Autonomous Driving Smart Car Based on Deep Learning

Zihao Nie[1] and Jian Qu[2]

[1,2]Faculty of Engineering and Technology, Panyapiwat Institute of Management, Nonthaburi, Thailand
E-mail: 6372100151@stu.pim.ac.th, jianqu@pim.ac.th

Received: December 17, 2021/ Revised: May 26, 2022 / Accepted: June 17, 2022

***Abstract*—Road tracking as an essential task in autonomous driving is crucial for artificial intelligence. Most research is conducted in virtual environments, but it is vital to conduct practical experiments on real cars. The current researchers use toy cars for road track, but the toy cars can only drive at a fixed speed and a fixed angle for steering, which leads to reproduction errors during the experiment. We built a smart car based on a scale model using Jetson Nano as a mainboard, which can adjust the speed and steering gain to improve road tracking performance and reduce reproduction errors. To analyze the impact of hyperparameters, we conducted experiments on 48 autonomous driving models and proposed optimal hyperparameter configuration schemes, and trained the optimal autonomous driving model BH-ResNet. In addition, we also research the effect of the speed and steering gain on the performance of the smart car and propose an optimal gain value. Moreover, we compare BH-ResNet with other existing models, and BH-ResNet outperforms other models, scoring the highest in both tracks, with 94 and 90. Furthermore, the BH-ResNet model can also achieve road tracks with superior performance in unseen scenes, and our proposed model has excellent applicability and practicality.**

***Index Terms*—Autonomous Driving, Convolutional Neural Networks, Deep learning, Deep residual network, Jetson Nano**

I. Introduction

Deep learning is one of the current breakthroughs in artificial intelligence, and the application of deep learning methods to autonomous driving research has solid practical and theoretical significance [1], [2].

Road tracking is the primary task of autonomous driving. In the existing autonomous driving research, most of the research remains in the virtual stage to save costs and ensure the safety of the experiment. Lin et al. chose a software simulator as the experimental environment platform to obtain the relative positions of the car and the road using deep neural networks as the computational framework. Finally, they achieved road track on the simulator track [3]. Although it is convenient and safe to conduct experiments in a virtual environment, this approach is not as accurate as training autonomous cars directly in the real world. Therefore, we propose to make a self-made car and an autonomous driving track in the real world, where the steps of data collection and model testing are done in the world.

Although more and more research is being done to achieve experiments in the real world using self-made cars, most research usually uses toy cars as research cars. Hossain et al. and Karni et al. conducted autonomous driving research by modifying abandoned toy cars, but toy cars are different from real cars [4], [5]. For example, a toy car cannot adjust the speed gain and steering gain like a real car. The toy car can only drive at a fixed speed and a fixed range for steering during the experiment, so the toy car often has reproduction errors, which affects the experimental results. We propose to achieve the autonomous driving experiment by building an autonomous driving smart car, which can replicate the driving situation of the actual car to the greatest extent. In addition, by fine-tuning the speed and steering gain, the autonomous driving performance of the smart car can be improved. For example, it can drive with low steering gain when there are many curves. The second is the use of sensors. Most of the existing research uses sensors to assist the smart car in achieving the autonomous driving experiment, but this makes the smart car not intelligent enough and independent. Banerjee et al. installed radar sensors on the smart car [6], and Yilmaz et al. installed many sensors such as infrared sensors and ultrasonic sensors [7]. However, humans do not use sensors when driving a car and only make judgments through hearing and vision. Therefore, the research needs to achieve the autonomous driving task of the smart car using fewer sensors. This article proposes that the smart car uses only one camera as a sensor for environmental perception to achieve the task of road tracking. As the brain of the smart car, the choice of the mainboard of the smart car is critical. The mainboard with powerful computing power can increase the performance of intelligent agents. Most of the research often uses

Arduino, and Raspberry Pi, as the mainboard of the intelligent agent. Yuenyong et al. used Arduino as the computing platform for reinforcement learning training in the research, but the computing power of Arduino is low, and a computer needs to be used as a backend for computing [8]. Do et al. used the Raspberry Pi for autonomous driving research, which was also limited by the low computing speed and performance of the Raspberry [9]. In order to solve the above problems, this article proposes to use Jetson Nano with superior computing power as the mainboard, which can make the smart car not attached to any back-end and can independently calculate and load the autonomous driving model, and the performance of the smart car is outstanding.

In addition, the construction of the track is also a critical step in achieving the road-tracking task. Both Zhang et al. and Li et al. built a circular track [10, 11], but these tracks are simple, which is not conducive to the test of the steering ability of the smart car. Therefore, we need a more complex track. This article makes two different tracks to meet the track diversity required for autonomous driving.

In achieving road tracking experiments, training deep neural networks and the optimal selection of structural parameters (different networks, batch size, epoch) is a challenging task. Do et al. constructed a new Convolutional Neural Network (CNN) to achieve the road tracking task by mapping the raw input image to a predicted steering angle through the CNN [9]. Rausch V et al. proposed an end-to-end control system based on Convolutional Neural Networks (CNN) for steering autonomous driving cars [12]. In the above research, the difference in the neural network will directly affect the accuracy of the smart car in achieving the automatic driving task. Therefore, this article proposes to use two popular neural networks, ResNet-18 and ResNet-50, to conduct experiments to research the impact of different neural networks on autonomous driving and select the optimal neural network to achieve the road tracking task.

In addition to the neural network, two hyper-parameters, batch size, and epoch, also affect model training results. Radiuk et al. explored the approach of improving the performance of convolutional neural networks and researched the effect of batch size on the network also researched the batch size and found the optimal batch size for training the Deep Q network on the shopping cart system [14]. However, they did not extend their research to autonomous driving. Similar to batch size, the choice of epoch will vary for different research. Chowdhuri et al. and Kocić et al. obtained the minimum error values in different epoch intervals, respectively [15], [16], so selecting different epoch intervals for different experiments is essential. We propose using different batch sizes and epochs to train the autonomous driving model to get the optimal batch size and epoch and their relationship with autonomous driving performance.

In summary, this article proposes to use a scale model to build an autonomous driving car in the real world that uses only one camera as an environment perception sensor, which can drive at an adjustable speed and steering range just like a real car. Furthermore, we choose Jetson Nano as the computing platform, which makes the autonomous driving smart car in this article an independent agent. Since hyperparameters play a vital role in the success of the model training stage, this article discusses and conducts a series of experiments on the effects of different hyperparameters on autonomous driving. Finally, we propose the optimal hyperparameter configuration scheme. In addition, we also discussed the influence of speed gain and steering gain on the smart car and proposed a set of optimal gain values, which further improved the performance of the smart car to achieve road tracks. Finally, we put the smart car in an unseen scene for experiments to verify the applicability of the optimal hyperparameter configuration scheme proposed in this article. In total, we compared 48 sets of hyperparameter configuration schemes, found possible optimal combinations, and trained the optimal model. In addition, we also compared five sets of speed gain values and five sets of steering gain values and found the optimal gain value. In addition, we also compared the optimal autonomous driving model with three groups of other models and achieved road-tracking experiments in unseen scenarios.

II. Literature Review

In recent years, autonomous driving technology has developed rapidly, and there has been much research on the hardware and hyperparameters of autonomous driving smart cars.

A. Hardware Improvement of Autonomous Driving Smart Cars

Research on autonomous driving is usually divided into virtual and real experiments. Most researchers use virtual platforms to ensure the safety and convenience of experiments. Lin et al. achieved road tracking in a virtual environment. After the trained virtual car obtains its position relative to the track, it can use this information as the basis for feedback control and eventually achieve road track in the simulator [3]. However, the results obtained in the virtual environment are usually affected by sunlight, shadows, chromatic aberrations, and noise when migrated to the real world, resulting in poor autonomous driving performance in the real world. Therefore, researchers have gradually used self-made cars to achieve autonomous driving research in the real world.

In this article, autonomous driving smart cars are roughly divided into two categories: toy cars and smart cars. Most research is based on toy cars and modified to build autonomous driving cars. As shown in Fig. 1(a), Hossain et al. aimed to build autonomous driving cars using very low-cost and readily available hardware, so they developed a low-cost mini rover using a toy car that could roam around the area it wanted to observe [4]. As shown in Fig. 1(b), Karni et al. also researched autonomous driving cars based on toy cars, aiming to achieve the task of road tracking [5]. However, the disadvantage of the toy car is that the experiment can only be carried out at a fixed speed and a fixed angle for steering. In the experiment, the actual operation effect of the toy car is often affected because the speed and steering gain cannot be adjusted. Therefore, we improved on this in our research. We propose to build an autonomous driving smart car that can adjust the speed and steering gain to ensure the improvement of the autonomous driving performance of the smart car.

Fig. 1. (a) Autonomous driving toy car by Hossain et al., (b) Autonomous driving toy car by Karni et al.

With the improvement of technology, most research often installs many sensors on the autonomous car to make the car easy to achieve the experimental task. The environmental perception part of the autonomous driving car is divided into two types: a combination of multiple sensors, either only using a camera as a sensor. Iqbal et al. to enable the car to achieve road tracking, infrared and ultrasonic sensors are installed on the autonomous driving car [17]. Banerjee et al. install radar sensors on the autonomous car, and radar sensors detect the safe distance between the vehicle and obstacles [6]. This article proposes to use only a camera as a sensor for environmental perception and the road tracking task, which can make the car imitate human behavior to the greatest extent.

Choosing an excellent mainboard is also an approach to improving autonomous driving performance. Many mainboards cannot achieve deep learning or reinforcement learning tasks independently due to their lack of computational power. A computer is needed as a back-end to assist in achieving the tasks. Yuenyong et al. chose Arduino as the mainboard of a small RC car to achieve reinforcement learning tasks, but the disadvantage is that Arduino is only a specific purpose microcontroller and cannot handle research that requires large-scale computing. They need to use a computer with a GTX 980Ti GPU as the back-end for data calculation and connect the computer to the car using Bluetooth [8]. Do et al. although the use of Raspberry Pi solves the problem that Arduino cannot handle large-scale computing, the performance of Raspberry Pi is also limited due to the lack of a powerful GPU [9]. Therefore, this article proposes to use Jetson Nano as a smart car computing platform, which supports most of the current deep learning frameworks and contains a powerful GPU, which allows the autonomous driving car to achieve all research independently.

As the test link of autonomous driving, the design of the autonomous driving track is critical. As shown in Fig. 2, Zhang et al. and Li et al. only test the road track by building a simple circular track task [10], [11], which is far from enough because the performance of the smart car when passing through the curve can better test the autonomous driving ability of the smart car. Therefore, this article designs two different complex tracks to increase the diversity of autonomous driving tracks and restore the authenticity of racing tracks in real life as much as possible.

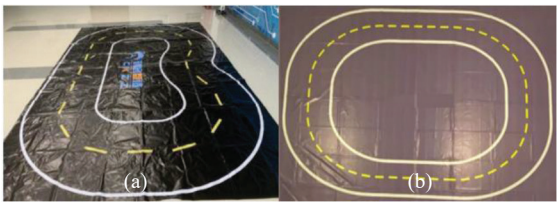


Fig. 2. (a) The track of Zhang et al., (b) The track of Li et al.

B. Research on Hyperparameters in Autonomous Driving

In addition to building an autonomous driving smart car, achieving the task of road tracking also requires setting the optimal hyperparameter configuration scheme before starting the training phase of the neural network. These hyperparameters include the neural network, batch size, epoch, and different datasets. These hyperparameters have a significant impact on the training of the model and the performance of autonomous driving.

1) Deep Neural Network

Autonomous driving technology requires many deep-learning algorithms to process complex data. Deep learning is a multi-layer perceptron that includes an input layer, multiple hidden layers, and an output layer, which can be composed of numerous processing layers. It is very good at finding complex structures in high-dimensional data. Deep learning uses backpropagation algorithms to instruct machines to change their internal parameters to find complex systems in large datasets [18]. Fig. 3 is a multi-layer neural network described by the backpropagation algorithm, which is a three-layer neural network consisting of an input layer with two input units, two hidden layers, and an output layer [19].

