Enhancing Vehicle Environmental Awareness via Federated Learning and Automatic Labeling

Chih-Yu Lin and Jin-Wei Liang

August 26, 2024

Abstract

Vehicle environmental awareness is a crucial issue in improving road safety. Through a variety of sensors and vehicle-to-vehicle communication, vehicles can collect a wealth of data. However, to make these data useful, sensor data must be integrated effectively. This paper focuses on the integration of image data and vehicle-to-vehicle communication data. More specifically, our goal is to identify the locations of vehicles sending messages within images, a challenge termed the vehicle identification problem. In this paper, we employ a supervised learning model to tackle the vehicle identification problem. However, we face two practical issues: first, drivers are typically unwilling to share privacy-sensitive image data, and second, drivers usually do not engage in data labeling. To address these challenges, this paper introduces a comprehensive solution to the vehicle identification problem, which leverages federated learning and automatic labeling techniques in combination with the aforementioned supervised learning model. We have validated the feasibility of our proposed approach through experiments.

1 Introduction

Vehicle environmental awareness refers to a vehicle's ability to identify and understand its surroundings using various sensing technologies. It is a critical aspect of enhancing road safety and serves as the basis for autonomous driving technology. To enable vehicles to have a more precise understanding of their surroundings, a wide range of sensors are installed in vehicles, including radar, LiDAR, GPS, cameras, and more. Additionally, through vehicle-to-vehicle communication, vehicles can exchange information with each other, allowing them to better comprehend the intentions of other vehicles in their vicinity. However, to make these data useful, they must be fused.

Data fusion is the technology that merges data from various sources to create more comprehensive and valuable information. The purpose of data fusion is to gain deeper and more holistic insights than what can be achieved using data from a single source alone. In Budi Christian et al. [2020], an approach to vehicle environmental awareness is proposed, which involves the fusion of image data and vehicle-to-vehicle communication data. To illustrate the problem it addresses, consider Fig. 1. We assume that vehicle x has a camera that captures images in front of vehicle. At the same time, vehicle x can receive messages from other surrounding vehicles. The challenge is how vehicle x can determine which vehicle in the image sends the message when vehicle x receives a message. This problem is referred to as the Vehicle Identification (VID) problem. We have proposed a neural network model called Mapping Decision Feedback Neural Network (MDFNN) to address the VID problemBudi Christian et al. [2020].

However, MDFNN is a supervised model. This means that after collecting vehicle-to-vehicle communication data and image data, a person must manually pair the vehicle-to-vehicle communication data with the image data to generate sufficient training data to train the MDFNN model. It can be expected that drivers are typically unwilling to perform this labeling task. Therefore, a more viable approach is to upload all the data to the cloud and have dedicated individuals perform

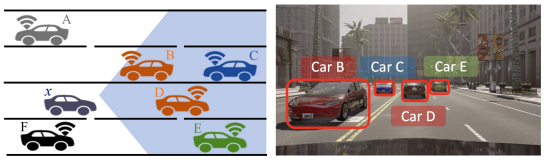


Figure 1: The vehicle identification (VID) problem.

the labeling and model training. However, dealing with a large volume of data not only consumes human resources but also requires a significant amount of network bandwidth for data uploading. Additionally, as data carries a certain degree of privacy, uploading data to the cloud raises privacy concerns.

Federated learning McMahan et al. [2017]Li et al. [2020]Yang et al. [2019] provides a potential solution to the privacy concerns mentioned above. Federated learning consists of numerous local nodes, also known as federated learning clients, each of which trains a local model based on its locally held data. Subsequently, all local nodes transmit their model parameters to a parameter server. The parameter server aggregates the parameters from these local models and constructs a global model. Then, this global model is returned to the local nodes. Since the global model is a combination of many local models, it is expected to perform better than a model trained by an individual local node. In this process, local nodes do not need to transmit sensitive data to the parameter server; they only need to send their local model parameters. As a result, data privacy is reasonably preserved.

We can apply federated learning to the aforementioned VID problem. For example, we can treat each vehicle as a local node, where vehicles only need to transmit model parameters to the parameter server, eliminating the need to send privacy-sensitive image data or vehicle-to-vehicle communication data to the server. However, when the model is a supervised model, the local nodes would typically require some labeled training data to train their local models. The challenge then becomes how to acquire labeled data. In the context of our in-vehicle network environment, expecting drivers to perform labeling is essentially impossible. Therefore, in the federated learning environment, local nodes sometimes need an automatic labeling method for their local data.

This paper presents a solution to the VID problem, which combines federated learning and an automatic labeling approach. We refer to this solution as FedMDFNN (Federated MDFNN). In FedMDFNN, we assume the presence of a federated learning parameter server in the cloud, with each vehicle regarded as a local node in the federated learning system. Each vehicle independently trains its local MDFNN based on its acquired data and uploads the relevant MDFNN parameters to the parameter server. The parameter server aggregates these parameters from all local nodes, resulting in new model parameters, which are then transmitted back to each vehicle. In FedMDFNN, each vehicle is not required to upload its private data, ensuring data privacy.

In addition, to address the challenge of labeling training data, this paper introduces an automatic labeling method using license plate recognition and data augmentation techniques. This automatic labeling approach makes FedMDFNN feasible. Finally, we have validated the proposed FedMDFNN through extensive experiments.

The remainder of this paper is organized as follows. Sect. 2 discusses some related work. Sect. 3 describes our proposed FedMDFNN. Sect. 4 details our simulation setup and performance evaluations. Conclusions are drawn in Sect. 5.

2 Related Work

2.1 Vehicle environmental awareness

Vehicle environmental awareness enables vehicles to understand and adapt to changes in road and environmental conditions. This relies on the real-time collection and processing of a vast amount of sensor data to identify key information such as road markings, obstacles, vehicles, pedestrians, etc.

Radar is a common automotive sensor that is used to detect the distance and speed of objects, such as other vehicles, obstacles, and pedestrians, in front of the vehicle Dickmann et al. [2016], Steinhauser et al. [2021], Hussain et al. [2020]. Radar can operate effectively even in adverse weather conditionsWang et al. [2021]. However, radar information alone cannot determine the intentions and driving states of vehiclesde Ponte Müller et al. [2016].

LiDAR (Light Detection and Ranging) is a sensor that uses laser beams to measure object positions and distances. It can generate precise three-dimensional models of the surrounding environment, including roads, buildings, and other obstaclesLiu et al. [2018]. Like radar, LiDAR alone cannot determine the intentions and driving states of vehicles.

Cameras are another common automotive sensor that captures images of the road, including road signs, traffic signals, lanes, and pedestrians. Through image processing and recognition technologies, the system can analyze these images and identify important elementsTsukamoto et al. [2020], Venator et al. [2020], Liu et al. [2022], Hasan et al. [2019]. Cameras can also be used to measure the distance and speed of vehicles in front, and even track vehiclesAli and Hussein [2016], Tang et al. [2018], Xu et al. [2016].

With the maturity of vehicle-to-vehicle communication technology, communication devices can also be considered as a type of sensor. Through the information transmitted by other vehicles, a vehicle can gain insight into the intentions and states of nearby vehicles, thus enhancing road safetyYang et al. [2016].

The various sensors mentioned above have their limitations when used individually. Therefore, we can achieve better vehicle environmental awareness by using multiple sensorsKocić et al. [2018]. When using two or more sensors simultaneously, we have to address the issue of data fusionHasanujjaman et al. [2023], Abdalwohab et al. [2021]. Wei et al. [2019] fuses radar sensors and communication data, while Budi Christian et al. [2020] and Lee et al. [2019] fuse cameras and vehicle-to-vehicle communication. The VID problem addressed in this paper can be considered as the fusion of image data and communication data.

Lastly, with the rapid advancement of artificial intelligence technology, many studies have used AI to analyze sensor data and formulate optimal driving strategiesStroescu et al. [2020, 2021], Eskandar et al. [2021], McCrae and Zakhor [2020].

2.2 Federated Learning

In traditional supervised machine learning, service providers need to collect user data and label the data to train a model. However, this approach can pose privacy risks. Additionally, transmitting large amounts of data to a server can consume significant bandwidth and time. Federated learning provides a solution to these problemsMcMahan et al. [2017]. In fact, common issues in federated learning such as data heterogeneity and participant selection have also been studiedZhang et al. [2023], Ma et al. [2022], Zhao et al. [2023].

