A Blockchain-Based Distributed Pruning Deep Compression Approach for Cooperative Positioning in Internet of Vehicles

Dajun Zhang*, Wei Shi*, Marc St-Hilaire*, and Ruizhe Yang†
*School of Information Technology, Carleton University, Ottawa, ON, Canada
†Beijing Laboratory of Advanced Information Networks, Beijing University of Technology, Beijing, China
Email: dajunzhang9038@gmail.com, wei.shi@carleton.ca marc_st_hilaire@carleton.ca, yangruizhe@bjut.edu.cn

2022 International Wireless Communications and Mobile Computing (IWCMC) | 978-1-6654-6749-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/IWCMC55113.2022.9825312

***Abstract*—Autonomous driving is a core application that greatly benefits from Internet of Vehicles (IoV). The calculation of the precise positions of Connected Autonomous Vehicles (CAVs) is mainly done using a Deep Neural Network (DNN) which requires significant computing power. Therefore, reducing the computational overhead and improving the efficiency are urgent problems to be solved. In this paper, we first propose a CAV cooperative learning architecture based on blockchain to improve the positioning accuracy of vehicles. Then, we introduce an error precision sharing model between CAVs. The proposed framework enables CAVs to train vehicle positioning accuracy models locally and exchange them via a blockchain network. Such a distributed training architecture further reduces the computing power required. Extensive simulation results show that the proposed scheme can also significantly improve the accuracy of the trajectory error compared to existing approaches.**

***Index Terms*—Internet of Vehicles, Connected Autonomous Vehicles (CAVs), Deep Neural Network (DNN), Blockchain, Cooperative learning.**

I. Introduction

For Connected Autonomous Vehicles (CAVs) applications such as autonomous driving, Advanced Driver Assistance Systems (ADAS) and location-based services, accurate vehicle positioning is of paramount importance. At present, the Global Positioning System (GPS) is a relatively common and well-known vehicle positioning solution. However, the attenuation of GPS positioning signals is particularly obvious in underground garages and dense areas, which leads to restrictions on the application scenarios of GPS. Generally, the accuracy required by GPS is 15m, which is far from meeting the safety accuracy requirements of automated vehicles. However, with the rapid development of new sensor technologies, such as LiDAR and camera, machine learning, and communication technologies, vehicle positioning accuracy can be significantly improved.

Various solutions have been proposed to improve the positioning accuracy of vehicles. In [1], the authors introduce a blockchain-based vehicle GPS positioning error evolution sharing framework, which aims to ensure the safety and credibility of the vehicle positioning accuracy. Similarly, the authors of [2] propose a new framework for the Internet of Vehicles (IoV) supporting blockchain with Cooperative Positioning (CP) to improve the accuracy, robustness, and security of the vehicle GPS positioning. Authors in [3] study the performance limits of Vehicle-to-Vehicle (V2V) relative positioning with multiple antenna arrays. They illustrate the importance of Angle of Arrival (AOA) and Time Difference of Arrival (TDOA) measurements to position estimation. In [4], a multi-vehicle (vehicle with rich sensors) coordinated positioning correction framework to improve the GPS positioning accuracy of ordinary vehicles is introduced. The authors of [5] combine and compare several solutions in terms of message representation and adaptive transmission policy to reduce overhead, channel congestion, and computational complexity. The work in [6] first studies the advantages of a vehicle-to-vehicle Real-time Relative Positioning (RRP) terrestrial communication system based on the Dedicated Short-Range Communication (DSRC). Then, it considers the position prediction technology applicable to V2V RRP. Finally, some existing schemes are compared.

These studies have used different methods to improve the accuracy of vehicle positioning. However, how to reduce the computational cost is still one of the challenges faced by CAVs. In addition, the safety and timeliness of the positioning scheme also need to be considered. As a result, we consider building a new CAV framework based on blockchain and supporting cooperative positioning to solve these challenges. The underlying IoV environment is composed of multiple CAVs, and each CAV is equipped with LiDAR to help obtain its positioning error. The information interaction between CAVs and Road-Side-Units (RSUs) is completed by blockchain nodes for information sharing and transaction verification. Here, the error accuracy model of each CAV is obtained by the deep pruning compression algorithm [7]. Each CAV improves and corrects its own positioning error by sharing the error accuracy model. Moreover, we also design a smart contract to clarify the data-sharing mechanism and security protection mechanism between CAVs. The main contributions are summarized below.