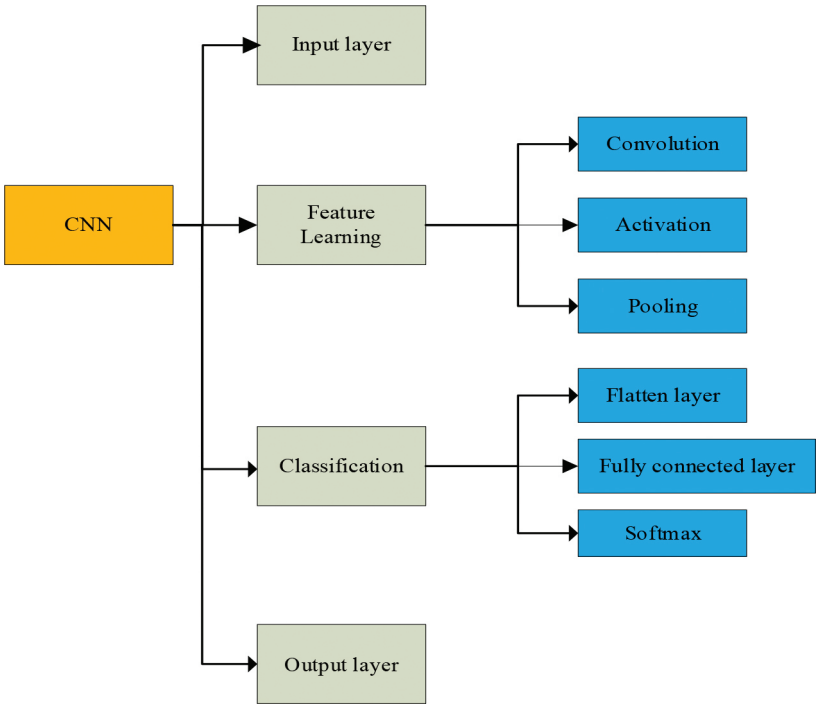


Fig. 3. Convolutional neural network architecture diagram

In deep learning, many deep neural networks are commonly used, among which convolutional neural networks are now more popular.

Convolutional Neural Network (CNN) is a deep learning neural network for image recognition and classification [20]. Each input image in the CNN model goes through a series of convolution layers, pooling layers, and fully connected layers and applies the softmax function to classify the objects. The steps of CNN are roughly divided into four steps: input, feature learning, classification, and output [21]. The feature learning step consists of the convolution layer, excitation layer, and pooling layer. The classification consists of the flattening layer, fully connected layer, and softmax classification layer. The specific steps of CNN are shown in Fig. 4.

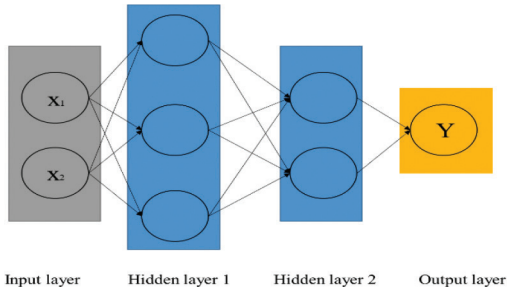
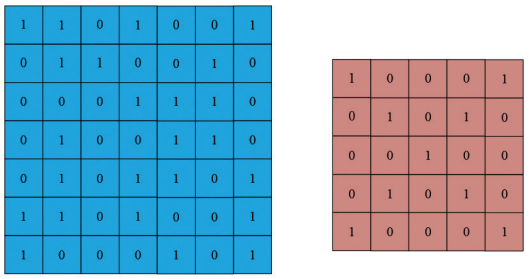


Fig. 4. Neural network architecture diagram

Convolution is the first layer and extracts features from the input image. Convolution uses small squares of input data to learn image features to preserve the relationship between pixels. The purpose of convolution operations is to extract high-level features from the input image, which has profound implications for image processing. For example, Fig. 5(a) is an image matrix with an input image of 7*7, and its image pixel values are 0 and 1. Fig. 5(b) is a 5*5 filter matrix called a convolution kernel.



(a) 7*7 image matrix (b) 5*5 filter matrix

Fig. 5. (a) The input image is an image matrix of 7*7, (b) The filter matrix of 5*5

As shown in Fig. 6(a), multiplying the convolution of the 7*7 image matrix by the 5*5 filter matrix becomes a “feature map”. The value in Fig. 6(b) is 4, which is obtained after one convolution. Convolving the image with different filters (convolution kernels) can perform edge detection and blurring operations. The filter will move to the right by a particular “step value” until the entire image is walked, completing the convolution process.

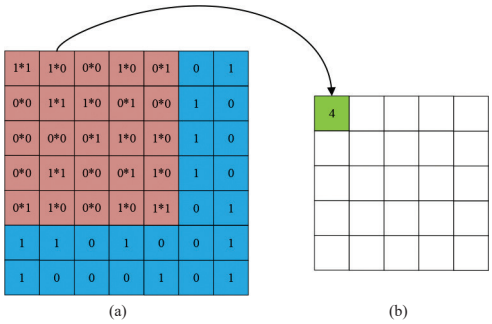


Fig. 6. (a) Process of convolution operation, (b) Results of the convolution operation

After the convolutional layer completes the convolution and extracts the information in the input image, it will perform a nonlinear mapping on the output of the convolutional layer through the excitation layer because the calculation of the convolutional layer is still linear, and the data in the real-world hope what is learned by CNN is a non-negative linear value.

Pooling is also known as spatial pooling. The pooling layer has two functions. First, by reducing the dimension of the feature map, the space size of the convolutional feature is reduced, and the computing power required to process the data is reduced. Second, maintain the process of effectively training the model, extracting essential features invariant to rotation and position. Its most common pooling methods are max pooling and average pooling [22]. The method of the max pooling layer is to use the maximum value of each region of the input part to perform max pooling and generate the max pooling layer, as shown (a) in Fig. 7. The method of the average pooling layer is to use the average value of each region of the input part to perform average pooling (b). Fig. 7 is an example of the average pooling layer.

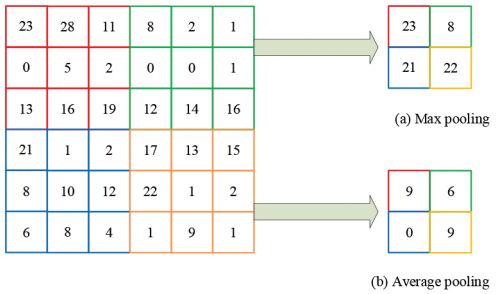


Fig. 7. Example of the calculation process of max pooling and average pooling

The last layer of the CNN starts classification. First, the matrix is converted into a vector by a flattening layer, then sent to a fully connected layer. Adding a fully connected layer is a common way to learn non-linear high-level features, represented by the output of a convolutional layer. The combined method finally classifies the output with a softmax or sigmoid and a classification function.

We summarize the processing steps of convolutional neural networks. First, the input image is provided to the convolutional layer. Second, the parameters are chosen, and filters with stride and padding are applied if necessary. Next, convolve the image and apply an activation function to the matrix. Then merge and reduce the dimensionality. Add as many convolutional layers as possible. Finally, flatten the output and send it to a fully connected layer, use the activation function to output the class, and merge the classified images.

Much research also uses different networks based on CNN for experiments in the current research. Do et al. proposed a 9-layer structure, including five convolutional layers, and four fully connected layers to form a new deep neural network, and build a monocular vision autonomous car with Raspberry Pi as the mainboard, using the end-to-end method to directly map the input image to the predicted steering angle as the output, and finally achieved the road tracking task [9]. Rausch et al. proposed a convolutional neural network consisting of three convolutional layers, two pooling layers, and one fully connected layer for end-to-end driving of the autonomous driving car. The trained terminal controller of the network directly transmits instructions through the mapping relationship between pixel data and steering commands, enabling the smart car to achieve the task of autonomous driving [12]. In the above research, the experimental results are directly related to deep neural networks. Therefore, we propose to use ResNet as the basic model, train different ResNet networks, explore the impact of different neural networks on autonomous driving, and select the optimal neural network to achieve the road tracking experiment.

2) Batch Size and Epoch

Batch size and epoch play an essential role in the model training process. Radiuk et al. researched a parameter of the training set: batch size. The goal was to find out the effect of the batch size on the performance of the neural network. They used the MNIST dataset and CIFAR-10 datasets to obtain consistent results and concluded that batch size affects experimental accuracy [13]. In the research of Choi , he fixed other hyperparameter values, and the neural network was trained for ten different batch sizes and obtained the logarithm of the quadratic relationship between the total training time and batch size [14]. However, they did not extend the results to other research. The research on epoch is also gradually increasing. In training the neural network of autonomous driving, Chowdhuri et al. neneedso choose the epoch that minimizes the average error of the network, which occurs at the 23rd epoch [15]. Kocić et al. proposed

an end-to-end deep neural network, J-Net, for autonomous driving, where J-Net provided the best driving performance when trained for sixth epochs [16]. We found that the epoch interval of the best training model is different, and the choice of epoch affects the fitting degree of the neural network. Therefore, this article will research batch size and epoch and propose their optimal hyperparameters for road-tracking experiments.

To sum up, this article will explore the influence of hyperparameters on model training and the actual operation effect of autonomous driving and propose an optimal configuration scheme of hyperparameters, which will ultimately enable the autonomous driving smart car to achieve the road tracking task. We will elaborate on the construction of the smart car and the autonomous driving track.

III. Our Approach

This article proposes a set of optimal hyperparameter configuration schemes and independently builds an autonomous driving smart car and achieves road tracking in the real world. In addition, we also conducted experiments to adjust the speed and steering gain and proposed a set of optimal gain values, which can further improve the performance of the smart car. We trained the optimal autonomous driving model using the optimal hyperparameter configuration and compared it with existing research and other neural network models. Finally, to verify the applicability of our model, we also test in unseen scenarios. The detailed flow chart of this research is shown in Fig. 8.

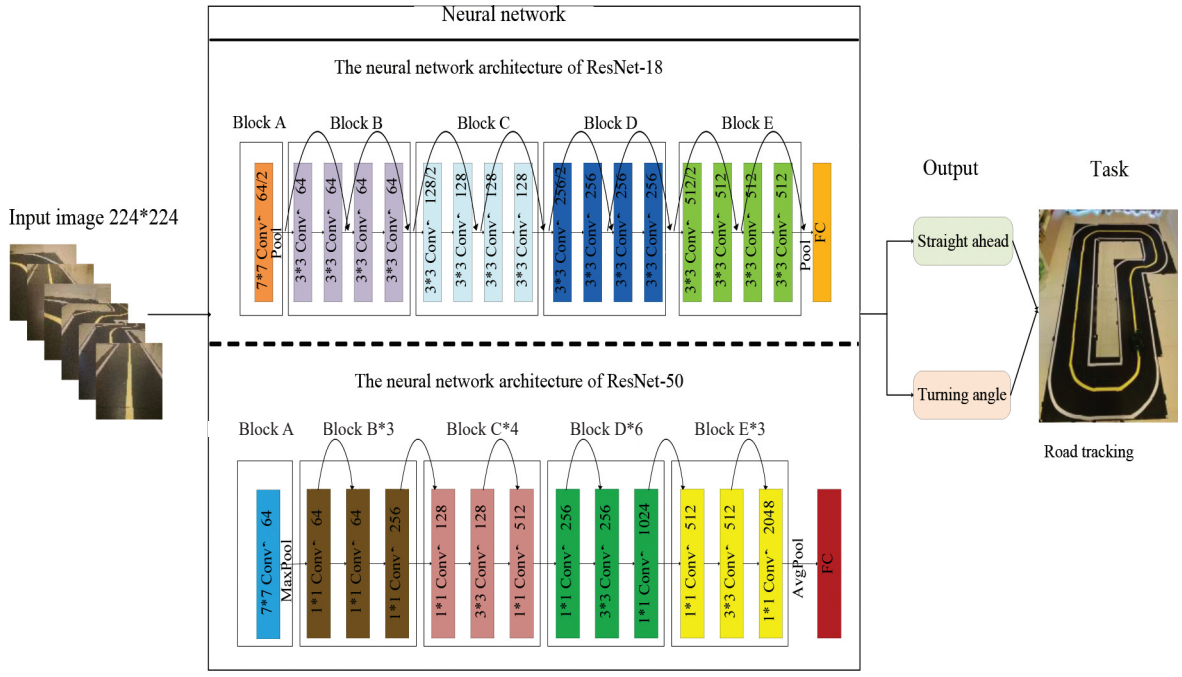


Fig. 8. Overview of the detailed process for achieving road tracking

A. Hardware settings of the Jetson Nano Autonomous Driving Smart Car

The hardware connection diagram of the Jetson Nano autonomous driving smart car is shown in Fig. 9. We chose Jetson Nano as the mainboard of the smart car, and the Jetson Nano is the center to send control signals to various components of the smart car so that the smart car can achieve the road tracking task.

