Optimization of Road Condition Decision at Intersection of V2X Control System based on Federated Learning and Edge Computing

Jiawei Tian
School of Automotive Engineering, Harbin Institute of Technology, Vehicle Engineering, Weihai, China

Abstract: As a kind of vehicular wireless communication technology, the Internet of vehicles (V2X) is a new generation of information and communication technology that connects vehicles with all objects on the road, where V stands for vehicles and X stands for objects that interact with the vehicle. At present, X mainly includes vehicles, people, traffic road infrastructure and network. With the help of modern wireless communication technology, the Internet of vehicles technology can realize the interaction between vehicles and between vehicles and the outside world, and can fully automatically remind drivers of possible safety risks, so as to effectively avoid traffic accidents. [1] With the rapid development of Internet of vehicles (IoV) technology, autonomous driving is evolving from single-vehicle intelligence to multi-agent vehicle-road collaboration mode, and its potential in improving road safety and traffic efficiency has attracted much attention. Current V2X applications, however, still faces significant challenges: network fluctuation result in higher data transmission delay, the cloud centralized computing model of great communication and calculate power burden; The traditional path planning model based on distributed learning is prone to make redundant or delayed decisions in dynamic scenarios such as complex intersections due to the large number of parameters and poor scene adaptability, and it is difficult to make the optimal path planning.

Aiming at the problem of the traditional distributed learning, this paper puts forward a kind of edge fusion calculation and federal study of the collaborative decision-making optimization framework. Edge computing reduces the end-to-end transmission delay by offloading data processing to road side units or vehicle terminals. Federated learning supports local training of multi-vehicle data and global model aggregation, which reduces redundant gradient calculation while protecting privacy and enables the control system to choose the optimal route. Based on this framework, this paper adopts CarSim associated with Simulink simulation platform, to build a typical urban intersection set, in view of the automatic vehicle decision-making process is optimized. In the experiment, the key indicators of the optimized vehicle such as driving curve, speed change rate, acceleration change rate (ride comfort) are analyzed, and the advantages of the proposed method are verified. Results show that the research for the lightweight, highly efficient of the autopilot system, privacy decision-making provides a new train of thought, has the important value of engineering application.

Keywords: Automatic Driving; Co-Simulation; Federated Learning; Experimental Analysis

1. Background

1.1 Federated Learning and Edge Computing

1.1.1 Federated learning

Federal study is research by Google in 2016 first put forward the concept of the technology can be finished in a case of not sharing data joint model, equivalent to the combination of distributed learning and encryption technology, and the aggregation is federal study is different from the optimization algorithm of distributed learning system, to solve the heterogeneous data of independent identically distributed and reduce data provides a new train of thought. [2] in the large amount of data and computing tasks has obvious advantages and the user does not need to consider when heavy data privacy, so local calculation can be applied to the vehicle terminal equipment. Figure 1 is the traditional distributed autonomous learning model structure diagram.

Terminal sensors complex road conditions of traffic data and output decision signals and feedback value, promote the system operation to complete safety and efficiency.

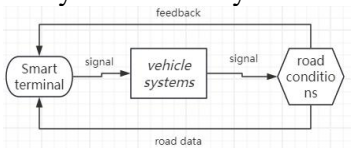

Figure 1. Schematic Diagram of the Distributed Learning Autonomous Driving Model

In Hu Bingxu [3] 's experiment, the average reward value was used to measure the performance of the federated learning and independent learning models, and the average reward value at a certain time was defined as the arithmetic average of the rewards and values in a certain number of previous rounds up to that time. The experimental results show that the MFRL algorithm based on federated learning can select stable and excellent cooperation objects for intelligent vehicles, so as to accelerate the training while ensuring the high performance of the model. Experiments with sunlin [4], and the study also found, on the premise of meet federal study constraints, will study the technique is applied to autonomous perception training algorithm, and reduce the training cost at the same time improve the generalization ability of network model, has important research value and significance.

Different from the classical maximum likelihood estimation method MLE, which transmits the data of different vehicles to the road side unit, the key parameters can be obtained by MLE based on federated learning, which can save communication resources for data transmission and avoid the problem of privacy leakage. Practice has proved, based on the learning schemes can save up to 79% of the cost, but similar solution accuracy and center [5]. FIG. 2 shows a schematic diagram of the federated aggregation model for autonomous driving with federated learning.