- We design a distributed vehicle trajectory accuracy error model that leverages the permissioned blockchain due to its low-cost, low-latency, and low-bandwidth-intensive characteristics. The proposed system aims to improve the accuracy, system timeliness and system security of the trajectory accuracy error positioning.

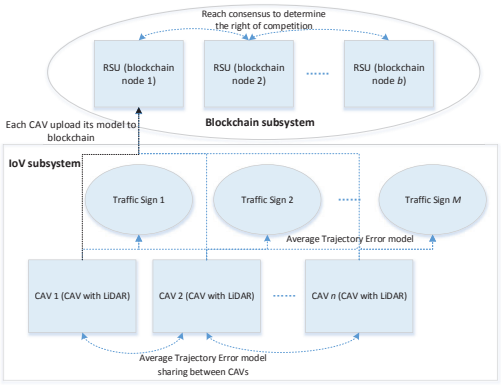

Fig. 1: The proposed framework with blockchain.

- We apply a pruning compression algorithm on deep neural network compression on each local training model. The algorithm improves the speed of local training by reducing unnecessary convolution kernels.
- Finally, we design a smart contract for vehicle trajectory accuracy error model sharing. Furthermore, a new transaction sorting mode is designed to ensure that the data requester obtains the transaction with the highest error accuracy. While the permissioned blockchain system ensures high performance and better scalability of the system architecture, the smart contracts ensure the security of the sharing mechanism.

The rest of this paper is organized as follows. Section II introduces the system model and Section III explains how deep compression neural networks are used for local training. In Section IV, we explain the cooperative learning method in CAV and report the performance of the architecture in Section V. Finally, we conclude the paper in Section VI.

II. System Model

A. Position Trajectory Error Model for CAV

As shown in Fig. 1, the IoV subsystem is composed of Traffic Sign (TS), RSU, and CAVs in the same road section, and mainly realizes the sharing of positioning accuracy between CAVs. Specifically, each RSU has sufficient storage space and computing power to process vehicle data requests, data selection, and data transmission initiated by the vehicle, reducing the amount of calculation for CAV local training. CAV nodes are connected to each other through DSRC communication. The blockchain subsystem, composed of RSUs, is responsible for processing corresponding transaction verification and accounting in a trusted and secure manner.

As shown in Fig. 2, we place a CAV in the center of the two-dimensional coordinate system. To evaluate the authenticity of the actual trajectory of the vehicle, we use a straight line ($Ax + By + C = 0$) to represent the trajectory of the vehicle. In this framework, the precise location of the TS is set by the specific method installed in the preset location (i.e., the coordinates of the RSU are known), and

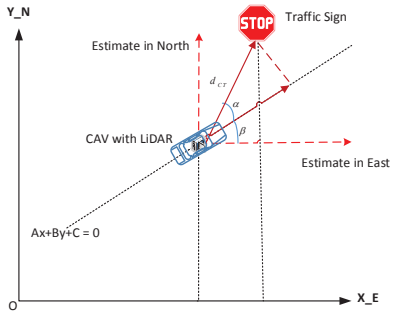

Fig. 2: The simplified calculated model of CAV.

all the location coordinate values (including GPS, LiDAR and TS) are based on this reference coordinate system (the GPS position coordinate value of the vehicle can be obtained through the corresponding coordinate transformation). Since the trajectory of the CAV is a horizontal road, we only focus on the position error in the X and Y directions.

For the straight line, we perform least-squares fitting to calculate the parameter A corresponding to the CAV location. First, the points of the vertical distance to the line are the sample points, which are defined as (x_z, y_z), and the number of samples is set to Z. We establish the variance error e of the fitting straight line: $e = \sum_{z=1}^{Z}(y_z - Ax_z - B)^2$.

To minimize e, we take the partial derivatives of A to make them equal to zero. Therefore, we have $\frac{\partial \sum_{z=1}^{Z}(y_z - Ax_z - B)^2}{\partial A} = 0$.

