User-Defined Privacy Preserving Data Sharing for Connected Autonomous Vehicles Utilizing Edge Computing

Tianyu Bai, Qing Yang, Song Fu
Department of Computer Science and Engineering
University of North Texas, Denton, TX, USA
TianyuBai@my.unt.edu,Qing.Yang@unt.edu,Song.Fu@unt.edu

ABSTRACT

In this paper, we present PRECISE, a novel privacy preserving data sharing framework for connected autonomous vehicles (CAVs). PRECISE allows users to define the objects or parts that they wish to protect privacy before sharing data with other vehicles. It leverages secure segmentation and inpainting technologies to protect sensitive data of vehicles. PRECISE explores the edges to offload resource-intensive deep learning workloads. To ensure data privacy in the processing on edge, PRECISE leverages additive secret sharing theory to define secure functions for deep neural networks (DNNs). Two secure DNN models, Secure SegNet and Secure Context Encoder, are introduced, along with detailed explanations of how to develop secure CNN layers and the secure functions used in building these layers. We have implemented a prototype of PRECISE and evaluated its performance. The experimental results demonstrate that PRECISE is lightweight, achieving secure segmentation in 3.47 seconds and secure inpainting in 0.99 seconds. The inference outputs from PRECISE remain the same as those from the original DNNs, while data privacy is protected. To the best of our knowledge, PRECISE is the first of its kind to provide user-defined privacy protection for sensor data sharing among CAVs.

ACM Reference Format:
Tianyu Bai, Qing Yang, Song Fu. 2023. User-Defined Privacy Preserving Data Sharing for Connected Autonomous Vehicles Utilizing Edge Computing. In *The Eighth ACM/IEEE Symposium on Edge Computing (SEC '23), December 6–9, 2023, Wilmington, DE, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3583740.3628436

1 INTRODUCTION

Connect Autonomous Vehicle (CAV) technology has witnessed widespread adoption, owing to its benefits across various domains. CAVs are equipped with a diverse range of sensors including LiDAR, radar, cameras, GPS, etc., enabling them to collect and interpret a wealth of information. A key aspect of CAV technology is the ability to facilitate data sharing among vehicles. By engaging in collaborative data sharing, CAVs can effectively extend their sensing range and enhance the accuracy of their perception systems. This enables CAVs to exchange real-time information, encompassing critical factors such as traffic conditions, road hazards, and accidents. As a result, CAVs can leverage this shared data to make more informed decisions, leading to improved overall performance, safety, efficiency, and reliability in autonomous driving applications.

Existing approaches to data sharing between vehicles present certain limitations, either by sharing raw data without adequate privacy protection or by sharing encrypted data to other vehicles. Both approaches have their drawbacks. When raw data is shared, privacy protection becomes a major concern as sensitive information is exposed. On the other hand, when encrypted data is shared, only a few sensitive objects in the data may need protection, but the entire images (2D or 3D) are encrypted. This lack of selectivity makes it difficult to discern which specific objects contain privacy-sensitive information and wastes computing resources. The encryption and decryption of complete images in transit may not fully address data privacy concerns, as sensitive content could still be at risk of exposure to receiver vehicles. Given these challenges, there is a critical need for novel approaches that enable user-defined data sharing, allowing for the protection of sensitive objects within the collected sensor data while ensuring the necessary collaboration and information exchange among vehicles.

Deep neural networks (DNNs) offer an effective solution to the issue, leveraging their ability to process vast volumes of sensor data rapidly. We utilize DNNs to identify user-defined privacy-sensitive objects effectively, remove them from the sensor data, and subsequently reconstruct the modified data using inpainting techniques. This involves leveraging segmentation and inpainting DNN models for object identification and data inpainting. The segmentation DNN operates at the pixel level, enabling it to detect objects in raw data, including potentially sensitive objects. On the other hand, the inpainting DNN reconstructs the privacy removed data, resulting in a modified version of the data that is suitable for sharing. By integrating these two models, we allow users to define the classification of sensitive objects within the data and enable effective data sharing while safeguarding privacy in CAV systems.

The continuous execution of DNN models puts a significant strain on the computing resources of autonomous vehicles, including CPU and memory. This strain becomes particularly challenging when vehicles encounter a high volume of data within a limited timeframe. To prioritize critical functions like driving and ensure the operational reliability and safety of the vehicle, CAVs must carefully allocate their finite resources. However, the emergence of Edge computing technology [26] and the advancements in 5G wireless network transportation offer a promising solution. By offloading the computationally intensive DNN tasks to edge servers, CAVs can effectively alleviate their computational burden and leverage

SEC '23, December 6–9, 2023, Wilmington, DE, USA

ACM ISBN 979-8-4007-0123-8/23/12…$15.00
https://doi.org/10.1145/3583740.3628436

the benefits of edge computing. This approach enables distributed processing, facilitating efficient data analysis and task execution while preserving the essential resources of the CAV itself.

However, edge servers cannot always be trusted as they have the potential to be compromised in certain scenarios. Thus, sending raw data to edge servers raises concerns about data privacy and security. Traditional data encryption methods offer protection only during data transmission, leaving the stored data vulnerable to unauthorized access or breaches at the edge servers. While technologies like homomorphic encryption [14, 17, 28, 29, 37] can enable secure cipher text processing, they often suffer from long latency, making them impractical for real-time applications.

Existing Multi-party computation (MPC) techniques [4, 5, 34] primarily protect privacy during sensor data processing. However, in certain scenarios, it is beneficial to share the original image while ensuring the removal of sensitive content. For example, in situations where edge devices need to evaluate road conditions or gather weather information for an area, preserving privacy while granting access to the original image can be highly valuable. There is a clear need for innovative approaches that offer fine-grained control over data privacy, enabling secure data sharing in a customizable manner, and has satisfied performance for real world applications.

In this paper, we present PRECISE (Privacy Preserving Data Sharing with Segmentation and Inpainting for CAV), a novel approach that leverages the power of encoder and decoder DNN models to address the challenges of privacy preservation in data sharing for CAVs. By harnessing the exceptional performance of these models in image segmentation and inpainting tasks, PRECISE enables CAVs to identify sensitive objects within collected sensor data and construct inpainted images where the sensitive objects are removed. These privacy preserving inpainted images are then securely transmitted to receivers, facilitating privacy preserved data sharing in both CAV-only and CAV-Edge collaboration scenarios.

To mitigate the risk of private data leakage to edge servers, PRECISE incorporates additive secret sharing, a well-established theory, to enhance the security of the context encoder and decoder DNN models. This enhancement enables the secure DNN models to process ciphertext, ensuring the privacy of the sensitive information. During the execution of PRECISE on edge servers, each server generates a partial segmentation and inpainted image result. These results are then transmitted to the receivers, who can combine and retrieve the privacy-removed data from the distributed outputs of the edge servers. By employing deep learning, edge computing, and secret sharing technologies, PRECISE provides an effective solution for user-defined privacy preserving data sharing among vehicles.

