

## Research article

# Multi-task Autonomous Driving Based on Improved Convolutional Neural Network and ST Loss in MTS and MOD Modes

Zihao Nie and Jian Qu\*

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

Received: 28 March 2022, Revised: 8 August 2022, Accepted: 28 October 2022

DOI: 10.55003/cast.2022.03.23.009

### Abstract

#### Keywords

deep learning;  
loss function;  
MT-ResNet26;  
MTS and MOD mode;  
multi-task autonomous driving

Multi-task autonomous driving is a research hotspot in autonomous driving. However, existing research has only achieved single-task or dual-task autonomous driving. Therefore, we propose two novel multi-task approaches: a multi-task shared model mode (MTS) and a multi-object dual-model mode (MOD). In addition, existing neural network architectures are underperforming in multi-task autonomous driving, so we propose a novel neural network architecture - MT-ResNet26. Moreover, to alleviate the problem of noise and class imbalance from data, we propose a new loss function - Stable Loss (ST Loss). Finally, our smart car can achieve multi-task road tracking, left-right turn sign recognition, automatic obstacle avoidance, stop, real-time acceleration and deceleration. In addition, we compare the existing multi-task autonomous driving model YS-VGG17\_MSE, which shows our MT-ResNet26\_ST is superior in loss value and actual performance. Meanwhile, we use our proposed approaches to train two classical neural networks—ResNet18\_MSE\* and DenseNet121\_MSE\*, so that they also achieve multi-task autonomous driving with our proposed approaches, showing the applicability of MTS and MOD. Furthermore, we compare MT-ResNet26\_MSE with MT-ResNet26\_ST, and the results show that the model using our novel ST Loss outperforms the model using the original loss function MSE. To sum up, it is shown that the performance of multi-task autonomous driving can be achieved and improved using our proposed neural network architecture and loss function. Furthermore, we propose optimized multi-task modes. OMTS and OMOD optimize and accelerate the models using semi-precision techniques based on the TensorRT. The results show that the optimized multi-task autonomous driving accuracy has been further improved.

\*Corresponding author: Tel.: (+66) 0863759307  
E-mail: jianqu@pim.ac.th

## 1. Introduction

Autonomous driving is an essential part of current artificial intelligence, and making autonomous cars achieve multi-tasks is a current hotspot. More and more research on autonomous driving has been conducted in recent years. Most of the existing research uses toy cars as research subjects. For example, Hossain *et al.* [1] used toy cars for experiments. However, because it was a toy car, there were reproduction errors such as speed deviation during turning which would occur in each experiment. The reason why the toy car could not perfectly replicate real cars due to the lack of many accessories such as gears (that would be found in real cars). Therefore, this research uses a scale model car, which can simulate the car to a greater extent and minimize reproduction error, thereby reducing the error in the experiment.

Moreover, humans usually use only sight and hearing to drive cars, but most research today often installs many sensors to allow multi-tasking by autonomous cars. For example, Yılmaz and Tariyan [2] installed infrared sensors, ultrasonic sensors, and other components on the smart car. Iqbal *et al.* [3] added infrared and ultrasonic sensors into the car. Their car could detect lanes, overtake other vehicles, avoid obstacles, and identify traffic lights. Banerjee *et al.* [4] installed a radar sensor to prevent the collision between the smart car and the obstacle. The radar sensor detected the distance between the vehicle and the obstacle. Therefore, designing a smart car with fewer sensors to achieve more tasks is particularly important for mimicking the human brain. This article proposes a smart car that only uses one camera to achieve all tasks, and puts more effort into the design of neural networks, loss functions, and approaches to achieve multi-tasking.

Furthermore, in the existing research, many cars are still non-independent units. The cars cannot complete deep learning or reinforcement learning tasks independently, and need to be hooked up to a computer, making artificial intelligence not sufficiently independent. For example, Yuenyong and Qu [5] used Arduino as the motherboard of the car to complete reinforcement learning tasks. Arduino was unable to handle many computing problems, requiring a computer to be connected through Bluetooth as a back-end device. The Arduino smart car itself could not achieve the task independently. To make the car an independent unit, in this work, we used Jetson Nano as the motherboard to support a smart car that can achieve all tasks independently.

In deep learning, neural networks are essential, and the structure of neural networks determines the quality of the research. Many neural networks have been proposed in the existing research. For example, Rausch *et al.* [6] proposed a complex convolutional neural network that consisted of three convolutional layers, two pooling layers, and a fully connected layer. The convolutional layer was used to extract the features of the track, the pooling layer was used to scale the extracted feature information, and the final layer used the steering angle as an output. The trained neural network maps the pixel data from the camera directly to the steering commands for the road tracking task. Similarly, Bechtel *et al.* [7] further deepened the neural network by proposing a convolutional neural network consisting of five convolutional layers and four fully connected layers. They finally applied the network to their smart car, which used the predicted steering angle values as the output to achieve road tracking.

However, the neural networks used by Rausch *et al.* [6] and Bechtel *et al.* [7] did not enable multi-task autonomous driving because their neural networks were too simple, and these shallow neural networks were not stable and accurate enough to be trained using only a small amount of data. We need deeper and more efficient networks to improve the training accuracy and achieve more tasks. However, the amount of data required for deep neural networks increases exponentially, and the complexity of the model increases significantly, both of which are not good for training and applying the model. Therefore, we proposed a novel neural network architecture, which we call multi-task-ResNet26 (MT-ResNet26), which can make a massive amount of data unnecessary while still allowing the car to complete various tasks with high accuracy.

Meyer and Thakurdesai [8] modified the KL Divergence loss function to build a new model between the prediction and label distribution by minimizing the relative entropy, which could disperse the relative entropy results, prevent overfitting the potential noisy labels, improve prediction distribution, and thus improve the target detection performance, thereby solving the problem caused by sensor noise and inaccurate target detection caused by incomplete data. Ren *et al.* [9] proposed a method for prediction using occupancy maps to solve the vehicle motion prediction problem. They proposed three new loss functions to construct a secure model for prediction. Two of these loss functions were modified based on the Sigmoid loss function to predict trajectories. The third loss function is adjusted based on the MSE loss function and was used to optimize the prediction of unseen vehicles. This approach predicted the earliest time a location within a specified range could be occupied by visible and invisible vehicles.

However, the loss functions of Meyer and Thakurdesai [8] and Ren *et al.* [9] could only solve specific problems and were not general. Furthermore, these loss functions solved only one task. When an AI agent achieves multiple tasks, it is often affected by noise and class imbalance problems in multi-tasks. Therefore, we proposed a new and more applicable loss function to reduce such problems, which we named Stable Loss Function (ST Loss).

In our research of multi-task autonomous driving, we divided the current research into single-task autonomous driving and multi-task autonomous driving. Most of the existing research is on single-task autonomous driving. For example, to achieve the task of road tracking, Do *et al.* [10] proposed an autonomous driving car model trained on a Raspberry Pi. The smart car could satisfactorily complete the task of road tracking. Al-Nima *et al.* [11] proposed a new neural network based on deep reinforcement learning, DRL-RT. It collected input states from the advancing car view and produced appropriate road-tracking actions. Ultimately, their methods made the car keep track on the roads in different weather environments. However, their approaches could not make the smart car achieve the object detection task.

Further research gradually combined detection tasks with road tracking to enable smart cars to achieve more tasks in autonomous driving. Kulkarni *et al.* [12] utilized the InceptionV2 model to recognize and detect traffic lights. Yang *et al.* [13] proposed a pedestrian detection and vehicle detection algorithm based on YOLOv2 optimized feature extraction to achieve efficient pedestrian detection and vehicle detection at the same time when the car was driving.

Fang *et al.* [14] also combined the detection and tracking of road signs. They divided detection and tracking into two phases. They developed two neural networks to extract the color and shape features of traffic signs from input scene images in the detection phase. In the tracking stage, a Kalman filter was used to track the traffic signs located in the previous step through an image sequence. The method performed well in object detection and road sign tracking.

However, most of the research could not perform a critical task in automatic driving - automatic obstacle avoidance. Many researchers gradually added automatic obstacle avoidance into the tasks of the smart car. Mitsch *et al.* [15] proposed a hybrid system model and theorem proving technique to achieve automatic obstacle avoidance of stationary and moving obstacles. Also, Kang *et al.* [16] researched automatic obstacle avoidance technology, and they established a behavior-friendly strategy for autonomous vehicles based on vehicle dynamics through Q-learning. In this case, the car learned by itself through repeated events. Finally, the autonomous car could smoothly avoid obstacles. Still, they could not make the car achieve the acceleration and deceleration task.

Their methods mainly performed single-task or dual-tasks and could not achieve multi-tasks. Therefore, this research proposes two approaches to facilitate multi-tasks. We call the first approach multi-task shared mode (MTS), the second approach we call the multi-object dual-model mode (MOD).

We used an embedded device, Jetson Nano, to deploy MTS and MOD to achieve multi-tasks. Because embedded devices are not built to accomplish computationally intensive devices, they often have relatively low inference speeds when deploying deep learning models, leading to

problems such as latency and increased power consumption [17]. Ding and Qu [18] had problems with model response time delays when deploying autonomous driving models due to resource constraints in embedded devices, and performed experiments with poor performance. Since the MTS and MOD modes perform multiple tasks while autonomous driving, the inference speed of the autonomous driving model is much lower than that of the single-task autonomous driving model. Therefore, we optimized our autonomous driving model using semi-precision technology (FP16) based on an TensorRT framework, and integrate it with multi-task mode. We thus propose a novel optimized multi-task mode, which we refer to as optimized MTS (OMTS) and optimized MOD (OMOD). OMTS and OMOD speed up the inference and computation of the autonomous driving model with good real-time performance to further improve the performance of the autonomous driving smart car to achieve multi-tasks.