Therefore, A can be obtained from the equation below:

$$A = \frac{\sum_{z=1}^{Z} x_z^2 \sum_{z=1}^{Z} y_z - \sum_{z=1}^{Z} x_z (\sum_{z=1}^{Z} x_z y_z)}{Z \sum_{z=1}^{Z} x_z^2 - (\sum_{z=1}^{Z} x_z)^2} \tag{1}$$

We define the distance from a CAV to the TS as d_{CT}, and the coordinates of the position of TS j in the two-dimensional coordinate system as P_{T_j} $(p_{T_{jx}}, p_{T_{jy}})$. The coordinates of CAV i are P_{Vi}^j $(p_{V_{ix}}^j, p_{V_{iy}}^j)$, and the position coordinates of vehicle i are measured by the on-board LiDAR based on the known position coordinates of TS j. P_{Vi}^j is expressed as:

$$\begin{cases} p_{V_{ix}}^j = p_{T_{jx}} - d_{CT} \cdot \cos(\alpha + \beta) \\ p_{V_{iy}}^j = p_{T_{jy}} - d_{CT} \cdot \sin(\alpha + \beta) \end{cases} \tag{2}$$

Here, α is the angle between the TS and the CAV driving track, which can be measured by the on-board LiDAR. β is the auxiliary angle, which is used to determine the position of the CAV (P_{Vi}^j). The slope of the trajectory can be obtained from (1), so the angle β can be expressed as $\beta = \arctan(A)$.

It is important to point out that the vehicle-mounted LiDAR in this article is on the top of the vehicle. We define the distance between the LiDAR and the GPS of the same vehicle as a constant d_{LG}. Therefore, we can get the relative precise

positioning $p_i^j(p_{ix}^j, p_{iy}^j)$ of CAV i as:

$$\begin{cases} p_{ix}^j = p_{V_{ix}}^j \\ p_{iy}^j = p_{V_{iy}}^j + d_{LG} \end{cases} \tag{3}$$

Based on the above definition, the trajectory error ΔT_i^j of CAV i is determined by the GPS positioning accuracy P_{Gi} and relative precise positioning p_i^j. Hence, we obtain $\Delta T_i^j = ||P_{Gi} - p_i^j||$. Considering the directionality of vehicle position, the vector of $\overrightarrow{\Delta T_i^j}$ is the value of the trajectory error ΔT_i^j and its direction information.

We define that there are M traffic signs in the proposed framework, that is $\mathcal{R} = \{R_1, R_2, ..., R_j, ..., R_M\}$. Hence, the average trajectory error of CAV i can be defined as:

$$\overrightarrow{\Delta T}_i^j = \frac{\sum_{j=1}^{M} \Delta T_i^j}{M} \tag{4}$$

III. Deep Compression Method for Local Distance Prediction

As mentioned above, the CAV uses its mounted LiDAR to obtain its position through TS (e.g. traffic signal) and compute its GPS track positioning error. Here, the key point that may affect positioning accuracy is the distance (d_{CT}) between CAV and TS. The optimal distance with the minimum trajectory positioning error can be determine through DNN training.

A. CNN Architecture

Considering that the proposed framework is mainly for straight-line driving trajectories, the distance d_{CT} between the vehicle and the TS depends on the driving speed, CAV's acceleration, the slope of the two-dimensional space, the driving time, and the location of the TS. We set the parameters as follows: input matrix S_{input} (we treat the input matrix as a picture of $N \times N$ pixels): the position of the TS P_{T_j} ($p_{T_{jx}}$, $p_{T_{jy}}$) on the road; the driving speed v_{C_i}; the spatial straight line slope a; CAV's acceleration, and the CAV driving time t_i. In particular, the driving time of a vehicle refers to driving in the TS road section. S_{input} feeds to the first convolution layer that convolves the input image with 32 filters, kernel size 2x2, stride size 1, and the same padding method. The output of the first convolution layer is convolved by the second convolution layer with 48 filters and the same stride size and padding method. The rectifier nonlinearity activation function (ReLU) is used as the activation function for two convolution layers. Moreover, the two max-pooling layers are connected to the first and second convolution layers. The fully connected layers that are used as two neural networks of all 512 units are designed to connect with the second max-pooling layer. Finally, the distance d_{CT} is returned by the output layer.