The main contributions of this paper are as follows.

- We present PRECISE, a privacy-preserving data sharing framework for connected autonomous vehicles. We describe the design details, showcasing the integration of SegNet and Context Encoder DNN models, known for their performance in segmentation and inpainting tasks.
- We demonstrate how DNN models in PRECISE can be enhanced with secure functions, ensuring privacy preservation on edge servers. One of the key components of PRECISE is the adoption of additive secret sharing based secure functions within the secure layers. We delve into the details of these secure functions, explaining their roles in preserving privacy during data processing on edge and sharing among vehicles.
- We have conducted a comprehensive series of experiments to assess the effectiveness and performance of PRECISE using a vehicle-edge test platform. In-vehicle operations were carried out on AStuff Spectra, while on-edge operations were performed on edge servers equipped with AMD Ryzen 7 processors. On the sender vehicle, we measured the time required to create secret shares from sensor data and encrypt those shares. On the receiver vehicle, we evaluated the time needed to combine outputs from edge servers and generate shared sensor data with sensitive objects removed. On the edge servers, we compared the execution times of individual layers in a deep neural network (DNN) - specifically, the layers with secure functions versus their original non-secure versions - and assessed the overall time and storage complexity.

2 BACKGROUND

2.1 Image Segmentation

Image segmentation [10] is a fundamental task in computer vision that plays a crucial role in various applications. It involves clustering pixels belonging to the same object in an image, providing a fine-grained understanding of the scene. Unlike object detection, which focuses on identifying objects using bounding boxes or region proposals, image segmentation operates at the pixel level, precisely delineating the location, boundaries, and classification of objects in images.

Image segmentation offers numerous advantages, and the field has witnessed significant advancements through the utilization of deep learning techniques, such as [9, 18, 36, 40, 41]. Firstly, it enables scene understanding by providing pixel-level object boundaries, facilitating high-level interpretation of images. This fine-grained localization and classification of objects are valuable for various computer vision tasks, including autonomous driving, object recognition, and medical image analysis.

Moreover, image segmentation enhances the performance of downstream tasks by enabling more precise object localization and reducing ambiguity. For example, in autonomous driving, accurate segmentation helps identify lane markings, traffic signs, and pedestrians, enabling safer and more reliable decision-making by autonomous vehicles. Additionally, segmentation results can serve as a crucial preprocessing step for higher-level vision tasks, such as object tracking, instance segmentation, and semantic understanding. It provides a semantic map of the scene, allowing subsequent algorithms to reason about objects, their interactions, and their contextual relationships.

2.2 Image Impainting

Image inpainting [6] is an essential task in computer vision that focuses on reconstructing damaged or missing pixels within an image. In the past, traditional approaches in computer vision relied on techniques such as texture synthesis and patch synthesis to repair damaged images. For instance, methods like Navier-Stokes

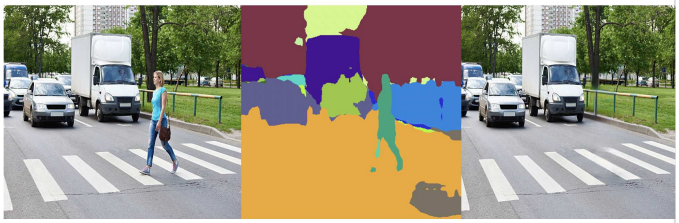

Figure 1: Images with secure segmentation and secure inpainting.

[11] and Fast marching [39] have been commonly used in this context.

In recent years, there has been a growing interest in leveraging deep learning approaches for image inpainting tasks [24, 25, 42, 43]. This surge in research has resulted in the development of various image inpainting convolutional neural networks (CNNs), including notable examples such as GLCIC [2], Patch-based Image Inpainting with GANs [13], and Deep Learning-based Copy-and-Paste [21]. Among these approaches, the Context Encoder [32] has gained particular prominence as an exceptional image inpainting CNN.

The advancements in deep learning-based image inpainting have opened up new possibilities in various applications, including photo restoration, image editing, and video processing. These techniques offer efficient and effective solutions for repairing damaged images and filling in missing regions, significantly improving the visual quality and usability of the inpainted images.

3 SYSTEM ARCHITECTURE

In an illustrating scenario (see Figure 1), a vehicle captures an image that includes various elements, such as buildings in the background, a pedestrian, and cars on the road. This information is of significant value to other CAVs within the infrastructure as it provides crucial insights into the current road conditions. Vehicle 1 recognizes the importance of sharing the captured image with other vehicles; however, it also acknowledges the presence of sensitive information, such as human faces or house numbers, within the image. To ensure privacy, Vehicle 1 aims to selectively protect this sensitive content and prevent its disclosure during data sharing. In this section, we will explain how the PRESICE framework can empower Vehicle 1 to define and safeguard the privacy of specific content within the captured image through the privacy preserved data sharing process.

3.1 PRECISE Architecture and Execution Flow

In a vehicle-only scenario, where Vehicle 1 possesses ample computational resources, Vehicle 1 takes the initiative to define the classification of sensitive objects, such as pedestrians, within the captured image. Subsequently, Vehicle 1 transmits the classified information to the PRECISE framework. Within Vehicle 1's local PRECISE module, an image segmentation deep neural network (DNN) is employed to detect objects present in the image. Based on the detection results, if any sensitive objects are identified, their pixel values are effectively removed. Subsequently, the PRECISE impainting CNN model generates a reconstructed image that excludes the sensitive data, which is then forwarded to the intended

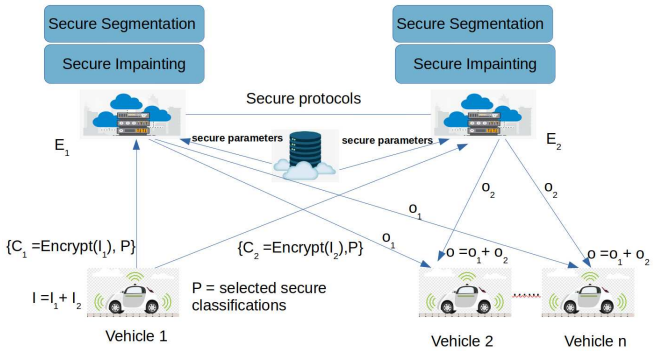

Figure 2: Image segmentation and image inpainting with privacy protection.

receivers. Through this process, Vehicle 1 is able to customize the privacy preservation of sensitive objects and securely share the modified image with other vehicles.