In summary, in this research, we built a scale model car with only one camera as a sensor and used a Jetson Nano motherboard, making the car a stand-alone unit. To enable the smart car to integrate multi-tasks into one AI, we proposed a novel neural network architecture - MT-ResNet26. Moreover, we proposed two approaches to make the smart car complete multi-task functional. Furthermore, we created a new loss function - ST Loss to reduce the effect of class imbalance and excessive noise caused by the multi-task function, increasing the robustness of the model and enabling the smart car to smoothly complete multi-tasks. In addition, we proposed optimized multi-task modes to alleviate the delay problem of the smart car in achieving multi-task functionality and improve the stability of multi-tasking.

## 2. Materials and Methods

### 2.1 Construction of the autonomous driving smart car

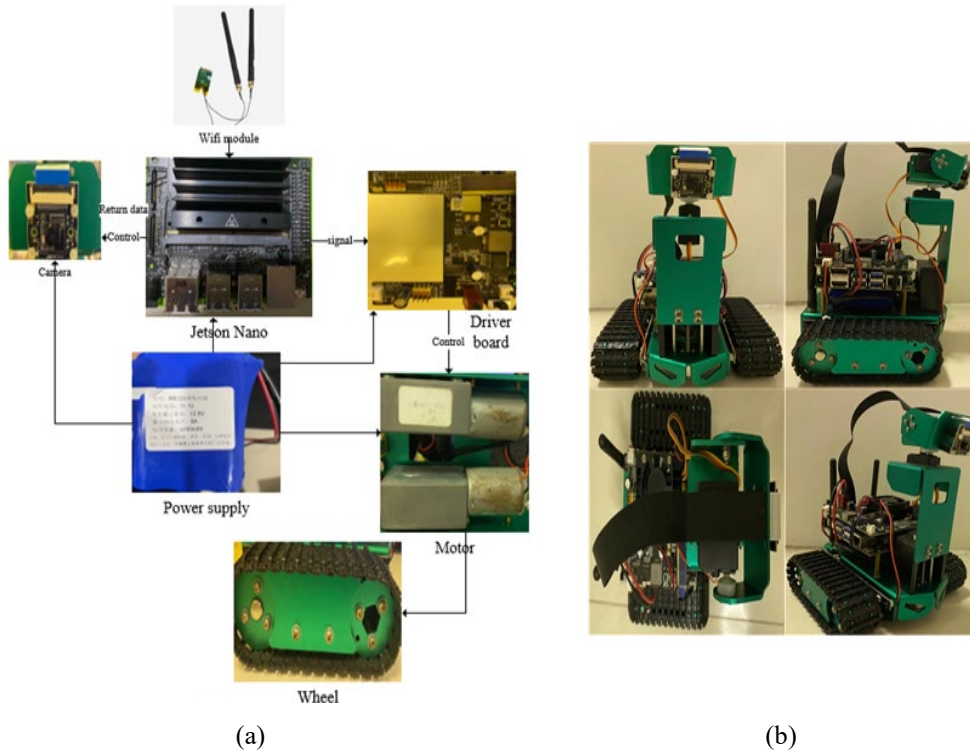
The smart car built in this research was different from the toy cars often used in past research. We use a scale model car, which could more accurately imitate real cars—and used a Jetson Nano motherboard that made the car a stand-alone agent.

The hardware framework of the Jetson Nano smart car is shown in Figure 1. The smart car used a 2200mAh battery pack as a power supply, and only used a camera as an input source, which transmitted data to the Jetson Nano for processing. The Jetson Nano was equipped with a driver board, which can transmit the processing signals of the Jetson Nano to the motor for control of the smart car.

### 2.2 Self-made neural network architecture--MT-ResNet-26

Different neural network architectures produce different effects depending on the experimental requirements. A deep neural network means better nonlinear expression ability in computer vision. It can fit more complex feature input. Moreover, as the number of network layers increases, the layer-by-layer feature learning ability is enhanced. However, the deep network also has problems such as gradient instability and network degradation, which lead to the decline of the learning ability of some shallow layers. As the depth of the neural network increases, the amount of data required to train the model increases exponentially. However, shallow neural networks cannot achieve multi-task accuracy.

Therefore, we proposed a novel neural network architecture, MT-ResNet26, for multi-tasking. We added several convolutional layers based on ResNet neural network architecture to increase the generalization ability of the neural network. At the same time, to speed up the training



**Figure 1.** (a) Jetson Nano smart car hardware framework, (b) Autonomous driving smart car overview

and convergence speed of the neural network and prevent overfitting during training, we added a BN layer after each convolutional layer.

Figure 2 shows the neural network architecture of MT-ResNet26, which has 24 convolutional layers, 26 BN layers, 1 maximum pooling layer, 1 average pool layer, and 1 FC layer.

The neural network we built can ensure the accuracy of the model. Moreover, to achieve multi-task function, the robustness of the model also needs to be stable, and the loss function is the critical factor affecting the robustness of the model; we therefore proposed a new loss function.

### 2.3 Our loss function-stable loss

In deep learning, loss functions are used to measure the performance of a model in predicting the desired outcome. Generally speaking, the larger the loss value, the worse the trained model will be. According to the different learning tasks, the loss function can be divided into regression loss and classification loss. Regression loss deals with the prediction problem of continuous values, and the autonomous driving problem belongs to the regression problem. We combined several regression loss functions in proposing a novel loss function - the stable loss function. ST Loss has good robustness and effectiveness that can alleviate the problems of noise and class imbalance.

The most widely used regression loss functions are MAE, MSE [19], and Smooth L1 loss function [20]. ST Loss combines their advantages to make the smart car achieve multi-tasking. Figure 3 shows the loss function diagram.

## MT-ResNet26

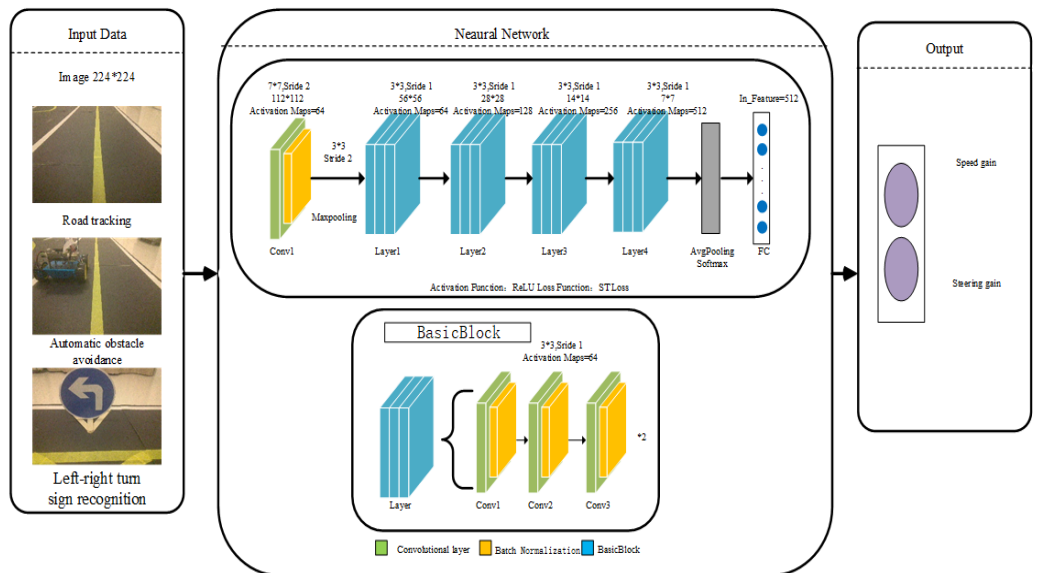


Figure 2. Network architecture diagram of MT-ResNet26

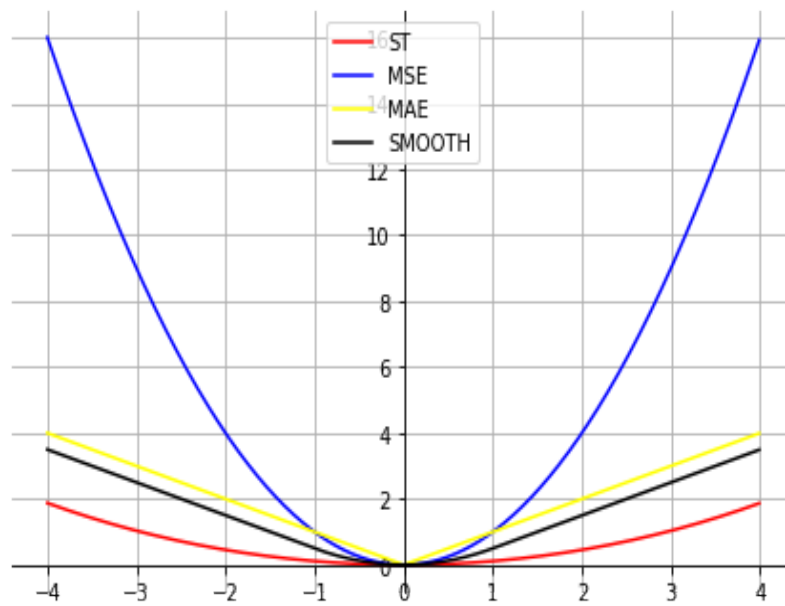


Figure 3. ST Loss, MSE, MAE, and Smooth L1 loss functions

From Figure 3, we can see the advantages of MSE. MSE is very sensitive to data, and is thus conducive to gradient update and function convergence. However, MSE is also easily affected by the error value, resulting in model accuracy degradation. The MAE loss function was proposed to solve this problem.