B. Pruning Filters Applied in CNN for Distance Prediction

In this paper, we directly remove the convolution kernel that has little effect on the accuracy of CNN. For the S_{input}, the number of input channels of the convolutional layer m is characterized as D_m. w_m and h_m are the width and height of the input feature map, respectively. When the convolutional layer receives the feature map with the input dimension $E_m \subset R^{D_m \times h_m \times w_m}$, it extracts the feature map with the dimension $E_{m+1} \subset R^{D_{x+1} \times h_{m+1} \times w_{m+1}}$ and takes it as input for the next layer. In particular, the multiplication and addition (MAdds) of the convolution kernel with the size of $D_{m+1} \times D_m \times k \times k$ in the convolutional layer m is $D_{m+1} \times D_m \times k^2 \times h_{m+1} \times w_{m+1}$. In the deep compression neural network used in our architecture, a certain convolution kernel in convolutional layer m is pruned, then the features extracted by this convolution kernel are also pruned at the same time, thereby reducing the number of computations to $D_m \times k^2 \times h_m \times w_m$. Furthermore, since the output feature map of the previous layer is cropped, the number of input channels of the next layer of convolution is reduced, so the number of the reduced computation is $D_{m+2} \times k^2 \times h_{m+2} \times w_{m+2}$. By analogy, we can conclude that if the convolution kernel with the smallest absolute value is cut, the dimension of the convolution kernel of this layer will be reduced accordingly, which will reduce the number of input channels to the next layer. Finally, deep pruning compression algorithm in our framework can effectively reduce the number of local training computations.

Therefore, the pruning compression strategy of CNN in this article cuts out m convolution kernels from the i^{th} convolutional layer as follows: 1) For each k_m (convolution kernel), calculate the sum of its weight absolute values $s_a = \sum_{m=1}^{D_m} \sum |\eta_m|$. Here, η_m represents the weights in the m^{th} convolutional kernel and D_m represents the number of input channels of the m^{th} convolutional layer. 2) Sort by K_a; 3) Cut the m convolution kernels with the smallest sum of absolute weights and the corresponding feature maps. 4) A new weight matrix for layer m and $m+1$ layer is created, and the remaining weight parameters are copied to the next layer model.

IV. Blockchain and Smart Contracts

A. Details of Blockchain in Proposed Framework

1) *Format of Each Block*: The block body records the transaction with the block confirmation. This paper records one type of transaction, referred to as the data demand and sharing records between CAVs.

2) *Redundant Byzantine Fault Tolerance (RBFT) Consensus Mechanism*: In our proposed framework, each RSU in the blockchain subsystem can act as a consensus node to participate in the verification of newly generated blocks. The primary RSU node packages the transaction into blocks and then verifies it and writes the verification result into the $Pre-Prepare$ message for network-wide broadcast so that it can include both sorting transaction information and the result of the block verification. The RSU node first checks the validity of the message after receiving the $Pre-Prepare$ message from the primary node. After the check is passed, the $Pre-Prepare$ message is broadcast to indicate that the node agrees with the ordering result of the primary node. The slave node will only start to verify the block after receiving the

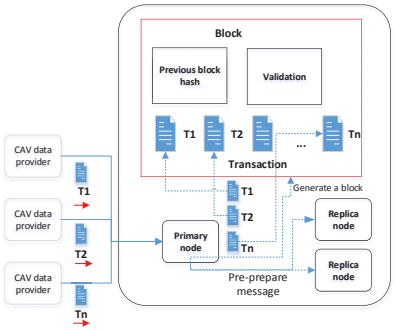

Fig. 3: Transaction sorting process.

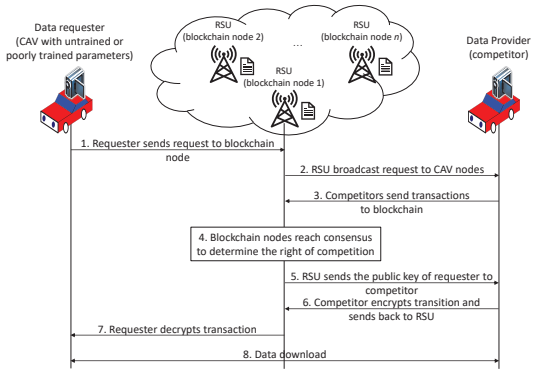

Fig. 4: Smart contract for local model parameter sharing.

$Pre-Prepare$ message. The verification result is compared with the primary node. If the comparison result is consistent, the broadcast $Commit$ message indicates that the node agrees with the verification result of the primary node. Otherwise, it directly initiates $ViewChange$ to indicate that the node believes that the master node has abnormal behavior. The RBFT consensus mechanism ensures that the smart contract can be carried out safely and effectively, ensuring the sharing of positioning accuracy between CAV nodes.