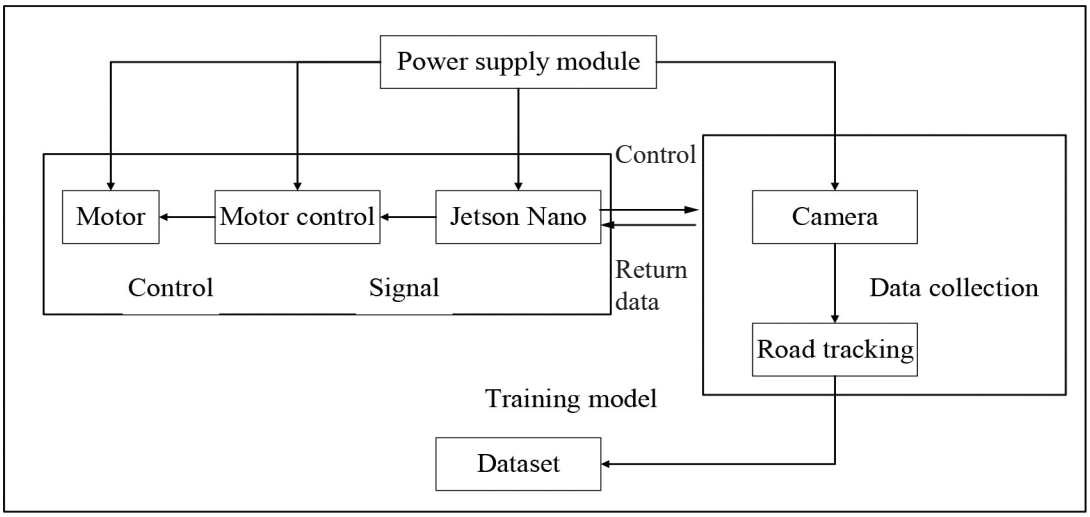


Fig. 9. The hardware framework of Jetson Nano smart car

1) Jetson Nano Mainboard

This article uses a smart car with only one camera as a sensor for research and collects data on the real track through the camera of the smart car for model training. Also, most of the research uses Raspberry Pi or Arduino as the mainboard for intelligent agents, but Arduino alone cannot do deep learning. The computing power and efficiency of the Raspberry Pi are low, and it cannot carry complex deep learning networks. We propose to use Jetson Nano as the mainboard of the smart car, which has a powerful GPU. The CPU of the Jetson Nano is a quad-core Cortex-A57, and the GPU is a graphics card of the NVIDIA Maxwell architecture. It has 128 CUDA units. We train the model using a ResNet network based on the Pytorch framework to recognize lane lines and output driving instructions [23], [24]. A picture of the Jetson Nano is shown in Fig. 10.

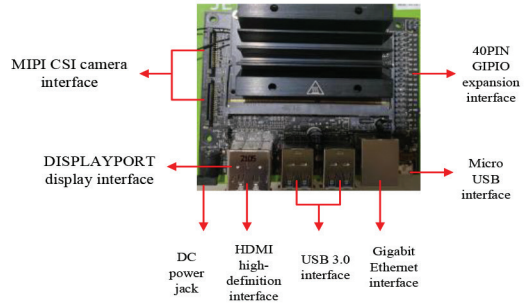


Fig. 10. Jetson Nano mainboard

The difference between the Raspberry Pi and the Jetson Nano is that the Jetson Nano has a higher performance and a more powerful GPU. As shown in Table I, Jetson Nano supports many deep learning frameworks, enabling us to use more complex deep learning models, obtain faster computing speeds, and reduce development time by 70%.

TABLE I
MAINBOARD INFORMATION TABLE

List	Raspberry Pi 4B	NVIDIA Jetson Nano
CPU	Quad-core ARM Cortex-A72 64-bit @ 1.5 GHz	Quad-core ARM Cortex-A57 64-bit @ 1.42 GHz
GPU	Broadcom VideoCore VI (32-bit)	NVIDIA Maxwell w/128 CUDA cores @ 921 Mhz
Memory	4 GB LPDDR4	4 GB LPDDR4
Net working	Gigabit Ethernt/ Wifi 802.1 1ac	Gigabit Ethernt/ M.2 Key E
Display	2x micro-HDMI (up to 4Kp60)	HDMI 2.0 and eDP 1.1
USB	2x USB 3.0,2x USB 2.0	4x USB 3.0, USB 2.0 Micro-B
Other	40-pin GPIO	40-pin GPIO
Video Encode	H264(1080P30)	H.264/H.265 (4Kp30)
Video Decode	H.265(4Kp60), H.264(1080P60)	H.264/H.265 (4Kp60,2x 4Kp30)
Camera	MIPI CSI port*1	MIPI CSI port*2

2) Materials Used in Jetson Nano Smart Car

Using the Jetson Nano mainboard as the core is an excellent choice for building an autonomous driving smart car. The smart car has the ability of independent computing. The smart car perceives the environment through high-definition cameras and uses the trained neural network to achieve the road tracking task. The hardware required to assemble the smart car is shown in Table II.

TABLE II
JETSON NANO SMART CAR MATERIAL TABLE

Serial Number	Part	Specifications/Remarks	Quantity
1	Jetson Nano	-	1
2	IC Expansion Board	-	1
3	Motor	370P	2
4	Servo	-	2
5	Smart Car Chassis	Including Servo and Camera Mount	1
6	Camera	Sony 8 Million HD Camera	1
7	Wireless Network Card	-	1
8	Track	-	1
9	Battery	-	1
10	Smart Car Crawler Gear	-	4

3) Jetson Nano Autonomous Driving Smart Car Achieved

The Jetson Nano autonomous driving smart car we built using the scale model is shown in Fig.11. The toy cars in the existing research have simple structures and can only drive with a fixed speed gain and a fixed steering gain. Therefore, toy cars often have reproduction errors during experiments, and toy cars have poor autonomous driving performance. The smart car in this article can improve the automatic driving performance of the smart car by adjusting the speed and steering gain. In addition, the experimental effects brought by different speeds and steering gains will also be different. For example, if the steering gain is large, the turning range of the smart car will be large when it is in a curve, and sometimes it will drive out of the track. Therefore, we will also research the speed and steering gain of the smart car and strive to propose an optimal gain value to improve the nomous driving performance.

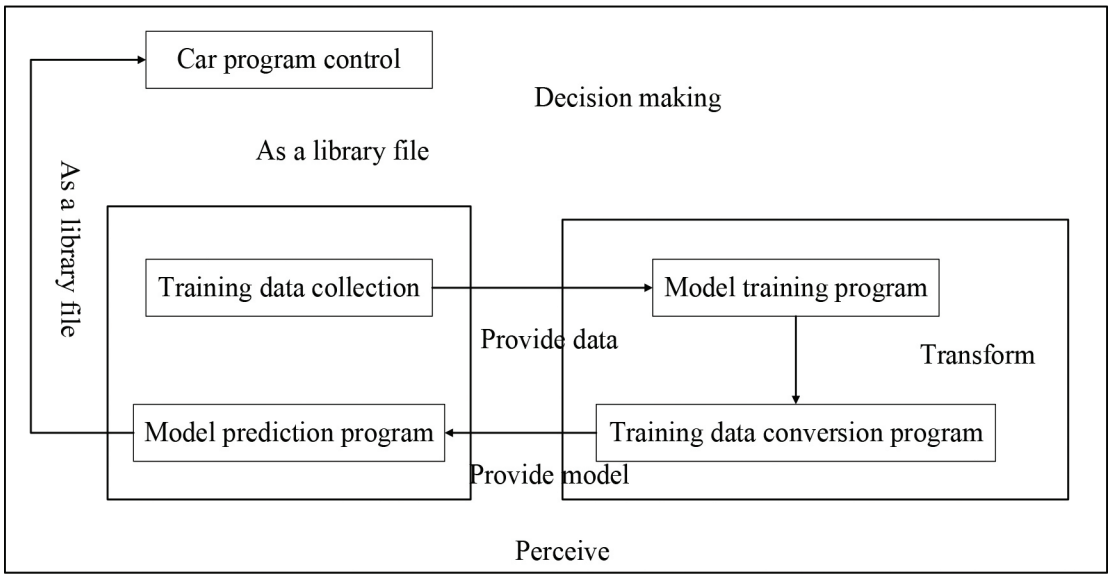


Fig. 11. Flowchart for achieving the road tracking of the Jetson Nano smart car

The achievement of the road tracking task of the Jetson Nano smart car is shown in Fig.12. It can be divided into four parts: the car control program, the training data acquisition program, the model training program, and the model prediction program.

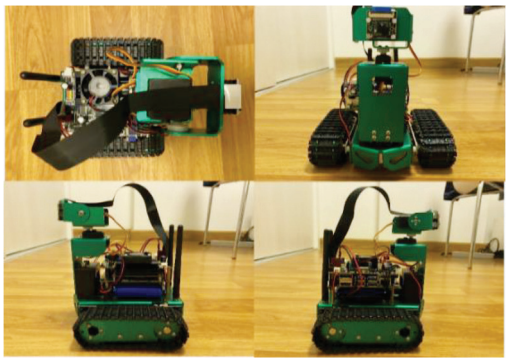


Fig. 12. Overview of the Jetson Nano autonomous driving smart car

B. Simulation Site Construction

We build a simulation site to simulate the driving of an autonomous driving car on the road. We added four corners and an S-curve to a standard circular road to restore the actual route and facilitate testing the steering of the Jetson Nano smart car when turning. The white line is the boundary line of the track, and the yellow line is the lane line of the track. The width of the entire track is 44 cm, and the distance between the yellow and white lines is 22 cm. Using two-lane lines in two colors can give the smart car better results in the road tracking experiment. Finally, we collect data and test the final model in the self-made racing track to determine the performance of road tracking under different neural networks, batch sizes, and different amounts of datasets and epochs. Fig.13 and Fig.14 are schematic diagrams of the smart car simulation site.

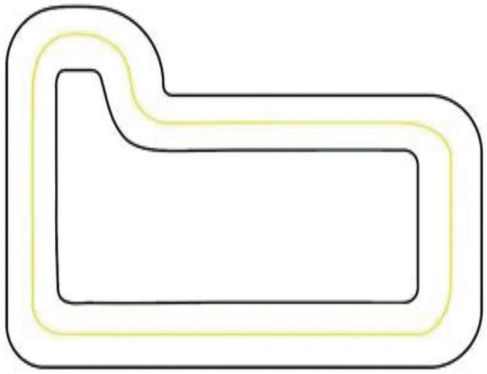


Fig.13. Simulation site model diagram

Fig.14. Actual map of the simulation site

C. Use of Neural Networks

The road tracking of the smart car is to collect a large amount of data and divides it into a training set and a test set. After deep neural network training, a model is formed, and the road tracking experiment is achieved. In this experiment, the lane lines taken by the camera of the smart car on the self-made track will be used as training data. The selected deep neural network is ResNet, and different ResNet neural networks are used to train under different epochs and batch sizes. Finally, the quality of the training results is compared. However, different ResNet neural networks achieve different effects. To improve the accuracy and impact of the model as much as possible while the smart car can be within the limits of Jetson Nano memory and computing power, we must rationally use different ResNet neural networks and choose an appropriate number of datasets, epochs, and batch sizes. This research compares the verification loss value of the ResNet-18 neural network and the ResNet-50 neural network with a certain number of datasets, epochs, and batch sizes and analyzes the impact of these hyperparameters on the verification loss value. Moreover, we will also load the trained model into the smart car for actual operation and analyze the real operation effect.

IV. Experimental Setup

The experimental process is divided into data collection, model training, and model testing. Finally, an autonomous driving model is trained to enable the Jetson Nano smart car to achieve road tracks.

In the data collection stage, we placed the smart car at different positions on the track and used the real-time camera as input for data collection. We collected six sets of datasets, respectively, and this is to compare the autonomous driving performance of the smart car with different numbers of datasets.

During the model training stage, different hyperparameters have a significant impact on the performance of the autonomous driving model. Therefore, we train the autonomous driving model using different deep neural networks, batch sizes, epochs, and different datasets and finally propose a set of optimal hyperparameter configuration schemes. Finally, we load the trained model on the smart car to achieve road tracks. We also analyzed the number of datasets and the impact of different hyperparameters on autonomous driving according to the actual operation effect and verification loss value. The experimental procedure diagram is shown in Fig.15.