Because the MAE loss function is an absolute value, MAE does not reduce the accuracy of the model even if outliers corrupt the training data. However, MAE is continuous but not differentiable at  $y - f(x) = 0$ , so it is more difficult to solve for the differentiation. Furthermore, the gradient of MAE always remains the same, which makes MAE unable to handle minimal loss values.

Smooth L1 combines the MAE and MSE loss functions by adding the parameter  $\delta$ , but Smooth L1 is a piecewise function, which leads to its continuous non-differentiable property. Moreover, Smooth L1 is needed to train the hyperparameter  $\delta$ . Thus, this process needs to be iterated continuously.

To solve the shortcomings of these loss functions and achieve multi-tasks, we proposed ST Loss and set three hyperparameters for ST Loss to change its robustness, thereby improving the model performance.

The general form of ST Loss is as follows:

$$f(x) = \frac{|\alpha-2|}{\gamma} (x^2)^{\frac{1}{\alpha}} + \cosh \frac{\beta}{\gamma} x - 1 \quad (1)$$

Among these terms,  $\alpha$ ,  $\beta$ ,  $\gamma$  are hyperparameters, and  $\gamma$  can be regarded as a scale parameter, which controls the bending scale of ST Loss in the neighborhood of  $x=0$ . The robustness of ST Loss is adjusted by  $\frac{|\alpha-2|}{\gamma}$  and  $\frac{\beta}{\gamma}$ , which can make ST Loss more robust. The trained model has high precision, which can reduce the effects of noise and class imbalance, thereby making the smart car able to achieve multi-tasking.

Since  $\alpha$  acts as a hyperparameter, we can see that the loss function has a similar form for different values of  $\alpha$ . When  $\alpha \rightarrow 2$ ,  $\gamma \rightarrow \alpha$ ,  $\beta=0$ , SL Loss  $\rightarrow 0.5\sqrt{x^2}$ , which is approximate to MAE, when  $\alpha = 1$ ,  $\beta=0$ , SL Loss  $= \frac{1}{\gamma} x^2$ , approximate at MSE.

From  $\alpha = 2$  to  $\alpha = 1$ , ST Loss smoothly transitions from MAE loss to MSE loss, so we can reduce the shortcomings of MAE and MSE for outliers and noise points by adjusting those three hyperparameters and enhancing the robustness of ST Loss. At the same time, ST Loss is different from Smooth L1. It is not a piecewise function and has continuous differentiability. Every point on the function has a specific value, making ST Loss highly precise.

We can draw the following inferences about ST Loss through continuous experimentation with three hyperparameters.

When  $\alpha$ ,  $\beta$ ,  $\gamma > 0$ , the image of ST Loss is smooth and suitable for gradient-based optimization; due to the continuous derivability of ST Loss, when  $x=0$ , ST Loss is still meaningful; with the decrease of  $\frac{|\alpha-2|}{\gamma}$  and  $\frac{\beta}{\gamma}$  on  $(0, 1)$  and  $\alpha$  approaching 1, the curve of ST Loss gradually becomes smooth, and the robustness of ST Loss gradually increases; the closer the curve of ST Loss is to 0, the more the neural network is more prone to overfitting. Through experiments and demonstrations, we finally obtained three hyperparameters, which were in order:  $\alpha = 1$ ,  $\beta = 2$ ,  $\gamma = 10$ .

The final ST Loss equation is as follows:

$$f(x) = 0.1x^2 + \cosh 0.5x - 1 \quad (2)$$



ST Loss inherits the characteristics of MSE, which is more sensitive to data, but at the same time is more robust to outliers, which enables ST Loss to effectively deal with the noise problem caused by multi-tasks and the imbalance class problem, which are in turn caused by multi-task datasets.

## 2.4 Approaches for multi-task autonomous driving

Road tracking, automatic obstacle avoidance, traffic sign recognition, and acceleration and deceleration are essential tasks in autonomous driving. Most of the current research is single-task or dual-task autonomous driving, which are insufficient for applying to real autonomous driving. The MT-ResNet26 and ST Loss proposed in this article enable the smart car to achieve multi-task driving.

The first approach is called the multi-task shared model mode (MTS). The MTS mode enables the smart car to simultaneously achieve three tasks: road tracking, automatic obstacle avoidance, and left-right turn sign recognition. MTS mode allows multiple tasks to share the same neural network (MT-ResNet26) and achieve different tasks at the output of the network.

The second approach is called multi-object dual-model mode (MOD). The MOD mode enables the smart car to achieve road tracking and multi-object detection tasks. We proposed a dual-model architecture to build the MOD mode. In this case, different tasks use different neural networks, by which MT-ResNet26 achieves the road tracking task, `ssd_mobilenet_V2` achieves the multi-object detection task. After recognizing the target object, the smart car will perform different actions according to the different targets. For example, when a stop sign is recognized, the smart car will stop driving. Moreover, two target objects: A and B are set in this research. When target A is detected, the smart car will accelerate, and the smart car will decelerate when target B is detected.

A general architecture diagram of the multi-tasks proposed in this article is shown in Figure 4.

### 2.4.1 MTS mode

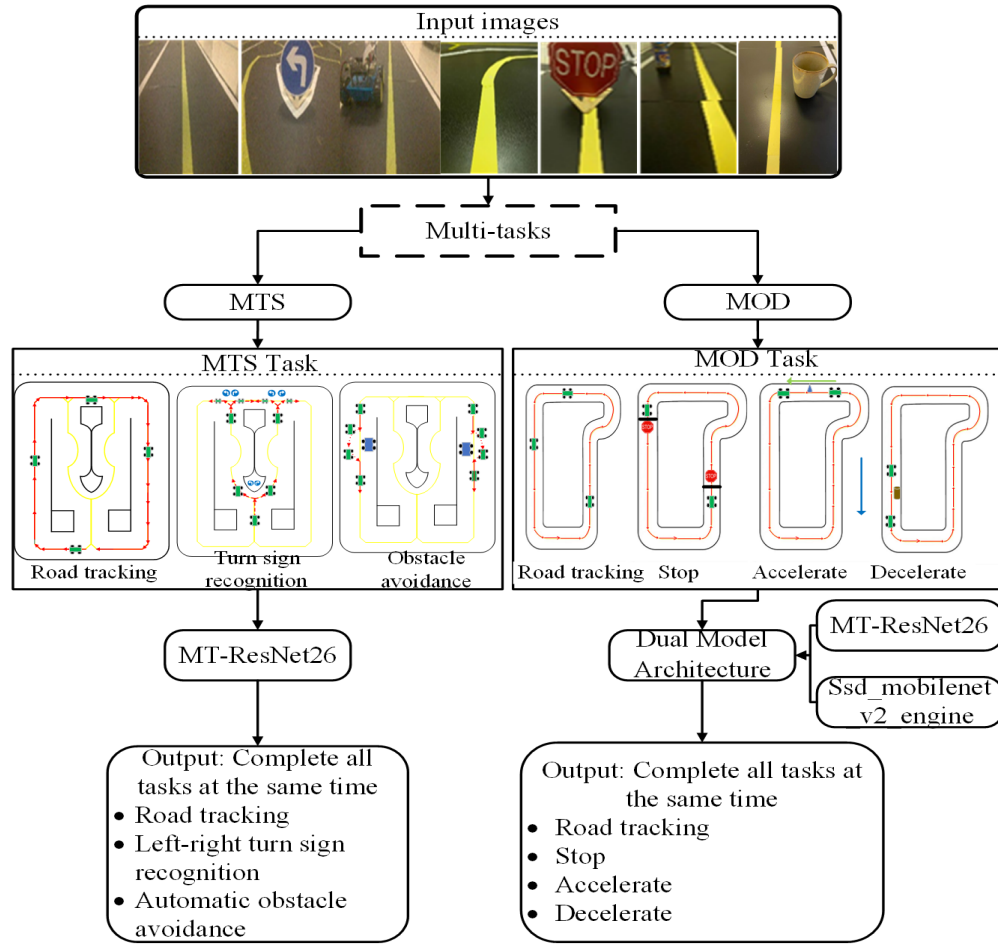
An architecture diagram of MTS mode is shown in Figure 5. The feature of MTS mode is that different tasks share the same neural network: MT-ResNet26. In the output of the neural network, the model uses a mapping relationship between the input picture and the output (speed gain and steering gain) to achieve different tasks.

The three tasks of road tracking, automatic obstacle avoidance, and left-right turn sign recognition belong to the same mapping relationship, and they are closely related. The smart car will only achieve different tasks when encountering obstacles and road signs. By using the same neural network, the model can learn some standard abstract low-level features. At the same time, we designed unique image labels for learning higher features according to the characteristics of each task. Then, different tasks provided additional useful information to each other while preserving the task characteristics, making the model more robust. Therefore, the MTS mode can reduce noise and reduce the risk of overfitting.

### 2.4.2 MOD mode

The second approach we proposed to achieve multi-tasking was the MOD mode. The MOD mode has a looser connection constraint on multi-tasks. When different tasks cannot transfer useful information to each other, the MOD mode is better suited. For example, the data between the road tracking task and the object detection task are independent and do not affect each other. The MOD mode features different tasks using different networks. The architecture diagram of MOD mode is shown in Figure 6.





