Edge-Assisted Privacy-Preserving Raw Data Sharing Framework for Connected Autonomous Vehicles

Jinbo Xiong, Renwan Bi, Mingfeng Zhao, Jingda Guo, and Qing Yang

Abstract

Data sharing among connected and autonomous vehicles without any protection will cause private information leakage. Simply encrypting data introduces a heavy overhead; most importantly, when encrypted data (ciphertext) is decrypted on a vehicle, the receiver will be fully aware of the sender's data, implying potential data leakage. To tackle these issues, we propose an edge-assisted privacy-preserving raw data sharing framework. First, we leverage the additive secret sharing technique to encrypt raw data into two ciphertexts and construct two classes of secure functions. The functions are then used to implement a privacy-preserving convolutional neural network (P-CNN). Finally, two edge servers are deployed to cooperatively execute P-CNN to extract features from two ciphertexts to obtain the same object detection results as the original CNN. We adopt the VGG16 model as a case study to illustrate how to construct P-CNN and employ the KITTI dataset to verify our solution. Experiment results demonstrate that P-CNN offers exactly the same classification results as the VGG16 model with negligible error, and the communication overhead and computational cost on the edge servers are less than existing solutions without leaking private information.

Introduction

An autonomous vehicle leverages a suite of sensors to accurately perceive its surrounding environment; for example, high-definition (HD) cameras are used to detect objects. In parallel with the advancement of autonomous driving technologies, we have witnessed a rapid development of wireless vehicular communication solutions [1, 2], such as vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), vehicle-to-everything (V2X), and cloud V2X (C-V2X)/5G communications. With the recent development of faster and more robust wireless network technologies [3], it is becoming possible for vehicles to share raw sensor data — for example, HD video streams or light detection and ranging (LiDAR) point cloud data — with each other. It has been proven that sharing raw LiDAR data among autonomous vehicles can help improve the 3D object detection accuracy on individual vehicles [4]. Transmitting and processing the massive amount of raw data shared among autonomous vehicles, however, is still an open problem. This problem becomes more challenging, compounded by the fact that the entire system may be subject to various security attacks. Therefore, it is crucial to implement an efficient but secure communication framework for connected autonomous vehicles (CAVs) to exchange raw data between each other.

The majority of existing security solutions for CAVs leverage data encryption techniques [5]; however, no matter whether symmetric- or asymmetric-key encryption is adopted, high key management overhead is inevitable. More importantly, after a receiving vehicle decrypts a ciphertext, the original raw data becomes available to the receiver, which may obtain private information about the sender. For example, by simply analyzing the background information in an image, it is possible to pinpoint the sender's location [6]. Another option is homomorphic encryption [7, 8], which does not require intermediate servers to decrypt a ciphertext; however, it is not suitable for CAVs due to the massive overhead in both computation and memory cost. In addition, the complex computations in homomorphic encryption usually costs a long data processing delay, which is prohibitive for a CAV that requires extremely low latency and near-real-time performance.

Recently, edge computing has been advocated to offload heavy computing tasks from mobile devices to edge servers, achieving an efficient system design [9, 10]. The new computing paradigm is perfectly suited for our privacy-preserving raw data sharing framework for CAVs. In our framework, a vehicle randomly splits/encrypts its sensor data into two (complementary) copies, which are sent to two different edge servers. Leveraging the additive secret sharing technique [11], the edge servers process the ciphertexts to learn useful information from them. The learning process is achieved by processing encrypted sensor data via a privacy-preserving convolutional neural network (P-CNN) model. The outputs of P-CNN are then sent to the receiver, which combines the received data to obtain the information contained in the original sensor data.

There are several advantages in our edge-assisted privacy-preserving raw data sharing framework, which are listed as follows.

Digital Object Identifier:
10.1109/MWC.001.1900463

Jinbo Xiong is with Fujian Normal University and the University of North Texas;
Renwan Bi and Mingfeng Zhao are with Fujian Normal University; Jingda Guo and Qing Yang are with the University of North Texas.

Lightweight: As heavy computational tasks (e.g., ciphertext-based image processing) are carried out on edge servers, the computational cost on individual vehicles is significantly reduced. As a result, a sender (vehicle) only needs to randomly split its raw image data into two pieces, and a receiver (vehicle) simply combines the results forwarded from edge servers.

Privacy-Preserving: Edge servers only possess a portion of an image; therefore, they are not able to extract the private information contained in the original image, assuming no collusion. Similarly, encrypted images are processed by edge servers, so a receiver only knows the processing results, and the privacy of the sender is protected.

Correct: Secure functions are designed in P-CNN to implement the mathematical operations involved in the original CNN. Two edge servers execute P-CNN to learn the features from images in a ciphertext format. After receiving the features from edge servers, a receiver can extract the same classification result as is generated from the original CNN.

