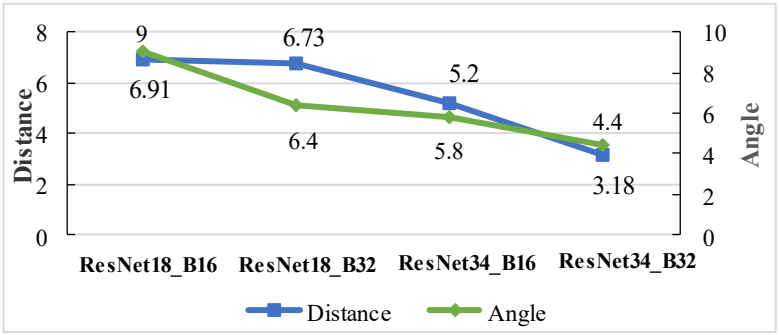


Fig. 27. Comparison of the four setups averages for Experiment 2.

Table 8. Experiment 2 Experimental results for each model.

Model	Curve											
	L_21		L_22		L_23		L_24		L_25		Average	
	Distance	Angle	Distance	Angle	Distance	Angle	Distance	Angle	Distance	Angle	Distance	Angle
Resnet18_B16	10.83	18	5.66	8	2.05	5	6.64	11	9.34	3	6.91	9
Resnet18_B32	8.78	12	5.93	3	2.56	1	7.42	16	8.95	0	6.73	6.4
Resnet34_B16	6.25	15	6.92	7	0.47	1	6.73	5	5.64	1	5.20	5.8
Resnet34_B32	6.61	9	1.31	5	2.93	0	1.68	4	3.37	4	3.18	4.4

It shows the following three results by combining Figs. 26-27, and Table 8.

First, all four setups can achieve road tracking in unfamiliar environments and adapt to new environments that have not

been trained. This means that the results of deep learning of the smart car had wide applicability and practicality. The result makes our study more meaningful because it is ordinary to achieve road tracking in a trained environment. More importantly, we have achieved normal driving of smart cars in unknown environments. We can achieve safe driving of smart cars in more road environments through training in limited road environments.

Second, when the smart car faced an unfamiliar environment, Resnet34_B32 performed best among the four setups. The average distance of its five positions was 3.18cm, and the average angle was 4.4°, which were both the smallest. It is worth noting that in the face of the new environment, the two models of Resnet34 were better than the two models of Resnet18. This means that the Resnet34 model has a stronger ability to predict unfamiliar environments and a stronger ability to adapt to the external environment and resist external noise.

Third, even in the face of unfamiliar environments, the experimental results again proved that for the models of the Resnet series, the larger the Batch_size was, the better the model was and the better it was for prediction. Taking Resnet34 as an example, when the Batch_size increased from 16 to 32, the average distance of five points decreased from 5.20cm to 3.18cm, and the average angle decreased from 5.8° to 4.4°.

4.3 Results and discussion of Experiment 3

In Experiment 1 and Experiment 2, we proved that the smart car could track the road in trained and untrained environments. In Experiment 3, we proved that the smart car could execute the recognition of traffic signs while performing road tracking. We randomly measured five sets of experimental data, as shown in Table 9. "6/6" means that the smart car could make six smooth turns in 6 random forks.

Table 9. Experimental results of Experiment 3.

Group	Fork						Car performs
	Left fork(A)	Left fork(B)	Left fork(A)	Left fork(B)	Left fork(A)	Left fork(B)	
Group1	Left	Left	Right	Left	Right	Right	6/6
Group2	Left	Right	Right	Right	Left	Right	6/6
Group3	Right	Left	Left	Right	Right	Left	6/6
Group4	Left	Left	Left	Left	Left	Left	6/6
Group5	Right	Right	Right	Right	Right	Right	6/6

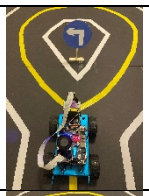
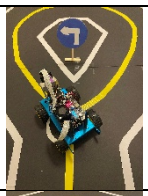
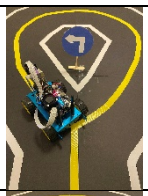
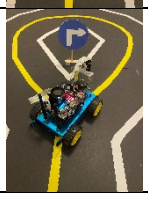
Left fork	Left				
	Right				

Fig. 28. Experiment 3 partial steering diagram.

Fig. 28 illustrates the turning diagram of Experiment 3. From Fig. 28, we can see that when the smart car arrived at two forks, it would make a turn according to the left and right turn signs and continue to drive along the lane line.

As we know from Table 9, all five sets of experimental data are 6/6. This means that when the smart car sees the left and right turn signs while road tracking, it can be recognize them and turn according to the direction without any problem. Furthermore, the accuracy rate is 100%.

5. Conclusion

The achievement of road tracking and traffic sign recognition are two key tasks for automatic driving and are the focus of our current experiments. In addition, we examined the effects of speed, model and Batch_size on automatic driving. We conducted three experiments and came up with the conclusions shown below.