The research on applying federated learning to vehicular networks has also been exploredTan et al. [2020]. An enhanced federated learning approach for Intelligent Transportation Systems (ITS) is presented in Zhao et al. [2023]. In Li et al. [2021], a novel federated learning framework named FedVANET is introduced for vehicle ad hoc networks (VANET). FedVANET is capable of handling Non-IID data and can be applied to dynamic topologies. Federated learning frameworks have been proposed to tackle the challenge of trajectory predictionBruna et al. [2022], Sakho and Othman [2023]. However, the above-mentioned papers do not delve deeply into the data labeling issue for federated clients.

The model proposed in this paper for the VID problem can be applied within a federated learning framework. Each vehicle can be regarded as a federated client, allowing each vehicle to avoid uploading sensitive image data. This not only ensures data privacy, but also saves the network bandwidth required for uploading image data. In particular, in this paper, we focus on discussing the data labeling issue for federated clients. In the assumed scenario, it is nearly impossible for vehicle drivers to perform labeling tasks. Therefore, we propose an automatic labeling approach to address the data labeling issue for federated clients. In the future, we will further consider the impact of data imbalance on the proposed method.

2.3 Automatic Labeling

Through federated learning, we address the issue of data privacy. However, when the model is a supervised model, we encounter the data labeling problem that is not discussed by most papers focusing on federated learning. In this scenario, we may need to complement with an automatic labeling method.

Zhang et al. [2021] provides a review of automatic labeling methods for video, audio, and text data. Lin et al. [2021] proposes a clustering method based on Gauss chaotic mapping particle swarm optimization (GCMPSO) to improve the accuracy of automatically labeling human activities. Hassan et al. [2021] presents an automatic labeling approach for automatically labeling daily activities, such as walking and driving behaviors.Lee et al. [2020] introduces an automatic labeling approach for labeling Synthetic Aperture Radar (SAR) images, using GPS data and open APIs supplied by government agencies. Simon et al. [2020] trains a model to filter automatically labeled data and uses the filtered data to train the final model, resulting in improved model performance.

In this paper, considering the assumed scenario where it is impractical for drivers to manually label data, we propose an automatic labeling method that combines license plate recognition with data augmentation. To the best of our knowledge, this paper is the first to propose an automatic labeling approach for the VID problem. With the introduction of automatic labeling, our proposed method eliminates the need for drivers to perform labeling tasks, making the overall approach more feasible.

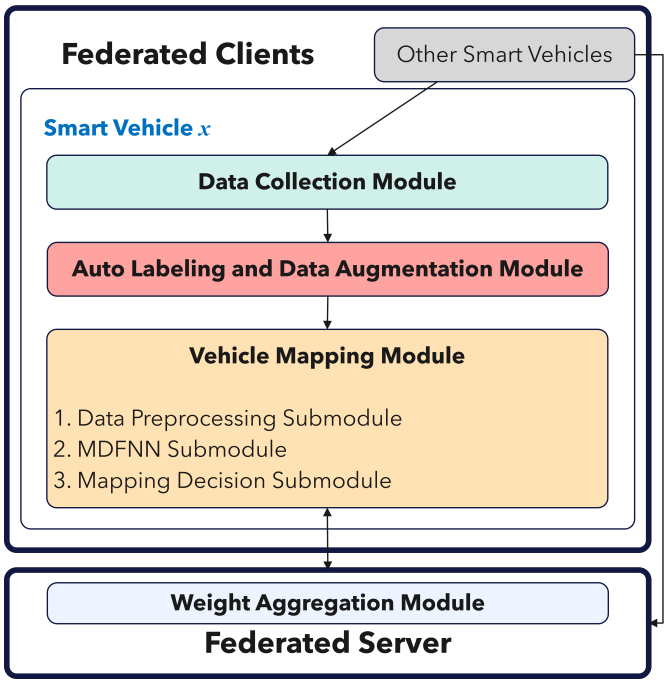


Figure 2: System architecture of FedMDFNN.

3 FedMDFNN

The VID problem addressed in this paper can be defined as follows. We assume that vehicle x is equipped with a front-view camera. Through this camera, vehicle x can use object recognition technology to find the bounding boxes of vehicles captured by the front-view camera. These vehicles are represented as $V^t = \{v_i^t \mid 1 \leq i \leq n\}$, where t represents the time when the image is captured, n represents the number of vehicles recognized by object recognition technology, and v_i^t represents vehicle i appearing in the capture range of the front-view camera of vehicle x at time t. In addition, we assume that vehicle x has received m messages at time t, represented as $B^t = \{b_j^t \mid 1 \leq j \leq m\}$, where b_j^t represents the message sent by vehicle j. The task of the VID problem is to find the pairing of V^t and B^t, represented as $P^t = \{(v_i^t, b_j^t) \mid v_i^t \in V^t \wedge b_j^t \in B^t\}$.

To address the VID problem, we propose a solution called FedMDFNN (Federated MDFNN), which is based on federated learning and automatic labeling. The system architecture of FedMDFNN is illustrated in Fig. 2. Following the architecture of federated learning, FedMDFNN is divided into two components: the federated server and the federated clients. The task of a federated server is to aggregate the model parameters from federated clients to build a global model. Federated clients represent participating vehicles and consist of three modules: (1) data collection module, (2) automatic labeling and data augmentation module, and (3) vehicle mapping module.

Firstly, the data collection module collects image data from the front-view camera and the rear-view camera of vehicle x. Through front-view camera and object recognition technology, vehicle x can acquire V^t. The rear-view camera is used for the data augmentation submodule, as described below. Furthermore, through vehicle-to-vehicle communication, vehicle x obtains B^t. To perform the pairing, vehicle x must also collect data from its own sensors, denoted by S^t. The details of the data collection module are described in Sect. 3.1.

Then, the automatic labeling and data augmentation module is utilized to generate labeled data. This module comprises two submodules: the automatic labeling submodule and the data augmentation submodule. The automatic labeling submodule attempts to establish initial pairings for V^t and B^t through license plate recognition. This initial pairing is denoted as P_{auto}^t, where $P_{auto}^t \subseteq P^t$. On the other hand, the data augmentation submodule aims to identify vehicles outside the horizontal field of view of the front-view camera but whose messages have been received by vehicle x, represented by $V_{Outside}^t = \{v_i^t \mid v_i^t \notin V^t \wedge b(v_i^t) \in B^t\}$, where $b(v_i^t)$ signifies the message sent by vehicle v_i^t. The specifics are elaborated in Sect. 3.2.

Finally, the vehicle mapping module primarily consists of a neural network model. It combines P_{auto}^t and $V_{Outside}^t$ to generate training data for the model. Once the model is trained, S^t and B^t serve as inputs, and the output of the model is the predicted positions of the vehicles in the image. Lastly, through the mapping decision submodule, B^t and V^t are paired. The details are explained

Table 1: Notation

Symbol	Description
V^t	The set of vehicles captured by the front-view camera of vehicle x at time t and recognized by object recognition software, including their bounding boxes.
B^t	The set of messages from other vehicles received by vehicle x at time t.
P^t	The vehicle pairing set.
S^t	The relevant sensor values of vehicle x at time t.
P^t_{auto}	The set of automatic pairings using license plate recognition technology at time t.
$V^t_{Outside}$	The set of vehicles at time t that did not appear in the front-view camera of vehicle x but whose messages were received by vehicle x.
V^t_{Rear}	The set of vehicles captured by the rear-view camera of vehicle x at time t and identified using license plate recognition.

in Sect. 3.3.

Table 1 summarizes the notation used in the paper.

3.1 Data Collection Module

The data collection module collects image data from the front-view camera and the rear-view camera of vehicle x. Through the front-view camera and object recognition technology, we can obtain the collection of vehicle bounding boxes V^t. The bounding box coordinates for each vehicle v_i^t in V^t are normalized and can be represented as $v_i^t.bb\_norm = [\frac{v_i^t.x_{TopLeft}}{w}, \frac{v_i^t.y_{TopLeft}}{h}, \frac{v_i^t.x_{BottomRight}}{w}, \frac{v_i^t.y_{BottomRight}}{h}]$, where w and h represent the width and height of the image, respectively. The image data from the rear-view camera is utilized for the data augmentation submodule, and the details are explained in Sect. 3.2.2.