The source code of P-CNN and the associated dataset are provided at https://github.com/CAVLab-tech/P-CNN-for-CAV.

Privacy-Preserving Raw Data Sharing Framework for CAVs

Data sharing among autonomous vehicles without any protection will be subject to privacy leakage. To tackle this issue, we leverage an additive secret sharing technique [11] to protect data. thus greatly reducing key management and encryption overhead.

System Architecture

In the proposed framework, there are three major components: autonomous vehicles ($\mathcal{V}$), edge servers ($\mathcal{S}$), and a trusted server ($\mathcal{T}$), as shown in Fig. 1. A sender vehicle perceives its surrounding environment, using various types of sensors, and generates real-time sensor data. In this article, we focus only on the image data captured by HD cameras on autonomous vehicles and forgo other types of sensor data. The proposed privacy-preserving techniques, however, could be applied to protect other types of raw data, including LiDAR and radar data.

Each image captured by a sender is encrypted into two encrypted images, also called ciphertexts. The two ciphertexts will be sent to two different edge servers, which employ a revised deep learning model to realize ciphertext-based image processing. As CNN-based solutions outperform others regarding object classification on autonomous vehicles, in this article, we concentrate our research on CNN-based object classification. We name our ciphertext-based object classification model P-CNN. After an edge server receives an encrypted image, the P-CNN model will "classify" the image and produce an output. During the process, a trusted server is involved to generate random parameters to support the secure interactions between the two edge servers. The outputs of P-CNN from two edge servers will be forwarded to a receiver vehicle, which recovers the original classification results from the ciphertexts.

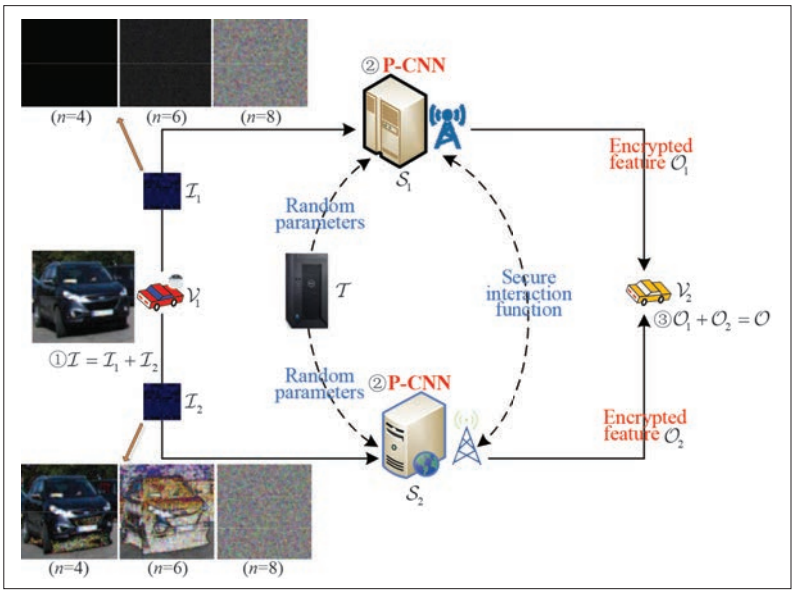

FIGURE 1. System architecture of the privacy-preserving raw data sharing framework for connected and autonomous vehicles.

Image Encryption

To illustrate how images are encrypted, we use $\mathcal{V}_1$ and $\mathcal{V}_2$ to denote the sender and receiver vehicles, respectively. Leveraging the additive secret sharing algorithm, $\mathcal{V}_1$ encrypts each frame of its data into two ciphertext images. Specifically, for each original image $\mathcal{I}$, $\mathcal{V}_1$ generates a random image $\mathcal{I}_1$ using a random generator. Following that, $\mathcal{V}_1$ obtains $\mathcal{I}_2$ by subtracting $\mathcal{I}_1$ from $\mathcal{I}$. Clearly, $\mathcal{I}_1$, $\mathcal{I}_2$ and $\mathcal{I}$ are in the same dimension. Images $\mathcal{I}_1$ and $\mathcal{I}_2$ are then sent to the two edge servers. To obtain stronger security protection, we define the random space of $\mathcal{I}_1$ as $[-2^{n-1}, 2^{n-1} - 1]$, where $n \geq 8$ is a security parameter. As shown in Fig. 1, if $n = 4$, the generated image is still recognizable. When n is increased to 6, we can barely see that there is a vehicle in the encrypted image $\mathcal{I}_2$. When $n = 8$, the two images $\mathcal{I}_1$ and $\mathcal{I}_2$ are totally unrecognizable, achieving better privacy protection.