3) *Transaction sorting process*: The CAV data requester submits the transaction to the blockchain system, and the ordering service node (primary node in RBFT) creates transaction blocks. These transaction blocks will eventually be distributed to all replica nodes for final verification and submission. In particular, the job of the ordering service node is to arrange the submitted transactions into batches then pack them into blocks in the order of error accuracy from high to low. The sorting process is shown in Fig. 3.

B. Smart Contracts

In our proposed architecture, different CAVs are mainly used to share the CAV positioning error of blockchain verification. If the CAV directly uploads the locally trained DNN model parameters (positioning accuracy), other nodes can easily use their data maliciously and cannot guarantee their own interests. Therefore, we designed a set of smart contracts to reduce the negative impact of malicious nodes, thereby improving the security of the system. The Smart Contract for Local Model Parameter (SCLMP) sharing is shown in Fig. 4. The contents of SCLMP are as follows:

1) ***Data request***: In our proposed architecture, some CAV nodes that have poor training results can request the nearest RSU node to obtain the best trained DNN model parameters. The request message includes the required data type, public key, and reward with their private signature.

2) ***Data sharing rights competition***: After receiving a request, the RSU node broadcasts this request to other participating CAV nodes, which consequently encapsulate their local training results into transactions and upload them to the blockchain system. The primary node performs transaction sorting. If a consensus is reached, the node with the highest error accuracy in the transaction sequence will share its data.

3) ***Data sharing***: When the RSU node receives a parameter message from the data provider, the RSU node sends the public key of the data requester to the data provider. The data provider uses the public key to encrypt the trained DNN model parameters and sends them to the RSU node.

4) ***Data reception and reward***: When the data requester receives a message sent by a RSU, the encrypted data will be decrypted by the data requester with the private key to obtain the required DNN parameters. In particular, once the data requester uses the link, the link will automatically become invalid to prevent abuse by other nodes. Meanwhile, the RSU node immediately sends the reward to the data provider.

5) ***Transaction record***: The RSU node records the summary of each transaction simultaneously. Then, the recorded transaction summary is periodically packaged into blocks and broadcast to the blockchain subsystem for further verification. If the blockchain system reaches a consensus, these blocks will be added to the blockchain.

V. Simulation Results and Discussions

We first evaluate the trajectory error accuracy of the proposed scheme. We then compare our results against Multi TS Cooperative Error Evaluation (MTCEE) [2] and GPS. We also analyze the timeliness and security performance.

A. Comparison of the Trajectory Error Accuracy

In CNN, the pruning strategy is set to start at 2,000 and end at 4,000 steps. We set a binary mask variable at each selected layer. On the premise that the shape of the weight tensor is consistent, the role of the mask variable is to determine the tensor participating in the weight update. In the Tensorflow training process, the update of the mask is done by adding a special operator. Its purpose is to control the CNN's selected layer weight sorting rules based on absolute values. When Tensorflow training is updated, the mask value whose absolute value of the weight is less than the threshold is set to 0. The mask will also be used in the back-propagation gradient descent, and the weight with a mask value of 0 will not update the weight value in the gradient descent. We control the absolute value of the weights and sort them by setting the

TABLE I: DNN Settings

Parameter	Value	Parameter	Value
Convolutional layer	4	Pooling layer	4
Batch size	32	Learning rate	0.00001
Discount factor	0.9	Total training steps	4000
Traffic sign number	4		

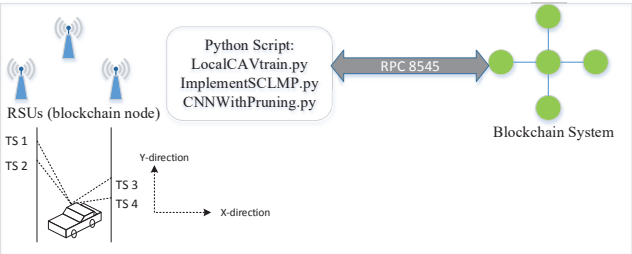

Fig. 5: Simulation structure of the proposed architecture.

mask variable. Table I shows the parameter settings that were used.

As shown in Fig. 5, the communication between each RSU (blockchain node) and the blockchain framework (including the establishment of smart contracts) is done via Remote Procedure Call (RPC). It contains the mapping of each account (inside the neighbor RSU) to messages, exchanges, and labels. Among them, *ImplementSCLMP.py* is a smart contract script that is used to extract the TensorFlow-based *LocalCAVtrain.py* file (local training results) into the contract script, and at the same time transmit the content of the contract to the blockchain node; *CNNWithPruning.py* aims to use a deep compression architecture to greatly reduce the computational cost and training delay of local training.