On the other hand, through vehicle-to-vehicle communication, vehicle x can receive communication data b_y^t, from other vehicles, with vehicle y as an illustrative example. These data include the following:

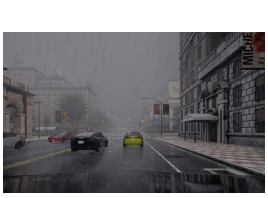
- GPS data of vehicle y at time t, represented by $v_y^t.lat$ and $v_y^t.lng$.
- Orientation sensor data of vehicle y at time t, represented by $v_y^t.ori$.
- Speed of vehicle y at time t, represented by $v_y^t.spd$.
- ID of vehicle y, represented by $v_y^t.id$. For privacy concerns, $v_y^t.id$ is obtained by hashing the license plate of vehicle y.
- State of vehicle y, which enables vehicle x to comprehend the driving conditions of surrounding vehicles.

All messages received from time $t-1$ to time t constitute the set B^t.

To facilitate pairing, vehicle x must also collect data from its own sensors, represented as $S^t = (v_x^t.lat, v_x^t.lng, v_x^t.ori,$
$v_x^t.spd)$. The VID problem can be viewed as matching V^t and B^t using S^t.

3.2 Auto Labeling and Data Augmentation Module

The automatic labeling and data augmentation module aims to automatically generate labeled training data. The automatic labeling submodule uses license plate recognition technology to establish pairings for V^t and B^t, creating a pairing set P^t_{auto}, where $P^t_{auto} \subseteq P^t$. Simultaneously, the data augmentation submodule identifies vehicles outside the horizontal field of view of the front-view camera of vehicle x (i.e., not in V^t), but whose messages have been received by vehicle x, denoted as $V^t_{Outside} = \{v_i^t \mid v_i^t \notin V^t \wedge b(v_i^t) \in B^t\}$. The purpose of both submodules is to generate training data, and the details are described separately.



(a) OCR failed due to bad weather.

(b) OCR failed due to license plate being obscured.

Figure 3: Examples of OCR failure.

Figure 4: Built-in license plates in the Carla simulatorDosovitskiy et al. [2017].

3.2.1 Auto Labeling Submodule

The auto labeling submodule uses license plate recognition technology to match V^t and B^t, aiming to establish a pairing set P^t_{auto}. The approach involves attempting to identify the license plate bounding box for $v^t_i \in V^t$ and using Optical Character Recognition (OCR) technology to extract the license plate number. Then, the extracted license plate number is hashed by a common hash function and compared with the vehicle ID carried by the transmitted packet, facilitating the matching of V^t and B^t.

It should be noted that due to various factors, such as different weather conditions, distances, and the state of the preceding vehicles, not all vehicles in V^t can be successfully matched using license plate recognition. For example, Fig. 3(a) illustrates a case where license plates cannot be successfully identified due to bad weather, and Fig. 3(b) shows a case where license plates cannot be recognized due to being obstructed by other vehicles. Therefore, the use of license plate recognition technology can only achieve partial matching, and the remaining vehicles can be matched using the FedMDFNN proposed in this paper.

We utilize the OCR function provided by the Google Cloud Vision APIGoogle to implement license plate recognition. Additionally, we employ a confusing character table, established through experiments, to refine the OCR recognition results. The reason for creating this table is that, during the license plate recognition process, we observed that certain confusing characters can affect the accuracy of recognition. For example, OCR software often confuses the number 0 with the English letter O or misinterprets the number 1 as the English letter I. Relying solely on an exact matching approach for character comparison would lead to the inability to automatically label a significant amount of data.

Below is an example that illustrates the process of establishing a confusing character table through experiments. We extracted 117 built-in license plates from the Carla simulatorDosovitskiy et al. [2017], as depicted in Fig. 4. To simulate real-world conditions more accurately, we adjusted the pixel dimensions of the license plates to 86×41 pixels, closely resembling the dimensions of license plates captured by dashcams. Based on the recognition results for these 117 license plates, we obtained the confusing character table, presented in Table 2. This table displays the number of occurrences and the percentage of OCR results.

Through Table 2, we can identify characters with error rates that exceed the threshold T. Taking the letter 'I' as an example, it appears 16 times on the 117 license plates. Among these occurrences, it is misrecognized as the numeric character '1' four times. Assuming T is set to 0.2, as $\frac{4}{16} > 0.2$,

Table 2: The confusing character table example.

Character	**OCR Result: the number of occurrences (percentage)**
0	0: 20 (61%), **O: 13 (39%)**
1	1: 42 (84%), T: 1 (2%), E: 1 (2%), I: 6 (12%)
2	2: 40 (91%), Z: 4 (9%)
3	3: 39 (95%), L: 1 (2%), 6: 1 (2%)
4	4: 50 (100%)
5	5: 39 (95%), S: 2 (5%)
6	6: 30 (88%), W: 1 (3%), 3: 1 (3%), G: 2 (6%)
7	7: 62(95%), A: 1(2%), T: 2(3%)
8	8: 66(96%), B: 3(4%)
9	9: 40(100%)
A	A: 19(100%)
B	B: 12(86%), L: 1(7%), 9: 1(7%)
C	C: 14(100%)
D	D: 5(71%), **0: 2(29%)**
E	E: 21(95%), C: 1(5%)
F	F: 8(100%)
G	G: 7(100%)
H	H: 16(100%)
I	I: 12(75%), **1: 4(25%)**
J	J: 12(100%)
K	K: 17(100%)
L	L: 20(100%)
M	M: 18(85%), 9: 1(5%), P: 1(5%), H: 1(5%)
N	N: 31(100%)
O	O: 8(42%), **0: 11(58%)**
P	P: 8(89%), M: 1(11%)
Q	Q: 1(33%), **0: 2(67%)**
R	R: 22(100%)
S	S: 2(22%), **5: 7(78%)**
T	T: 13(93%), 7: 1(7%)
U	U: 17(100%)
V	V: 10(91%), W: 1(9%)
W	W: 4(80%), M: 1(20%)
X	X: 7(100%)
Y	Y: 9(100%)
Z	Z: 3(100%)

Table 3: Character Conversion Table

Key	Value
0	#1
O	#1
D	#1
Q	#1
1	#2
I	#2
5	#3
S	#3

we consider the letter 'I' and the digit '1' as a pair. This process generates a character pairing set $CP = \{(c_1, c_2) \mid err(c_1, c_2) > T \vee err(c_2, c_1) > T\}$, where $err(c_1, c_2)$ represents the probability of character c_1 being recognized as c_2. With the character pairing set CP, we can create a key-value table named character conversion table. The algorithm for building the character conversion table is shown in Algo. 1.

Algorithm 1 Character Conversion Table Generator

$CCT = \emptyset$
$value = 1$
for each $(c_1, c_2) \in CP$ **do**
  if $CCT.hasKey(c_1)$ **then**
    $CCT = CCT \cup (c_2, CCT.getValue(c_1))$
  else if $CCT.hasKey(c_2)$ **then**
    $CCT = CCT \cup (c_1, CCT.getValue(c_2))$
  else
    $CCT = CCT \cup (c_1, value)$
    $CCT = CCT \cup (c_2, value)$
    $value = value + 1$
  end if
end for

For a pair (c_1, c_2) in CP, if either character is already in the character conversion table, the other character is added to the table using the same value. If neither character is present in the character conversion table, a new value is assigned. Based on the confusing character table (Table 2), the character pairing set CP, and the character conversion table generator algorithm, we obtain Table 3.

Through the character conversion table, when a character from a license plate appears in the list of keys, we replace that character with its corresponding value. For example, suppose that we receive a message with the license plate number '5CRD321.' According to Table 3, this license plate number is transformed into '#3CR#132#2.' Consequently, even if the OCR recognition software mistakenly interprets it as 'SCRO32I,' with the transformed license plate '#3CR#132#2,' we can still successfully match it. To distinguish between converted and unconverted numeric characters, we prepend a '#' symbol to the converted values. In this way, for example, even after conversion, we can correctly distinguish between '1ABCEF' and '2ABCEF'. Finally, as mentioned earlier, the vehicle's ID is generated by the license plate number and a hash function. Before performing the hash calculation, it is necessary to convert the license plate number based on Table 3 so that the receiving vehicles can make comparisons.