Image Processing with P-CNN

After receiving ciphertext images, the two edge servers execute ciphertext-based image processing to extract ciphertext features, and then "classify" ciphertext images. As a result, the two edge servers will obtain two results, denoted as $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively. It is worth mentioning that the information contained in $\mathcal{O}_1$ and $\mathcal{O}_2$ is meaningless at this moment because they do not correctly tell what types of objects are detected.

To design P-CNN, we need to make changes on four types of layers within a CNN: the convolution (CONV) layer, activation (ACT) layer, pooling (POOL) layer, and full connection (FC) layer. We know that the CONV and FC layers consider different coefficients (weights and biases) in their computation, while the ACT and POOL layers do not. To implement P-CNN, we first train a CNN (classification) model using a publicly available dataset. Then we reconstruct another neural network matching the trained CNN model, where the new CONV and FC layers invoke the same weight and bias coefficients as the trained CNN model.

To design P-CNN, we first need to understand the major operations/computations involved in the four types of layers in VGG16. The CONV layer is mainly responsible for extracting local features from images. While features are extracted, the underlying computation can be viewed as a linear combination of the data from images and the convolution core.

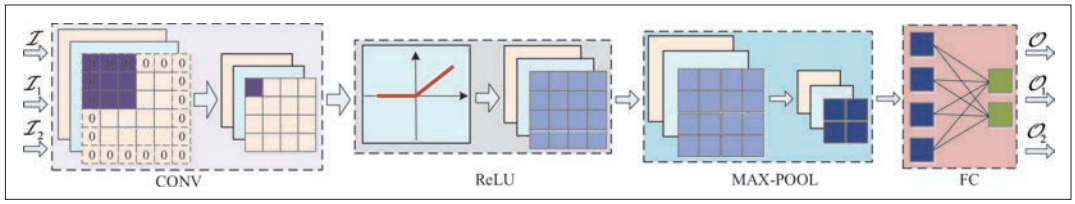

FIGURE 2. Major layers in the VGG16 model: CONV layer, ReLU layer, MAX-POOL layer, and FC layer.

Within the new network, a series of secure functions are introduced to implement the ACT and POOL layers.

Object Classification Results

Upon receiving $\mathcal{O}_1$ and $\mathcal{O}_2$ from the edge servers, vehicle $\mathcal{V}_2$ combines $\mathcal{O}_1$ and $\mathcal{O}_2$ to decrypt the information provided by the two P-CNN models on edge servers. According to the additive secret sharing algorithm, simply adding $\mathcal{O}_1$ and $\mathcal{O}_2$ yields the same classification result as the original CNN produces by processing the original image $\mathcal{I}$.

Design of Privacy-Preserving CNN

Overview of VGG16

Unlike previous works [12], which consider simple CNN models consisting of 1, 2, or 7 CONV layers, we utilize the VGG16 [13] model to implement P-CNN. We choose VGG16 because the majority of the state-of-the-art image classification algorithms are based on VGG16. VGG16 consists of 16 hidden layers (13 CONV layers and 3 FC layers), 15 ACT layers, and 5 POOL layers, as shown in Fig. 2.

Functions in VGG16

To design P-CNN, we first need to understand the major operations/computations involved in the four types of layers in VGG16. The CONV layer is mainly responsible for extracting local features from images. While features are extracted, the underlying computation can be viewed as a linear combination of the data from images and the convolution core (i.e., the weights and biases).

The ACT layer, on the other hand, is used to increase the nonlinear expression ability of the model. Therefore, nonlinear computations are involved in this layer. The activation function used in VGG16 is called the linear correction unit (ReLU). In essence, the purpose of ReLU is to set the negative element in the input to zero.

The POOL layer is responsible for feature selection, also known as feature map compression; that is, a down-sampling area is replaced by a single value. VGG16 adopts MAX-POOL, which selects the largest element from a 2×2 matrix to represent this matrix.

The FC layer is used to map features to labeled samples (i.e., making the final classification decision). As both anterior and posterior neurons in this layer are connected, including weights and biases, FC can be interpreted as a special CONV layer.

Dataset and Training

To ensure P-CNN works in practice, it is important to select an appropriate real-world dataset to train and test our model.

As our framework is designed for CAVs, we choose the KITTI dataset [14], which is considered the most popular vision benchmark suite for autonomous vehicle technologies. We process the KITTI dataset by cropping out the vehicles and pedestrians from each frame provided in the dataset.

With the hand-crafted training dataset, we retrain the classical VGG16 model by altering the last FC layer to produce only two outputs: vehicles and pedestrians. We set the required precision of P-CNN at no less than 94 percent because humans are able to recognize different objects while driving with a precision of about 94 percent. After 10,000 iterations training on our own dataset, VGG16 is able to offer up to 96.93 percent classification precision. The retrained VGG16 model will be modified to provide privacy preservation, which is discussed in later sections.