As shown in Fig. 6, the traditional positioning error that relies solely on GPS is the largest. The remaining solutions improve the accuracy of the trajectory error. We can see that our proposed scheme (SCLMP) performs better than existing schemes in terms of error accuracy. The reasons can be summarized as follows: 1) Our proposed scheme uses convolutional nerves on the basis of existing schemes. The network can more accurately identify local features (i.e., positioning accuracy parameters), thereby further improving the position information of TS and RSU, so the error accuracy is better. 2) Our local error accuracy training part introduces the pruning compression method. We aim to reduce the impact of parameter redundancy on the training effect, accelerate the network convergence, and have a higher accuracy rate. 3) In the blockchain consensus part, we adopted the permissioned consensus algorithm (RBFT). This consensus algorithm is faster. At the same time, the authority consensus algorithm requires that the identities of all participating nodes are known, ensuring accuracy and security in the accuracy sharing process. In summary, the proposed scheme has an average error of 2.87 meters as compared to 4.03 meters for MTCEE.

In order to further clarify the performance of the positioning accuracy, we respectively compared the positioning

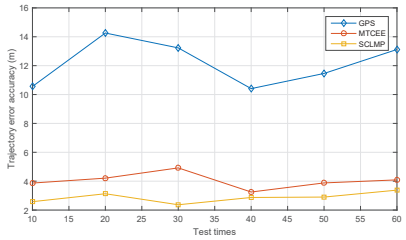

Fig. 6: Trajectory error accuracy comparison between different approaches.

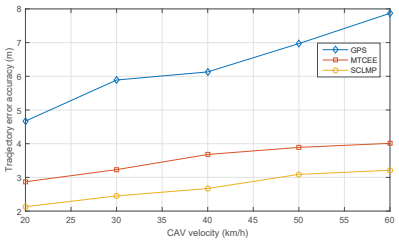

Fig. 7: Trajectory error accuracy comparison with respect to different vehicle speeds.

error accuracy of the three schemes at 20, 30, 40, 50, and 60km/h vehicle speeds. According to Fig. 7, as the driving speed increases, the local training speed between CAVs and the speed of interaction with the blockchain decreases. This also affects the calculation accuracy of the trajectory error. However, at different speeds, SCLMP consistently receives fewer positioning errors than MTCEE.

B. Other Performance Analysis

1) *Security*: We analyze the security by analyzing the data security of the proposed architecture and the influence of malicious nodes on the system. First, the trained CAV nodes need to provide their trajectory accuracy error to the blockchain system and obtain the right to provide data through competition (SCLMP), thereby protecting the security of the data requested by the data holder. Second, if there are malicious nodes that provide false trajectory error accuracy, due to the nature of the blockchain, the data written to the blockchain cannot be tampered with. Therefore, false data providers can be directly removed from the blockchain system to ensure data accuracy and security.

2) *Robustness*: Fig. 8 illustrates the comparison of the positioning accuracy of the different schemes with 4 TSs, a speed of 50km/h and different numbers of data providers (collaborators). From the figure, we can see that with the increase of the number of data providers, the positioning accuracy obtained by the data requester has been improved to varying degrees. Obviously, as the number of collaborators increases, the data requester can correct its errors by multi-point cooperation in the positioning accuracy. However, our proposed SCLMP scheme can obtain more precise localization errors than the other two schemes. According to Fig. 8,

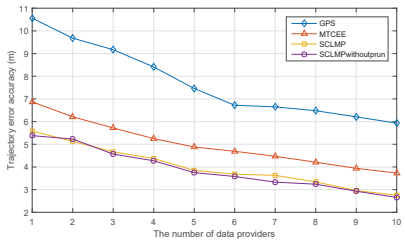

Fig. 8: Trajectory error accuracy comparison between different number of collaborators.

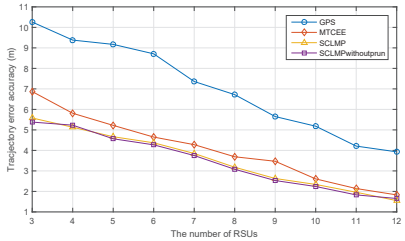

Fig. 9: Trajectory error accuracy comparison between different number of RSUs.

when the number of data requesters is reduced from 6 to 5, the positioning error accuracy of GPS and MTCEE increases to 7.57m and 5.03m respectively. However, our proposed SCLMP scheme provides a better accuracy with 4.15m. In summary, for the same number of cooperations, SCLMP has better error accuracy than GPS and MTCEE, indicating that the scheme has strong adaptability to environmental changes and maintains high robustness.