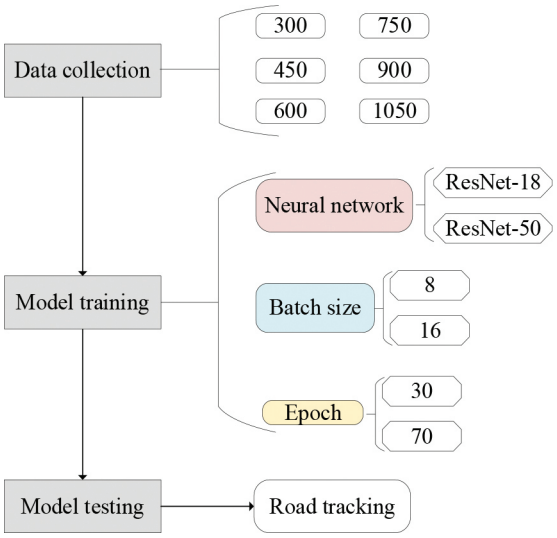


Fig. 15. Experimental procedure diagram

In addition, we also research the effect of different speed and steering gains on the performance of road tracking and propose a set of optimal gain values. Finally, to verify the applicability of our optimal autonomous driving model, we will achieve the road tracking task in unseen and untrained scenarios.

A. Data collection

The first step in this experiment is data collection. The quality of the training dataset will directly affect the performance of the autonomous driving model. The steps of data collection are shown in Fig.16.

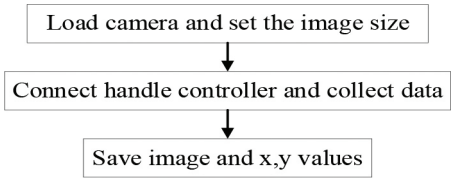


Fig. 16. Diagram of data collection steps

We initialize and display our camera. We choose to use a 224x224 pixel image as input. We set the photo to this size to minimize the memory of the dataset and speed up the training of the model.

Then the approach we take photos with is handle shooting. We create an instance of the PlayStation controller and collect images through the buttons on the controller.

We will place the smart car on different positions of the track according to the lane line and move the "x" and "y" sliders to mark the "green dot" in the center of the lane line during the road tracking experiment. The position marked by the green dot is the target position to be reached when the smart car drives. After the action is completed, press the "L1" button on the handle to save. At the same time, we will create a component to display the real-time image feed, the number of collected images, and the value of the storage target. It can be seen in Fig. 17 that the number of collected data is 1050 photos, and the method of moving the green dots can be seen in Fig. 18.

Fig. 17. Live preview of data collection (1050 represents the number of photos collected as 1050)

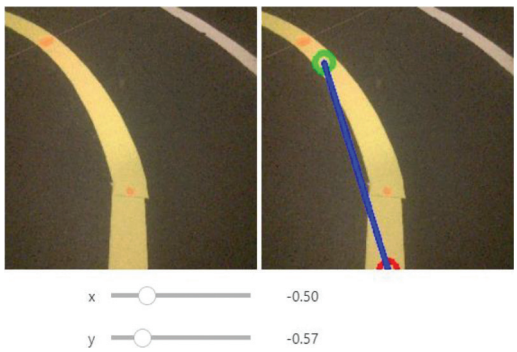


Fig. 18. The method of collecting points (drag X can change the left and right position of the green dot, and drag Y can change the distance of the green dot)

Finally, the collected data set is automatically saved to the corresponding new folder after collecting the corresponding data. When we train, we will transfer the data to the PC to load the image and parse the x and y values in the filename. Fig.19 is an example of the contents of the data folder. Each photo in the dataset folder is named with its x and y coordinates.

Fig. 19. Example of the contents of the data folder

B. Model Training

We can train the optimal autonomous driving model through model training, and this step is essential. The model training part aims to use the trained artificial neural network to reproduce the values when collecting the data: the x and y values. The steps of model training are shown in Fig. 20.

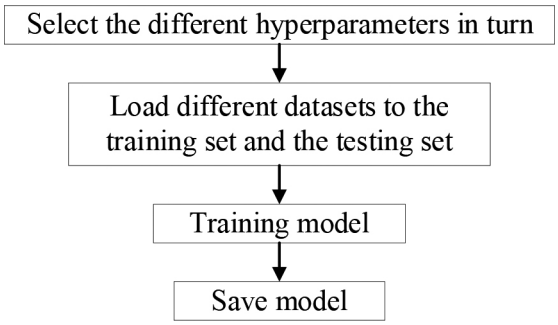


Fig. 20. The steps of model training

In training the model, the dataset will be divided into the training set and testing set. This research split the data into 90% as the training set and 10% as the testing set. The training set is used to train the autonomous driving model, and the testing set will be used to verify the accuracy of our trained model. Then we load the data in batches and shuffle the data. Most of the current research usually sets batch size=8, epoch=30 for training. Therefore, our initial batch size is set to 8, and the initial epoch is set to 30. we increase the batch size and epoch to 16 and 70, respectively, depending on the GPUs available in memory. This allows us to research the effect of different batch sizes and epochs on the autonomous driving model.

Next, we use the now popular ResNet-18 and ResNet-50 neural networks to train the model and transfer it to the GPU to run the model training through "CUDA". Once the model is trained, it will generate a model file that we will use for road tracking.

Table III shows the 48 hyperparameter settings in this experiment. They are trained with different neural networks, different numbers of datasets, different batch sizes, and epochs.

TABLE III
HYPERPARAMETER SETTINGS for ROAD TRACKING MODELS

Neural Network	Dataset	Batch Size	Epoch
ResNet-18	300	8	30
	450		
	600		
ResNet-50	750	16	70
	900		
	1050		

Through model training, we can get the validation loss value of each model. The validation loss value can reflect the performance of the model. We also load the trained autonomous driving model on the smart car for experiments, which can more accurately judge the impact of different hyperparameters on autonomous driving. We can get accurate results by combining the verification loss value with the actual operation effect of the smart car.

C. Model Testing

Through the training of the models, we were able to obtain the validation loss values for each model separately. The validation loss values can show the performance of the models on the data, and in order to get more accurate results, we also need to combine the real effects of the smart car operation, so we started the final step of the experiment by loading the trained models on the smart car and testing the models on a real track.

We can reduce the reproduction errors of toy cars often by setting the speed and steering gain. We drive the smart car with an initial speed and initial steering angle of 0.65 and 0.21, according to the design of the smart car. After obtaining the influence of hyperparameters on the autonomous driving model and proposing the optimal hyperparameter configuration scheme, we conduct research on the influence of the gain value on the autonomous driving model by adjusting the speed and steering gains and propose an optimal gain value to improve the road tracking performance.

When the smart car is driving in a straight line, no matter what dataset, neural network, batch size, or epoch, the smart car can achieve excellent performance. However, when the smart car is driving on a curve, the performance of different autonomous driving models can be demonstrated.

Therefore, to accurately compare the actual performance of autonomous driving models trained with different hyperparameter configuration schemes, we selected three points as measurement points at the S-turn of the self-made racing track.

We will judge the performance of the smart car according to two judgment criteria. 1) The distance between the center of the smart car and the measurement point when the smart car is driving. The smaller the distance between the centerline of the smart car and the measurement point, the better road tracking performance. 2) When the smart car is driving, the angle between the center of the smart car and the measurement point. The smaller the angle that the centerline of the smart car deviates from the measurement point, the better the performance of road tracks. The autonomous driving model it loaded will be better. In addition, we also marked a standard block on the map, and we can scale the standard block to the actual distance to get the accurate distance and angle. As shown in Fig. 21, P1, P2, and P3 are three measurement points, respectively. The white squares marked with green fonts are the standard blocks we reserved. Fig. 21(a) is the actual track map with the measurement points and the standard block marked. Fig. 21(b) is the track model diagram with marked measurement points and the standard block. Moreover, we also marked the detailed dimensions of the track in the track model diagram.

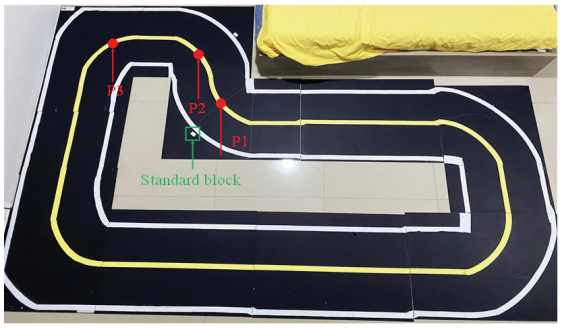


Fig. 21(a). The actual track map with the measurement points and standard blocks marked

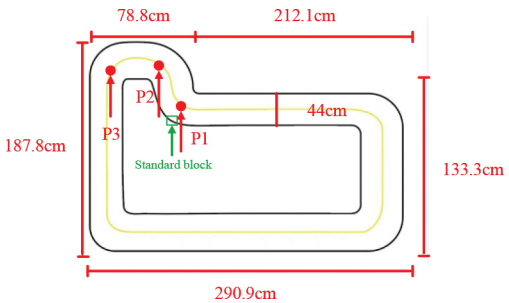


Fig. 21(b). The track model diagram with marked measurement points and the standard block, and detailed dimensions of the track

We shot the video with a camera and tripod during the experiment, took screenshots when the center of the smart car passed the measurement point, and measured. During the experiment, we will keep the light consistent and the measurement height and position unchanged to ensure the accuracy of the experiment. The specific experimental situation is shown in Fig. 22.

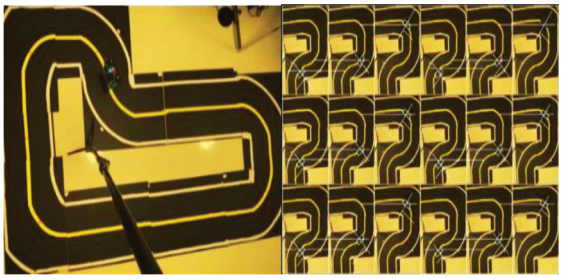


Fig. 22. Specific experimental situation diagram

We will obtain the performance of 48 models trained by different hyperparameter configuration schemes in actual operation through the above experiments. The next chapter will discuss obtained validation loss values and actual measurements.

V. Experimental Results and Analysis

This chapter will describe and analyze the impact of different neural networks, datasets, batch sizes, and epochs on the validation loss and the actual operation effect of smart cars. Finally, we propose an optimal hyperparameter configuration scheme and train an optimal autonomous driving model.

A. The Impact of Different Neural Networks, Different Datasets, Batch Sizes, and Epochs on the Verification Loss Value

The validation loss value can reflect the performance of the model. Table IV shows the validation loss values of our 48 sets of autonomous driving models trained with different hyperparameter configuration schemes to analyze the effect of different hyperparameters on the validation loss values.

TABLE IV
THE VALIDATION LOSS VALUES OF 48 SETS OF AUTONOMOUS DRIVING MODELS TRAINED WITH DIFFERENT HYPERPARAMETER CONFIGURATION SCHEMES

Neural Network	Dataset	Batch Size	Epoch	Loss	Neural Network	Dataset	Batch Size	Epoch	Loss
ResNet-18	300	8	30	0.011001	ResNet-50	300	8	30	0.013148
			70	0.010553				70	0.005804
		16	30	0.011973			16	30	0.011661
			70	0.007873				70	0.006367
	450	8	30	0.006913		450	8	30	0.01226
			70	0.008783				70	0.009035
		16	30	0.012878			16	30	0.01102
			70	0.006626				70	0.011696
	600	8	30	0.011351		600	8	30	0.008344
			70	0.008674				70	0.008863
		16	30	0.011769			16	30	0.011322
			70	0.011649				70	0.008776
	750	8	30	0.008598		750	8	30	0.007925
			70	0.011314				70	0.009001
		16	30	0.01077			16	30	0.011043
			70	0.009004				70	0.009863
	900	8	30	0.011724		900	8	30	0.018298
			70	0.014531				70	0.015377
		16	30	0.019155			16	30	0.018096
			70	0.012837				70	0.011401
	1050	8	30	0.0214		1050	8	30	0.018373
			70	0.015585				70	0.019586
		16	30	0.01889			16	30	0.015242
			70	0.021052				70	0.019499

1) The Impact of Different Neural Networks on Verification Loss Value

Based on the experimental results of 48 sets of autonomous driving models, we judge the influence of different neural networks on the validation loss value. Fig. 23 is a plot of validation loss results trained with ResNet-18 and ResNet-50. The horizontal ordinate is a different setting. For example, "P300_B8_E30" means that the dataset is 300 photos, batch size=8, epoch=30, the blue line represents the model trained with the ResNet-18 neural network, and the red line represents the model trained with the ResNet-50 neural network.