Secure Functions in P-CNN

Secure functions are designed to implement the operations performed in the original VGG16 model so that P-CNN can process ciphertexts and obtain useful information. As shown in Fig. 2, the original input $\mathcal{I}$ is processed by VGG16 via four major layers to produce an output $\mathcal{O}$. Our main objective is to let P-CNN process $\mathcal{I}_1$ and $\mathcal{I}_2$ to obtain $\mathcal{O}_1$ and $\mathcal{O}_2$ such that combining $\mathcal{O}_1$ and $\mathcal{O}_2$ yields $\mathcal{O}$.

Our secure functions are designed based on the additive secret sharing technique which is a popular and lightweight secure algorithm offering practical and low-cost operations. In the following, we introduce two classes of fundamental secure functions: *secure non-interactive functions* and *secure interactive functions*. While the former is used to implement the operations involved in the CONV/FC/POOL layers in VGG16, the latter is for the ACT layers.

Secure Non-Interactive Functions: Secure non-interactive functions are designed to securely implement the CONV and FC layers. Because both CONV and FC layers conduct a linear combination of input values, they can be re-implemented easily using our security functions. Considering the nature of operations involved in a CONV/FC layer (i.e., addition, subtraction, and scalar multiplication), we need to design three corresponding secure functions: secure addition (SecAdd), secure subtraction (SecSub), and secure scalar multiplication (SecSMul) functions.

Given two input matrices, $\mathcal{T}$ is responsible for randomly splitting the matrix into two components and sending them to two edge servers. The idea of the SecAdd/SecSub function is that the edge servers carry out addition or subtraction operations on their received matrices. In contrast, the SecSMul function is designed to securely calculate the product between a fixed matrix and a single input matrix consisting of two component matrices. The two edge servers multiply the fixed matrix and their component matrix. The above security functions

can get the same result as the normal calculation. Most importantly, the calculating process of the security functions does not leak any information to two edge servers.

It is easy to verify that the linear calculations in CONV/FC layers can be distributed by means of the above secure non-interactive functions. It is worth noting that although the POOL layer seems to be a nonlinear operation, it can be implemented by SecSub to ensure correctness and security.

Secure Interactive Functions: Secure interactive functions are designed to securely implement the ReLU function in the ACT layer. This is because the input matrix is split into two sub-matrices, and each of them is held by only one edge server. It is necessary to allow $\mathcal{S}1$ and $\mathcal{S}2$ to execute a secure comparison (SecComp) function to judge whether the input matrix is positive or not.

As shown in Fig. 3, SecComp consists of three sub-functions that provide unique and indispensable calculations. First, $\mathcal{T}$ randomly splits the input matrix into two equal-dimensional matrices and sends them to two non-collusive edge servers. Then the edge servers send the random matrix, generated by $\mathcal{T}$, to convert the input into the addition results of the two sets of complementary components matrices in SecBitExtra. The advantage of this operation is that both servers can calculate the symbol-bit matrices of input independently, without worrying about disclosing the sharing information they possess. Therefore, SecBitAdd is designed to realize secure carry addition calculation using binary bit. In the calculation process, the conversion from decimal to binary is redundant, because all servers will calculate the input as binary by default.

Obviously, a set of binary bits can be divided into three situations: 0 & 0, 0 & 1, and 1 & 1. Because the addition of 0 & 0 and 0 & 1 does not produce carry, the addition result can be directly obtained by the exclusive OR (XOR) calculation. However, the calculation of 1 & 1 can produce extra carry. There is no doubt that discarding carry will lead to wrong calculation results. Therefore, SecBitMul is necessary for providing all addition carries operated by the two edge servers. Similarly, the idea is to transfer the security parameters (multiplication triples) to realize secure bit by-wise multiplication. Subsequently, SecBitAdd calls SecBitMul iteratively so that all carries are handled in the bit-wise addition calculation process. The iteration loop does not terminate until all elements of the carry matrix are zero matrix. In this way, both servers can obtain the addition result component. By simple comparison, the two servers can extract the symbol-bit components of the input. Admittedly, if the XOR result of two symbol-bits is 1, it means the input is negative; otherwise, the input is positive. It is clear that the two edge servers do not need to know the complete input in the whole calculation process.

Implementation of P-CNN

In this section, we detail how the operations/calculations in different layers of VGG16 are implemented by the aforementioned secure functions.

For a CONV/FC layer, we divide any input matrix into two equal-dimensional matrices, and perform secure linear calculations on these matrices accordingly. In these layers, we use SecAdd/SecSub and SecSMul to ensure the security of all linear operations. To guarantee the same result as the original CONV/FC layer, we only need to delegate $\mathcal{S}_1$ to perform the specific calculation with bias and $\mathcal{S}_2$ to perform same calculation without bias.