Fig. 9 demonstrates the correlation between the system positioning accuracy and the number of consensus nodes (RSUs) under a fixed speed of 50km/h, 4 TSs and 6 cooperators. The increase in the number of consensus nodes (RSUs) in the blockchain leads to an increase in system delay, energy consumption between nodes, and the positioning accuracy of the system is also be affected, which leads to a negative impact to the blockchain system. However, according to Fig. 6 to Fig. 9, the strong advantages of the blockchain system in terms of robustness, security and adaptability cannot be ignored. SCLMP outperforms the two existing schemes on error accuracy even when the latency is increased, which demonstrates its high robustness.

3) *Timeliness*: Fig. 10 shows the total training set of MTCEE and SCLMP where the corresponding curves converge above 90%. The training latency rests on the number of training episodes. From this figure, we can see that the training delay of our proposed SCLMP scheme is always lower than MTCEE. Our proposed solution demonstrates two improvements on its timeliness: i) local training speed is increased due to the deep compression scheme performed; ii) a CAV can request the pre-trained CNN model parameters through the blockchain, therefore, eliminating a long training delay. This figure further confirms that our proposed cooperative learning scheme leads to a shorter delay.

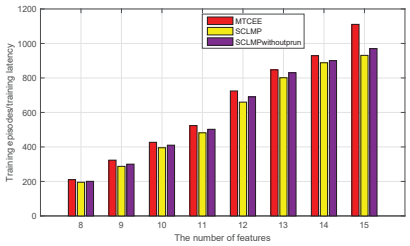

Fig. 10: Training latency comparison.

According to Figs. 8 and 9, the accuracy of the SCTPE scheme is very close to the one of SCTPE method without pruning. This is because the pruning compression scheme adopted by the local training of SCTPE can recover the accuracy through a short-term retraining (less than the original training time). As shown in Fig. 10, the applications of pruning and compression leads to great reduction on the computing cost of the RSU and improvement on the training efficiency, thereby ensuring the superiority of the overall performance of the proposed SCTPE system.

VI. Conclusions and Future Work

In this paper, we proposed a new framework based on blockchain to improve the accuracy of CAV positioning. The architecture includes the blockchain subsystem and the CAV subsystem. In addition, we propose a local distance prediction method based on the deep compression method to get the trajectory error accuracy model of a CAV. Finally, we combined the characteristics of the blockchain and proposed SCLMP that shares the error model between CAVs. Compared with traditional methods, the experiments show the effectiveness and accuracy of our proposed method. For future work, we plan to apply the proposed framework in embedded or mobile systems such as Internet of Things.

References

[1] C. Li, Y. Fu, F. R. Yu, T. H. Luan, and Y. Zhang, "Vehicle position correction: A vehicular blockchain networks-based GPS error sharing framework," *IEEE Trans. Intelligent Transp. Sys.*, vol. 22, no. 2, pp. 898–912, 2020.

[2] Y. Song, Y. Fu, F. R. Yu, and L. Zhou, "Blockchain-enabled internet of vehicles with cooperative positioning: A deep neural network approach," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3485–3498, 2020.

[3] A. Kakkavas, M. H. C. Garcia, R. A. Stirling-Gallacher, and J. A. Nossek, "Multi-array 5G V2V relative positioning: Performance bounds," in *Proc. IEEE GLOBECOM'18*, 2018, pp. 206–212.

[4] X. Kong, H. Gao, G. Shen, G. Duan, and S. K. Das, "Fedvcp: A federated-learning-based cooperative positioning scheme for social internet of vehicles," *IEEE Trans. Computational Social Systems*, 2021.

[5] G.-M. Hoang, B. Denis, J. Härri, and D. T. Slock, "On communication aspects of particle-based cooperative positioning in GPS-aided VANETs," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 20–25.

[6] K. Ansari, "Cooperative position prediction: Beyond vehicle-to-vehicle relative positioning," *IEEE Trans. Intell. Transp.*, vol. 21, no. 3, pp. 1121–1130, 2019.

[7] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.