In the vehicle-edge platform, the PRECISE framework becomes more intricate as it necessitates protecting data privacy during edge server processing. The architecture of the PRECISE framework in the vehicle-edge platform comprises various components: the data owner (Vehicle 1, responsible for capturing the image), the receivers (other interested CAVs), the edge server (responsible for performing secure image segmentation and inpainting), and the trusted server (providing secure protocols and key generation). This architecture, as illustrated in Fig. 2, ensures the privacy and confidentiality of the image data at every stage of processing and transmission. It establishes robust data protection within the framework, ensuring that sensitive information remains secure throughout the entire process.

PRECISE consists of the following key components.

Data Owner ($Vehicle1$): This represents the vehicle that captures and owns the image data.

Receivers (other vehicles): These connected autonomous vehicles (CAVs) are interested in the captured image. Their objective is to utilize the shared data from $Vehicle1$ to gather information about the surrounding environment.

Edge Servers (E_1, E_2): the edge server is a vital component of the PRECISE framework, performing essential functions. It receives the input data from CAV_1 and utilizes two secure deep learning neural networks: secure segmentation and secure inpainting. Then the edge server transmits the processed output to the intended receivers for further use.

Trusted Server T: The trusted server in the PRECISE framework assumes the responsibility of generating encryption and decryption keys for data security. Additionally, it generates a set of random bit arrays that are utilized by the secure protocol implemented in the deep learning models of the edge server. This random bit arrays play a crucial role in maintaining the security and privacy of the data throughout the computational process.

During the secure data sharing process, the following steps occur:

The edge server transmits a set of object classes G to the data owner $Vehicle1$. G contains commonly found object classes in street views.

$Vehicle1$ takes the initiative to select a specific class of objects, denoted as P, from the captured image that it considers private.

This selection process is similar to the vehicle-only scenario, where Vehicle 1 defines the classification of sensitive objects.

The captured image I is then randomly split into two secret shares, denoted as I1 and I2, following the I_2: $I = I_1 + I_2$.

$Vehicle1$ employs a predetermined data encryption scheme, such as AES256 or RSA, along with encryption keys acquired from the trusted server T, to encrypt the secret shares into ciphertext C_1 and C_2. CAV1 then transmits these encrypted shares to their respective edge servers E_1 and E_2.

Upon receiving the encrypted shares and data from CAV1, each edge server applies the decryption key obtained from the trusted server T to decrypt the shares and retrieve the secret share as well as the selected privacy object class P.

The edge servers perform secure segmentation on the decrypted shares and obtain partial segmentation results. These partial results are exchanged between the edge servers to identify the location of pixels belonging to the privacy object class P. If no pixels are found in P, the edge server directly sends the partial segmentation output and the secure share to the intended receivers.

If there are pixels belonging to the privacy object class P, the edge server removes the corresponding pixel values from the secret share. The privacy-removed secret share is then processed using secure inpainting, which generates a partial inpainting result.

Finally, edge servers send the partial inpainting results to the receivers. The receivers can combine the partial outputs received from the edge servers to reconstruct a privacy-removed image that was captured by $Vehicle1$.

The secure segmentation and inpainting flow guarantees the preservation of selected objects' privacy while facilitating the sharing of $Vehicle1$'s image data with other entities. Through encryption, secure processing, and distributed computation, PRECISES achieves customized privacy-preserving data sharing among CAVs.

3.2 Attack Model

In our framework, the edge servers (E_1 and E_2) are classified as "Curious But Honest" entities. This implies that although they carry out their computational tasks diligently, there is a possibility of them attempting to explore user information. In other words, an edge server may analyze the input image with malicious intent to extract details regarding the user's privacy. This behavior could involve examining specific patterns or features in the image to uncover sensitive information, even if it hasn't been explicitly shared or identified as private. Such actions would violate the user's privacy and compromise the confidentiality of their data.

Moreover, in scenarios where multiple edge servers collaborate, there is an increased risk of privacy breaches. These edge servers have the potential to collude and share information obtained from various stages of the computation process. Through the sharing of intermediate results or exchanging knowledge about the data, they could collectively deduce the private information contained within the image. This collusion among edge servers poses a significant threat to privacy and can compromise the confidentiality of the user's data.

To mitigate the aforementioned concerns, the PRECISE framework incorporates secure segmentation and secure inpainting convolutional neural networks (CNNs) that rely on additive secret

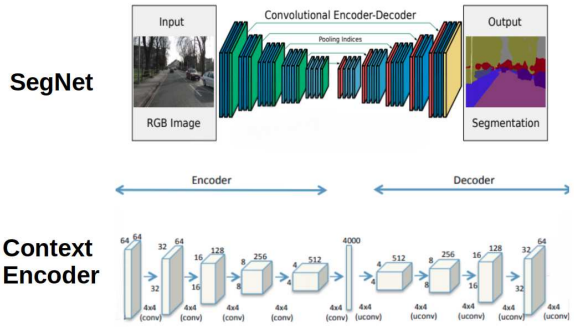

Figure 3: SegNet and Context Encoder Decoder.

sharing-based secure functions. These functions guarantee the protection of the image data and private information during computation. PRECISE also allows the CAV to execute some CNN layers locally and send only the encrypted feature map secrets to the edge servers. This approach limits the collaboration and inference among the edge servers, enhancing privacy protection. The HBC scenario emphasizes the need for robust privacy-preserving mechanisms, including secure computation and secure protocols, to safeguard sensitive information against privacy breaches by curious but honest edge servers.

4 PRIVACY PROTECTION FOR IMAGE SEGMENTATION AND INPAINTING DNNS

4.1 Segmentation and Impainting DNNs

One prominent deep learning approach for image segmentation is SegNet[3]. SegNet's architecture (Figure 3top)is built upon the popular VGG16 model, employing a similar topology but excluding the fully connected layers. The SegNet encoders comprise 13 convolutional layers, 13 leaky ReLU activations, 13 batch normalization layers, and 5 max pooling layers. Conversely, the decoders consist of 13 secure transposed convolutional layers, 13 secure ReLU activations, 13 batch normalization layers, and 5 max unpooling layers.

This design choice results in a lightweight encoder that improves training efficiency, making it well-suited for real-time and resource-constrained applications. By leveraging the encoder-decoder structure, SegNet enables efficient and accurate segmentation by capturing and reconstructing detailed spatial information.

The Context Encoder [32] CNN (Figure 3 bottom) comprises an encoder and a decoder component. Encoder consists of 5 convolutional layers, 4 batch normalization and 5 leaky RELU. Decoder consists of 5 transpose Convolutional layers, 4 batch normalization, and 4 leaky RELU. The role of the encoder is to transform the input image into a condensed latent feature representation. This latent representation captures crucial information about the image, enabling the model to grasp the content and context of the input. The decoder, on the other hand, takes this latent representation as input and generates the missing or damaged pixels to complete the inpainting process. Also one channel-wise fully-connected layer are injected to connect encoder and decoder.