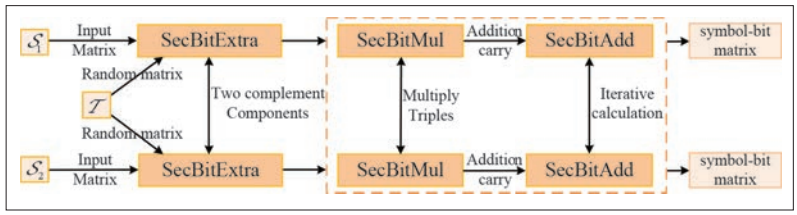

FIGURE 3. The details of SecComp function, including the SecBitMul, SecBitAdd, and SecBitExtra sub-functions, executed on two edge servers.

For the ACT layer, the ReLU function will set all elements in the input matrix to zero if they are less than zero; otherwise, the elements are kept unchanged. To implement the ReLU function, two edge servers interactively execute the SecComp function to determine the correct value of each element in the input matrix. As shown in Fig. 4a, $\mathcal{S}_1$ and $\mathcal{S}_2$ are able to determine the symbol of all elements in a plaintext matrix by exchanging the symbol-bits.

The secure execution of the POOL layer is illustrated in Fig. 4b. Here, $\mathcal{S}_1$ and $\mathcal{S}_2$ need only one round of interaction to achieve the maximum pooling securely. Specifically, $\mathcal{S}_1$ and $\mathcal{S}_2$ exchange the differences matrix between any two pooling regions by executing the SecSub function. Furthermore, by comparing the differences matrix, $\mathcal{S}_1$ and $\mathcal{S}_2$ find the maximum location of the whole pooling region and output the correct max-pooling result.

To integrate our secure functions into the VGG16 model, we download the model's parameter files and all training-related configuration files from the Model Zoo of Caffe's official website. Except for the last FC layer, we set the weights and bias parameters of all other layers to be 0, ensuring that the parameters of these layers are directly invoked in our training process. We only need to modify the last FC layer in the training profile to ensure that the original parameters of that layer are not loaded. In fact, we need to modify the parameter *num_output* to 2 in the last FC layer. Meanwhile, we fine-tune the initialized hyperparameter settings in solver files, based on the size of our datasets and the targeted classification precision.

Experiment and Results

The main objective of our experiments is to confirm whether the P-CNN framework produces the same classification results as the original CNN model. Meanwhile, we make comprehensive experiments to verify the practicability of applying the P-CNN framework on edge servers.

Correctness Experiment

First, we consider that the main factor affecting the performance of SecComp function is the bit-width l. This is because the iterations number of the addition loop in SecBitAdd is related to l. When l equals 8, 16, or 32, the error of SecComp can be ignored. Also, there is no error in theory for SecAdd/SecSub/SecSMul functions, which do not involve any approximation operation.

Furthermore, we randomly select an image from the testing dataset and compare the output

Focusing on object detection, we propose a privacy preserving raw data sharing framework for CAVs. The key innovation of our framework is the introduction of P-CNN model which processes encrypted images, shared from vehicles, to produce meaningful classification results, when they are combined.

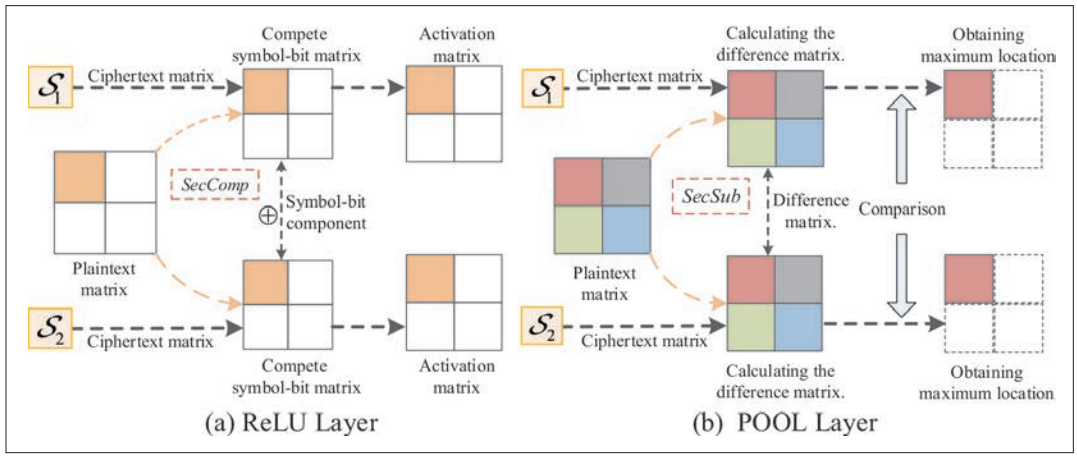

FIGURE 4. Illustrations of how ACT and MAX-POOL functions are implemented by using the SecComp and SecSub functions.