The base station server complete local updated transfer model to the model, then the model of input the updated model to the server (x1, x2, xn), base station server aggregation calculated global model x ', then repeat step iteration. This way of learning will be calculated from a central server to local vehicle, through continuous training model to minimize the loss function. The calculation accuracy is improved, the privacy problem is solved, and the computational cost is reduced.

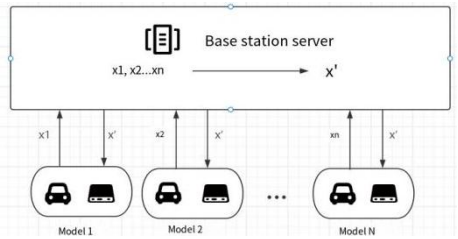

Figure 2. Schematic Diagram of Federated Aggregation Model for Federated Learning

1.1.2 Edge computing

Edge computing is a computing method based on cloud computing, which can collect and analyze data in the local device terminal or local area network close to the data generation, rather than the need to transmit data to the centralized cloud for calculation. It is often used to process time-sensitive information. Therefore, edge computing can be applied to the vehicle terminal of autonomous driving vehicles. The edge computing model can not only reduce the data transmission bandwidth, but also better protect private data and reduce the risk of privacy leakage of sensitive data on the terminal. Therefore, with the development of the Internet of Everything, edge computing model will become an emerging application support platform for the Internet of Everything [6].

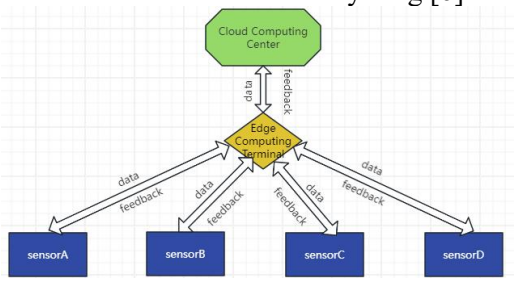

Figure 3. Schematic Diagram of Edge Computing Model Structure

Figure 3 shows the structure diagram of the edge computing model. The local terminal collects vehicle data obtained by sensors and transmits it to the edge computing center for local calculation and analysis, then interacts with the cloud computing center, and outputs feedback signals to the terminal at the same time, which has high efficiency and strong privacy.

1.2 Vehicle-road Cooperative Control System

With the advent of 5G era, combined with the general trend of Internet of things + artificial intelligence, it has developed rapidly in many vertical industry applications, and the Internet of Vehicles to Everything (V2X) is one of them. Car road synergy is connected to the application

of one way, it is based on wireless communication and sensing edge detection, artificial intelligence and computing technology, vehicle and road information through various channels, through the car-car, the car-way communication interaction and sharing of information, and in all the time and space, on the basis of dynamic traffic information collection and integration, to carry out the active safety control and coordinated management roads, To fully realize the effective cooperation between cars and roads, ensure traffic safety, improve traffic efficiency, and form a safe, efficient and environmentally friendly road traffic system, which provides strong support for application scenarios such as assisted driving, automatic driving, traffic management and public travel. [B]However, there are many problems in the practical application of V2X system. The large amount of data generated by the control system based on the traditional learning mode will produce large delays in the cloud computing process, and the driving data does not have strong privacy.

1.3 Optimization Scheme of Control System

This article is based on the average algorithm and vehicles through the intersection crossroads car model, using Carsim software modeling with the Simulink module, and training simulation data with the mathematical principle of vehicles through the intersection traffic. The simulation results such as the curve of vehicle speed change and acceleration change are analyzed to draw relevant conclusions.

The behavior of vehicles through the intersection can be divided into two categories: (1) the vehicle does not need to stop and wait, just decelerate without stopping, that is, the vehicle passes through the intersection through deceleration and uniform speed condition; (2) the vehicle needs to stop and wait for a certain time before accelerating and starting to pass through the intersection, that is, the vehicle passes through deceleration, acceleration and uniform speed condition. This study considered the first (1).