By harnessing the power of deep learning techniques, Context Encoder can generate inpainted images that exhibit enhanced realism and visual appeal. The model has the ability to learn intricate

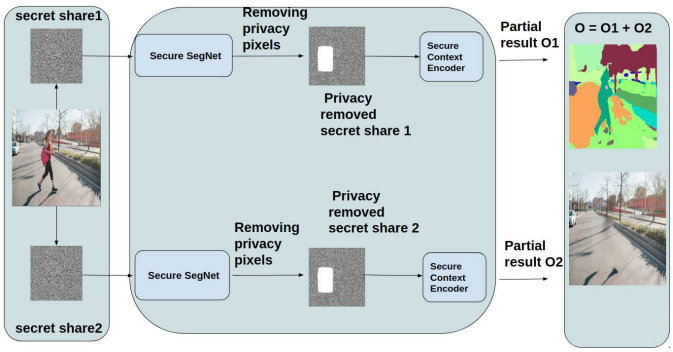

Figure 4: Key components in PRECISE. The left box represents the data owner vehicle, the middle box denotes edge servers, and the right box represents the receiver vehicle.

patterns and structures from large datasets, enabling them to produce coherent and semantically meaningful image inpainting.

The selection of SegNet and Context Encoder as the base models for image segmentation and inpainting in the PRECISE framework is driven by several factors:

High Accuracy: Both SegNet and Context Encoder have demonstrated good accuracy in their respective tasks. They achieve this by employing the encoder-decoder pattern, which allows them to capture detailed spatial information and reconstruct accurate outputs. The accurate segmentation and inpainting results are crucial for preserving privacy while maintaining the quality and integrity of the shared images.

Lightweight Architecture: SegNet and Context Encoder are designed with a lightweight architecture, making them suitable for real-time applications and resource-constrained environments. The efficient encoder-decoder structure enables faster training and inference times, ensuring that the privacy-preserving operations can be performed efficiently even on devices with limited computational resources.

4.2 DNN Models in PRECISE

In PRECISE, we utilize four privacy preserving DNN models, comprising two SegNet models and two Context Encoder models. The two secure SegNet models and two secure Context Encoder models are identical in structure. One set of secure SegNet and secure Context Encoder is deployed on one edge server, while the other set is deployed on a separate edge server. These secure segmentation and inpainting models maintain the same architecture as the original SegNet and Context Encoder, including the same types, number, and sequence of CNN layers. However, each CNN layer is enhanced with secure functions, transforming them into secure CNN layers capable of processing secret shared values instead of the original inputs. In the subsequent section, we will provide a detailed explanation of each secure CNN layer, accompanied by its respective implementation methodology.

Figure 4 shows the processing flow of secure SegNet and secure Context Encoder models for an image containing sensitive data. The image, denoted as F, is captured by a Connected Autonomous Vehicle (CAV), and the data owner has specified that the privacy classification is pedestrian. The data owner does not want this information to be accessed by other parties, while other non-sensitive information in the image can be shared with other CAVs within the infrastructure.

To protect the privacy of the pedestrian information, Image F is randomly split into two secret shares, f_1 and f_2. The secure SegNet models on the edge servers, E_1 and E_2, process the respective secret shares independently. They exchange partial segmentation results, $sec_seg(f_1)$ and $sec_seg(f_2)$, and combine them to retrieve the overall segmentation result, $seg(f) = sec_seg(f_1) + sec_seg(f_2)$.

Next, E_1 and E_2 scan the segmentation result, $seg(f)$, to identify the locations of pixels belonging to humans. They mark these locations and remove the corresponding values from secret shares.

The privacy-removed secret shares are then sent to the secure Context Encoder. E_1 and E_2 generate inpainting outputs, o_1 and o_2, respectively, as well as the segmentation result, $seg(f)$. These results are sent to the receivers.

The receivers can simply combine the inpainting outputs, $o = o_1 + o_2$, to recover the privacy-removed image, which no longer contains the sensitive pedestrian information.

5 USER-DEFINED PRIVACY PROTECTION FOR SEGMENTATION AND IMPAINTING DNNS

The PRECISE framework incorporates two deep learning models: secure Segmentation and secure Inpainting, both of which rely on CNN layers as their fundamental building blocks. To ensure accurate computations using ciphertext as input, secure versions of SegNet and Context Encoder, known as secure SegNet and secure Context Encoder, are employed. In PRECISE, the additive secret sharing relationship in the input data necessitates the independent implementation of each secure CNN layer. Consequently, for a given secure CNN layer on edge server E_i, it takes input f_i and produces output o_i. The overall input f and output o of the secure CNN layers can be expressed as the summation of their respective components, $f = \sum_{i=1}^{n} f_i$ and $o = \sum_{i=1}^{n} o_i$. This formulation ensures that the input and output of the secure CNN layers remain equivalent to those of traditional CNN layers.

The design of secure CNN layers for secure object segmentation and secure image inpainting is a crucial and intricate aspect of the PRECISE framework. The primary objective is to develop secure layers that yield identical output to their traditional CNN layer counterparts. This necessitates the utilization of secure functions to execute additive secret sharing schema-based CNN linear and non-linear computations.

The CNN layers utilized in the SegNet and Context Encoder models can be classified into two groups based on their computation type: linear computation layers and non-linear computation layers. Linear computation layers encompass the convolutional layer, transpose convolutional layer, and fully connected layer, and do not require any modifications. On the other hand, the non-linear computation layers, including leaky ReLU, max pooling, batch normalization, and max unpooling layers, necessitate the implementation of specially designed secure CNN layers. In the subsequent paragraphs, we will provide a comprehensive explanation of each individual secure CNN layer, highlighting their correspondence to their counterparts in the original CNN model.

5.1 Secret Sharing

Secret sharing [35] is a widely used cryptographic scheme in modern privacy preservation applications. It serves as a key-less method to protect the privacy of data. The scheme revolves around two primary operations: dividing and combining.

The dividing function is responsible for generating secret shares from the original secret. It takes the secret as input and produces a set of shares: $d(s) = s_1, s_2, ..., s_n$. Each participant in the scheme holds one of these shares. The combining function, in the case of additive secret sharing, employs addition as the operation. It defines how the secret can be reconstructed from the secret shares: $s = s_1+s_2+...+s_n$. To recover the original secret, a minimum required number of secret shares must be collected and provided to the combining function. The original secret can only be reconstructed when the minimum required number of shares is gathered and combined. In additive secret sharing, this minimum number is equal to the total number of participants involved in the scheme. Secret sharing theory has been extensively utilized in research to construct privacy-preserving Convolutional Neural Networks (CNNs) [1, 16, 23, 27, 37].