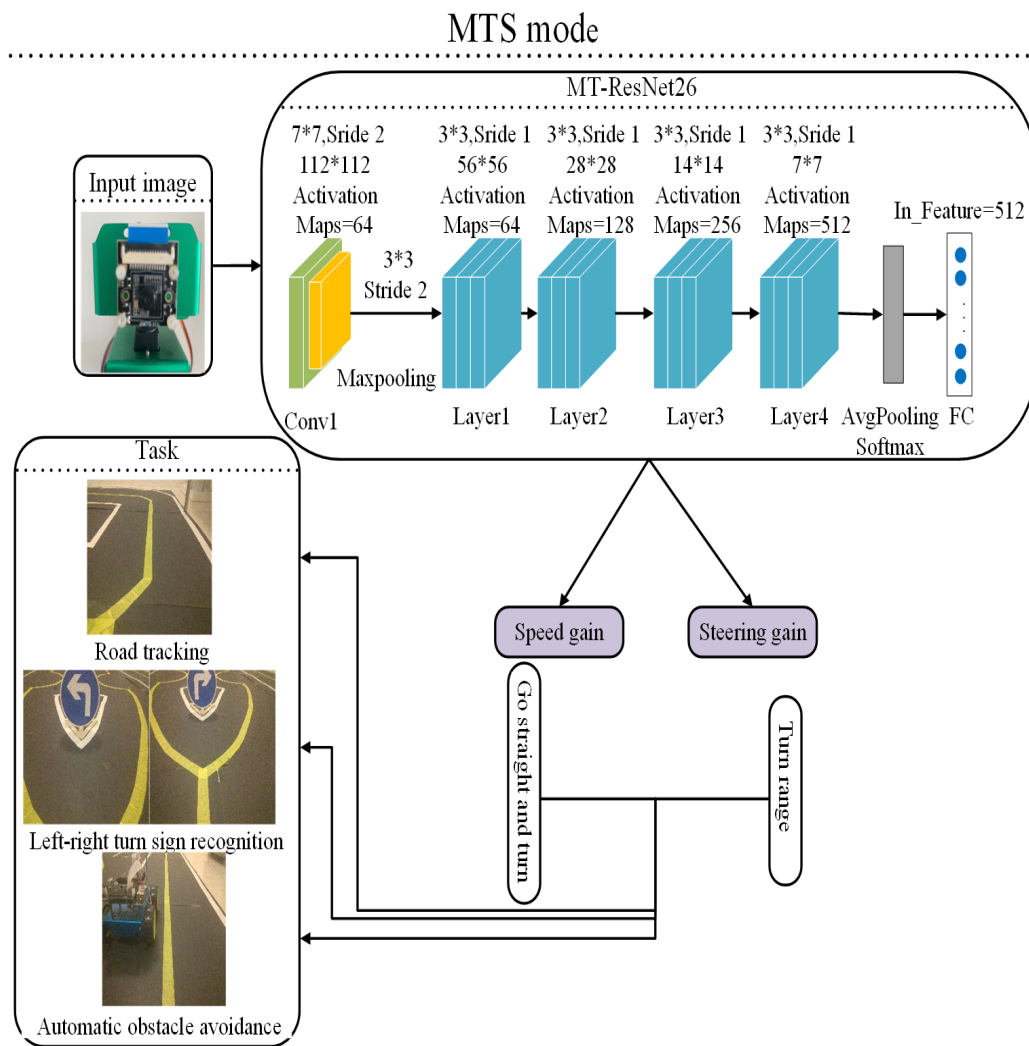
**Figure 4.** Multi-task architecture diagram

The road tracking model achieves the road tracking task of the smart car. Since the two models in the dual-model mode affect each other and have more noise, we used MT-ResNet26 as the neural network and ST Loss as the loss function to train the road tracking model to ensure the robustness and accuracy of the model.

The multi-object detection model is responsible for object detection, and the smart car recognizes different object targets to perform different actions. We used TensorFlow object detection API to train the ssd\_mobilenet\_v2 model on the coco dataset to construct an object detection model.

Since two models need to be combined into one package simultaneously when constructing a dual-model architecture, the operational speed of the model is affected, and there will be a considerable delay in the actual operation of the smart car. Therefore, we used TensorRT to optimize ssd\_mobilenet\_v2 to reduce the latency of the model.

In the MOD mode, when the input of the camera is the lane line, the road tracking model completes the road tracking task of the smart car; when the input of the camera is targets A and B or a stop sign, the multi-object detection model detects the target and the smart car will achieve the task of accelerating, decelerating or stopping according to the corresponding target.



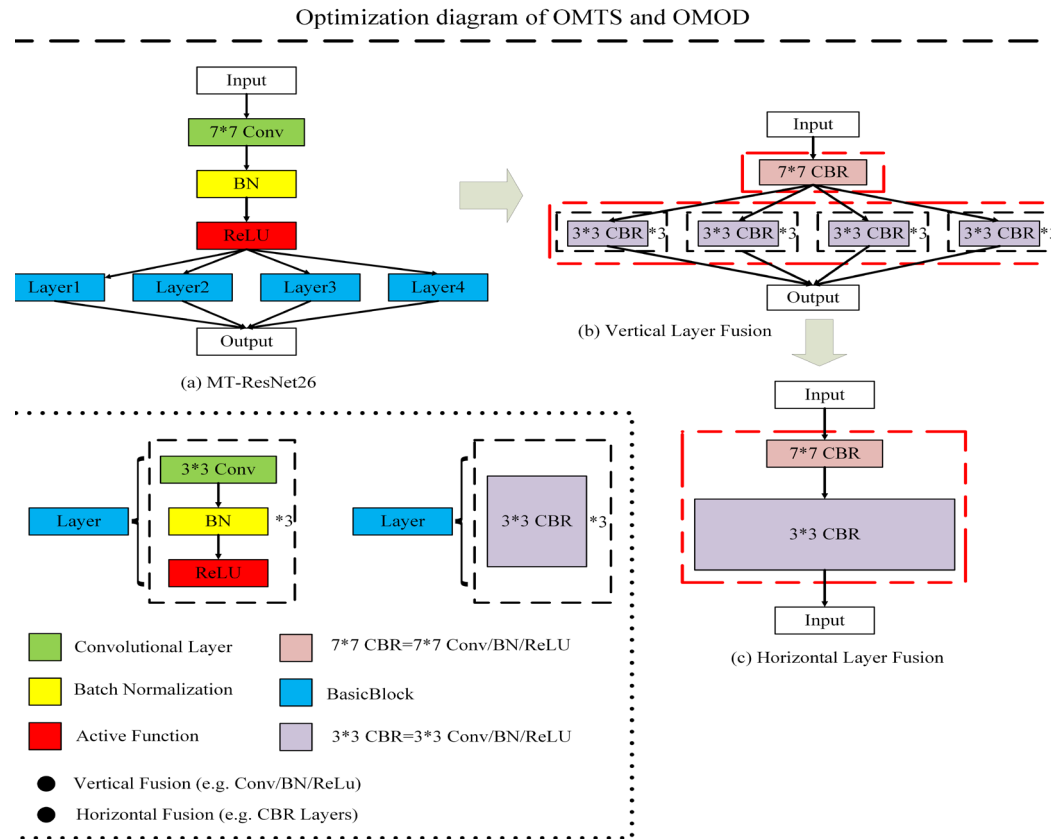
**Figure 5.** MTS mode architecture diagram



optimized multi-task modes, which we refer to as optimized MTS (OMTS) and optimized MOD (OMOD), respectively.

Our proposed optimized multi-task mode compresses, optimizes, and reconstructs the computational graph of the neural network through the TensorRT framework, thus speeding up the inference of the model and reducing the latency. Figure 7 shows the optimization approach of the optimized multi-task mode (OMTS and OMOD). Taking the optimization of our proposed MT-ResNet26 network architecture as an example, firstly, the optimized multi-task mode eliminates some redundant layers to reduce the amount of data circulation and reduce the computation. Secondly, as shown in Figure 7 (b), vertical fusion is performed on MT-ResNet26 by combining compatible layers consisting of convolutional, BN, and ReLU layers into a single CBR layer. Finally, horizontal fusion, i.e. similar layer merging, is performed on MT-ResNet26, as shown in Figure 7 (c), where twelve 3\*3 CBR layers are fused into the same CBR layer. This optimization method can produce a highly compressed MT-ResNet26, which we call MT-ResNet26\_TRT.

In addition, the optimized multi-task mode uses a semi-precision technique (FP16) for low-precision inference on the highly compressed MT-ResNet26\_TRT, which significantly reduces the use of computational resources, speeds up the inference of the autonomous driving model, and mitigates latency conditions, thus improving the accuracy rate of multi-tasks autonomous driving.



**Figure 7.** Optimization diagram of OMTS and OMOD

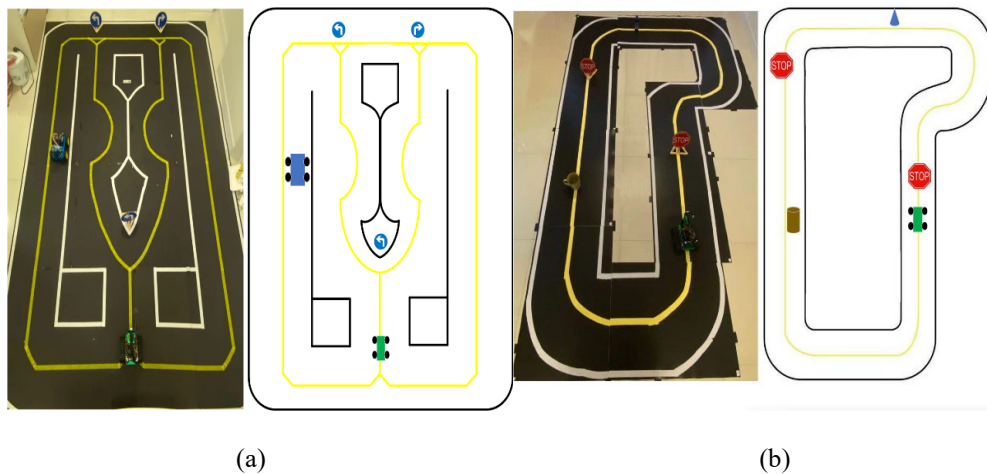
## 2.6 Experimental setup

### 2.6.1 Simulation environment construction

Most of the existing research uses a relatively simple circular track for experiments. However, we encounter various road conditions in real life, so relying only on a circular track cannot certainly replicate the “real use” of automatic driving. We need more complex and diverse road conditions. Therefore, we constructed different tracks according to different multi-tasks (MTS mode and MOD mode) as shown in Figure 8.