Table 1. Car Model Parameters

Vehicle mass	Upwind area	Air drag coefficient	Coefficient of rolling resistance	Gravitational acceleration	Drivetrain efficiency
1600kg	2.0 m	0.3	0.012	9.8	0.9
Distance between vehicle and intersection	Starting acceleration	Maximum vehicle speed	Low driving speed	No stop process time setting	
200m	$2m/s^2$	15m/s	5m/s	15s	

Table 1 is the simulation selection of automobile related parameters. Based on the above table, the mathematical model is built as follows: in the ideal state, consider the case (1), when the intersection appears in front of the vehicle and the waiting time is short, the vehicle can realize the deceleration and pass without stopping. The vehicle will decelerate to a certain speed for a period of time, and then accelerate after passing the intersection.

The driving distance of the vehicle is

$$X_t=v_0t_{u-b}+\frac{1}{2}a_b(t_{b-u}-t_{u-b})^2+v_{b-u}(t_{u-a}-t_{b-u}) \quad (1)$$

Where X_t is the driving distance of the vehicle; v_0 is the initial speed; t_{u-b} is the start time of braking; v_{b-u} is vulgar constant speed; t_{u-a} is the end time of vulgar constant speed driving. Figure 4 and Figure 5 below show the road condition model built by using Carsim.

2. Core Algorithms

2.1 Overview of the Algorithm

In this paper, we adopt the federated averaging algorithm, which is one of the most commonly used federated learning optimization algorithms. Its essential idea is to take a local stochastic descent gradient method for each federated user to optimize the local model [F], and perform parameter aggregation operation on the aggregator. The core goal is to learn the mapping relationship between vehicle state and control quantities (throttle opening, brake pressure), and adapt to the simulation scenario of vehicle "uniform driving → intersection deceleration → uniform passing → acceleration to recover the original speed". Finally, it was deployed in MATLAB off-line training and Simulink/CarSim co-simulation environment. The core architecture of the algorithm is as follows:

1) Server: The Host is responsible for global model initialization, weight aggregation and update.

Edge node (Client) : Vehicle A, responsible for local dataset training and model weight uploading.

2) Core constraint: when the brake pressure is non-zero, the throttle opening is forced to 0; Throttle opening ∈[0,0.8] and brake pressure ∈[0,100] (normalized to [0,1] during training).

Figure 4. The Car is Running at a Uniform Speed Without Passing the Intersection

Figure 5. A car slows Down and Drives Without Stopping at an Intersection

2.2 Neural Network Model Structure

Based on the principle of T³OMVP method mentioned in the relevant experiment of Yuan Zheng [7], this paper adopts a lightweight fully connected neural network (MLP) to adapt to the real-time control of vehicles. The mathematical description of the model structure is as follows:

2.2.1 Input layer and output layer

The input layer is a 3D feature vector corresponding to the vehicle state parameters, and the mathematical expression is:

$$X=[t,v,v_{ref}]^T \quad (2)$$

Where: : current simulation time (s); tv: current vehicle speed (kph); v_{ref} : vehicle target driving speed (kph).

The output layer is a two-dimensional control vector, corresponding to the vehicle actuator commands, and the mathematical expression is:

$$Y=[\text{throttle},\text{brake}]^T \quad (3)$$

Where:

throttle: throttle opening, value range;[0,0.8]

brake: brake pressure, normalized to in training and restored to in simulation.[0,1][0,100]

Hidden layers and activation functions

The model consists of two fully connected hidden layers, and the ReLU activation function is used. The mathematical mapping relationship of each layer is as follows:

1. Input layer → hidden layer 1 (64 neurons) :

$$H_1=\text{ReLU}(W_1\cdot X+b_1) \quad (4)$$

Hidden layer 1 → Hidden Layer 2 (32 neurons) :

$$H_2=\text{ReLU}(W_2\cdot H_1+b_2) \quad (5)$$

Hidden layer 2 → Output layer:

$$Y=W_3\cdot H_2+b_3 \quad (6)$$

Where: is the weight matrix of each layer, and the dimensions are, and;$W_1,W_2,W_3$64×332×642×32 b_1,b_2,b_3 Are the bias vectors of each layer respectively, and the dimensions are, and;64×132×12×1