Secure functions play a vital role in the PRECISE CNN models, as they adhere to the additive secret sharing philosophy. These secure functions are the building blocks of the PRECISE framework, particularly in the design of secure CNN layers for object segmentation and image inpainting. The primary objective of these secure layers is to produce the same output as their traditional CNN counterparts. To achieve this, secure functions are utilized to implement the linear and non-linear computations of CNN layers within the additive secret sharing scheme. This ensures the privacy preservation of data while maintaining the desired functionality of the CNN models.

5.2 Secure Leaky ReLU

Leaky ReLU derives from ReLU, a classic activation layer in CNN. Among various neurons in deep learning neural networks, not all neurons are needed to be activated and involved in the computation. An activation layer decides whether a neuron will be activated or not. Leaky ReLU enforces negative feature values to be replaced with a predefined small coefficient to multiply with the feature values while positive feature values remain the same.

Algorithm 1 Secure Leaky ReLU

Input: feature map secret share $F_i, N_i, B_i, a, t_i, y_i, u_i$
Output: F's leaky ReLU result on F_i

Edge server receives secure parameters$N_i, B_i, a, t_i, y_i, u_i$ from trusted server
for each feature f in feature map secret share F_i **do**
    $s_i = \text{SSE}(F_i, N_i, B_i, a, t_i, y_i, u_i)$
    Edge servers exchange s_i to compute $s = s_1 \bigoplus s_2 .. \bigoplus s_n$
    if $s == 1$ **then**
        return $negativeslope * f$
    end if
    return f
end for

In the PRECISE framework, the secure Leaky ReLU operation is defined in Algorithm 1. While the computation of Leaky ReLU is straightforward in a regular CNN, in PRECISE, the input of secure Leaky ReLU must be the secret share of the raw data. To ensure the confidentiality of the raw data during computation, a secure symbol extraction (SSE) technique is introduced. SSE generates a partial result s_i on each edge server, and by exchanging s_i and computing $s = s_1 \bigoplus s_2 .. \bigoplus s_n$, the first symbol digit of the bitwise representation of the raw data can be obtained without leaking the actual data value. If the symbol digit is equal to one, indicating a negative value, the secret share is multiplied by a predefined negative slope coefficient. This ensures that the secure Leaky ReLU operation preserves the privacy of the raw data while performing the required computations.

5.2.1 Secure Symbol Extraction. Secure symbol extraction plays a crucial role in the secure RELU layer of the PRECISE framework. Its purpose is to obtain the symbol digit of the input feature value without revealing the actual feature value itself. Secure symbol extraction serves as the foundation for the secure RELU layer, which replaces negative feature values with zero and activates neurons corresponding to positive values.

To perform secure symbol extraction, each edge server utilizes the secret share of the feature map, denoted as F_i, along with additional secure bit arrays from the trusted server. Since the feature value is divided among the edge servers using additive secret sharing, the original feature value can be represented as $F = \sum_{i=1}^{n} F_i$. The objective of the secure symbol extraction function is to return a boolean value based on whether F is greater than zero or not. This enables the secure RELU layer to make decision on activating or deactivating neurons while preserving the privacy of the original feature values.

In Algorithm 2, the trusted server first generates three sets of random bit arrays: N, B, and V. These arrays have a length of L, which corresponds to the bit-wise length of the feature value. Each set contains n random values, where n is the total number of secret shares. The random values in N and B are correlated in the following manner: $N_1 \bigoplus N_2 .. \bigoplus N_n = \sum_{i=1}^{n} B_i$. The edge servers involved in the computation are indexed from 1 to n, representing the total number of secret shares. The trusted server randomly selects an index value, denoted as a, to designate a specific edge server for a different computation schema. This selected edge server will generate bit arrays for the subsequent secure operations using this alternate schema.

The remaining edge servers (those not selected) compute c_i by subtracting B_i from their corresponding secret share F_i and generate a random value X_i. The edge servers then send the values X, c to the selected server, which computes X_i by performing bitwise addition on all the received X, c values, as well as numeric addition. As a result, each edge server produces its own X_i, and the feature map value F is divided into two parts: the generated bit arrays X, N.

The randomness present in the X, N arrays ensures that even identical feature values can produce different intermediate secure parameters. This characteristic enhances PRECISE's resistance to cipher text-based attacks and strengthens its overall security.

Algorithm 2 Secure Symbol Extraction

1: Input: Feature map secret share F_i, random value N_i, B_i, index number a, random value t_i, y_i, u_i
2: Output: F partial symbol value
3:
4: **if** index of server is not equals to a **then**
5: compute $c_i = F_i - B_i$ and generate a random bit array X_i, then forward to edge server with index a
6: **else**
7: compute $X_i = (c_1 + c_2 .. + c_n) \bigoplus X_1 \bigoplus X_2 .. \bigoplus X_(n - 1)$
8: **end if**
9: **if** index of server is not equals to a **then**
10: o_i = secure multiplication$(X_i, N_i, t_i, y_i, u_i, true)$
11: **else**
12: o_i = secure multiplication$(X_i, N_i, t_i, y_i, u_i, false)$
13: **end if**
14: edge servers collaborate compute $o = \sum_{i=1}^{n} o_i$
15: $s_i = X_i \bigoplus N_i$
16: **while** $o_i \neq 0$ **do**
17: computes $s_i = s_i \bigoplus o_i$
18: **if** index of server is not equals to a **then**
19: o_i = secure multiplication$(s_i, o_i, t_i, y_i, u_i, true)$
20: **else**
21: o_i = secure multiplication$(s_i, o_i, t_i, y_i, u_i, false)$
22: **end if**
23: **end while**
24: **if** $s_i < 0$ **then**
25: return 1
26: **end if**
27: return 0

The SSE is designed to retrieve the symbol digit value of F using intermediate parameters X, N without exposing F to any parties. Let's examine the relationship between F and X, c. We can observe that X and c are the results of bit-wise addition $(\bigoplus) of \mathrm{X}_i$ and c_i respectively. The relationship between F and X, c can be expressed as follows: $F = F_1 + F_2 + \cdots + F_n = (c_1 + B_1) + (c_2 + B_2) + \cdots + (c_n + B_n)$ Here, $\sum_{i=1}^{n} c_i = X_1 \bigoplus X_2 \bigoplus \cdots \bigoplus X_n$ and $\sum_{i=1}^{n} B_i = N_1 \bigoplus N_2 \bigoplus \cdots \bigoplus N_n$. Consequently, we have: $F = (X_1 \bigoplus X_2 \bigoplus \cdots \bigoplus X_n) + (N_1 \bigoplus N_2 \bigoplus \cdots \bigoplus N_n)$.