1. We found that speed, model and Batch_size are the key factors affecting automatic driving performance. For the two neural network models we studied, too fast or too slow speeds resulted in poor performance of automatic driving. Under different experimental settings, S0.6 is the optimal speed setting. In Experiments 1 and 2, the results show ResNet34 to be superior to ResNet18. Furthermore, the larger the Batch_size is in the same model, the better the results will be. The complete experimental results show that ResNet34 with Batch_size=32 is the best model, and it performs well in both Environment 1 and Environment 2.

2. The smart car can perform road tracking in trained and untrained environments. In addition, in Environment 1 and Environment 2, the smart car was able to complete its driving in good condition, driving smoothly without touching the line.

3. With only one camera, the smart car can accomplish both road tracking and traffic sign recognition tasks. However, this is not completely driverless, and we will conduct more research in the future.

Acknowledgements

The first author designed and performed the experiments. The second author guided and advised on the experiments, and the two authors jointly drafted the manuscript. Thus, the first and second authors contributed 50% each to this work.

The first author received scholarship support from CPALL for conducting this research in PIM.

References

[1] Shreyas V, Bharadwaj SN, Srinidhi S, Ankith K, Rajendra A. Self-driving cars: An overview of various autonomous driving systems. Advances in Data and Information Sciences 2020:361-71.

[2] Van Brummelen J, O'Brien M, Gruyer D, Najjaran H. Autonomous vehicle perception: The technology of today and tomorrow. Transportation research part C: emerging technologies 2018;89:384-06.

[3] Li B, Hou B, Yu W, et al. Applications of artificial intelligence in intelligent manufacturing: a review. Frontiers of Information Technology & Electronic Engineering 2017;18(1):86-96.

[4] ÖZTÜRK G, KÖKER R, ELDOğAN O, Durmuş K. Recognition of vehicles, pedestrians and traffic signs using convolutional neural networks. 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT). IEEE, 2020: 1-8.

[5] Zhang Q, Du T, Tian C. Self-driving scale car trained by deep reinforcement learning. arXiv preprint arXiv: 1909.03467, 2019.

[6] Li X, Ma H, Wang X, Zhang X. Traffic light recognition for complex scene with fusion detections. IEEE Transactions on Intelligent Transportation Systems, 2017; 19(1): 199-208.

[7] Yuenyong S, and Jian Q. Generating Synthetic Training Images for Deep Reinforcement Learning of a Mobile Robot. Journal of Intelligent Informatics and Smart Technology 2017:16-20.

[8] Iqbal A, Ahmed SS, Tauqeer MD, Sultan A; Abbas SY. Design of multifunctional autonomous car using ultrasonic and infrared sensors. International symposium on wireless systems and networks (ISWSN). IEEE, 2017: 1-5.

[9] Deac MA, Al-doori RWY, Negru M, Blaga BCZ, Dănescu R. Miniature autonomous vehicle development on raspberry pi. 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP). IEEE, 2018: 229-36.

[10] Krauss R. Combining Raspberry Pi and Arduino to form a low-cost, real-time autonomous vehicle platform. American Control Conference (ACC). IEEE, 2016: 6628-33.

[11] Jain AK. Working model of self-driving car using convolutional neural network, Raspberry Pi and Arduino. 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, 2018: 1630-5.

[12] Lee KL, Lam HY. Development of deep learning autonomous car using Raspberry Pi. Progress in Engineering Application and Technology. 2021, 2(1): 534-48.

[13] Du X, Ang MH, Rus D. Car detection for autonomous vehicle: LIDAR and vision fusion approach through deep learning framework. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 749-54.

[14] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM 2017; 60(6):84-90.

[15] LeCun Y. Generalization and network design strategies. Connectionism in perspective. 1989;19:143-55.

[16] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, et al. Backpropagation applied to handwritten zip code recognition. Neural computation 1989;1(4):541-51.

[17] Wu J. Introduction to convolutional neural networks. China: National Key Lab for Novel Software Technology Nanjing University; 2017.

[18] Lavin A, Gray S. Fast algorithms for convolutional neural networks. Proceedings of the IEEE conference on computer vision and pattern recognition 2016: 4013-21.

[19] Rausch V, Hansen A, Solowjow E, Liu C, Kreuzer E, Hedrick J. K. Learning a deep neural net policy for end-to-end control of autonomous vehicles. American Control Conference (ACC). IEEE, 2017: 4914-9.

[20] Do TD, Duong MT, Dang QV, Le MH. Real-time self-driving car navigation using deep neural network. 2018 4th International Conference on Green Technology and Sustainable Development (GTSD). IEEE, 2018: 7-12.

[21] Li Y, Qu J. Intelligent road tracking and real-time acceleration-deceleration for autonomous driving using modified convolutional neural networks. CURRENT APPLIED SCIENCE AND TECHNOLOGY, 2022: 26 pages-26 pages.

[22] Gupta P, Singh V, Parashar A. Smart autonomous vehicle using end to end learning. Journal of Innovation in Computer Science and Engineering, 2020, 9(2): 7-11.

[23] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recogni-tion. Proceedings of the IEEE conference on computer vision and pattern recogni-tion 2016:770-8.