The ReLU activation function: is used to enhance the nonlinear fitting ability of the model and avoid the disappearance of the gradient.ReLU(x)=max(0,x)

2.3 Mathematical Principle and Model Training

2.3.1 Dataset Definition

The two-car dataset generated offline was used for training without running Simulink/CarSim, and the dataset was mathematically defined as follows:

Host host vehicle dataset::$D_{host}=\{(X_i,Y_{host,i}) \mid i=1,2,...,N\}$

Vehicle A dataset::$D_A=\{(X_i,Y_{A,i}) \mid i=1,2,...,N\}$

Where is the total number of data points (obtained by simulation time 20s and step size 0.001s),,

(brakepressurenormalized).

NN=20001$Y_{host,i}$

=[throttle$_{host,i}$,brake$_{host,i}$/100]^T$Y_{A,i}$

=[throttle$_{A,i}$,brake$_{A,i}$/100]^T

2.3.2 Definition of algorithm hyperparameters

Federated training core hyperparameters:

R : total number of federated training rounds (taken by this algorithm);R=10

α,β: Weight coefficient of center node and edge node (satisfied, taken by this algorithm);$\alpha+\beta=1\alpha=0.5,\beta=0.5$

E: the number of local training iterations (set by the training option and taken by the algorithm);E=5

η : learning rate (adaptively adjusted by Adam optimizer);

2.3.3 Complete flow of federated training

Step 1: Initialize the global model parameters

Initialize the global weight matrix with the bias vector:,, where the superscript 0 denotes the initial value. $W_{global}=[W_1^0,W_2^0,W_3^0]b_{global}=[b_1^0,b_2^0,b_3^0]$

Step 2: Federated Training Iterations (common

rounds)R

For each round, do the following:$r \in [1,R]$

1) Issue global model: The central node issues the current global model to the Host master vehicle and vehicle A.(W^r_{global},b^r_{global})

2) Local training: The two vehicles are trained independently based on the local data set, and the local model parameters are updated: Local training loss function (mean square error) : where is the model prediction output. $L=\frac{1}{N}\sum_{i=1}^{N} // Y_i-\hat{Y}_i //^2 \hat{Y}_i$

(1) Host: Based on the training, the local model parameters are obtained $D_{host}(W^r_{host},b^r_{host})$

(2) Vehicle A: Get local model parameters based on training$D_A(W^r_A,b^r_A)$

(3) Weight upload: The two vehicles only upload the locally trained model parameters to the central node, and do not share the original data.(W^r_{host},b^r_{host})(W^r_A,b^r_A)

(4) Federated weight aggregation (core step) :

The central node uses the weighted average method to aggregate local parameters and update the global model. The mathematical formula is as follows: Weight matrix aggregation: bias vector aggregation: where, is the global model parameters of the first round. $W^{r+1}_{global}=\alpha\cdot W^r_{host}+\beta\cdot W^r_A b^{r+1}_{global}=\alpha\cdot b^r_{host}+\beta\cdot b^r_A W^{r+1}_{global} b^{r+1}_{glo}$

Step 3: The training is terminated

When the iteration reaches the total number of rounds, the training is stopped and the final global model is saved for the subsequent Simulink/CarSim simulation deployment. $R(W^R_{global},b^R_{global})$

2.3.4 Mathematical expression of control constraints

In order to ensure the safety of vehicle driving, the model output should meet the following constraints:

1) Brake priority constraint: when the brake pressure, the throttle opening is forced to set to 0:brake>0

$$throttle=\begin{cases} 0, & \text{if brake}>0 \\ \hat{throttle}, & \text{else} \end{cases} \quad (7)$$

Where is the original throttle opening predicted by the model.$\hat{throttle}$

2) Control quantity range constraints:

0≤throttle≤0.8, (training phase);0≤brake≤1

0≤throttle≤0.8, (simulation stage);0≤brake≤100

3. Core code and Model

3.1 Data and network construction

MATLAB core implementation code:

matlab

```
% input feature X = [t, v, v_ref]^TX
 = [t', v_current, v_ref];

 % output label Y = [throttle, brake]^T (brake normalized)
y_host = [throttle_host, brake_host/100];
y_A    = [throttle_A,    brake_A/100];

 % neural network structure (corresponding to mathematical mapping in Section 2.2)
layers = [
featureInputLayer(3) % Input layer: 3D feature
  fullyConnectedLayer(64) % hidden Layer 1: W1 (64×3), b1 (64×1)
reluLayer % ReLU activation function
  fullyConnectedLayer(32) % hidden Layer 2: W2 (32×64), b2 (32×1)
reluLayer % ReLU activation function
   fullyConnectedLayer(2) % Output layer: W3 (2×32), b3 (2×1)
      regressionLayer];          % regression layer (corresponding to loss function L)

% training parameter Settings
options = trainingOptions('adam',...
  'MaxEpochs', 5, ...          % number of local iterations E=5
'MiniBatchSize', 256,...
'Verbose', false, ...
      'OutputAsSingle', false);      % Output double type, adapted to Simulink
```

3.2 Algorithm Core Functions

Federated aggregation algorithm core function

matlabfunction

```
  model_avg = federated_average_compatible(m1, m2, w1, w2)
 % Functions: Realize federated_weighted average, w1=α, w2=β (0.5, 0.5) % in this algorithm
  to extract the local model parameters of the two cars (W1, b1; W2, b2; W2) %. W3, b3)
 % m1: Host local model, m2: A vehicle local model

 % extract m1 (Host) parameters
  W1_1 = m1.layers (2).Weights; % W1 (hidden layer 1 weights)
 b1_1 = m1.Layers(2).Bias; % b1 (hidden layer 1 bias)
  W1_2 = m1.Layers(4).Weights; % W2 (hidden layer 2 weights)
 b1_2 = m1.Layers(4).Bias; % b2 (hidden layer 2 bias)
  W1_3 = m1.Layers(6).Weights; % W3 (output layer weights)
 b1_3 = m1.Layers(6).Bias; % b3 (output layer bias)

 % extract m2 (car A) parameters
```

```
  W2_1 = m2.layers (2).Weights;
  b2_1 = m2.Layers(2).Bias;
        W2_2 = m2.Layers(4).Weights;
        b2_2 = m2.Layers(4).Bias;
        W2_
 3 = m2.Layers(6).Weights;       b2_3
 = m2.Layers(6).Bias;
   % weighted average (corresponding formula: W_global = α·W_host + β·W_A)
   W_avg1 = w1 * W1_1 + w2 * W2_1; After % aggregation W1
    b_avg1 = w1 * b1_1 + w2 * b2_1; After % polymerization b1
   W_avg2 = w1 * W1_2 + w2 * W2_2; After % polymerization W2
    b_avg2 = w1 * b1_2 + w2 * b2_2; After % polymerization b2
   W_avg3 = w1 * W1_3 + w2 * W2_3; After % polymerization W3
    b_avg3 = w1 * b1_3 + w2 * b2_3; % after aggregation b3%

  Build aggregated global model
  layers_avg = [
 featureInputLayer(3)
  fullyConnectedLayer(64, 'Weights', W_avg1, 'Bias', b_avg1)
      reluLayer
         fullyConnectedLayer(32, 'Weights', W_avg2, 'Bias', b_avg2)
 reluLayer
          fullyConnectedLayer(2, 'Weights', W_avg3, 'Bias', b_avg3)
      regressionLayer];

  model_avg = assembleNetwork(layers_avg);     % Generate global model
 end
```

3.3 Federate the Main Training Loop

```
matlabnum_rounds
  = 10;    % total federated training rounds R=10%
  Initialize global model (initialize with first Host training results) global_model
    = trainNetwork(X, y_host, layers, options); % federated training iterations

  (r from 1 to R) for round
  = 1:num_rounds % Local training: two cars
  train independently model_host
  = trainNetwork(X, y_host, layers, options); % Host local training model_A =
 trainNetwork(X, y_A, layers, options); %
  A car local training % federated aggregation:
   Call core function with weight α=0.5, β=0.5 global_model =


            federated_average_compatible(model_host, model_A, 0.5, 0.5); fprintf(' federated_rounds %d/%d complete \n', round, num_rounds); end% save the final global model (
 for Simul
 ink
  deployment) save ('federated_control_model.mat
 ', 'global_model');
   fprintf('✅ federated model trained! Saved as federated_control_model.mat\n');
```