Each edge server E_i possesses a pair of arrays X_i and N_i, enabling them to efficiently compute $s_i = X_i \bigoplus N_i$. However, due to security concerns, the secure protocol prevents the exchange of N_i between edge servers. This precaution is necessary to thwart potential attacks where an adversary eavesdrops on the network and combines X with N to recover the original feature value. Therefore, the SSE aims to compute F solely based on bit-wise operations, allowing each edge server to independently compute the result without the need for exchanging N_i.

Indeed, the carry digit value plays a crucial role in the Secure Symbol Extraction process. Bitwise addition (XOR operation) does not consider carry values from previous locations, so the correct carry value must be determined and added to each digit before moving to a higher index position.

To replace the numeric addition $X + N$ with bit-wise addition $X \bigoplus N$, we need to compute the correct carry values between the combined bit arrays X and N. The carry value for X and N can be obtained through the expression $X \bigotimes N$.

When performing the bit-wise addition in the Secure Symbol Extraction algorithm, it is necessary to maintain the updated carry value for each iteration (lines 16 to 23) if the carry value is not empty. This ensures the correctness of the Secure Symbol Extraction process at every digit's location and guarantees the accurate retrieval of the symbol digit without revealing the original feature value.

In the Secure Multiplication, a secure protocol is employed to enable the exchange of transformed bit arrays X_i and N_i without compromising privacy. The approach is inspired by the work of Damgård et al [12]. The algorithm takes two sets of input data. The first set consists of H_i, G_i, which are the two bit arrays obtained from the Secure Symbol Extraction algorithm, representing the computed carry value $H \bigotimes G$. The second set includes random values t_i, y_i, u_i generated by the trusted server. These random values are injected into the input arrays H_i and G_i to introduce data randomization and prevent the leakage of input array privacy during the exchange of intermediate computation parameters.

By leveraging this secure protocol, the PRECISE framework ensures that the transformed bit arrays X_i and N_i can be exchanged between edge servers without revealing any sensitive information. This allows for the computation of $X \bigotimes N$ while maintaining the confidentiality of the original feature values and protecting against potential privacy breaches.

5.3 Secure Max Pooling

The max pooling layer serves two key purposes in a deep learning neural network: first, it identifies the most significant feature values within a pooling region, and second, it reduces the magnitude of a given feature map by eliminating redundant features. In traditional max pooling, a straightforward maximum operation is used to select the maximum feature. In the context of secure max pooling 3, a two-dimensional array is constructed to store the pairwise differences between values in the flattened pooling region f for each secret share F_i on the respective edge servers. Specifically, the array t_i (created on the i_{th} edge server) with indices x and y represents the value $f[x] - f[y]$. The edge servers then exchange and combine the difference arrays, resulting in the combined array $t = t_1 + t_2 + \ldots + t_n$. For the q_{th} row in t, the values represent the differences between $f[q]$ and all other feature values. If all these differences are positive, it indicates that $f[q]$ is the maximum element within the pooling region. It is worth noting that the array t_i captures the differences between the secret share's values (F_i) on edge server i, while the combined array t represents the differences between the actual feature values F.

5.4 Secure Batch Normalization

Batch normalization [20] is a technique used to normalize the output of previous CNN layers by subtracting the mean value and dividing by the standard deviation of the batch. It enables independent learning of each CNN layer and accelerates the training process. The mean of the batch values, denoted as $E(F)$, can be easily computed by adding the means of the two secret shares: $E(F) = \frac{\sum_{i=1}^{n} F_i}{n} = \frac{\sum_{i=1}^{n} F_{1i}}{n} + \frac{\sum_{i=1}^{n} F_{2i}}{n}$. However, directly combining

Algorithm 3 Secure Max Pooling

1: Input: feature map secret share F_i
2: Output: partial maximum feature values
3:
4: **for** each pooling region f in F_i **do**
5: Create t_i with a size $[w(f) * h(f)]^2$
6: Flatten pooling region feature values into single dimension
7: **for** each feature value in f **do**
8: array $t_i[x][y] = f[x] - f[y]$ $x, y \in [0, [w(f) * h(f)]^2]$
9: **end for**
10: E_i exchanges t_i with other edge servers to compute $t = \sum_{i=1}^{i=n} t_i$
11: **for** each row with index q in t **do**
12: **if** all values in the row are greater than 0 **then**
13: return $f[q]$
14: break;
15: **end if**
16: **end for**
17: **end for**

Algorithm 4 Secure Batch Normalization

1: Edge server E_i, $i \in 1, 2$
2: Input: feature map secret share F_i
3: Output: F batch normalization
4:
5: E_i compute mean $c_i = \frac{\sum_{i=1}^{n} f_i}{n}$, where n is total number of element in F_i
6: Edge server collaborate compute $c = c_1 + c_2$
7: E_2 computes $v_2 = \frac{\sum_{i=1}^{n} (f_1 - c)^2}{n}$
8: E_1, E_2 flatten F_1, F_2
9: **for** for element with in F_1 with index m; $m < n$; $m = m + 2$ **do**
10: $R = F_1[m] \div F_1[m + 1]$
11: **end for**
12: E_1 forward R to E_2
13: E_2 create array d with random number with size $\frac{n}{2}$
14: **for** for element with in F_2 with index j; $j < n$; $j = j + 2$ **do**
15: $u[j] = F_2[j] + d[\frac{n}{2}]$
16: $u[j + 1] = F_2[j + 1] + (-d[\frac{n}{2}]) * R[\frac{n}{2}]$
17: **end for**
18: E_2 sends u to E_1
19: E_1 create $z = 0$
20: **for** each element in F_1, u with index l **do**
21: $z = z + 2 * F_1[l] * u[l]$
22: **end for**
23: E_1 computes $v_1 = \sum_{i=1}^{n} (f_2 - c)^2 - n * c^2 + z$
24: E_1, E_2 collaborate compute $V = \sqrt{\frac{(v_1 + v + 2)}{n}}$
25: return $\frac{F_i - c}{V}$

the results of the edge servers does not provide the batch variation. To address this, edge server E_1 computes $var(F_1)$ and E_2 computes $var(F_2)$. The difference between $var(F)$ and $var(F_1) + var(F_2)$ is given by $2(F_1 * F_2) - (nE(F))^2$. In algorithm 4, lines 9-23 aim to compensate for this difference while preserving the secret sharing principle. Intermediate parameters u are created and exchanged between the two servers. u is derived from F_2 by adding random numbers based on the ratio in F_1. The random value d is eliminated when combining $2F_1[l] * u[l] + 2F_1[l + 1] * u[l + 1]$. Additionally, E_1 and E_2 exchange v_1 and v_2 to compute the batch normalization.

5.5 Secure Max Unpooling

Max unpooling is a technique in CNNs that recovers the original input size from a downsampled feature map. In secure max unpooling, the indices of the maximum secret share values obtained during secure max pooling are preserved and utilized to accurately reconstruct the original positions of the pooled values. This ensures that the spatial information is preserved throughout the process.