results from each layer in the VGG16 and P-CNN. As shown in Fig. 5a, the errors of CONV/ACT/POOL layers are on the level of 10^{-4}, that is, the outputs are almost the same as those in VGG16. Moreover, we observe that the errors of the penultimate FC layer are significantly reduced. This is important as the relatively larger errors in other layers (i.e., the CONV, ACT and POOL layers) are compensated for by the accurate calculations in the FC layer. The tiny errors (i.e., 10^{-6}) of the FC layer are negligible because the output results of VGG16 are usually integers, with different integers representing different classification results.

Overhead Experiment

Communication Overhead: All the images consist of 224 × 224 pixels, and each pixel is stored in 4 bytes. During the encryption phase, P-CNN just needs to split 750 images by generating a random Numpy matrix. The total message size is 143.55 MB and the message size of each image is 196.00 kB, as shown in Fig. 5b. If a homomorphic encryption scheme (i.e., CryptoNets [7]) is adopted, each pixel is encrypted as 5 polynomials, and each coefficient in the polynomial requires 24 bytes. Therefore, the message size of each image is 5.74 MB, and the total message size is as high as 4306.64 MB. Similarly, the communication overhead in the decryption phase of P-CNN is far lower than that of CryptoNets [7].

Computational Cost: The runtime of each layer in CNN is shown in Fig. 5c. For the CONV layer, the runtime of 28 × 28 input size and 5 convolution kernels in CryptoNets [7] is 30 s, and the runtime of 28 × 28 input size and 20 convolution kernels in CryptoDL [8] is 13.08 s. In contrast, the runtime of 224 × 224 input size and 64 convolution kernels in P-CNN is only 0.05 s. For three ACT layers (i.e., 20 × 28 × 28, 50 × 8 × 8, and 500 × 1), the runtime in CryptoDL is 9.76 s, and the runtime in Huang [12] is 1.13 s, while that in P-CNN is only 0.04 s. It is worth emphasizing that the runime with a Numpy matrix is 10 times faster than the embedded original SecComp. Also, the runtime of two POOL layers [12] is 1.72 s, while that in P-CNN is only 0.07 ms. Lastly, for the FC layer, the runtime from 100 to 5 in CryptoNets and CryptoDL is 1.60 s and 34.32 s, while that in P-CNN is only 0.02 s. Obviously, a homomorphic encryption scheme has no advantage in linear and nonlinear computation. Noted that, the layers embed proposed secure function is more time-saving than the scheme using the same addition secret sharing.

Impressively, the total runtime of P-CNN using VGG16 is 32.19 s. Compared to the previously designed functions which process elements one by one (i.e., 1173.29 s), the computational efficiency has improved 36.4 times. The experiment result shows that P-CNN, especially the secure functions, has great advantages and strong practicability.

Summary and Future Work

Focusing on object detection, we propose a privacy preserving raw data sharing framework for CAVs. The key innovation of our framework is the introduction of the P-CNN model, which processes encrypted images shared from vehicles to produce meaningful classification results when they are combined. In P-CNN, the operations in the original CNN model are implemented by our secure functions, based on the additive secret sharing technique. Experiment results demonstrate that P-CNN offers exactly the same classification results as the VGG16 model, but does not require vehicles to send raw image data that may contain private information. The main objective of this article is to explore the possibility of executing CNN models with encrypted images; however, many research challenges still need to be addressed before we can fully implement secure and privacy-preserving raw data sharing for CAVs. Open research issues that require substantial research efforts are summarized as follows.

Optimization of P-CNN

When the operations in VGG16 are implemented by our secure functions, we discover that the computational cost in P-CNN is mainly caused by the addition cycle in SecBitAdd. Because the minimum processing unit is a matrix, the reason for the termination of the addition cycle is often to reach the maximum iterations number rather than the carry matrix turning into a zero matrix. Therefore, it is urgent to design an adaptive cycle mechanism. Additionally, if a more complex network is adopted (e.g., ResNet50), there will be more stacked layers and more complex nonlinear functions involved. In this case, how to design fast