On track (a), we achieved road tracking, automatic obstacle avoidance, and left-right turn sign recognition tasks. There were three left-right turning intersections on track (a). We set different driving routes according to different turning directions. The smart car drove on an untrained custom route to test the adaptability of the multi-task autonomous driving smart car in unseen environments. In Figure 8 (a), the green model car is the smart car, the blue model car is an obstacle, and the blue turn sign at the corner is the left-right turn sign.

On track (b), we achieved road tracking and multi-object detection tasks. To verify the accuracy of the model, we added several corners because on the corners, the output of the model (speed gain and steering gain) has a more significant impact on the smart car. In Figure 8 (b), the green model car was the smart car. The red mark was the stop target, the blue cone was the acceleration target, and the brown cylinder was the deceleration target.



**Figure 8.** (a) MTS Mode custom track, (b) MOD Mode custom track

### 2.6.2 Data collection

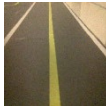
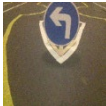






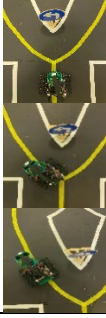

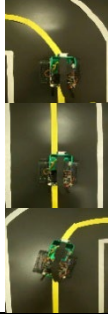

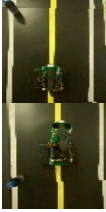
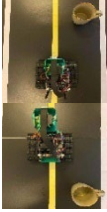
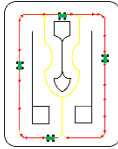
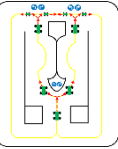
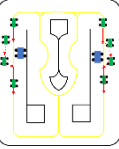
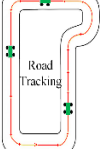
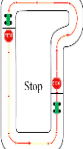
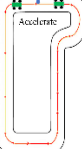
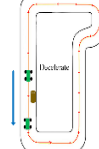
We used the smart car to collect data for the MTS mode and the MOD mode on track (a) and track (b), respectively. The size of the data image is 224\*224. The MTS mode needed to achieve three different tasks. We collected the data in sequence according to the task classification, and divided the dataset into road tracking dataset 1, left-right turn sign recognition dataset, and automatic obstacle avoidance dataset.

The MOD mode needed to collect datasets for MT-ResNet26 and ssd\_mobilenet\_v2. MT-ResNet26, and was responsible for the road tracking task. We named the dataset of MT-ResNet26

as road tracking dataset 2. Ssd\_mobilenet\_v2 was responsible for multi-object detection tasks. We used TensorFlow object detection API to directly train the model using the coco dataset without collecting the dataset separately. We used "stop" as the stop target, target A (bottle) as the acceleration target, and target B (cup) as the deceleration target.

Moreover, we only collected part of the images on the track for each task so that the smart car could operate in an unseen environment. The collection method is shown in the data collection route in Table 1. Details of the collected data are shown in Table 1.

**Table 1.** Overview of data collection

Mode	MTS			MOD			
Model	MT-ResNet26			MT-ResNet26	Ssd_mobilenet_v2		
Task	Road tracking1	Left-right turn sign recognition	Automatic obstacle avoidance	Road tracking2	Stop	Accelerate	Decelerate
Camera field of view							
top view							
Number	1750	900	350	2000	Dataset from coco		
Data collection route							

### 2.6.3 Model training

To train the autonomous driving model, we built the training environment in Jetson Nano, and Table 2 shows our environment configuration information.

**Table 2.** Environment configuration

Environment configuration					
Request	TensorRT	Pytorch	CUDA	CUDNN	Jetpack
Version	6.0.1.10	1.4.0	10.0.326	7.6.3.28	4.3

To test the applicability of the neural network and loss function proposed in this research, we needed to use different neural networks and loss functions to train multi-task autonomous driving models and compared the results between them.

We used the same hyperparameter setting scheme for all models to accurately test the improvement of our neural network and loss function on the model. Table 3 shows the hyperparameter settings. We used an embedded device as our motherboard with limited memory, so we chose the mini-batch size and mini-epoch for model training within the memory allowance of the Jetson Nano, with a batch size of 16 and epoch of 70. In addition, we chose the now very popular Adam as our optimizer because it is simple to use, computationally efficient, and has a small memory requirement [21].

**Table 3.** Hyperparameter setting

Hyper-parameters		
Optimizer	Batch Size	Epoch
Adam	16	70

We trained two multi-task autonomous driving models with MT-ResNet26 as the neural network. We called these models MT-ResNet26\_MSE and MT-ResNet26\_ST; MT-ResNet26\_MSE used the original loss function MSE, and MT-ResNet26\_ST used the ST Loss proposed in this research. Through these two sets of models, we tested the improvement of ST Loss on autonomous driving.

To better compare with existing autonomous driving methods, we trained the existing autonomous driving models. In the research of Suo *et al.* [22], they proposed a new neural network based on the VGG16 neural network and achieved the road tracking task. We trained their autonomous driving model and called it YS-VGG17\_MSE.

Furthermore, we used a multi-task approach as we proposed to train two existing classical models—ResNet18 proposed by He *et al.* [23] and DenseNet121 proposed by Huang *et al.* [24], which we then referred to as ResNet18\_MSE\* and DenseNet12\_MSE\*, \* representing the models using our proposed multi-task approach. Aside from testing the applicability and universality of the two approaches proposed in this research, this process also compared MT-ResNet26\_MSE to test the multi-task improvement of MT-ResNet26.

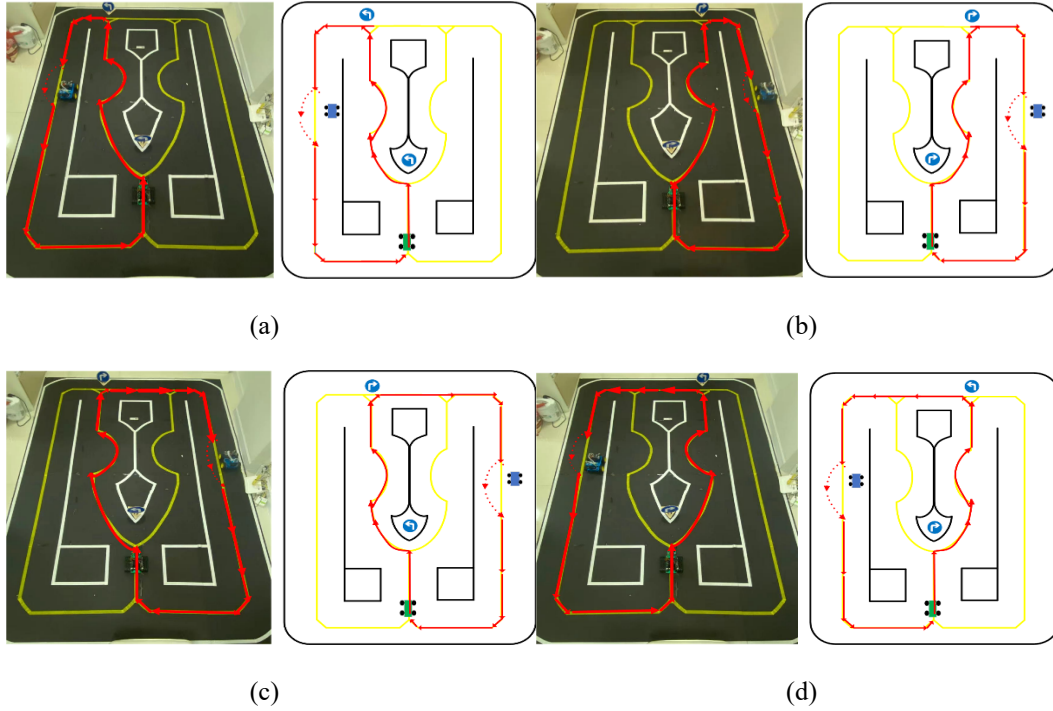
#### 2.6.4 Model testing

We loaded the MT-ResNet26\_ST, MT-ResNet26\_MSE, YS-VGG17\_MSE, ResNet18\_MSE\*, and DenseNet12\_MSE\* models into the autonomous driving car for model testing. The model testing used two maps to test the performance of the multi-task autonomous driving model and the feasibility of the two multi-task approaches. The test track for MST mode is shown in Figure 9, and the test track for MOD is shown in Figure 10.

We randomly set the track to better test the performance of autonomous driving smart car in unseen scenarios.

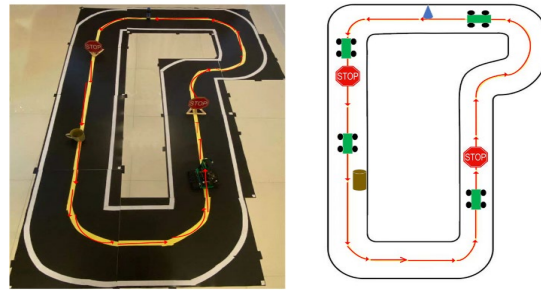


In the MTS mode track, we randomly placed the left-right turn signs at the left-right turns of the track and freely placed the obstacle positions. A total of four tracks were set, as shown in Figure 9.



**Figure 9.** MST mode test track

In the MOD mode track, we arbitrarily placed the target positions to set the track, as shown in Figure 10.