According to the license plate recognition method described above, we can match V^t and B^t, and establish an initial pairing set P^t_{auto}. As mentioned above, not all vehicles in V^t can successfully undergo pairing using the license plate recognition method. Therefore, an issue for discussion is what proportion of vehicles in V^t can be successfully matched using this method. If this proportion is high, it implies that a simple license plate recognition method is sufficient for pairing, and there may not be a need for a complex vehicle mapping model. On the contrary, if this proportion is low, the training dataset that we can generate might be insufficient. Based on our simulation experiments, we find that the license plate recognition method can successfully match approximately 24% of vehicles in a single photo.

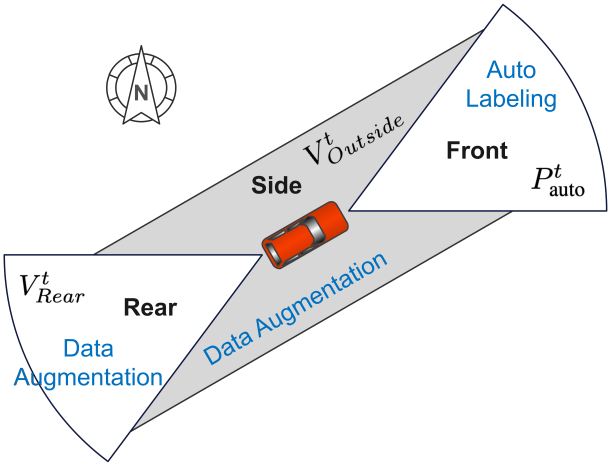


Figure 5: The front, rear, and sides of the vehicle. P^t_{auto} is acquired through the automatic labeling submodule, while V^t_{Rear} and $V^t_{Outside}$ are obtained via the data augmentation submodule.

3.2.2 Data Augmentation Submodule

The purpose of the data augmentation module is to identify vehicles, denoted as $V^t_{Outside} = \{v^t_i \mid v^t_i \notin V^t \wedge b(v^t_i) \in B^t\}$, that are not in V^t but have received their messages.

We employ two methods for data augmentation. The first method involves using a rear-view camera. Assuming that the fields of view of the front-view and rear-view cameras do not overlap, we can apply the license plate recognition method mentioned earlier to recognize images captured by the rear-view camera. This results in $P^t_{Rear} = \{(v^t_i, b^t_j) \mid v^t_i \in V^t_{Rear} \wedge b^t_j \in B^t\}$, where V^t_{Rear} represents vehicles that are captured by the rear-view camera (i.e., the vechicles located in the rear area in Fig. 5) and can be recognized using the license plate recognition method.

The second method of data augmentation considers vehicles on the sides, as illustrated in the gray areas in Fig. 5. For vehicle x, we can determine whether vehicle v is within the horizontal field of view of x's front-view camera using the following formula:

$$x.ori - \frac{\alpha}{2} \leq x.loc.bearing(v.loc) \leq x.ori + \frac{\alpha}{2}, \tag{1}$$

where α denotes the horizontal angle of view of the camera, $x.ori$ denotes the orientation of vehicle x, and loc denotes the location obtained from GPS. If the formula holds, it indicates that vehicle v is within the horizontal field of view of vehicle x's camera. The set of vehicles that we aim to collect consists of those outside the horizontal field of view of the x's front-view camera, but their messages are received by x, denoted as $V^t_{Outside} = \{v^t_i \mid v^t_i \notin V^t \wedge b(v^t_i) \in B^t\}$. It should be noted that, considering the inherent errors in GPS, we refer to the past k seconds of data for vehicle v. If during these k seconds, vehicle v is consistently outside the horizontal field of view of vehicle x's front-view camera, only then do we conclude that vehicle v is indeed not within the horizontal field of view of the front-view camera of vehicle x.

In the end, we include the vehicles in V^t_{Rear} for which vehicle x has received their messages into $V^t_{Outside}$ to obtain the final $V^t_{Outside}$. It is important to note that the vehicles in V^t_{Rear} cannot be part of the training data if vehicle x has not received their messages. Additionally, there is a high probability that V^t_{Rear} is a subset of $V^t_{Outside}$ (i.e., $V^t_{Rear} \subseteq V^t_{Outside}$). Therefore, if vehicles lack a rear-view camera and rely solely on GPS coordinates for orientation, we can still obtain $V^t_{Outside}$. The vehicles in $V^t_{Outside}$ constitute part of the training data.

3.3 Vehicle Mapping Module

As mentioned above, some vehicles cannot be successfully matched through license plate recognition, and the vehicle mapping module is designed for these cases. In our previous work, we have proposed a model called MDFNN for vehicle mappingBudi Christian et al. [2020]. In this paper, we have made minor modifications to MDFNN. More importantly, compared to our previous work, there are two major differences:

- In previous work, training data labeling was done manually. In this paper, training data are generated through the auto labeling and data augmentation module, enhancing the feasibility of the proposed solution.

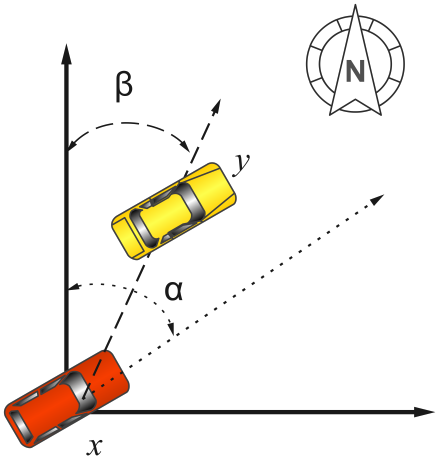


Figure 6: Illustration of orientation conversion method.

- In this paper, we have incorporated federated learning, where vehicles do not need to upload images and communication data with privacy concerns. This ensures the privacy of the data and reduces the substantial bandwidth required for uploading image data.

For a neural network model, input data typically undergo preprocessing, including normalization. Sect. 3.3.1 details the data preprocessing method. The design of MDFNN is described in Sect. 3.3.2, and the mapping decision submodule, responsible for final pairing, is explained in Sect. 3.3.3.

3.3.1 Data Preprocessing Submodule

Sensors that can be used to determine the position of vehicles in the image include GPS sensors, speedometers, and orientation sensors. However, preprocessing of the sensor data is necessary to facilitate model training. The data preprocessing submodule handles the latitude and longitude information from GPS, along with the values from the orientation sensor and the speedometer, as follows.

First, the latitude and longitude values are converted to the decimal degrees format. In the longitude part, east longitude is represented as positive, while west longitude is represented as negative. In the latitude part, north latitude is represented as positive, and south latitude is represented as negative. After conversion, the difference between the latitude and longitude positions obtained from the received message and the vehicle's own latitude and longitude positions is calculated. This difference is then transformed into a value between -1 and 1. For example, if the GPS signal extracted from the message after conversion is (+23.973875, +120.982025), and the vehicle's GPS signal after conversion is (+23.973828, +120.982038), then the difference is (0.000047, -0.000013). Finally, we normalize the latitude and longitude differences separately to ensure that they fall within the range of -1 and 1.

The orientation sensor value also undergoes a conversion process, as depicted in Fig. 6. Assume that the angle obtained by vehicle x through its own orientation sensor is α, and β is calculated using the GPS information of x and y. Through the following formula, we can calculate γ:

$$\gamma = \begin{cases} (\alpha - \beta)/180, & 180 \geq \alpha - \beta \geq -180 \\ (\alpha - \beta + 360)/180, & \alpha - \beta < -180 \\ (\alpha - \beta - 360)/180, & \alpha - \beta > 180 \end{cases} \tag{2}$$

The value of γ, if positive, indicates that vehicle y is on the left side of vehicle x; conversely, if the value of γ is negative, it indicates that vehicle y is on the right side of vehicle x. Furthermore, the value of γ ranges from -1 to 1.