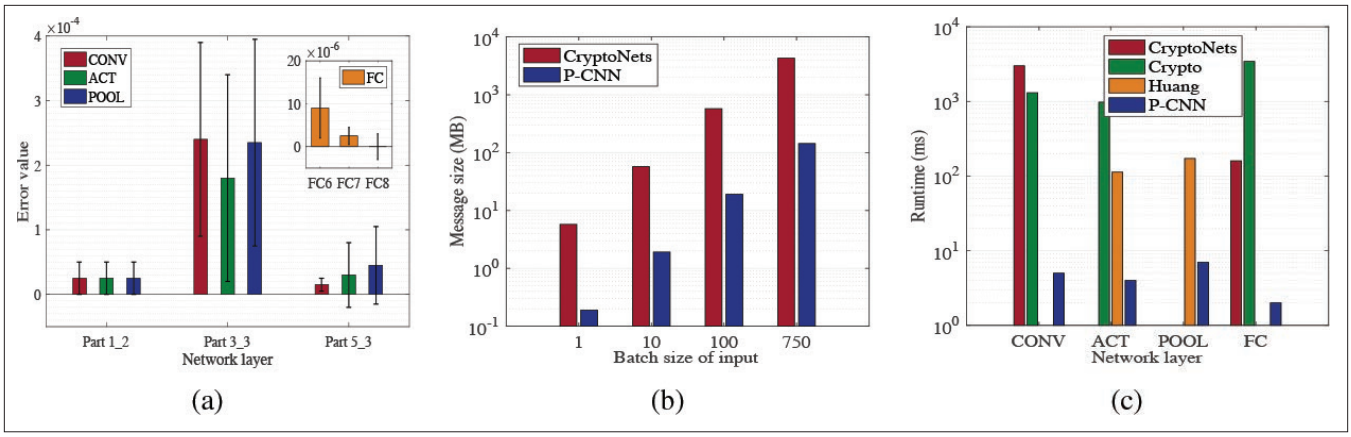

FIGURE 5. Experimental results: a) the error bar of each layer using the security functions in VGG16; b) the communication overhead in encryption phase (batch size from 1 to 750); c) the computational cost of each layer in VGG16.

but effective secure functions to implement these nonlinear operations becomes a big challenge problem.

Secure Object Detection

Besides object classification, object detection is also an important problem for autonomous vehicles. The goals of object detection on autonomous vehicles are to accurately find the location and size of objects within an image frame, as well as the object types (e.g., car, pedestrian, cyclist, or road sign). Existing solutions make use of a CNN plus a region proposal network (RPN) to finish the object detection task. Although P-CNN can be used to implement the CNN model here, it is not clear how to effectively re-implement the RPN by secure functions. Our P-CNN model considers only two types of object: vehicle and pedestrian. However, when more classes are considered, the structure of the last FC layers (of a CNN) becomes complicated, thus increasing computational cost.

Secure Data Fusion at the Edge

In practice, there will be more than one vehicle sending encrypted images to the edge servers to achieve privacy-preserving raw data sharing. The huge amount of images received on edge servers must be processed in real time; however, the computational overhead involved in P-CNN prohibits executing the model several times within a short period of time. Therefore, it would be beneficial to combine/fuse data from different vehicles to generate a mega image that contains all the information from individual images. As such, the classification results of P-CNN from the two edge servers would include all the objects identified by these contributing vehicles. Because only one message is broadcasted from the edge servers to nearby vehicles, the network traffic is significantly reduced. It is possible to achieve this goal because raw-data-level and feature-level data fusions for CAV have been studied in the literature [4, 15]. Similar approaches could be adopted to fuse encrypted images as well. Then the research challenges lie in the perfect alignment of pixels from different images, as well as the fusion (e.g., addition, average, or maxout) of pixels as they are in an encrypted format.

Secure Deep Learning

We propose a mechanism to implement the operations of VGG16 to convert it into a model that is capable of handling encrypted images. Further, the CNN model can be applied in radar data processing. The goal is to accurately estimate the real location of objects, and even to predict the speed, acceleration, landing point, and other information on the objects. Moreover, similar mechanisms can be designed from other types of deep learning models, such as the long short-term memory (LSTM) model, which possesses an artificial recurrent neural network (RNN) architecture. This model can be used to process and mine meaningful information of acoustic data, such as the maximum sound pressure level and transmission frequency characteristics, and then judge the specific location of the voice body. In fact, these data are all in the form of real numbers or integers, so it is reasonable to give full play to the idea of adding secret sharing for encryption.

Besides object classification for autonomous vehicles, secure deep learning can be realized from other applications of image classification. For example, in face recognition, secure functions can be defined to implement those used in DeepID, DeepFace, CNN-3DMM, and Faceness-Net. As face images contain very sensitive information, it is crucial to employ secure and privacy-preserving protection. For generic image classification, other models like LeNet, AlexNet, VGG19, GoogLeNet, and ResNet could also be converted to become a secure deep learning model. Research challenges will be how to design suitable security algorithms to approximately implement these used in the deep neural networks.

Acknowledgments

The authors would like to thank Dr. Ximeng Liu, Dr. Jie Lin, Dr. Biao Jin, and anonymous reviewers for their valuable comments. This work is supported in part by the National Natural Science Foundation of China (61872088, 61872090, U1905211, 61402109); and the Natural Science Foundation of Fujian Proince (2019J01276). Jinbo Xiong and Renwan Bi are the co-first authors and have made the same contribution.