**Figure 10.** MOD mode test track

In addition, to further improve the multi-task performance of the smart car, we optimized the above autonomous driving models to obtain MT-ResNet26\_ST\_TRT, MT-ResNet26\_MSE\_TRT, YS-VGG17\_MSE\_TRT, ResNet18\_MSE\*\_TRT, and DenseNet12\_MSE\*

\_TRT, respectively, and loaded these onto the autonomous driving smart car to test the optimization effects of the OMTS and OMOD modes on two sets of tracks.

### 3. Results and Discussion

#### 3.1 Model evaluation loss value and variance

A lot of research today uses the evaluation loss values to reflect the performance of the model [25, 26], i.e. the smaller the loss value of the model, the better the model. In addition, the loss variance can represent the stability of the loss values of the model. Therefore, we trained the MT-ResNet26\_ST, MT-ResNet26\_MSE, YS-VGG17\_MSE, ResNet18\_MSE\* and DenseNet121\_MSE\* models five times, and recorded their loss values and loss variances, respectively. The loss values and variances of the models are shown in Table 4.

**Table 4.** Model evaluation loss value and variance

MTS				MOD			
Model	Loss	AVG Loss	Variance of Loss	Model	Loss	AVG Loss	Variance of Loss
ResNet-18_MSE*	0.004677	0.004648	1.00E-08	ResNet-18_MSE*	0.005505	0.005557	5.97E-09
	0.004553				0.005435		
	0.004798				0.005603		
	0.004691				0.005651		
	0.004522				0.005593		
DenseNet-121_MSE*	0.005423	0.005548	1.34E-08	DenseNet-121_MSE*	0.005240	0.005417	1.95E-08
	0.005688				0.005660		
	0.005435				0.005343		
	0.005512				0.005451		
	0.005681				0.005391		
YS-VGG17_MSE	0.007157	0.00736	1.77E-08	YS-VGG17_MSE	0.004962	0.005691	3.67E-07
	0.007356				0.005396		
	0.007436				0.005402		
	0.007553				0.005978		
	0.007296				0.006715		
MT-ResNet26_MSE	0.004565	0.004511	3.99E-09	MT-ResNet26_MSE	0.003871	0.003798	2.76E-09
	0.004421				0.003707		
	0.004581				0.003804		
	0.004453				0.003811		
	0.004537				0.003795		
MT-ResNet26_ST	0.000674	0.000685	3.07E-10	MT-ResNet26_ST	0.000510	0.000547	5.63E-10
	0.000718				0.000561		
	0.000684				0.000578		
	0.000679				0.000553		
	0.000668				0.000531		

\* Represent the model using our proposed multi-task approach

As shown in Table 4, ResNet-18\_MSE\*, DenseNet-121\_MSE\*, YS-VGG17\_MSE, and MT-ResNet26\_MSE models were trained with the same loss function but with different neural networks. By comparing the above four groups of models, it can be seen that MT-ResNet improved the model. In MTS mode, MT-ResNet26\_MSE had the lowest average loss of 0.004511, and YS-VGG17\_MSE had the highest average loss of 0.00736. In MOD mode, MT-ResNet26\_MSE also had the lowest average loss of 0.003798, and YS-VGG17\_MSE had the highest average loss of 0.005691. Furthermore, MT-ResNet26\_MSE and MT-ResNet26\_ST use different loss functions. By comparing these two models, the evaluation loss value of the model trained with ST Loss was significantly reduced. MT-ResNet26\_ST had the lowest average losses, 0.000685 and 0.000547, in the MTS and MOD modes, respectively. In summary, it is apparent that the proposed neural network and loss function effectively reduced the evaluation loss value of model training.

In addition, as seen in Table 4, the variance of all the autonomous driving models was extremely low, and the loss variance of MT-ResNet26\_ST was the lowest and most stable.

### 3.2 Actual performance of multi-task autonomous driving

In the MTS mode test, we divided multi-task autonomous driving into three tasks: road tracking (task I), obstacle avoidance (task II), and left-right turn sign recognition (task III). To judge the performance of the model, we set the total score of multi-tasks autonomous driving as 100 points, with task I accounting for 30 points and task II and task III each accounting for 35 points. If the smart car achieved the corresponding task during the experiment, it got the corresponding score. Points were deducted if mistakes were made during the experiment. Each time the smart car touched the white line, each time 1 point was deducted from the total score of task I. Each time the smart car could not avoid obstacles, each time 2 points were deducted from the total score of task II; each time the smart car did not recognize the of turn left and right sign, two points were deducted from the total score of task III.

In the MOD mode test, we divided multi-task autonomous driving into three tasks: road tracking (task I), acceleration and deceleration (task II), and stopping (task III). Score settings were the same as for MTS mode. Each time the smart car touched the white line during the experiment, 1 point was deducted from the total score of task I. Each time the smart car could not stop, accelerate or decelerate, 2 points were deducted from the total scores of task II and task III, respectively.

In order to evaluate the performance of various multi-task autonomous driving models more clearly, we introduced the concept of a perfect score, which indicated that the autonomous driving car achieved all tasks without touching the white line, and we proposed the concept of accuracy rate, which referred to the degree of accuracy with which the multi-task autonomous driving model can achieve multi-tasks. We defined it as:

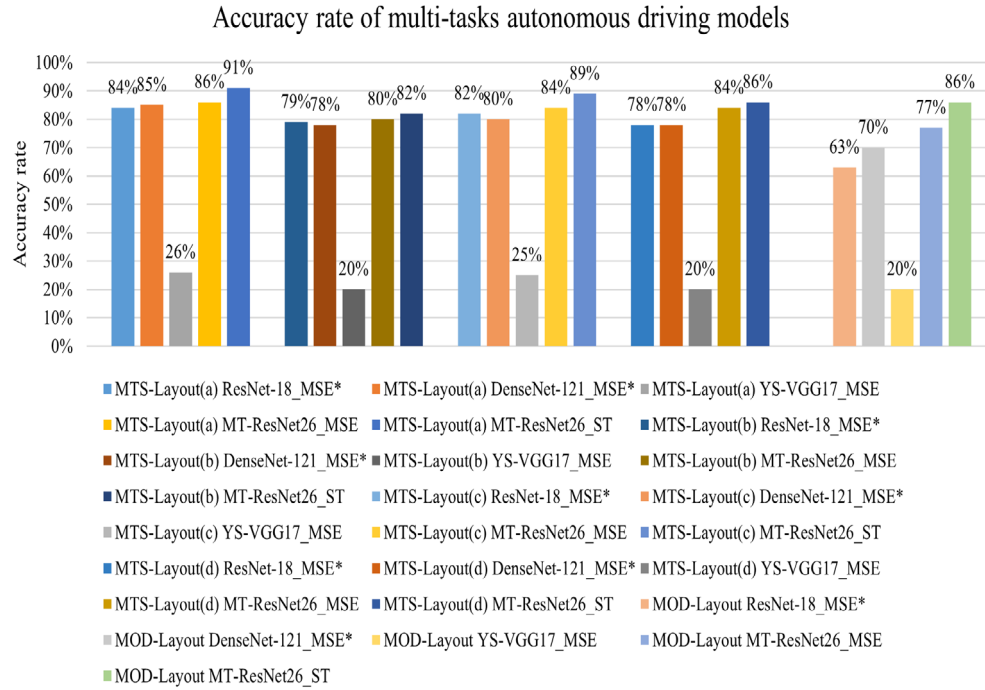
$$Accuracy\ rate = \frac{Sum_{actual\ score}}{Sum_{ideal\ score}} \times 100\% \quad (3)$$

We completed three laps of multi-task autonomous driving on the track. The final score and the accuracy rate were judgements of the performance of multi-task autonomous driving. The final score and the accuracy rate of different multi-task autonomous driving models are shown in Table 5 and Figure 11.

The experiment videos can be viewed at this URL: <https://github.com/MTAD1/Experimental-video>

**Table 5.** Multi-task autonomous driving model scores

<b>MTS</b>					
	<b>Model</b>	<b>Task I</b>	<b>Task II</b>	<b>Task III</b>	<b>Total Score</b>
Layout(a)	ResNet-18_MSE*	24	31	29	84
	DenseNet-121_MSE*	27	29	29	85
	YS-VGG17_MSE	26	0	0	26
	MT-ResNet26_MSE	28	27	31	86
	MT-ResNet26_ST	29	31	31	91
Layout(b)	ResNet-18_MSE*	21	27	31	79
	DenseNet-121_MSE*	22	31	25	78
	YS-VGG17_MSE	20	0	0	20
	MT-ResNet26_MSE	24	29	27	80
	MT-ResNet26_ST	24	31	27	82
Layout(c)	ResNet-18_MSE*	24	27	31	82
	DenseNet-121_MSE*	26	31	23	80
	YS-VGG17_MSE	25	0	0	25
	MT-ResNet26_MSE	24	31	29	84
	MT-ResNet26_ST	27	31	31	89
Layout(d)	ResNet-18_MSE*	26	27	25	78
	DenseNet-121_MSE*	24	29	25	78
	YS-VGG17_MSE	20	0	0	20
	MT-ResNet26_MSE	22	35	27	84
	MT-ResNet26_ST	26	33	27	86
<b>MOD</b>					
	<b>Model</b>	<b>Task I</b>	<b>Task II</b>	<b>Task III</b>	<b>Total Score</b>
Layout	ResNet-18_MSE*	15	19	29	63
	DenseNet-121_MSE*	18	21	31	70
	YS-VGG17_MSE	20	0	0	20
	MT-ResNet26_MSE	21	25	31	77
	MT-ResNet26_ST	24	29	33	86



**Figure 11.** Accuracy rate of multi-task autonomous driving models

For the actual operational effectiveness of the multi-task autonomous driving smart car, we scored the performance of different multi-task autonomous driving models to evaluate the performance of different models under multi-task scenario and the results are shown in Table 5 and Figure 11. Because the YS-VGG17\_MSE proposed by Suo *et al.* [22] could only manage the road tracking task, the YS-VGG17\_MSE model has the lowest scores and accuracy rate in both MTS and MOD modes. ResNet-18\_MSE\*, DenseNet-121\_MSE\* achieve multi-tasking after using the method proposed in this research, but the neural network and loss function used were not accurate and robust, and the scores and accuracy rate were lower than MT-ResNet26\_MSE and MT-ResNet26\_ST. After training the model with MT-ResNet26, the scores and accuracy rate of MT-ResNet26\_MSE significantly improved, but were still lower than that of MT-ResNet26\_ST with ST Loss. Finally, the scores and accuracy rate of MT-ResNet26\_ST were the highest in both MTS and MOD modes.