Finally, the vehicle speed collected by the speedometer is normalized to fall between 0 and 1.

3.3.2 MDFNN Submodule

In our previous work, we proposed two types of MDFNN: the Grid-based MDFNN (Grid-MDFNN) and the Bounding box-based MDFNN (BBX-MDFNN)Budi Christian et al. [2020]. We have made slight modifications to the design of the MDFNN, and here, we only provide a brief overview of the BBX-MDFNN.

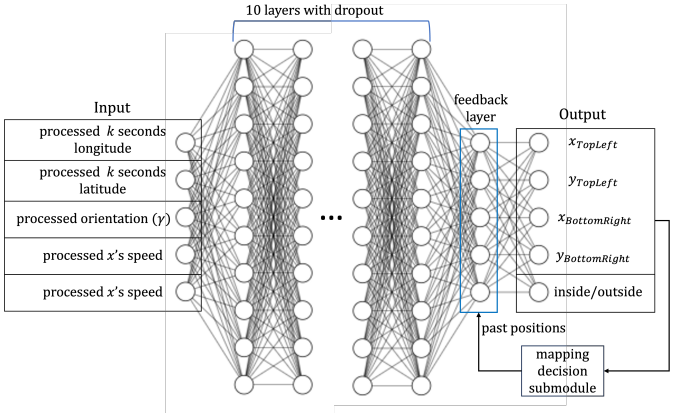


Figure 7: The neural network architecture of BBX-MDFNN.

The neural network architecture of BBX-MDFNN is depicted in Fig.7. The inputs include processed latitude and longitude values from the past k seconds, speeds of vehicle y (message sender) and vehicle x (message receiver), and the processed orientation value γ. The output is the estimated bounding box of vehicle y in the image, represented by a vector of length 5 denoted as O_{bbx}. The first four elements (O^1_{bbx} to O^4_{bbx}) of the vector represent the predicted bounding box position, while the last element (O^5_{bbx}) indicates whether vehicle y is present in the image. The model consists of 10 hidden layers, and to prevent overfitting during training, we apply a 30% dropout. Preceding the output layer, there is a feedback layer where the bounding boxes of vehicle y, calculated by the mapping decision module (explained in Section3.3.3), are input to collaboratively determine the final output.

The loss function used by BBX-MDFNN is as follows:

$$Loss_{bbx} = \frac{1}{4}\sum_{i=1}^{4}(y_i - \hat{y}_i)^2 + \mu(y_{inside} - \hat{y}_{inside})^2, \tag{3}$$

where y_i represents the ground truth of the bounding box, $\hat{y}_i$ represents the predicted bounding box by BBX-MDFNN, y_{inside} represents the ground truth of whether the vehicle y is inside the image, $\hat{y}_{inside}$ represents the output value of BBX-MDFNN for whether the vehicle is inside the image, and μ is an adjustable weight.

Finally, we explain how the automatically generated training data are labeled in the BBX-MDFNN. For vehicles v in P^t_{auto}, we use the bounding box ($v.bb_norm$) marked by the image recognition software for vehicle v as labeled data, with O^5_{bbx} set to 1. For vehicles in $V^t_{outside}$, all five elements in O_{bbx} are set to 0.

3.3.3 Mapping Decision Submodule

We assume that for the vehicles in B^t, the set of bounding boxes estimated by BBX-MDFNN is denoted as E^t. Therefore, the original VID problem that pairs between B^t and V^t is transformed into the problem that pairs between E^t and V^t. E^t and V^t may not match perfectly, and the number of vehicles in E^t and V^t may also be inconsistent. Therefore, we need a mapping decision submodule to determine the final pairing.

Algo. 2 outlines the mapping decision algorithm, which can be explained in four parts.

- Filtering out vehicles: The goal of the 1-6 lines of the algorithm is to filter out the vehicles not in the image. In the BBX-MDFNN, each element e in E^t is a vector of length 5, where the last element, $e.inside$, represents the probability of the vehicle being inside the image. Only vehicles with $e.inside$ greater than a threshold are considered for pairing, and E^t_{inside} represents the set of vehicles considered for pairing.

- Creating the score table: To compare the similarity between E^t_{inside} and V^t, we define a score function, where a higher score indicates greater similarity. In BBX-MDFNN, $e_j \in E^t$ represents the estimated bounding box with its first four elements, and $v_i \in V^t$ represents the bounding box itself. The score function is determined by calculating the Intersection over

Algorithm 2 Mapping Decision Algorithm

Require: E^t and V^t
Ensure: P^t
$E^t_{inside} \leftarrow \emptyset$ ▷ Filtering out vehicles.
for each $e \in E^t$ **do**
  if $e.inside > Threshold_{inside}$ **then**
    $E^t_{inside} \leftarrow E^t_{inside} \cup e$
  end if
end for
for each $e \in E^t_{inside}$ **do** ▷ Score table.
  for each $v \in V^t$ **do**
    $ST[e][v] \leftarrow Score(e, v)$
  end for
end for
for each $(e, v) \in ST$ **do** ▷ Confidence table.
  $C[e][v] \leftarrow Confidence(e, v)$
end for
$P^t \leftarrow \emptyset$ ▷ Mapping
while True **do**
  $(max_e, max_v) \leftarrow FindMaximumPair(C)$
  if $Score(max_e, max_v) \neq 0$ **then**
    $P^t \leftarrow P^t \cup (max_e, max_v)$
    Set all elements in the row max_e to 0.
    Set all elements in the column max_v to 0.
  else
    break
  end if
end while

Table 4: An example of the score table.

	v_1	v_2	v_3
e_1	0.3	0.7	0.1
e_2	0.1	0.83	0.8
e_3	0.62	0.35	0.4

Union (IoU) and the center point distance, as expressed by the following formula:

$$Score_{bbx}(e_j, v_i) = (1-\omega)(\frac{Intersection(e_j, v_i)}{Union(e_j, v_i)}) + \omega(\frac{DiagonalLen - dist(e_j, v_i)}{DiagonalLen}), \tag{4}$$

where, ω is an adjustable weight, $Intersection(e_j, v_i)$ represents the intersection area of the bounding boxes e_j and v_i, $Union(e_j, v_i)$ represents the union area of the bounding boxes e_j and v_i, $DiagonalLen$ denotes the diagonal length of the image, and $dist(e_j, v_i)$ represents the center point distance between the bounding boxes e_j and v_i. Based on the score function, we can establish a score table, as shown in Table 4, represented by ST.

- Converting the score table to the confidence table: With the score table, we can now perform the mapping based on the scores. In the example from Table 4, we would get $\{(e_2, v_2), (e_3, v_1), (e_1, v_3)\}$. However, this combination may not be the best, as the score for (e_1, v_3) is only 0.1. In fact, if e_j assigns high scores to multiple bounding boxes simultaneously, it indicates that e_j may not be confident about which bounding box is correct. Conversely, if e_j assigns a high score to only one bounding box, it suggests that e_j is confident about this choice. Therefore, $\{(e_1, v_2), (e_2, v_3), (e_3, v_1)\}$ could be a better combination. Consequently, we define a confidence function to establish a confidence table. The formula for the confidence function is as follows:

$$Confidence(e_j, v_i) = \frac{Score(e_j, v_i)}{\sum_{k=1}^{|V^t|} Score(e_j, v_k)}. \tag{5}$$

Table 5: An example of the confidence table.

	v_1	v_2	v_3
e_1	0.27	0.64	0.09
e_2	0.06	0.48	0.46
e_3	0.45	0.26	0.29

According to this confidence function, we can convert the score table in Table 4 into the confidence table as shown in Table 5.

- Mapping: With the confidence table, we can now perform the mapping based on confidence values. Note that rows and columns that have been mapped will not participate in the subsequent mapping. For detailed specifics, please refer to lines 15-26 in Algorithm 2.

Finally, it should be noted that for the paired vehicle e_j, the corresponding value of v_i is input into the feedback layer of MDFNN at the next time step. Conversely, for vehicles that are not paired, the values are set to 0 and then input into the feedback layer of MDFNN.