6 PERFORMANCE EVALUATION

We have implemented a prototype of the PRECISE. To evaluate the performance of privacy-preserving perception networks using PRECISE, we conducted experiments on a vehicle-edge testbed. The testbed includes a Polaris GEM e4 electric vehicle equipped with Sekonix cameras, a Velodyne LiDAR, a Delphi radar, GPS, and IMU sensors. The on-vehicle processing unit utilized is the AStuff Spectra. Each edge server in the testbed is equipped with an AMD Ryzen 7 processor with 6 cores running at 3.2 GHz, 16 GB DRAM, and operates on Ubuntu Linux v20.04 and Python v3.8.

For our experimental evaluation, we employed the widely-used COCO dataset [22], which is commonly used for object detection tasks. The COCO dataset contains 118,000 images spanning over 80 object categories. We selected a subset of 1,200 images from the training dataset. Within this subset, we used over 50 distinct categories, with a specific focus on commonly encountered roadside objects, such as pedestrians, vehicles, and traffic lights. The images were preprocessed by cropping into a size of 224x224 pixels with 3 channels and applying padding as needed.

In the training phase, the models were trained for 120 epochs, with a batch size of 32 and a learning rate of 0.01. These training parameters were carefully selected to ensure effective learning and convergence of the models. The trained models were tested on a vehicle equipped with cameras. Evaluation of the models was conducted using both captured images and videos. The testing scenarios encompassed a diverse range of environments, including residential areas, local roads, intersections, and highways. More specifically, our dataset consists of 50 high-resolution videos, each with an average duration of 10 minutes. We extracted 500 frames from these videos. For each video, we identified object classes that were considered privacy-sensitive, including humans, street name signs, and houses. These object classes were then encoded into metadata tags, with a unique ID and a description of the sensitive objects.

6.1 Perception Accuracy

To access inference accuracy of the PRECISE framework, we conducted an evaluation on a wide range of object classifications in the COCO dataset, comprising over 40 categories. The segmentation results generated by PRECISE demonstrate precise and accurate outputs, as depicted in Figure 5. Additionally, we are comparing the

Figure 5: User-defined privacy protection by PRECISE.

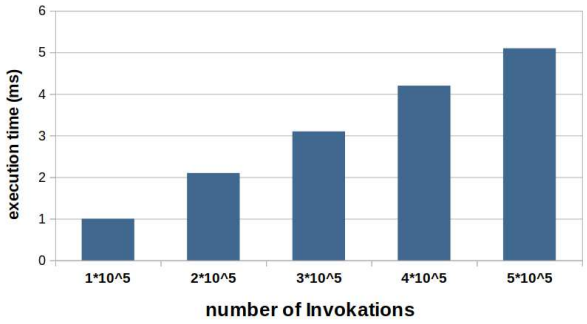

Figure 6: Execution time of secure bit multiplication.

difference between the original CNN model and PRECISE secure CNN model. The difference is in scale of e^{-6} which is negligible to impact the inference result.

6.2 Security

The threat model introduced in section III emphasizes two significant risks: first, the edge server is Honest But Curious, so executing segmentation and impainting with raw image will leak the privacy of CAV captured data.

Second, an attacker is assumed to be capable of effectively monitoring the network transmission data or even further able to read data on edge servers. PRECISE sends encrypted secret shares to edge servers instead of raw images. The randomness in secret share enhances PRECISE's resistance to cryptography attacks, for example, chosen ciphertext attacks and eavesdropping attacks. Additionally, additive secret sharing-based secure operation schema does not rely on key-based encryption but splits and hides plain text information into secret shares. Thus PRECISE does not suffer from key management risks.

In vehicular edge computing infrastructure, edge servers could be compromised simultaneously and collaborate to detect private information. PRECISE could increase its privacy protection resistance by letting CAV conduct partial segmentation CNN layers. In such a way, CAV produces secret shares of feature maps instead of the raw captured image. Even if edge servers collaborate to combine secret shares from CAV, only feature map data could be revealed. Thus, the privacy of raw image data remains protected.

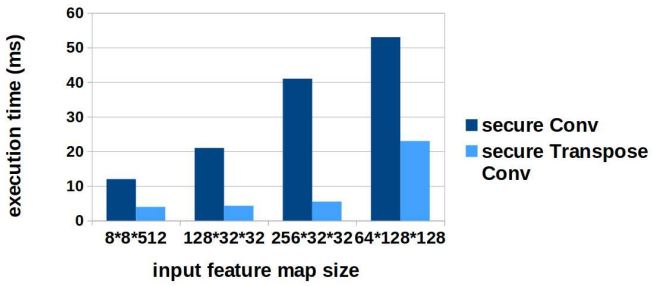

Figure 7: Execution time of secure convolutional layer and secure transpose convolutional layer.

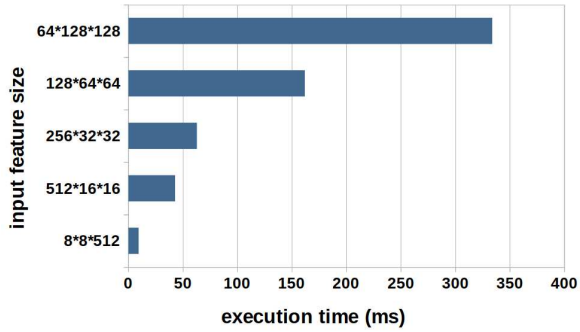

Figure 8: Execution Time secure Leaky ReLU.

6.3 Execution Time

In the context of secure bit multiplication, the execution time is depicted in Figure 6. The fundamental computations involved in secure bit multiplication primarily revolve around the secure manipulation of parameters through bitwise addition and bitwise multiplication. Notably, bitwise operations demonstrate a relatively swift execution, thereby mitigating the substantial growth in overall latency as the number of invocations escalates. This characteristic highlights the efficiency and scalability of secure bit multiplication, allowing for seamless processing even in scenarios involving numerous invocations by Secure Symbol Extraction.

The relationship between input feature map size and execution time in secure leaky ReLU is illustrated in Figure 8. The primary computation involved in secure leaky ReLU is secure symbol extraction, which internally relies on secure bit multiplication. As the size of the input feature map increases, the execution time of secure leaky ReLU exhibits an exponential growth pattern. However, it is worth noting that the model structure remains fixed, and the input image size is defined as 128x128x3 (height, width, and channel). Consequently, the overall execution latency remains predictable and feasible for real-time inference. It is important to consider that, while the attainment of a higher level of security comes at a cost, the extended computation latency of secure leaky ReLU still allows for timely inference.