When combining the model evaluation loss value and the actual operation of the smart car, our proposed MT-ResNet26\_ST model achieved multi-task autonomous driving better, which shows that MT-ResNet26 and ST Loss can effectively improve model training and multi-task performance. At the same time, MT-ResNet26\_ST achieved multi-task autonomous driving in an unseen environment, which shows that MT-ResNet26 and ST Loss enhanced the robustness of the model and reduced interference due to noise from the data. Furthermore, our proposed multi-task approaches can be applied to different models, enabling them to achieve multi-task autonomous driving, which shows the generality of our proposed approaches.

### 3.3 Testing of the optimized multi-task mode (OMTS and OMOD)

To solve the problems of slow model inference and delay caused by embedded devices, we proposed an optimized multi-task mode approach to optimize the autonomous driving model using semi-precision techniques through the TensorRT framework. We tested the inference speed of different autonomous driving models before and after optimization using datasets collected in MTS mode (3000 images) and MOD mode (2000 images), respectively. The inference speeds of the autonomous driving models in OMTS mode and OMOD mode are shown in Table 6.

**Table 6.** The inference speeds of different autonomous driving models before and after optimization

OMTS			
Model	Pytorch(s)	TensorRT(s)	Optimized speed / Original speed(x)
ResNet-18_MSE*	0.043622	0.005933	7.3524
DenseNet-121_MSE*	0.107876	0.027502	3.9224
YS-VGG17_MSE	0.040207	0.008584	4.6839
MT-ResNet26_MSE	0.032854	0.002548	12.894
MT-ResNet26_ST	0.028271	0.002192	12.8973
OMOD			
Model	Pytorch(s)	TensorRT(s)	Optimized speed / Original speed(x)
ResNet-18_MSE*	0.037718	0.004526	8.3336
DenseNet-121_MSE*	0.104753	0.027442	3.8172
YS-VGG17_MSE	0.03139	0.00635	4.9433
MT-ResNet26_MSE	0.030146	0.002284	13.1987
MT-ResNet26_ST	0.0271	0.001964	13.7962

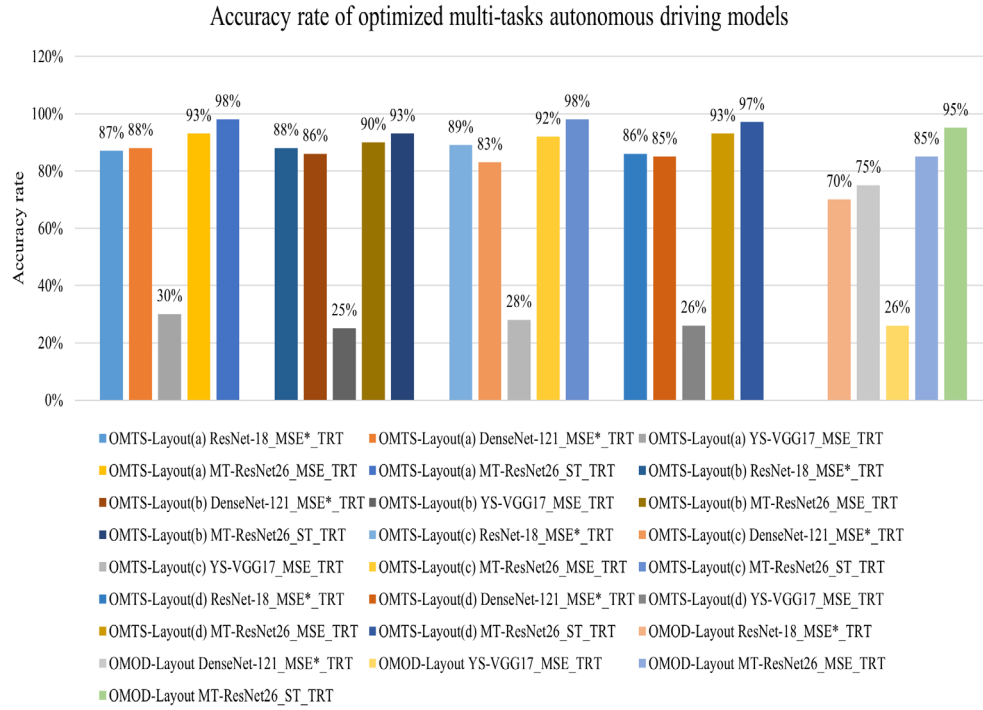
From Table 6, it can be seen that the different autonomous driving models were optimized (FP16) with considerable improvement in the inference speed. Among them, the autonomous driving models using DenseNet-121 and YS-VGG17 as neural networks had the slightest improvements in OMTS and OMOD modes, with only a 3–4 times improvement in inference speed. In contrast, the autonomous driving models using MT-ResNet26 showed the most significant improvement. The model inference speed was improved by more than ten times. In addition, the models using ST Loss as the loss function had more considerable improvement than the MSE loss function. Furthermore, MT-ResNet26\_ST had the fastest inference speeds of 0.002192 s and 0.00194 s. This indicates that the autonomous driving models trained using our proposed MT-ResNet26 and ST Loss had a huge optimizable space and the best performance to achieve multi-tasks. We loaded all optimized autonomous driving models onto the smart car to verify our theory for multi-task testing. The final score and the accuracy rates of the different optimized multi-task autonomous driving models are shown in Table 7 and Figure 12. (Experiments video can be viewed at this URL: <https://github.com/MTAD1/Experimental-video-TRT>)

We evaluated the performance of the different multi-task autonomous driving models after optimization. As shown in Table 7 and Figure 12, all the autonomous driving models improved on both optimized multi-task modes. YS-VGG17\_MSE\_TRT, although improved in achieving the road tracking task, still had the lowest scores and accuracy rates due to its inability to achieve other tasks. The final score and the accuracy rate of MT-ResNet26\_TRT\_ST were the highest in both OMTS and OMOD modes due to the excellent performance of MT-ResNet26 and ST Loss.

**Table 7.** Optimized multi-task autonomous driving model score

<b>OMTS</b>					
	<b>Model</b>	<b>Task I</b>	<b>Task II</b>	<b>Task III</b>	<b>Total Score</b>
Layout(a)	ResNet-18_MSE*_TRT	27	29	31	87
	DenseNet-121_MSE*_TRT	28	31	29	88
	YS-VGG17_MSE_TRT	30	0	0	30
	MT-ResNet26_MSE_TRT	29	33	31	93
	MT-ResNet26_ST_TRT	30	35	33	98
Layout(b)	ResNet-18_MSE*_TRT	26	31	31	88
	DenseNet-121_MSE*_TRT	26	31	29	86
	YS-VGG17_MSE_TRT	25	0	0	25
	MT-ResNet26_MSE_TRT	28	31	31	90
	MT-ResNet26_ST_TRT	29	33	31	93
Layout(c)	ResNet-18_MSE*_TRT	29	29	31	89
	DenseNet-121_MSE*_TRT	27	29	27	83
	YS-VGG17_MSE_TRT	28	0	0	28
	MT-ResNet26_MSE_TRT	28	33	31	92
	MT-ResNet26_ST_TRT	30	35	33	98
Layout(d)	ResNet-18_MSE*_TRT	28	31	27	86
	DenseNet-121_MSE*_TRT	27	31	27	85
	YS-VGG17_MSE_TRT	26	0	0	26
	MT-ResNet26_MSE_TRT	27	35	31	93
	MT-ResNet26_ST_TRT	29	35	33	97
<b>OMOD</b>					
	<b>Model</b>	<b>Task I</b>	<b>Task II</b>	<b>Task III</b>	<b>Total Score</b>
Layout	ResNet-18_MSE*_TRT	18	21	31	70
	DenseNet-121_MSE*_TRT	21	25	29	75
	YS-VGG17_MSE_TRT	26	0	0	26
	MT-ResNet26_MSE_TRT	25	29	31	85
	MT-ResNet26_ST_TRT	27	33	35	95





**Figure 12.** Accuracy rate of optimized multi-task autonomous driving model

We also investigated the improvement of accuracy for different multi-task autonomous driving models. Table 8 shows that the multi-task autonomous driving model using MT-ResNet26 as the neural network had more accuracy after optimization than the other models. In addition, when ST Loss was used as the loss function, the accuracy improvement of the model in achieving multi-tasks was further enhanced. Among them, MT-ResNet26\_TRT\_ST had the highest multi-task accuracy and showed enormous improvement. This indicated that our proposed MT-ResNet26 and ST Loss not only enhanced the robustness of the model and improved the multi-task achievement accuracy of the model but also had a large room for optimization, which further improved the performance of the model.