3.4 Federated Learning based MDFNN (FedMDFNN)

In our federated learning framework, each vehicle is treated as a federated client. After training its local MDFNN model, each vehicle uploads the MDFNN model parameters to the federated server. Upon receiving the model parameters from the participating vehicles, the federated server aggregates the model parameters and returns the aggregated parameters to the participating vehicles. This process enables the participating vehicles to update their local parameters. Subsequently, the participating vehicles continue training with the updated local parameters, and this process repeats. After a certain number of rounds, a satisfactory global model is generated. This trained global model is then returned to each participating vehicle and used for identifying the positions of vehicles in the images.

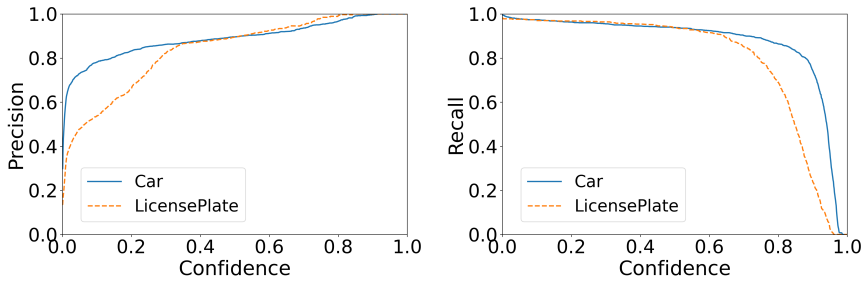
In the above process, participating vehicles do not need to transmit sensor values or image data to the server. They only need to send the parameters of their local MDFNN models. This not only ensures that sensitive data does not need to be uploaded, but also saves the network bandwidth required for uploading image data.

4 Experiments

4.1 Simulation Environment and Dataset Generation

We conducted experiments and validated our proposed method using the Carla simulatorDosovitskiy et al. [2017]. We utilized five maps provided by Carla for experimentation, namely Small Town, Middle Town, Highway, Square Town with Multiple Lanes, and Big City. Experiments were conducted on each map under 14 different weather conditions. A total of 100 vehicles were deployed on the roads, and these vehicles varied their speeds over time while adhering to traffic rules on the simulated maps (e.g., traffic lights). One of the vehicles, denoted as x, was equipped with front and rear cameras. All vehicles recorded their orientation, speed, and GPS coordinates. We assumed a sensor data acquisition rate of every 0.5 seconds. In the end, we obtained a total of 3,500 front camera image data, 3,500 rear camera image data, and 350,000 sensor data entries. When vehicle x received a message, it could calculate the distance to the sending vehicle, since the message contained the GPS information of the sending vehicle. Vehicle x processed only messages from vehicles within a distance of 50 meters.

During the experimentation, we utilized YOLOv5Redmon et al. [2016] to detect the bounding boxes of vehicles and license plates. The vehicle recognition model provided by YOLOv5 is trained for real-world vehicles. However, the vehicles in Carla differ slightly from their real-world counterparts. In addition, YOLOv5 does not offer a model for license plate recognition. Consequently, we retrained models for both vehicle and license plate recognition. The performance of our trained models is shown in Fig. 8. Fig. 8(a) illustrates the relationship between precision and confidence, while Fig. 8(b) shows the relationship between recall and confidence. We observed that as the confidence threshold increased, the recognition accuracy improved, but the number of successfully identified objects decreased. In other words, although precision could be enhanced, recall decreased significantly. Therefore, for subsequent experiments, we set the confidence threshold to 0.7 to strike a balance between precision and recall.



(a) Precision (b) Recall

Figure 8: The performance of the retrained model.

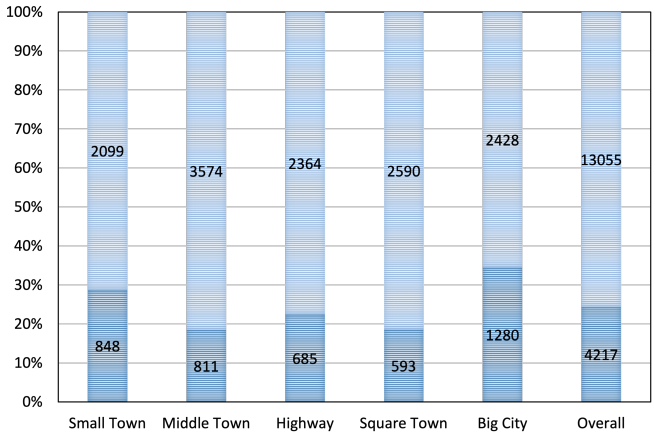


Figure 9: The performance of the automatic labeling submodel.

4.2 Performance of the Automatic Labeling Submodule

Firstly, we are interested in the proportion of vehicles in a single image that can be labeled using the automatic labeling method proposed in this paper. If this proportion is too low, it implies that we may not be able to gather sufficient data for local training. Conversely, if this proportion is too high, it suggests that using the automatic labeling method alone is sufficient for pairing vehicles, and there might not be a need for FedMDFNN.

Fig. 9 shows the results. In general, approximately 24% vehicles can be paired using the automatic labeling method, while the remaining 76% vehicles can be paired using FedMDFNN. Sect. 4.3 shows that the data generated by the automatic labeling method are sufficient to train a good MDFNN model. Additionally, this experiment also shows that FedMDFNN is required to pair more vehicles.

We also conducted an experiment for the character conversion method proposed in Sect. 3.2. Without character conversion, only about 16% vehicles can be successfully paired. This indicates that the proposed character conversion indeed increases the proportion of paired vehicles.

The data of these successfully paired vehicles can serve as the training dataset required for training the MDFNN model. In the next section, these automatically labeled data are used to train MDFNN and validate the model performance.

4.3 Performance of the Automatic Labeling and Data Augmentation Module

In this experiment, three datasets are used separately to train the MDFNN model:

- $\mathcal{D}_{AL}$: This dataset is generated using the automatic labeling with the front camera and data augmentation with the rear camera, meaning that it includes only the data from the front and rear cameras.
- $\mathcal{D}_{ALDA}$: This dataset is generated using the complete automatic labeling and data augmentation module.
- $\mathcal{D}_{Manual}$: The labeling data for this dataset are completed manually.

Table 6: Number of entries in dataset.

Dataset	Inside	Outside	Total
$\mathcal{D}_{AL}$	1734	1890	3624
$\mathcal{D}_{ALDA}$	1734	16171	17905
$\mathcal{D}_{Manual}$	7249	16956	24205

Table 7: The performance of Grid-MDFNN under different datasets.

Dataset	CR_{ic}	CR_{inside}	$CR_{outside}$	CR_{total}
$\mathcal{D}_{AL}$	50.35%	65.23%	92.46%	76.76%
$\mathcal{D}_{ALDA}$	53.42%	67.36%	94.14%	78.96%
$\mathcal{D}_{Manual}$	55.84%	68.18%	94.21%	79.91%

The three datasets only label data for Small Town, Middle Town, Highway, and Big City. $\mathcal{D}_{Manual}$ undoubtedly has the most labeled data, while the number of entries in the $\mathcal{D}_{AL}$ dataset is the lowest. The numbers of entries in the datasets are shown in Table 6. The models trained on these three datasets are validated using the Square Town map.

To compare the performance of these three datasets, we define CR_{ic}, CR_{inside}, $CR_{outside}$, and CR_{total} as follows:

$$CR_{ic} = \frac{P_{correctly}}{N_{inside}} \tag{6}$$

$$CR_{inside} = \frac{P_{inside}}{N_{inside}} \tag{7}$$

$$CR_{outside} = \frac{P_{outside}}{N_{outside}} \tag{8}$$

$$CR_{total} = \frac{P_{correctly} + P_{outside}}{N_{inside} + N_{outside}}, \tag{9}$$

where $P_{correctly}$ represents the number of vehicles correctly paired in the image by MDFNN; N_{inside} represents the number of vehicles labeled as inside the image; P_{inside} represents the number of vehicles that are actually inside the image, and identified as inside by MDFNN; $P_{outside}$ represents the number of vehicles actually outside the image, identified as outside the image by MDFNN; $N_{outside}$ represents the number of vehicles labeled as outside the image.