As shown in Fig. 23, when the model is trained with ResNet-50, the validation loss value is generally lower than the model trained with ResNet-18. As the depth of the network deepens, the degree of abstraction of features is higher. The accuracy of the trained model will increase, and the validation loss will decrease,

The maximum value appears in "ResNet-18_P1050_B80_E30", the validation loss value is 0.0214, and the minimum value appears in "ResNet-50_P300_B8_E70", and the validation loss value is 0.005804.

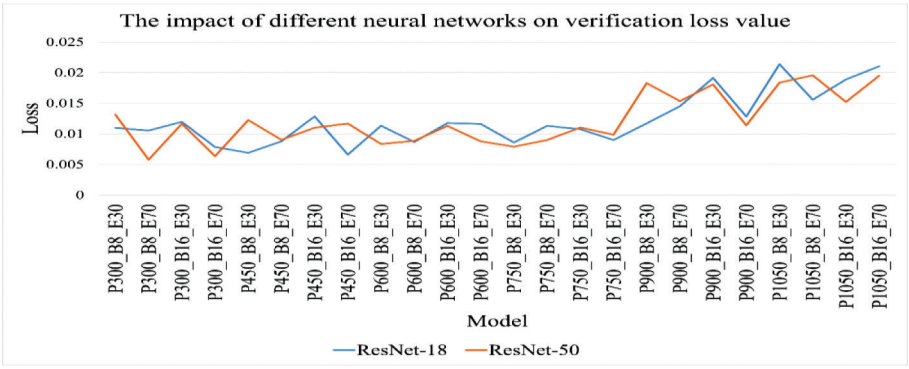


Fig. 23. The results of verification loss for different neural networks

2) The impact of different numbers of datasets on verification loss value

The amount of data in the dataset also affects model training. In this research, six datasets with different numbers of data were selected for training, and the impact of dataset on the model was explored. Fig. 24 shows the validation loss results when training with six datasets.

As shown in Fig. 24, when the dataset is 300 photos, 450 photos or 900 photos, 1050 photos, the validation loss value will fluctuate wildly. Furthermore, the validation loss is significantly reduced when using the ResNet-50 neural network. Through experiments, the number of datasets for the ResNet-18 neural network is not inversely proportional to the validation loss value, but for the ResNet-50 neural network, the larger the number of datasets, the smaller the validation loss value. Therefore, the size of the dataset has a more significant impact on the deep network. As the network depth deepens, the amount of data required to train the model also needs to increase.

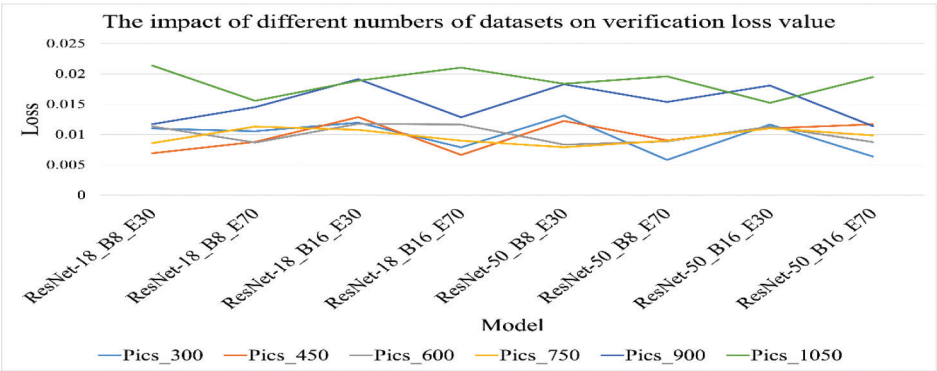


Fig. 24. The results of verification loss for different numbers of datasets

3) The impact of Different Batch Sizes on Verification Loss Value

Batch size is a crucial hyperparameter in deep learning. Batch size will affect memory utilization, processing speed, and model accuracy. Therefore, the choice of batch size also affects the quality of the model.

Most research usually sets the batch size to 8. We increase the batch size during training to 16 according to the available GPU in memory, set batch size=8 and 16 for model training, respectively, and then judge the impact of batch size for model training and validation loss.

Fig. 25 shows the validation loss results for batch size=8 and 16 training. It can be seen that the small batch size has little effect on the model validation loss.

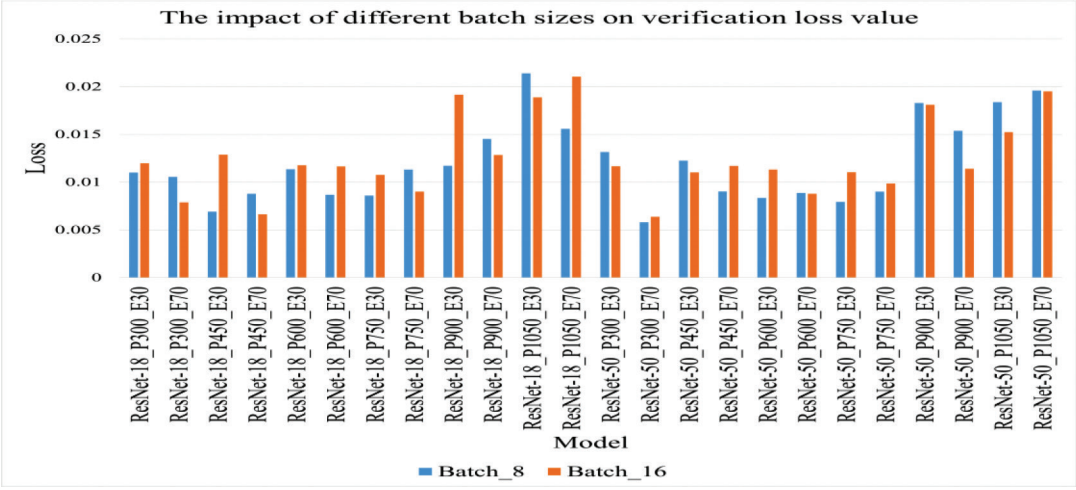


Fig. 25. The results of verification loss for different batch sizes

4) The Impact of Different Epochs on Verification Loss Value

Epoch refers to the number of times to train all data. Most research usually sets the epoch to 30 for model training. We also increased the epoch value to 70 according to the available GPU conditions and analyzed the impact of different epochs on the model validation loss value.

Fig. 26 is a plot of the validation loss results for training with epoch=30 and 70. As shown in Fig. 26, the validation loss value of the model with epoch=70 is smaller than epoch=30. Therefore, on the premise that the model does not enter the overfitting state, the more iterations of weight update, the more epoch, the smaller the validation loss of the model, and the better the effect of training the model.

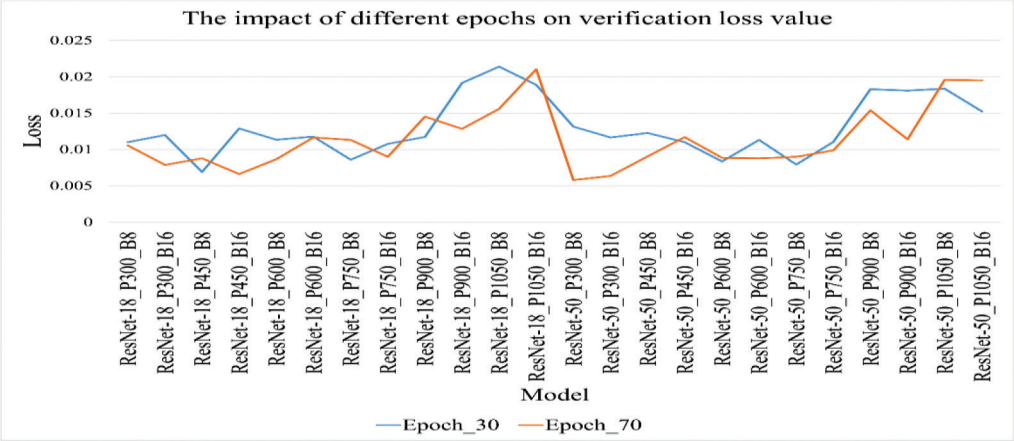


Fig. 26. The results of verification loss for different epochs

5) Summary of Validation Loss Values of Models Trained Under Different Hyperparameter Configuration Schemes

This experiment trains 48 autonomous driving models according to different hyperparameter configuration schemes. Fig. 27 shows the validation loss values for 48 autonomous driving models. From Fig. 27, it can be seen that the validation loss values of the autonomous driving model trained with ResNet-50 are smaller than those of the model using ResNet-18. Among them, when the dataset is 300 photos, batch size=8, epoch=70, the validation loss is the smallest, 0.005804.

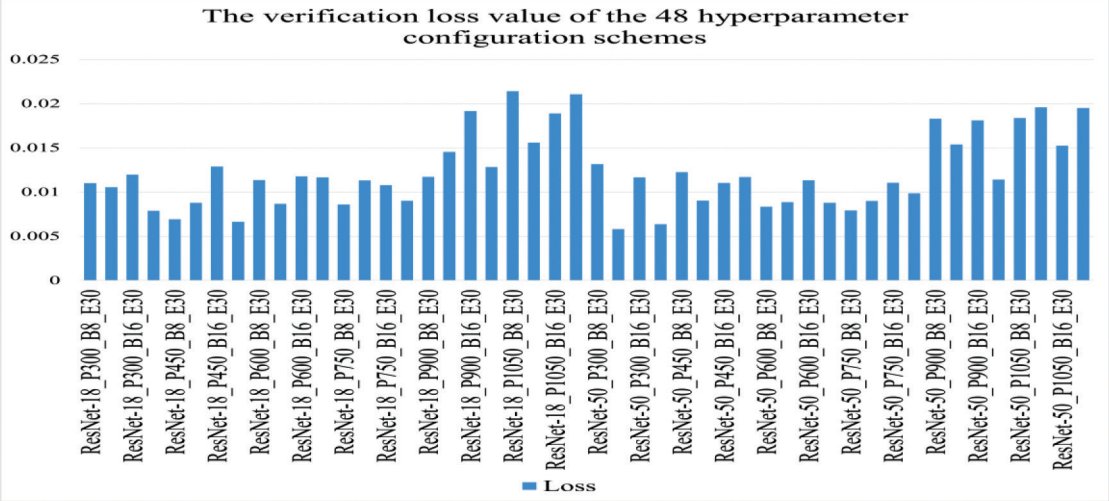


Fig. 27. The validation loss values for 48 autonomous driving models

Although the validation loss value can reflect the performance of the model, when the models are trained with ResNet-18, the difference in the validation loss value of each model is very small.

Therefore, to get accurate results, we need to evaluate further the quality of the autonomous driving model based on the actual road tracking performance after loading the model into the smart car. Finally, an optimal hyperparameter configuration scheme is proposed.

B. The Impact of Different Neural Networks, Different Datasets, Batch Sizes, and Epochs on the Actual Performance of the Smart Car

We load the trained model on the autonomous driving smart car. We analyze the performance of autonomous driving models trained by different hyperparameter configuration schemes by comparing the average offset distance and average offset angle of the smart car when passing through 3 measurement points and analyze different neural networks, different datasets, batch sizes, and epochs on the actual performance of the smart car.

Table V shows the actual operating results of the smart car after loading different autonomous driving models. The neural network used in Table V (A) is ResNet-18, and the neural network used in Table V (B) is ResNet-50.