4. Carsim & Simulink Co-simulation

4.1 Simulation Flow

After the training is completed, based on the main simulation process based on federated learning training method in Wu Wentao [8] 's experiment, the global model is deployed to Simulink/CarSim to realize closed-loop control. The process steps are as follows:

1) Model loading: The model is loaded through the Predict module in Simulink, and the loaded model corresponds to the mathematical mapping.

federated_control_model.mat$Y=W^R_{global}\cdot H_2+b^R_{global}$

2) Input signal processing: The output of CarSim (current vehicle speed), the output of Simulink Clock module (time), and the output of constant module (target vehicle speed) are combined into a 3D input through Mux module and fed into the model.$vtv_{ref}X=[t,v,v_{ref}]^T$

3) Output signal processing: model output, converted to double Type by Data Type Conversion module, in which the brake pressure is multiplied by 100 to restore to the actual range.Y=[throttle,brake]brake[0,100]

4) Closed-loop control: the processed throttle and brake signals are sent to the CarSim vehicle model to realize the closed-loop cycle. Vehicle State → Model Prediction → Control Execution → State Feedback

5) Simulation output: extract the displacement, velocity and acceleration of CarSim output, draw (displacement-time), (velocity-time) and (acceleration-time) curves, and verify the effect of the algorithm.$xvax-tv-ta-t$

4.2 Simulation Results

Carsim was used to build the intersection scene and model the dynamics of two vehicles, and the input and output were defined. Generate the truth value of the speed control of the host vehicle and vehicle A generate_data.m; The federated aggregation algorithm is used to train the network weights, the two-layer neural

network is used, the input and output and the training steps are in learn.m, and the federated_control_model.mat is finally generated and saved

simlink loads the neural network model, performs online simulation, and generates simulation results in Figs. 6, 7, and 8 as follows.

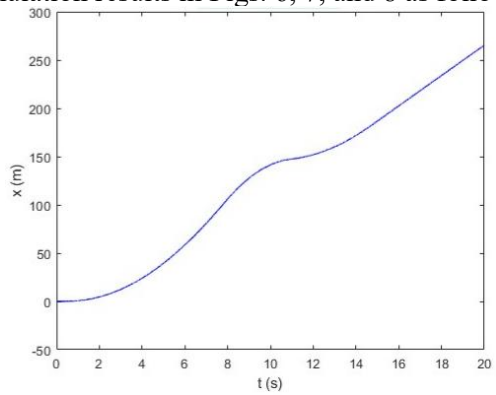

Figure 6. Car Driving Curve (x-t)

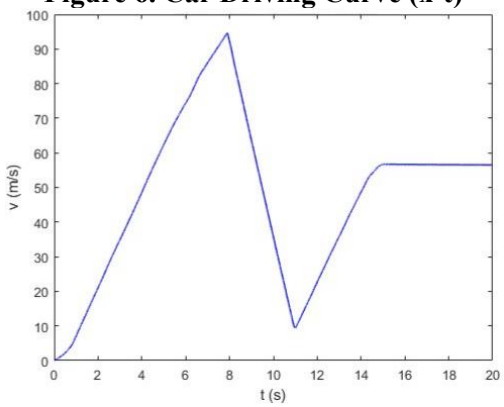

Figure 7. Change Diagram of Car Driving Speed (v-t)

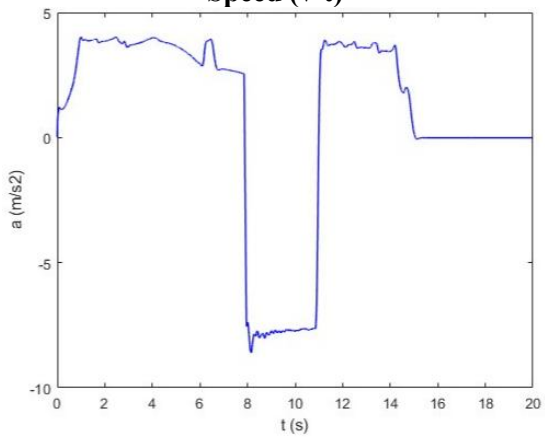

Figure 8. Change Diagram of Car Driving Acceleration (a-t)

The above three figures are the data results obtained under the road condition (1) in 1.3 under the co-simulation, which respectively show the driving curve, speed and acceleration change of the car. The comprehensive analysis shows that the driving curve of the car is smooth, which ensures the ride comfort and passability of the car. The speed change is basically linear, and the acceleration change interval is obvious, which ensures the driving stability and driving comfort of the car.