References

[1] D. Rawat *et al.*, "Enhancing Vanet Performance by Joint Adaptation of Transmission Power and Contention Window Size," *IEEE Trans. Parallel and Distributed Systems*, vol. 22, no. 9, 2011, pp. 1528–35.

[2] Q. Yang *et al.*, "ACAR: Adaptive Connectivity Aware Routing for Vehicular Ad Hoc Networks in City Scenarios," *Mobile Networks and Applications*, vol. 15, no. 1, 2010, pp. 36–60.
[3] V. Va *et al.*, "Millimeter Wave Vehicular Communications: A Survey," *Foundations and Trends in Networking*, vol. 10, no. 1, 2016, pp. 1–113.
[4] Q. Chen *et al.*, "Cooper: Cooperative Perception for Connected Autonomous Vehicles Based on 3D Point Clouds," *Proc. 39th IEEE Int'l. Conf. Distributed Computing Systems*, 2019.
[5] R. Lu *et al.*, "ECpp, Efficient Conditional Privacy Preservation Protocol for Secure Vehicular Communications," *Proc. 27th 2009 IEEE INFOCOM*, 2009, pp. 1229–37.
[6] Z. Xiong *et al.*, "Privacy-Preserving Auto-Driving: A GAN-Based Approach to Protect Vehicular Camera Data," *Proc. 19th IEEE Int'l. Conf. Data Mining*, 2019.
[7] P. Xie *et al.*, "Crypto-Nets: Neural Networks Over Encrypted Data," arXiv preprint arXiv:1412.6181, 2014.
[8] E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep Neural Networks Over Encrypted Data," arXiv preprint arXiv:1711.05189, 2017.
[9] W. Shi *et al.*, "Edge Computing: Vision and Challenges," *IEEE Internet of Things J.*, vol. 3, no. 5, 2016, pp. 637–46.
[10] J. Ni, X. Lin, and X. S. Shen, "Toward Edge-Assisted Internet of Things: From Security and Efficiency Perspectives," *IEEE Network*, vol. 33, no. 2, Mar./Apr. 2019, pp. 50–57.
[11] I. Damgard *et al.*, "Unconditionally Secure Constantrounds Multi-Party Computation for Equality, Comparison, Bits and Exponentiation," *Proc. TCC*, vol. LNCS 3876. Springer, 2006, pp. 285–304.
[12] K. Huang *et al.*, "A Lightweight Privacy-Preserving CNN Feature Extraction Framework for Mobile Sensing," *IEEE Trans. Dependable and Secure Computing*, 2019.
[13] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556v6, 2015.
[14] A. Geiger *et al.*, "Vision Meets Robotics: The Kitti Dataset," *The Int'l J. Robotics Research*, vol. 32, no. 11, 2013, pp. 1231–37.
[15] Q. Chen *et al.*, "F-Cooper: Feature Based Cooperative Perception for Autonomous Vehicle Edge Computing System Using 3D Point," *Proc. Fourth ACM/IEEE Symp. Edge Computing*, 2019.

Biographies

Jinbo Xiong [M'13] (jbxiong@fjnu.edu.cn) received his Ph.D. degree in computer system architecture from Xidian University in 2013. He is a professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, and the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China. Currently, he is a visiting scholar with the Department of Computer Science and Engineering at the University of North Texas, Denton. His research interests include connected and autonomous vehicles, secure deep learning, mobile crowdsensing, privacy protection, and the Internet of Things.

Renwan Bi (brw2806@163.com) is currently pursuing an M.S. degree in software engineering in the College of Mathematics and Informatics, Fujian Normal University. His research interests include connected and autonomous vehicles, deep learning neural networks, secure multiparty computation and, cryptography security.

Mingfeng Zhao (zmf1900953654@163.com) is currently pursuing an M.S. degree in software engineering at the College of Mathematics and Informatics, Fujian Normal University. His research interests include mobile data security and privacy protection.

Jingda Guo (jingdaguo@my.unt.edu) is currently pursuing a Ph.D. degree in computer science from the Department of Computer Science and Engineering, University of North Texas. His research interests include artificial intelligence, computer vision, and autonomous driving.

Qing Yang [M'11,SM'17] (Qing.Yang@unt.edu) received his B.S. and M.S. degrees in computer science from Nankai University and Harbin Institute of Technology, China, in 2003 and 2005, respectively. He received his Ph.D degree in computer science from Auburn University in 2011. He is currently an assistant professor in the Department of Computer Science and Engineering at the University of North Texas. His research interests include the Internet of Things, trust models, and network security and privacy.