TABLE V (A)
THE ACTUAL OPERATION RESULTS OF THE SMART CAR (RESNET-18)

Neural Network	Dataset	Batch Size	Epoch	Distance (P1 P2 P3) cm			Average Distance	Angle (P1 P2 P3)°			Average Angle
ResNet-18	300	8	30	5.19	2.61	2.24	3.35	16	6	1	7.67
			70	6.2	2.5	0.72	3.14	3	8	5	5.33
		16	30	5.4	1.94	0.88	2.74	10	1	4	5
			70	4.56	0.47	4.36	3.13	7	7	5	6.33
	450	8	30	5.66	1.16	2.66	3.16	1	11	5	5.67
			70	7.51	1.06	0.75	3.11	10	26	19	18.33
		16	30	7.38	1.1	2.8	3.76	3	1	7	3.67
			70	5.86	0.64	4.35	3.62	1	0	11	4
	600	8	30	4.42	0.91	0.77	2.03	4	2	6	4
			70	3.99	1.66	2.4	2.68	9	4	0	4.33
		16	30	8.46	3.77	1.86	4.7	1	6	2	3
			70	3.91	1.88	1.05	2.28	5	3	3	3.67
	750	8	30	5.24	2.09	3.04	3.46	4	2	6	4
			70	5.99	1.81	1.3	3.03	5	0	4	3
		16	30	6.84	2.65	0.66	3.38	8	2	5	5
			70	6.39	1.13	0.62	2.71	9	0	11	6.67
	900	8	30	6.59	2.75	0.44	3.26	6	3	5	4.67
			70	5.6	1.34	1.44	2.79	7	7	2	5.33
		16	30	5.11	2.47	1.4	2.99	17	1	11	9.67
			70	5.6	3.18	1.25	3.34	6	5	4	5
	1050	8	30	7.66	1.92	1.09	3.56	3	11	3	5.67
			70	6.37	1.99	1.48	3.28	11	5	3	6.33
		16	30	6.28	1.6	0.9	2.93	8	2	6	5.33
			70	5.68	1.83	0.66	2.72	9	8	12	9.67

TABLE V (B)
THE ACTUAL OPERATION RESULTS OF THE SMART CAR (RESNET-50)

Neural Network	Dataset	Batch Size	Epoch	Distance (P1 P2 P3) cm			Average Distance	Angle (P1 P2 P3)°			Average Angle
ResNet-50	300	8	30	4.98	36.45	95.36	45.6	15	96	3	38
			70	7.32	33.9	65.65	35.62	3	79	25	35.67
		16	30	5.63	22.34	32.22	20.06	10	18	7	11.67
			70	3.3	26.41	51.6	27.1	61	44	41	48.67
	450	8	30	7.59	18.09	24.76	16.81	0	10	17	9
			70	2.16	23.52	88.96	38.21	22	76	153	83.67
		16	30	2.17	5.89	10.07	6.04	9	1	33	14.33
			70	4.52	6.32	13.45	8.1	43	4	74	40.33
	600	8	30	4.39	2.16	42.72	16.42	14	11	139	54.67
			70	3.61	4.35	45.93	17.96	25	28	145	66
		16	30	2.59	4.31	15.88	7.59	5	19	14	12.67
			70	2.07	22.42	10.36	11.62	21	27	118	55.33
	750	8	30	0.92	19.66	23.54	14.71	27	30	44	33.67
			70	1.72	12.71	7.76	7.4	1	115	39	51.67
		16	30	3.01	27.93	17.05	16	19	14	61	31.33
			70	3.46	33.76	96.78	44.67	86	1.3	53	46.77
	900	8	30	7.19	23.37	57.02	29.19	39	67	15	40.33
			70	5.78	23.76	60.94	30.16	54	131	114	99.67
		16	30	8.89	11.37	13.4	11.22	13	39	14	22
			70	5.75	4.79	24.67	11.74	71	13	38	40.67
	1050	8	30	4.54	5.13	20.14	9.94	27	16	62	35
			70	3.25	14.77	3.11	7.04	7	4	34	15
		16	30	7.12	6.39	4.43	5.98	19	19	23	20.33
			70	1.36	23.61	6.01	10.33	33	0	74	35.67

1) The Impact of Different Neural Networks on the Actual Operation Effect of the Model

We judge the impact of different neural networks on model quality by analyzing the actual effect of smart car operation. Fig. 28 is the actual operation of the model trained with ResNet-18 and ResNet-50. The horizontal ordinate is the autonomous driving model trained with different hyperparameter configuration schemes, the ordinate is the average offset distance when the smart car is driving, and the second coordinate is the average offset angle.

According to Fig. 28, we can see that although the validation loss value of ResNet-50 is generally smaller than that of ResNet-18, the actual performance of the model trained with ResNet-18 is significantly better than that of ResNet-50. Moreover, as the number of photos in the dataset increases, the actual performance of ResNet-50 gradually improves. Therefore, in experiments with small datasets, even if a deeper network can bring better nonlinear expression capabilities, the actual effect of the ResNet-50 neural network is still far worse than that of the ResNet-18 neural network.

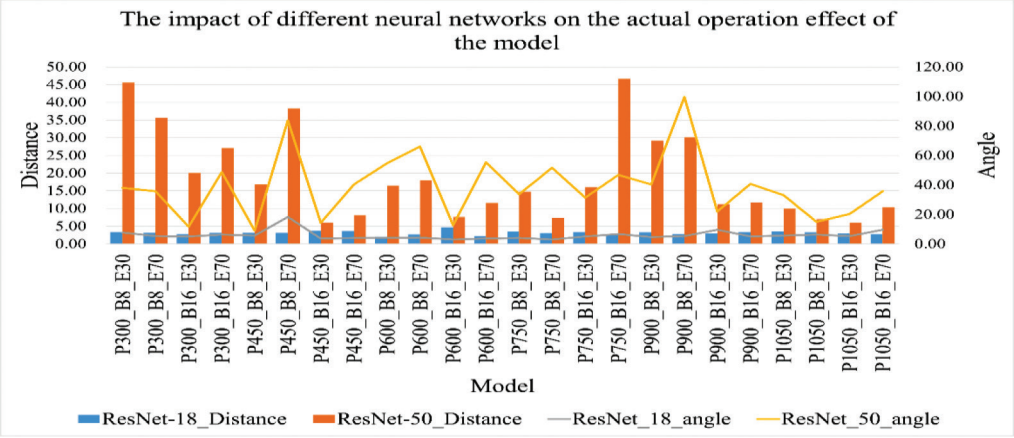


Fig. 28. Actual operation effect diagram of different neural network

The neural network in the hyperparameter configuration scheme of this article will choose ResNet-18 as the neural network.

2) The Impact of Different Numbers of Datasets on the Actual Operation Effect of the Model

Fig. 29 shows the impact of different numbers of datasets on the actual operation. From Fig. 29, we can see that when using ResNet-18 as the neural network, regardless of the number of photos in the dataset, the average offset distance and average offset angle are small, and the fluctuations are flat. However, when the neural network training is changed to ResNet-50, the average offset distance and offset angle increase sharply, and the road tracking experiments do not perform excellently. As the number of photos increases, the average distance and angle gradually decrease, and the actual operating effect gradually improves. This is due to the small number of our datasets, and ResNet-50 requires large-scale data to train a model with excellent performance. This article chooses to use the dataset of 600 photos with the best overall performance as the training dataset for the optimal hyperparameter configuration scheme.

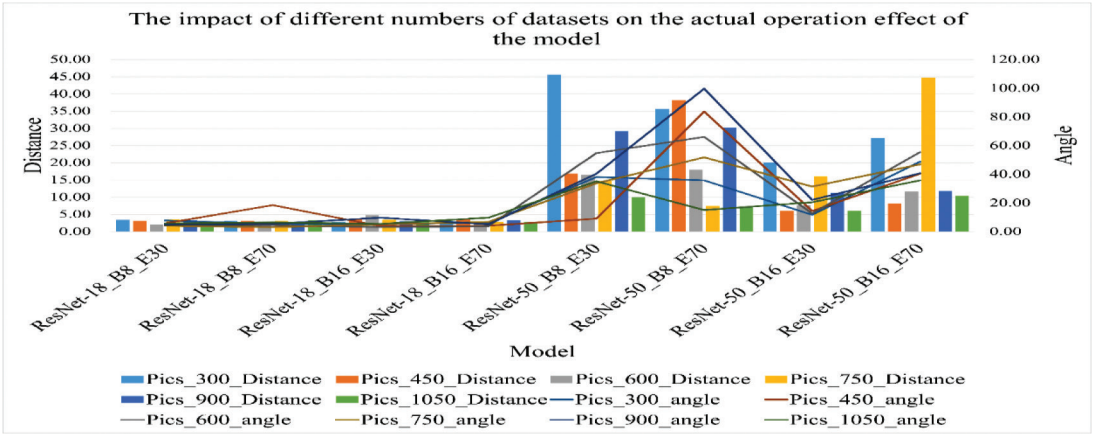


Fig. 29. Actual operation effect diagram of different numbers of datasets

3) The Impact of Different Batch Sizes on the Actual Operation Effect of the Model

From the analysis of batch size in the previous section, we can conclude that the small batch size has little effect on the model validation loss value. In order to further verify the conclusion, we analyzed the effect of different batch sizes on the actual operation of the smart car. Fig. 30 shows the actual driving effect of the model under different batch sizes. The histograms and curves also represent the actual average offset distance and angle of the model under different batch sizes.

As shown in Fig. 30, when the neural network is ResNet-18, the batch size has little effect on the average offset distance and offset angle. However, when the neural network is ResNet-50, the model trained with batch size=16 performs better than the model trained with batch size=8. Moreover, as the number of datasets increases, the actual effect of the model using batch size=16 becomes more and more stable.

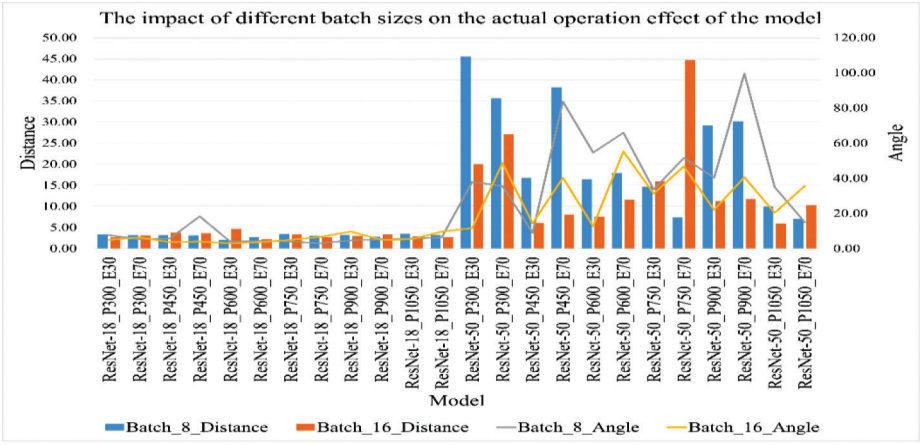


Fig. 30. Actual operation effect diagram of different batch sizes

Therefore, combined with the analysis of the verification loss value and the actual performance, the batch size=8 in the hyperparameter configuration scheme of this article is used for training with the ResNet-18 neural network. Under the condition of ensuring the training accuracy and actual performance, the training time and GPU consumption are reduced.

1) The Impact of Different Epochs on the Actual Operation Effect of the Model

Fig. 31 shows the effect of different epochs on the actual performance of the smart car. As shown in Fig. 31, the actual performance of the model trained with the ResNet-18 neural network is less affected by the epoch. However, when the model was trained with ResNet-50, the average offset distance and angle of the smart car gradually decreased with increasing epoch. Furthermore, as the number of photos increases, the impact of epochs on how well the model runs is slowly decreasing.

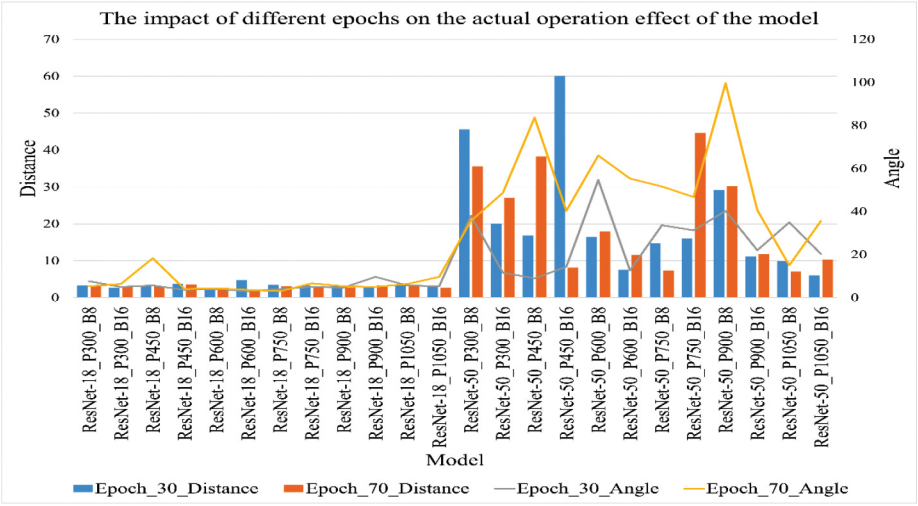


Fig. 31. Actual operation effect diagram of different epochs

Therefore, combined with the selection of neural network, dataset, and batch size, we set the epoch in the hyperparameter configuration scheme of this article to 8, which can effectively reduce the model training time and memory loss while ensuring the model accuracy and road tracking performance.