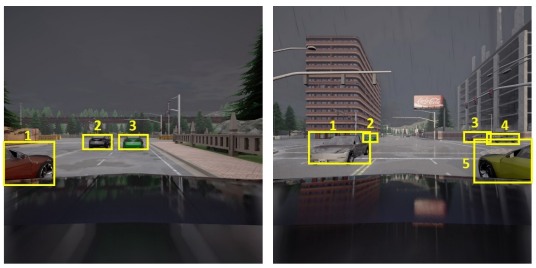
The performance of Grid-MDFNN and BBX-MDFNN under different datasets is shown in Table 8 and Table 7. As expected, the dataset generated using manually labeled data performs the best, as it has the most data. However, models trained on datasets generated using automatic labeling with data augmentation still achieve good results. Compared to using manually labeled datasets, in BBX-MDFNN, CR_{total} decreases from 85.63% to 83.56%, a slight decrease of 2.07%.

We then analyze the reason for the lower CR_{inside} compared to $CR_{outside}$. The vehicle recognition model that we used cannot identify all vehicles. According to Fig. 8(b), when we set the confidence value to 0.7, the recall value is 89.87%, indicating that about 10% of vehicles cannot be successfully identified by the vehicle recognition model (i.e., false negative cases). In cases where the number of bounding boxes is not sufficient, some vehicles cannot be paired, and thus, they are not counted in P_{inside}.

We conducted further analysis on cases of pairing errors and found that weather is not the most crucial factor affecting pairing accuracy. The number of bounding boxes has a more significant impact. As the number of bounding boxes increases, it indicates that the bounding boxes are closer or even overlap, making pairing more challenging. Fig.10 illustrates an example. In Fig.10(a), there are only three vehicles, and all three are correctly paired. However, in Fig. 10(b), there are five vehicles. Due to the close proximity and overlapping bounding boxes of vehicles 1 and 2, the retrained vehicle recognition model only identifies the bounding box of vehicle 1. As a result,

Table 8: The performance of BBX-MDFNN under different datasets.

Dataset	CR_{ic}	CR_{inside}	$CR_{outside}$	CR_{total}
$\mathcal{D}_{AL}$	61.33%	77.74%	92.11%	80.63%
$\mathcal{D}_{ALDA}$	65.29%	79.63%	94.42%	83.56%
$\mathcal{D}_{Manual}$	72.08%	79.87%	93.68%	85.63%



(a) The case of easy to pair.

(b) The case of hard to pair.

Figure 10: Analysis of pairing errors cases.

Table 9: The performance of FedMDFNN

Model	CR_{ic}	CR_{inside}	$CR_{outside}$	CR_{total}
Grid-FedMDFNN	48.11%	64.46%	92.18%	75.75%
Grid-MDFNN	53.42%	67.36%	94.14%	78.96%
BBX-FedMDFNN	57.32%	75.27%	91.30%	78.63%
BBX-MDFNN	65.29%	79.63%	94.42%	83.56%

only vehicle 1 is correctly paired, while vehicles 3 and 4 are erroneously paired with each other's bounding boxes. Vehicle 5 is correctly paired.

4.4 Performance Analysis of the FedMDFNN

In this section, we compare the performance of the federated FedMDNN with the centralized MDFNN. In the experiment, we used FlowerBeutel et al. [2020] to implement the federated environment. We distributed the $\mathcal{D}_{ALDA}$ dataset to 2 client nodes. Each client locally trained its model and uploaded the model parameters to the federated server. The server then performed parameter aggregation. The final globally aggregated model was validated for performance using the Square Town map. Table 9 presents the results.

From the experimental results, we observe that both BBX-FedMDFNN and Grid-FedMDFNN exhibit performance gaps compared to centralized BBX-MDFNN or Grid-MDFNN. In the case of Grid-FedMDFNN, CR_{total} decreases from 78.96% to 75.75%, while in the case of BBX-FedMDFNN, CR_{total} decreases from 83.56% to 78.63%, representing decreases of 3.21% and 4.93%, respectively. One reason for this is that in FedMDFNN, the amount of data used by each client to train the local model is only half of the data used to train centralized MDFNN. However, FedMDFNN ensures data privacy, coupled with the automatic labeling and data augmentation proposed in this paper (which is the reason for using the $\mathcal{D}_{ALDA}$ dataset in this experiment). Compared to centralized MDFNN, FedMDFNN demonstrates greater feasibility.

5 Conclusions

This paper proposes FedMDFNN, which combines federated learning and automatic labeling to address the vehicle identification problem. Through the federated learning framework, drivers only need to upload local model parameters to aggregate and obtain the global model. This approach ensures user privacy as only parameters are uploaded, reducing network traffic consumption. Furthermore, we integrate automatic labeling and data augmentation to acquire the dataset needed for the FedMDFNN model, eliminating the need for manual labeling. This greatly enhances the overall feasibility of the approach, because typically drivers are reluctant to label data. In the future, we aim to explore the integration of object tracking technology to obtain more training data and enhance the model's performance. Additionally, as federated learning is extensively studied, we will also consider the common Non-IID issues in federated learning and their impact on the FedMDFNN.

References

Albert Budi Christian, Chih-Yu Lin, Cheng-Wei Lee, Lan-Da Van, and Yu-Chee Tseng. A Neural Network-based Multisensor Data Fusion Approach for Enabling Situational Awareness of Vehicles. In *2020 International Conference on Pervasive Artificial Intelligence (ICPAI)*, pages 199–205, 2020. doi:10.1109/ICPAI51961.2020.00044.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020. doi:10.1109/MSP.2020.2975749.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), Jan. 2019. ISSN 2157-6904. doi:10.1145/3298981. URL https://doi.org/10.1145/3298981.

Juergen Dickmann, Jens Klappstein, Markus Hahn, Nils Appenrodt, Hans-Ludwig Bloecher, Klaudius Werber, and Alfons Sailer. Automotive radar the key technology for autonomous driving: From detection and ranging to environmental understanding. In *2016 IEEE Radar Conference (RadarConf)*, pages 1–6, 2016. doi:10.1109/RADAR.2016.7485214.

Dagmar Steinhauser, Patrick Held, Bernhard Thöresz, and Thomas Brandmeier. Towards Safe Autonomous Driving: Challenges of Pedestrian Detection in Rain with Automotive Radar. In *2020 17th European Radar Conference (EuRAD)*, pages 409–412, 2021. doi:10.1109/EuRAD48048.2021.00110.

Muhamamd Ishfaq Hussain, Shoaib Azam, Farzeen Munir, Zafran Khan, and Moongu Jeon. Multiple Objects Tracking using Radar for Autonomous Driving. In *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–4, 2020. doi:10.1109/IEMTRONICS51293.2020.9216363.

Yizhou Wang, Zhongyu Jiang, Yudong Li, Jenq-Neng Hwang, Guanbin Xing, and Hui Liu. RODNet: A Real-Time Radar Object Detection Network Cross-Supervised by Camera-Radar Fused Object 3D Localization. *IEEE Journal of Selected Topics in Signal Processing*, 15(4):954–967, 2021. doi:10.1109/JSTSP.2021.3058895.

Fabian de Ponte Müller, Estefania Munoz Diaz, and Ibrahim Rashdan. Cooperative positioning and radar sensor fusion for relative localization of vehicles. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 1060–1065, 2016. doi:10.1109/IVS.2016.7535520.

Jingyun Liu, Qiao Sun, Zhe Fan, and Yudong Jia. TOF Lidar Development in Autonomous Vehicle. In *2018 IEEE 3rd Optoelectronics Global Conference (OGC)*, pages 185–190, 2018. doi:10.1109/OGC.2018.8529992.

Yukihiro Tsukamoto, Masahiro Ishizaki, Akihito Hiromori, Hirozumi Yamaguchi, and Teruo Higashino. Multi-Lane Detection and Tracking Using Vision for Traffic Situation Awareness. In *2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–6, 2020. doi:10.1109/WiMob50308.2020.9253415.

Moritz Venator, Erich Bruns, and Andreas Maier. Robust Camera Pose Estimation for Unordered Road Scene Images in Varying Viewing Conditions. *IEEE Transactions on Intelligent Vehicles*, 5 (1):165–174, 2020. doi:10.1109/TIV.2019.2955375.