**Table 8.** Degree of accuracy improvement of multi-task autonomous driving model

Model	OMTS				OMOD
	Layout(a)	Layout(b)	Layout(c)	Layout(d)	Layout
ResNet-18_MSE*_TRT	3%	9%	7%	8%	7%
DenseNet-121_MSE*_TRT	3%	8%	3%	7%	5%
YS-VGG17_MSE_TRT	4%	5%	3%	6%	6%
MT-ResNet26_MSE_TRT	7%	10%	8%	9%	8%
MT-ResNet26_ST_TRT	7%	11%	9%	11%	9%

## 4. Conclusions

We built an autonomous driving smart car that only used one camera as a sensor and proposed two approaches of MTS and MOD to achieve multi-task autonomous driving. We found that deep neural networks had higher requirements for datasets, while shallow neural networks could not accurately achieve multi-task autonomous driving, so we proposed a new deep neural network architecture, MT-ResNet-26. In order to achieve the multi-task challenge, we found that the existing loss function could not handle the noise and class imbalance from the data, so we proposed a novel loss function - ST Loss, which enabled the smart car to achieve multi-task autonomous driving efficiently.

To test the improvement of our proposed neural network architecture, loss function, and multi-task approaches, we used the MT-ResNet26 proposed in this research as the neural network and trained MT-ResNet\_MSE and MT-ResNet\_ST with the original MSE and ST Loss as the loss functions, respectively. Moreover, to compare our multi-task approach with existing autonomous driving methods, we trained an autonomous driving model YS-VGG17\_MSE proposed by Suo *et al.* [22]. We found that YS-VGG17\_MSE was far worse than our method for multi-task autonomous driving. Furthermore, we used our proposed approach to improve the existing classic models - ResNet18 and DenseNet121, and the trained ResNet-18\_MSE\* and DenseNet-121\_MSE\* achieved multi-tasks after using our method, which simultaneously verified the applicability of our proposed method. Finally, we compared them against MT-ResNet\_MSE and MT-ResNet\_ST proposed in this research to demonstrate the multi-task performance improvement of the proposed neural network and loss function.

For the loss values and variances, the model trained using the MT-ResNet26 deep neural network had lower values than other neural networks. In addition, the accuracy of the model could be further improved by applying our proposed ST Loss to the model.

For actual operation, our approach was better than the existing methods. After applying our method to the classical model, the classical model could also achieve multi-tasks, reflecting the applicability of our approach. Furthermore, MT-ResNet26 and ST Loss effectively improved the operation of multi-task autonomous driving. In addition, we tested the multi-task autonomous driving performance under unknown scenarios by constructing different track routes. The experiments show that the MT-ResNet26\_ST model using our multi-task approach performed well on untrained routes.

To further improve the performance in the multi-task scenario, we proposed optimized multi-task modes, namely the OMTS and OMOD modes. OMTS and OMOD modes were based on the TensorRT framework using FP16 techniques to accelerate the inference of multi-task autonomous driving models. Experiments demonstrated that the performance of optimized multi-task models was improved. In addition, the MT-ResNet26\_TRT\_ST model had the best multi-task performance and the most considerable accuracy improvement after optimization, which also indicated that our proposed MT-ResNet26 and ST Loss had superior optimizability.

## 5. Acknowledgements

The first and last authors each contributed 50% equally to this work. The first author received scholarship support from CPALL for conducting this research in PIM

## References

- [1] Hossain, N., Kabir, M.T., Rahman, T.R., Hossen, M.S. and Salauddin, F., 2010. A real-time surveillance mini-rover based on OpenCV-Python-JAVA using Raspberry Pi 2. *Proceedings of the 2005 IEEE International Conference on Control System, Computing and Engineering (ICCSCCE)*, Penang, Malaysia, November 27-29, 2015, pp. 476-481.
- [2] Yılmaz, E. and Tariyan Özyer, S., 2019. Remote and autonomous controlled robotic car based on Arduino with real time obstacle detection and avoidance. *Universal Journal of Engineering Science*, 7(1), DOI: 10.13189/ujes.2019.070101.
- [3] Iqbal, A., Ahmed, S.S., Tauqeer, M.D., Sultan, A. and Abbas, S.Y., 2017. Design of multifunctional autonomous car using ultrasonic and infrared sensors. *Proceedings of the 2017 International Symposium on Wireless Systems and Networks (ISWSN)*, Lahore, Pakistan, November 19-22, 2017, pp. 1-5.
- [4] Banerjee, A., Bolar, V., Chaurasia, A., Maurya, S. and Gite, Y., 2020. Autonomous driving vehicle. *International Research Journal of Engineering and Technology (IRJET)*, 7(4), 6048-6050.
- [5] Yuenyong, S. and Qu, J., 2017. Generating synthetic training images for deep reinforcement learning of a mobile robot. *Journal of Intelligent Informatics and Smart Technology*, 2(3), 16-20.
- [6] Rausch, V., Hansen, A., Solowjow, E., Liu, C., Kreuzer, E. and Hedrick, J.K., 2017. Learning a deep neural net policy for end-to-end control of autonomous vehicles. *Proceedings of the 2017 American Control Conference (ACC)*, WA, USA, May 24-26, 2017, pp. 4914-4919.
- [7] Bechtel, M.G., McEllhiney, E., Kim, M. and Yun, H., 2018. Deeppicar: A low-cost deep neural network-based autonomous car. *Proceedings of the 2018 IEEE 24<sup>th</sup> International Conference on Embedded and Real-time Computing Systems and Applications (RTCSA)*, Hakodate, Japan, August 28-31, 2018, pp. 11-21.
- [8] Meyer, G.P. and Thakurdesai, N., 2020. Learning an uncertainty-aware object detector for autonomous driving. *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, October 24, 2020 - January, 24, 2021, pp. 10521-10527.
- [9] Ren, X., Yang, T., Li, L.E., Alahi, A. and Chen, Q., 2021. Safety-aware motion prediction with unseen vehicles for autonomous driving. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Virtual, October 10-17, 2021, pp. 15731-15740.
- [10] Do, T.D., Duong, M.T., Dang, Q.V. and Le, M.H., 2018. Real-time self-driving car navigation using deep neural network. *Proceedings of the 2018 4<sup>th</sup> International Conference on Green Technology and Sustainable Development (GTSD)*, Ho Chi Minh City, Vietnam, November 23-24, 2018, pp. 7-12.
- [11] Al-Nima, R.R.O., Han, T. and Chen, T., 2019. Road tracking using deep reinforcement learning for self-driving car applications. *Proceedings of the International Conference on Computer Recognition Systems*, Polanica-Zdroj, Poland, May 20-22, 2019, pp. 106-116.
- [12] Kulkarni, R., Dhavalikar, S. and Bangar, S., 2018. Traffic light detection and recognition for self driving cars using deep learning. *Proceedings of the 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBE)*, Pune, India, August 16-18, 2018, pp. 1-4.
- [13] Yang, Z., Li, J. and Li, H., 2018. Real-time pedestrian and vehicle detection for autonomous driving. *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China, June 26-30, 2018, pp. 179-184.
- [14] Fang, C.Y., Chen, S.W. and Fuh, C.S., 2003. Road-sign detection and tracking. *IEEE Transactions on Vehicular Technology*, 52(5), 1329-1341, DOI: 10.1109/TVT.2003.810999.

- 
- [15] Mitsch, S., Ghorbal, K. and Platzer, A., 2013. On provably safe obstacle avoidance for autonomous robotic ground vehicles. *Proceedings of the Robotics: Science and Systems IX*, Berlin, Germany, June 24-28, 2018, pp. 14-21.
  - [16] Kang, D.H., Bong, J.H., Park, J. and Park, S., 2017. Reinforcement learning strategy for automatic control of real-time obstacle avoidance based on vehicle dynamics. *Journal of Korea Robotics Society*, 12(3), 297-305, DOI: 10.7746/jkros.2017.12.3.297.
  - [17] Tao, L., Hong, T., Guo, Y., Chen, H. and Zhang, J., 2020. Drone identification based on CenterNet-TensorRT. *Proceedings of the 2020 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, Paris, France, October 27-29, 2020, pp. 1-5.
  - [18] Ding, S. and Qu, J., 2022. Smart car with road tracking and obstacle avoidance based on Resnet18-CBAM. *Proceedings of the 2022 7<sup>th</sup> International Conference on Business and Industrial Research (ICBIR)*, Bangkok, Thailand, May 19-20, 2022, pp. 582-585.
  - [19] Willmott, C.J. and Matsuura, K., 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, 30(1), 79-82, DOI: 10.3354/cr030079.
  - [20] Sutanto, A.R. and Kang, D.K., 2021. A Novel Diminish Smooth L1 Loss Model with Generative Adversarial Network. *Proceedings of the 12<sup>th</sup> International Conference on Intelligent Human Computer Interaction*, Daegu, South Korea, November 24-26, 2020, pp. 361-368.
  - [21] Mehta, S., Paunwala, C. and Vaidya, B., 2019. CNN based traffic sign classification using Adam optimizer. *Proceedings of the 2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, Madurai, India, May 15-17, 2019, pp. 1293-1298.
  - [22] Suo, Y., Chen, S. and Zheng, M., 2020. Developing an Autonomous Driving Model Based on Raspberry Pi. *Proceedings of the 53<sup>rd</sup> Annual Midwest Instruction and Computing Symposium*, Milwaukee Wisconsin, USA, April 3-4, 2020, pp. 56-66.
  - [23] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, USA, June 27-30, 2016, pp. 770-778.
  - [24] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q., 2017. Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July 21-26, 2017, pp. 4700-4708.
  - [25] Kocić, J., Jovičić, N. and Drndarević, V., 2019. An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms. *Sensors*, 19(9), 2064-2090, DOI: 10.3390/s19092064.
  - [26] Li, Y. and Qu, J., 2022. Intelligent road tracking and real-time acceleration-deceleration for autonomous driving using modified convolutional neural networks. *Current Applied Science and Technology*, 22(6), 199-225.