2) Summary of the Actual Operation Effects of Models Trained Under Different Hyperparameter Configuration Schemes

Fig. 32 shows the average offset distances and angles for 48 autonomous driving models trained according to the different hyperparameter configuration schemes. As shown in Fig. 32, the model trained using ResNet-18 as a neural network performs excellent, but the actual operation effect drops significantly after using ResNet-50 to train the model. In addition, with the increase in the number of photos and the increase in batch size, the performance of the model with ResNet-50 as the neural network is gradually getting better. It can be seen that the ResNet-18 neural network can be trained using small-batch datasets, and the number of datasets, batch size, and epoch have little effect on its autonomous driving performance.

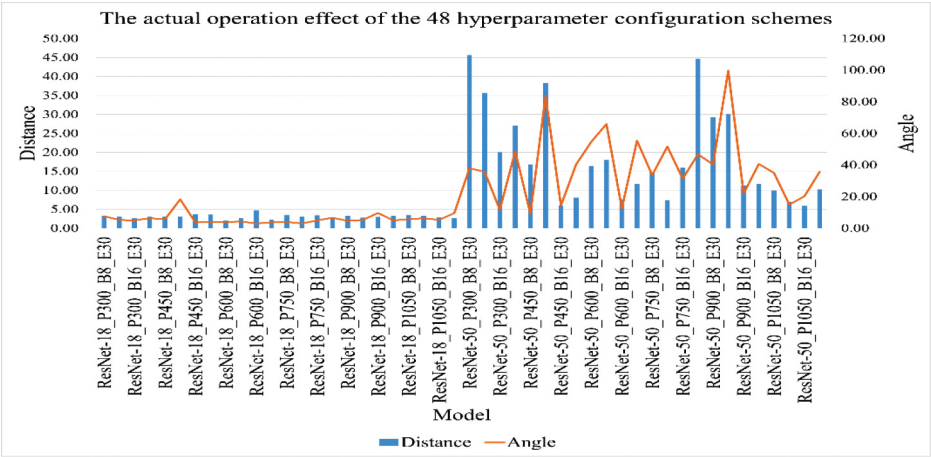


Fig. 32. The Actual operation effect for 48 autonomous driving models

Combining the influence of different hyper-parameters on the verification loss value and the actual operation effect of the smart car, we propose a set of optimal hyperparameter configuration schemes in which the neural network is ResNet-18, the batch size is 8, the epoch is 30. The dataset of 600 photos for training and the optimal autonomous driving model is obtained. The validation loss value of the optimal autonomous driving model is 0.011351. When the road tracking experiment was achieved, the average offset angle from the measurement point was 2.03cm, and the average offset angle was 4°.

3) Correlation Analysis of Verification Loss Value and Actual Operation Effect of Smart Car

In this section, we will focus on analyzing the relationships between the validation loss value and the actual performance of the model. Fig. 33 and Fig. 34 are the relationships between the verification loss value and the actual operation of the smart car when the neural network is ResNet-18 and ResNet-50, respectively.

As shown in Fig. 33 and Fig. 34, when the validation loss value is extreme (such as the minimum or maximum value), the change of the actual operating effect fluctuates wildly, and the performance is not excellent. When the verification loss value tends to the median, the variation and fluctuation of the actual operation effect are small, and the performance is excellent. Furthermore, the choice of neural network has a significant impact on the performance of the model.

The verification loss value can reflect the model training results, but it cannot directly determine the actual performance of the model. We need to make a comprehensive judgment on the performance of the model combined with the verification loss value and the actual operation effect.

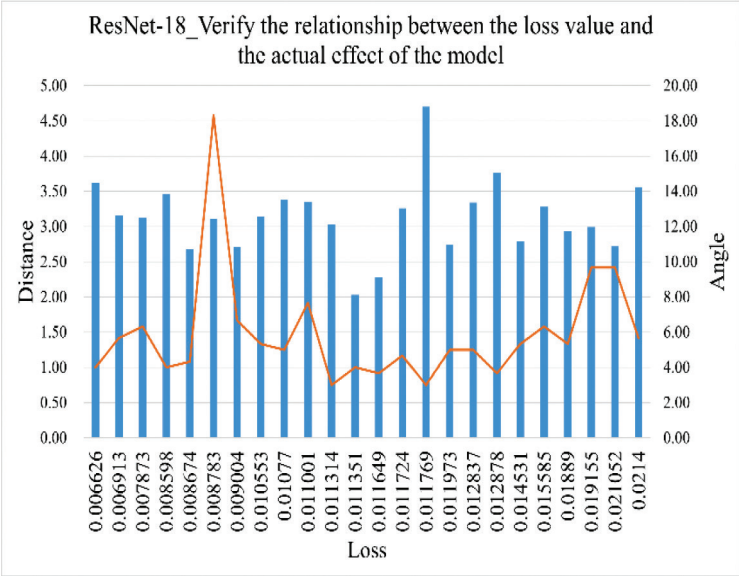


Fig. 33. ResNet-18_Verify the relationship between the loss value and the actual operation effect of the model

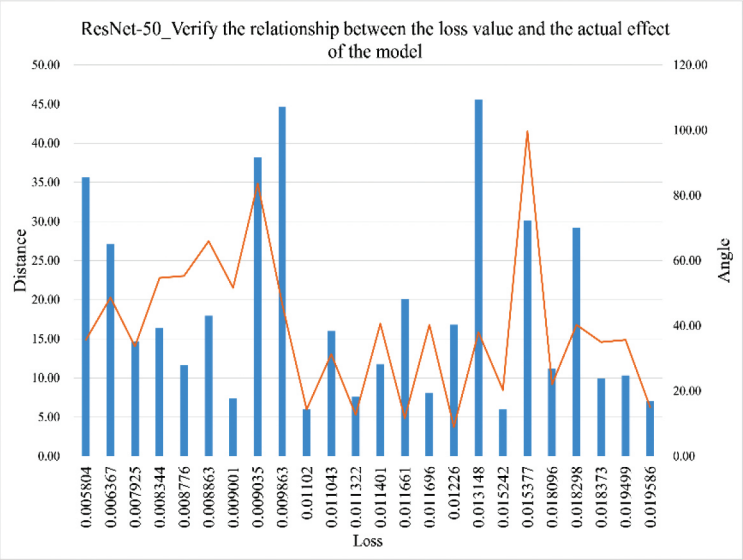


Fig. 34. ResNet-50_Verify the relationship between the loss value and the actual operation effect of the model

C. The Impact of Different Speed Gain and Steering Gain on Road Tracking

The main difference between an autonomous driving smart car and a toy car is that the smart car can reduce reproduction errors and improve the performance of autonomous driving by adjusting the speed gain and steering gain. Therefore, we will research the effect of gain value on autonomous driving performance by adjusting the speed and steering gain values and find the optimal gain value in the experimental results. The model used in the experiment is the optimal autonomous driving model trained with our proposed optimal hyperparameter configuration scheme. The neural network is ResNet-18, the batch size is 8, the epoch is 6, and the dataset is 600 photos.

Since speed gain and steering gain significantly impact autonomous driving, excessive speed or steering cause the smart car to drive off the track. Therefore, we propose a new evaluation criterion that can more intuitively represent the performance of autonomous driving models. We set the road tracking task as a 100-point scoring system. If the smart car touches the white line, 1 point will be deducted from the total score. 5 points will be deducted from the total score if the smart car drives off the track. We achieved three laps of road tracking experiments on the track and finally calculated the total score. The higher the final total score, the better the performance of the model, and we call this criterion the TO criterion. (touch the white line and off-track)

When the speed gain is below 0.35, the smart car cannot start the motor at this speed. When the speed exceeds 0.95, the smart car will drive out of the track and cannot achieve road tracks. Therefore, we choose six groups of speed gains for research in the range of speed gain from 0.35 to 0.95. The steering gain is set to the default value of 0.21. Table VI (A) shows the road tracking performance of different speed gains.

TABLE VI (A)
ROAD TRACKING PERFORMANCE FOR DIFFERENT SPEED GAINS

Speed Gain	Touch White Line	Off Track	Total Score
0.35	11	1	84
0.5	8	0	92
0.65	6	0	94
0.8	10	1	85
0.95	13	2	77

As shown in Table VI (A), with the increase of the speed gain, the driving state of the smart car changes from bad to good and then from good to bad. When the speed gain is 0.65, the performance of the smart car to achieve road tracking is the best. Therefore, 0.65 is the optimal speed gain value for the smart car.

When the steering gain is below 0.11, the smart car loses the ability to turn. When the steering gain is above 0.31, the smart car is very sensitive to steering and spins in place. Therefore, we select six groups of steering gains in the range of 0.11 to 0.31 for the research. The speed gain was set to the optimal speed gain value we obtained, 0.65. Table VI (B) shows the road tracking performance with different steering gains.

TABLE VI (B)
ROAD TRACKING PERFORMANCE WITH DIFFERENT STEERING GAINS

Steering Gain	Touch White Line	Off Track	Total Score
0.11	12	1	83
0.16	10	0	90
0.21	6	0	94
0.26	8	2	82
0.31	7	3	78

As shown in Table VI (B), when the steering gain is 0.21, the smart car has the highest performance score. 0.21 is the optimal steering gain value for the smart car. When the steering gain is 0.31, the smart car will many times drive out of the track.

In summary, we found the optimal gain values, where the speed gain is 0.65, and the steering gain is 0.21. Our optimal autonomous driving model performs best at this gain value.

D. Validation Experiment of the Optimal Hyper-parameter Configuration Scheme

This article proposes a set of optimal hyper-pa-rameter configuration schemes and trains the optimal autonomous driving model, which we call BH-ResNet. We compare this model with the excellent model proposed by Gupta P et al., which we call the GP-VGG model [25]. In addition, we also compare BH-ResNet with popular neural networks, DenseNet-121 proposed by Huang et al. [26], and AlexNet proposed by Krizhevsky et al. [27]. The performance of our BH-ResNet is validated by comparison with these existing models. Among them, the optimal speed gain and the optimal steering gain are applied in the existing method, and the evaluation standard we use is the TO criterion.

TABLE VII
COMPARISON TABLE OF BH-RESNET AND EXCELLENT MODELS

Model	Verification Loss Value	Touch White Line	Off Track	Total Score
BH-ResNet	0.011351	6	0	94
GP-VGG	0.121545	14	4	66
DenseNet-121	0.011744	12	1	83
AlexNet	0.011654	8	2	82

The comparison table of BH-ResNet and the excellent model is shown in Table VII. In the road tracking experiment, the validation loss value of BH-ResNet was lower than that of other models, and the validation loss of the GP-VGG model was the highest. In addition, according to the score of the TO scoring standard, we can see that BH-ResNet has the highest score of 94 points. Combining the validation loss value and the TO scoring criteria, the BH-ResNet model performs best.

E. Road Tracking Experiment in Unseen Scenes

Autonomous driving cars are challenging to train in all possible environments, so an excellent autonomous driving model can perform road-tracking tasks even in unfamiliar environments. Therefore, we designed a new track, which is a new environment that the smart car has not seen or trained. Fig. 35 is an unseen scene environment.

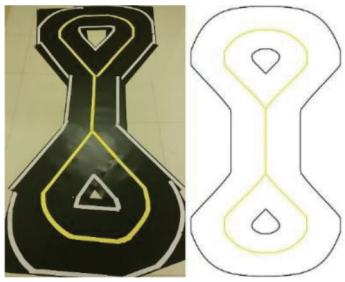


Fig. 35. An unseen scene environment

We conduct road tracking experiments in this unseen scene to verify the applicability of BH-ResNet. In addition, the three excellent models in the previous section are also involved in the experiments, which can accurately verify the performance of our BH-ResNet. The judging standard we use is the TO criterion. Table VIII shows the experimental results under the unseen scenario.