Yunuo Liu, Xiangqi Meng, and Xinquan Huang. Route Planning Strategy of intelligent car Based on Camera Sensor. In *2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, pages 729–732, 2022. doi:10.1109/IPEC54454.2022.9777323.

Moh. Khalid Hasan, Md. Shahjalal, Mostafa Zaman Chowdhury, Nam Tuan Le, and Yeong Min Jang. Simultaneous Traffic Sign Recognition and Real-Time Communication using Dual Camera in ITS. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 517–520, 2019. doi:10.1109/ICAIIC.2019.8668986.

Abduladhem Abdulkareem Ali and Hussein Alaa Hussein. Distance estimation and vehicle position detection based on monocular camera. In *2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA)*, pages 1–4, 2016. doi:10.1109/AIC-MITCSA.2016.7759904.

Zheng Tang, Gaoang Wang, Hao Xiao, Aotian Zheng, and Jenq-Neng Hwang. Single-Camera and Inter-Camera Vehicle Tracking and 3D Speed Estimation Based on Fusion of Visual and Semantic Features. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 108–1087, 2018. doi:10.1109/CVPRW.2018.00022.

Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-End Learning of Driving Models from Large-Scale Video Datasets. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3530–3538, 2016. URL https://api.semanticscholar.org/CorpusID:11913930.

Jiaxin Yang, Benoît Pelletier, and Benoît Champagne. Enhanced autonomous resource selection for LTE-based V2V communication. In *2016 IEEE Vehicular Networking Conference (VNC)*, pages 1–6, 2016. doi:10.1109/VNC.2016.7835937.

Jelena Kocić, Nenad Jovičić, and Vujo Drndarević. Sensors and Sensor Fusion in Autonomous Vehicles. In *2018 26th Telecommunications Forum (TELFOR)*, pages 420–425, 2018. doi:10.1109/TELFOR.2018.8612054.

Muhammad Hasanujjaman, Mostafa Zaman Chowdhury, and Yeong Min Jang. Sensor Fusion in Autonomous Vehicle with Traffic Surveillance Camera System: Detection, Localization, and AI Networking. *Sensors*, 23(6), 2023. ISSN 1424-8220. doi:10.3390/s23063335. URL https://www.mdpi.com/1424-8220/23/6/3335.

Mohamed Abdalwohab, Weibin Zhang, Abdeldime M. S. Abdelgader, and Ibraheem Abdelazeem. Deep learning based camera and radar fusion for object detection and classification. In *2021 IEEE 4th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, pages 322–326, 2021. doi:10.1109/AUTEEE52864.2021.9668695.

Shouyang Wei, Yuan Zou, Xudong Zhang, Tao Zhang, and Xiaoliang Li. An Integrated Longitudinal and Lateral Vehicle Following Control System With Radar and Vehicle-to-Vehicle Communication. *IEEE Transactions on Vehicular Technology*, 68(2):1116–1127, 2019. doi:10.1109/TVT.2018.2890418.

Tzu-Kuang Lee, Yu-Chiao Kuo, Shih-Hsuan Huang, Guan-Sheng Wang, Chih-Yu Lin, and Yu-Chee Tseng. Augmenting Car Surrounding Information by Inter-Vehicle Data Fusion. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2019. doi:10.1109/WCNC.2019.8885487.

Ana Stroescu, Liam Daniel, and Marina Gashinova. Combined Object Detection and Tracking on High Resolution Radar Imagery for Autonomous Driving Using Deep Neural Networks and Particle Filters. In *2020 IEEE Radar Conference (RadarConf20)*, pages 1–6, 2020. doi:10.1109/RadarConf2043947.2020.9266686.

Ana Stroescu, Liam Daniel, Dominic Phippen, Mikhail Cherniakov, and Marina Gashinova. Object Detection on Radar Imagery for Autonomous Driving Using Deep Neural Networks. In *2020 17th European Radar Conference (EuRAD)*, pages 120–123, 2021. doi:10.1109/EuRAD48048.2021.00041.

George Eskandar, Alexander Braun, Martin Meinke, Karim Armanious, and Bin Yang. SLPC: A VRNN-based approach for stochastic lidar prediction and completion in autonomous driving. pages 721–725, 08 2021. doi:10.23919/EUSIPCO54536.2021.9616229.

Scott McCrae and Avideh Zakhor. 3d object detection for autonomous driving using temporal lidar data. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2661–2665, 2020. doi:10.1109/ICIP40778.2020.9191134.

Weishan Zhang, Zhicheng Bao, Yuru Liu, Liang Xu, Qinghua Lu, Huansheng Ning, Xiao Wang, Su Yang, Fei-Yue Wang, and Zengxiang Li. CFSL: A Credible Federated Self-Learning Framework. *IEEE Internet of Things Journal*, 10(24):21349–21362, 2023. doi:10.1109/JIOT.2023.3286398.

Xiaodong Ma, Jia Zhu, Zhihao Lin, Shanxuan Chen, and Yangjie Qin. A state-of-the-art survey on solving non-IID data in Federated Learning. *Future Generation Computer Systems*, 135: 244–258, 2022. ISSN 0167-739X. doi:https://doi.org/10.1016/j.future.2022.05.003. URL https://www.sciencedirect.com/science/article/pii/S0167739X22001686.

Jianxin Zhao, Xinyu Chang, Yanhao Feng, Chi Harold Liu, and Ningbo Liu. Participant Selection for Federated Learning With Heterogeneous Data in Intelligent Transport System. *IEEE Transactions on Intelligent Transportation Systems*, 24(1):1106–1115, 2023. doi:10.1109/TITS.2022.3149753.

Kang Tan, Duncan Bremner, Julien Le Kernec, and Muhammad Imran. Federated Machine Learning in Vehicular Networks: A summary of Recent Applications. In *2020 International Conference on UK-China Emerging Technologies (UCET)*, pages 1–4, 2020. doi:10.1109/UCET51115.2020.9205482.

Beibei Li, Yukun Jiang, Wenbin Sun, Weina Niu, and Peiran Wang. FedVANET: Efficient Federated Learning with Non-IID Data for Vehicular Ad Hoc Networks. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2021. doi:10.1109/GLOBECOM46510.2021.9685068.

G. La Bruna, C. Risma Carletti, R. Rusca, C. Casetti, C. F. Chiasserini, M. Giordanino, and R. Tola. Edge-assisted Federated Learning in Vehicular Networks. In *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*, pages 163–170, 2022. doi:10.1109/MSN57253.2022.00038.

T. M. Sakho and J. Ben Othman. FedVANET-TP: Federated Trajectory Prediction Model for VANETs. In *2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 1–6, 2023. doi:10.1109/WINCOM59760.2023.10322884.

Shikun Zhang, Omid Jafari, and Parth Nagarkar. A Survey on Machine Learning Techniques for Auto Labeling of Video, Audio, and Text Data. *ArXiv*, abs/2109.03784, 2021. URL https://api.semanticscholar.org/CorpusID:237440255.

Bo-Yan Lin, Che-Nan Kuo, and Yu-Da Lin. A Clustering-Based Gauss Chaotic Mapping Particle Swarm Optimization for Auto Labeling in Human Activity Recognition. In *2021 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 238–242, 2021. doi:10.1109/TAAI54685.2021.00052.

Iqbal Hassan, Abtahi Mursalin, Robin Bin Salam, Nazmus Sakib, and H M Zabir Haque. AutoAct: An Auto Labeling Approach Based on Activities of Daily Living in the Wild Domain. In *2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 1–8, 2021. doi:10.1109/ICIEVicIVPR52578.2021.9564211.

Jaewon Lee, Dalwon Jang, and JongSeol Lee. Semi-Automatic SAR Image Land Cover Labeling Pipeline. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1644–1646, 2020. doi:10.1109/ICTC49870.2020.9289591.

Dror Simon, Miriam Farber, and Roman Goldenberg. Auto-Annotation Quality Prediction for Semi-Supervised Learning with Ensembles. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3984–3988, 2020. doi:10.1109/CVPRW50498.2020.00465.

Google. Google cloud vision api. URL https://cloud.google.com/vision/.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi:10.1109/CVPR.2016.91.

Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Hei Li Kwing, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390*, 2020.