5. Summary

With the rapid development of intelligent connected vehicles and V2X vehicle-road collaboration technologies, autonomous driving is evolving from single-vehicle intelligence to multi-agent cooperation. Traditional centralized computing in the cloud has problems such as high transmission delay, large computing power load, and the risk of data privacy leakage. Traditional distributed learning is prone to redundant decision making and delayed response in complex intersection scenarios, which is difficult to meet real-time and safety requirements. The fusion architecture of federated learning and edge computing can complete model collaborative training without data leaving the local area, and sink computing tasks to the edge nodes, which effectively reduces communication overhead and improves decision-making efficiency, and provides a feasible technical route for V2X control system optimization.

In this paper, the non-stop traffic at urban intersections is taken as the simulation scene, and the CarSim &Simulink co-simulation method is used to construct a neural network decision model based on the federal average algorithm, so as to realize the mapping control of vehicle state to throttle opening and brake pressure. The local model training, parameter uploading, weighted aggregation and global model iterative update are completed in the experiment, and the system performance is verified by closed-loop simulation. The displacement-time, velocity-time and acceleration-time curves show that the optimized vehicle's driving trajectory is smooth, the speed change is linearly stable, and the acceleration fluctuation interval is reasonable. The key indicators such as braking distance, ride comfort and trajectory tracking accuracy are significantly improved, which effectively solves the problem of decision redundancy and control lag. It effectively solves the problem of decision redundancy and control lag. It improves the efficiency of intersection traffic and driving safety while ensuring data privacy.

Through experimental simulation, this paper

fully verifies the optimization value of the fusion framework of federated learning and edge computing for V2X intersection decision-making. In the future, algorithms such as 6G communication, heterogeneous federated aggregation and multi-objective optimization can be combined to expand the adaptability of complex scenarios such as multi-vehicle mixed traffic and unsignalized intersections, promote the practicability of the integrated vehicle-road-cloud collaborative decision-making system, and provide theoretical and technical support for the large-scale development of intelligent transportation and autonomous driving.

References

[1] Yang Yitong, Wu Jingjing, Zhang Lili. Design and implementation of V2X based Internet of Vehicles experimental system [J]. Modern Electronics Technology,2024,47(24):33-37.

[2] WANG Jianzong, KONG Lingwei, HUANG Zhangcheng, et al. A Survey of Federated Learning Algorithms [J]. Big Data,2020,6(06):64-82.

[3] HU Bingxu. Research on Intelligent Vehicle Obstacle Avoidance Algorithm Based on Improved Federated Learning [D]. University of Electronic Science and Technology of China,2023.[C] QIU Fan. Research on resource allocation of C-V2X Internet of Vehicles based on multi-agent federated learning [D]. South Central University for Nationalities,2023.

[4] SUN Lin. Research on Efficient Federated Learning Algorithm for Autonomous Driving [D]. Xidian University,2024.

[5] QIU Fan. Research on Resource allocation of C-V2X Internet of Vehicles Based on Multi-agent Federated Learning [D]. South-central University for Nationalities,2023.

[6] SHI Weisong, Sun Hui, Cao Jie, et al. Edge computing: a new computing model in the era of Internet of Everything [J]. Journal of Computer Research and Development,2017,54(05):907-924.

[7] Wang Y, Wang Y, et al. A novel approach to multi-agent cooperative decision making based on Federated Reinforcement Learning [J]. Beijing University of Posts and Telecommunications,2024. (in Chinese)

[8] WU Wentao. Research on Autonomous Driving Strategy of Internet of Vehicles Based on Federated Learning [D]. Shanxi University,2023.