TABLE VIII
EXPERIMENTAL RESULTS IN THE UNSEEN SCENARIO

Model	Touch White Line	Off Track	Total Score
BH-ResNet	10	0	90
GP-VGG	17	6	53
DenseNet-121	15	3	70
AlexNet	12	5	63

Table VIII shows that the BH-ResNet model can also achieve the road tracking task even though the number of times touching the white line increases in unseen scenes. This means that our proposed BH-ResNet model can handle unseen environments, and the model has broad applicability and utility. Furthermore, the effects of the remaining three models are inferior, which shows that the BH-ResNe model has more ability to predict unfamiliar environments and a stronger ability to adapt to the external environment and resist external noise. After being trained in a limited environment, the smart car can be achieved road tracking in more unseen scenarios.

F. Summarize of Discussion

1) Using Jetson Nano as the mainboard of the smart car can allow the car to load more complex and efficient deep network models, and the calculation speed is excellent. The smart car can achieve all tasks independently without needing a computer as a back-end for processing operations. If the Jetson Nano is mounted on the toy car, it still cannot get superior performance, which is caused by the reproduction error brought by the hardware defect of the toy car. Therefore, we built an autonomous driving smart car based on a scale model in the real world, which can adjust the speed gain and steering gain, improves the performance of autonomous driving, and effectively reduces the problem of reproduction error of toy cars, which improves the accuracy of the experiment.

2) The hyperparameter setting is a crucial factor affecting the performance of smart cars. For different neural networks, we found that the validation loss values for models trained with ResNet-50 were generally lower than those trained with ResNet-18. The validation loss value fluctuates wildly for different datasets when the dataset is 300 or 1050 photos. When the dataset is 600 photos, the validation loss value is small and stable, and we need to choose the right amount of data according to the neural network used. The small batch size has little effect on the model validation loss for different batch sizes. For different epochs, on the premise that the model does not enter the overfitting state, the more iterations of weight update, the smaller the model validation loss.

3) The verification loss value can reflect the performance of the model. In order to make the experimental results more reliable, we need to make judgments based on the actual operation effect of the smart car. Therefore, we loaded 48 autonomous driving models into smart cars for actual measurements. The model trained by ResNet-18 performs significantly better for different neural networks than the model trained by ResNet-50. For different numbers of datasets, the size of the dataset has less impact on the model with ResNet-18 as the neural network but has a more significant impact on the model with ResNet-50 as the neural network, and the larger the number of datasets, the better the effect. For different batch sizes, when the neural network is ResNet-50, the batch size has a more significant impact on the results. For different epochs. The increase in epochs did not substantially improve the performance of the model.

4) Choosing the correct hyperparameters is very important. Based on the research on the verification

loss value and the actual operation effect of the smart car, we propose a set of optimal hyperparameter configuration schemes. The neural network is ResNet-18, the batch size is 8, and the epoch is 30. The optimal autonomous driving model BH-ResNet is obtained by training with a dataset of 600 photos.

5) Our smart car can adjust the speed gain and steering gain like a real car, so we researched the influence of the gain value on the smart car and finally proposed a set of optimal gain values. The optimal speed gain value is 0.65, and the optimal steering gain value is 0.21. The optimal gain value can significantly improve the performance of the smart car.

6) We compare BH-ResNet with three existing groups of excellent models, and we find that the BH-ResNet model outperforms other models in both validation loss value and actual operating effect, which also verifies the superiority of our model. In addition, the BH-ResNet model can achieve road-tracking experiments in unseen scenes. This demonstrates the practical utility of the model.

VI. Conclusion

Road tracking is a critical task in autonomous driving research. In the research, we use a scale model to build an autonomous driving smart car with adjustable speed gain and steering gain, equipped with a Jetson Nano which includes a high-performance GPU to achieve the road tracking task. Furthermore, we propose a set of optimal hyperparameter configuration schemes and train the optimal autonomous driving model BH-ResNet, which is proven to achieve road tracking tasks with excellent performance.

In the model training part, we tested the effect of different hyperparameters on the model validation loss value. We found that batch size has less effect on validation loss, and the different neural networks and datasets have more effect on validation loss. When the neural network is ResNet-50, the batch size is 8, the epoch is 70, and the dataset is 300, the validation loss is the smallest value of 0.005804. In the actual experiment part, we found that although the validation loss of ResNet-50 is lower than that of ResNet-18, the actual performance is far worse than that of ResNet-18. In addition, batch size and epoch have less impact on the model.

Integrating the research of validation loss values and actual operation effects, we proposed a set of optimal hyperparameter configuration schemes with the neural network of ResNet-18, a batch size of 8, an epoch of 30, and a set of 600. We trained the optimal autonomous driving model BH-ResNet.

In addition, we found that when the speed gain and steering gain increase, the number of times the smart car drives out of the track increases, and when the speed gain and steering gain decrease, the number of times the smart car touches the white line increases. When the speed gain is 0.65 and the steering gain is 0.21, the performance of the smart car to achieve the road tracking task is the best.

We compared BH-ResNet with DenseNet-121, Alexnet, and GP-VGG and found that all models can achieve road track, but DenseNet-121, Alexnet, and GP-VGG all have problems with touching the line or driving off the track when turning, with a total score of 83, 82, and 66, respectively. BH-ResNet has the highest score of 94. Compared with GP-VGG, the performance of the BH-ResNet model is improved by 42.4%.

A good model should have the ability to handle the unseen environment. Therefore, we designed a new track and proved that BH-ResNet could still achieve road tracking with high performance in an unseen environment. The existing models all showed many touchlines and driving off the track, with DenseNet-121 scoring 70, AlexNet scoring 63, and GP-VGG scoring 53. Our BH-ResNet has the highest score of 90. Compared with GP-VGG, the BH-ResNet model outperforms 69.8% in unseen environments.

Acknowledgement

The first author conducted the experiment and drafted the manuscript. The last author guided and advised the experiment and co-drafted the manuscript. The first and last authors each contributed 50% equally to this work.

References

[1] H. Fujiyoshi, T. Hirakawa, and T. J. I. R. Yamashita, "Deep Learning-Based Image Recognition for Autonomous Driving," *IATSS Research*, vol. 43, no. 4, pp. 244-252, Dec. 2019.

[2] S. Grigorescu, B. Trasnea, T. Cocias et al., "A Survey of Deep Learning Techniques for Autonomous Driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362-386, Apr. 2020.

[3] W. Y. Lin, W. H. Hsu, and Y. Y. Chiang, "A Combination of Feedback Control and Vision-Based Deep Learning Mechanism for Guiding Self-Driving Cars," in *Proc. 2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, 2018, pp. 262-266.

[4] N. Hossain, M. T. Kabir, T. R. Rahman et al., "A Real-Time Surveillance Mini-Rover Based on Opencv-Python-JAVA Using Raspberry Pi 2," in *Proc. 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 2015, pp. 476-481.

[5] U. Karni, S. S. Ramachandran, K. Sivaraman et al., "Development of Autonomous Downscaled Model Car Using Neural Networks and Machine Learning," in *Proc. 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 2019, pp. 1089-1094.

[6] A. Banerjee, V. Bolar, A. Chaurasia et al., "Autonomous Driving Vehicle," *International Research Journal of Engineering and Technology*, vol. 7, no. 4, pp. 6048-6050, Apr. 2020.

[7] E. Yılmaz and S. T. Özyer. (2019, Jan.). Remote and Autonomous Controlled Robotic Car Based on Arduino with Real-Time Obstacle Detection and Avoidance. *Universal Journal of Engineering Science*. [Online]. *7(1)*, pp. 1-7. Available: http://earsiv.cankaya.edu.tr:8080/handle/handle/20.500.12416/4265

[8] S. Yuenyong and Q. Jian, "Generating Synthetic Training Images for Deep Reinforcement Learning of a Mobile Robot," *Journal of Intelligent Informatics and Smart Technology*, vol. 2, pp. 16-20, Mar. 2017.
[9] T. D. Do, M. T. Duong, Q. V. Dang et al., "Real-Time Self-Driving Car Navigation Using Deep Neural Network," in *Proc. 2018 4th International Conference on Green Technology and Sustainable Development*, 2018, pp. 7-12.
[10] Q. Zhang, T. Du, and C. Tian. (2019, Dec. 20). *Self-Driving Scale Car Trained by Deep Reinforcement Learning*. [Online]. Available: https://arxiv.org2abs/1909.03467
[11] Y. Li, J. Qu, and Technology, "Intelligent Road Tracking and Real-time Acceleration-deceleration for Autonomous Driving Using Modified Convolutional Neural Networks," *Current Applied Science and Technology*, vol. 22, no. 6, p. 26, Mar. 2022.
[12] V. Rausch, A. Hansen, E. Solowjow et al., "Learning a Deep Neural Net Policy for End-to-End Control of Autonomous Vehicles," in *Proc. 2017 American Control Conference*, Seattle, WA, USA, 2017, pp. 4914-4919.
[13] P. M. Radiuk, "Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets," *Information Technology and Management Science*, vol. 20, no. 1, pp. 20-24, Dec. 2017.
[14] M. Choi, "An Empirical Study on the Optimal Batch Size for the Deep Q-Network," in *Proc. International Conference on Robot Intelligence Technology and Applications*, 2017, pp. 73-81.
[15] S. Chowdhuri, T. Pankaj, and K. Zipser, "Multinet: Multi-Modal Multi-Task Learning for Autonomous Driving," in *Proc. 2019 IEEE Winter Conference on Applications of Computer Vision*, 2019, pp. 1496-1504.
[16] J. Kocić, N. Jovičić, and V. J. S. Drndarević, "An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms," *MDPI*, vol. 19, no. 9, p. 2064, Apr. 2019.
[17] A. Iqbal, S. S. Ahmed, M. D. Tauqeer et al., "Design of Multifunctional Autonomous Car Using Ultrasonic and Infrared Sensors," in *Proc. 2017 International Symposium on Wireless Systems and Networks (ISWSN)*, 2017, pp. 1-5.
[18] K. Yu, L. Jia, Y. Chen et al., "Deep Learning: Yesterday, Today, and Tomorrow," *Journal of Computer Research and Development*, vol. 50, no. 9, pp. 1799-1804, Sep. 2013.
[19] Y. LeCun, Y. Bengio, and G. J. N. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436-444, May. 2015.
[20] J. Gu, Z. Wang, J. Kuen et al., "Recent Advances in Convolutional Neural Networks," *Pattern Recognition*, vol. 77, pp. 354-377, May. 2018.
[21] A. Karpathy, G. Toderici, S. Shetty et al., "Large-Scale Video Classification with Convolutional Neural Networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725-1732.
[22] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a Convolutional Neural Network," in *Proc. 2017 International Conference on Engineering and Technology*, 2017, pp. 1-6.
[23] S. J. I. S. Cass, "Nvidia Makes it Easy to Embed AI: The Jetson Nano Packs a Lot of Machine-Learning Power into DIY Projects," *IEEE Spectrum*, vol. 57, no. 7, pp. 14-16, Jul. 2020.
[24] R. Febbo, B. Flood, J. Halloy et al., (2020, Jul. 26). *Autonomous Vehicle Control Using a Deep Neural Network and Jetson Nano*. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3311790.3396669
[25] P. Gupta, V. Singh, A. J. J. Parashar et al., "Smart Autonomous Vehicle Using End to End Learning," *Journal of Innovation in Computer Science and Engineering*, vol. 9, no. 2, pp. 7-11, Jan. 2020.
[26] G. Huang, Z. Liu, L. Van Der Maaten et al., "Densely Connected Convolutional Networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July 21-26, 2017, pp. 4700-4708.
[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, vol. 60, no. 6, pp. 84-90, Jun. 2017.

Zihao Nie is currently studying for the Master of Engineering (Engineering and Technology), at Panyapiwat Institute of Management, Thailand. His research interests are Research direction is artificial intelligence, autonomous driving

Jian Qu is a full-time lecturer at the Faculty of Engineering and Technology, Panyapiwat Institute of Management. He received Ph.D. with Outstanding Performance award from Japan Advanced Institute of Science and Technology, Japan, in 2013. He received 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, Thammast University, Thailand, in 2010. His research interests are natural language processing, intelligent algorithms, machine learning, machine translation, information retrieval, and image processing.