Due to the additive nature of secret sharing in the PRECISE framework, secure convolutional layers and secure transpose convolutional layers exhibit identical computational characteristics to their original counterparts in models such as SegNet and Context Encoder. Figure 7 presents the execution time comparison between convolutional layers and transpose convolutional layers. It is worth

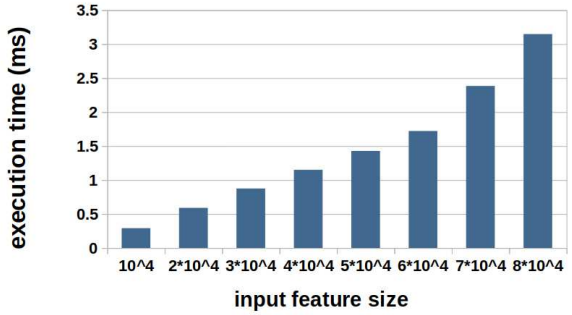

Figure 9: Execution time of secure Max pooling.

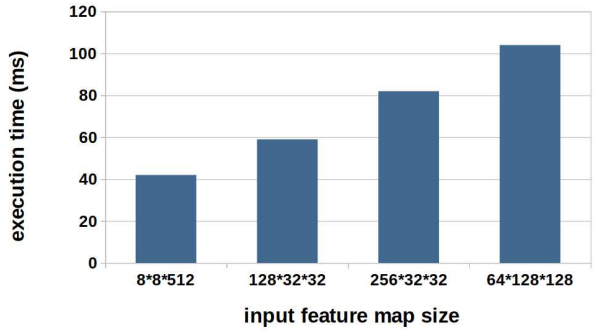

Figure 10: Execution time of secure batch normalization.

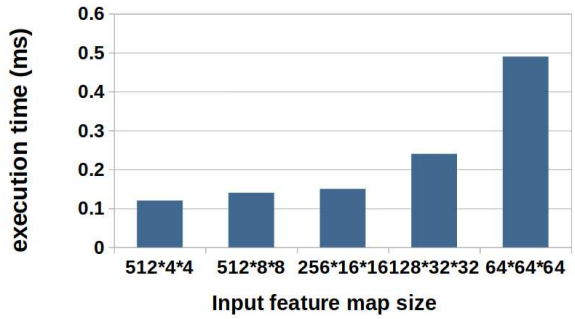

Figure 11: Execution time of secure Max unpooling.

noting that convolutional layers are the primary contributors to performance latency in the original CNN model.

Additionally, Figure 9, 11, and 10 demonstrate the execution time of secure maxpooling, secure max unpooling, and secure batch normalization, respectively. Secure maxpooling exhibits linear execution time growth as the input feature map size increases, which can be attributed to the iterative computation implementation of the secure function. On the other hand, secure max unpooling demonstrates minimal execution time due to the utilization of saved indexes from secure maxpooling, where only zero value filling within the stride is necessary. Secure batch normalization involves multiple computation steps but does not require looping, resulting in relatively small computation time complexity.

The comparison of inference times between the original CNN models and the PRECISE secure CNN models is depicted in Figure 12 and 13. The experiments were conducted solely on CPU due to the presence of sequential computation logic and complex bitwise computations within certain secure functions, which may result

	SegNet		secure SegNet
convolutional and transpose convolutional	0.54s	secure convolutional and secure transpose convolutional	0.57s
leaky ReLU	0.017s	secure leaky ReLU	1.9s
max pooling	0.021s	secure max pooling	0.12s
max unpooling	0.0002s	secure max unpooling	0.0002s
batch normalization	0.0031s	secure batch normalization	0.884s
total	0.6092s	total	3.4742s

Figure 12: Comparison of execution time between Secure SegNet and the original SegNet.

	Context Encoder		secure Context Encoder
convolutional and transpose convolutional	0.034 s	secure convolutional and secure transpose convolutional	0.034s
leaky ReLU	0.004s	secure leaky ReLU	0.747s
batch normalization	0.0058s	secure batch normalization	0.21s
total	0.0438s	total	0.991s

Figure 13: Comparison of execution time between secure Context Encoder and the original Context Encoder.

in slower performance if GPU utilization is attempted without algorithm optimization.

Among the various secure CNN layers, secure Leaky ReLU stands out as the dominant factor contributing to execution time latency. In secure SegNet, secure Leaky ReLU accounts for 1.9 seconds, representing 54% of the total execution time, while in secure Context Encoder, it amounts to 0.747 seconds, comprising 75.3% of the total execution time. This can be attributed to the intricate secure computation logic involved in secure symbol extraction, which necessitates bitwise computations, looping over outputs from secure bit multiplication, and secure parameter generation.

The total execution time for secure SegNet is 3.4742 seconds, surpassing that of secure Context Encoder at 0.991 seconds. This discrepancy can be attributed to the more complex and deeper neural network architecture employed in secure SegNet. With 13 convolutional layers and 13 transpose convolutional layers, as well as larger intermediate feature map sizes, secure SegNet offers a greater number and variety of CNN layers compared to secure Context Encoder, which includes 6 convolutional layers and 5 transpose convolutional layers. Additionally, secure SegNet incorporates maxpooling and max unpooling layers, which are absent in the context encoder model.

Due to the limited existing research on secure data sharing incorporating privacy-preserved segmentation and inpainting, we conducted a comparative analysis of PRECISE's secure segmentation and inpainting functionalities against previous works, as

	Object Removal time
Edge mask	1.16 s
Flores et al.	31.6s
Nodari et al.	21.4s
PRECISE	0.991s

	Secure Segmentation
HSI	46s
BUNET	1078s
PRECISE	3.47s

Figure 14: Top: Performance comparison between PRECISE and existing object removal approaches: Flores [15], Nodari [30], and Edge Mask [38]. Bottom: Performance comparison between PRECISE and secure object segmentation approaches: HSI [7], and BUNET [8].

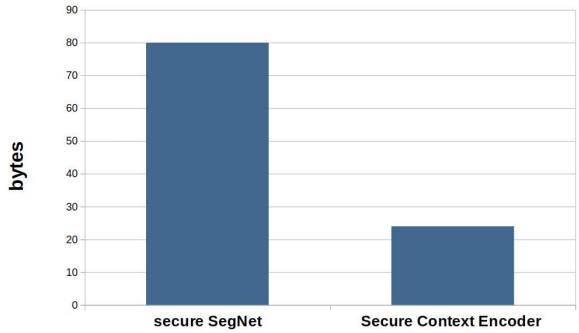

Figure 15: Storage overhead from data produced by secure functions (in bytes).

illustrated in Figure 14. The top table presents the results, highlighting the effectiveness and efficiency of PRECISE's inpainting technique. It successfully removes sensitive objects from images and achieves faster execution times compared to the works of Flores [15] and Nodari [30]. While edge mask achieves the shortest execution time, it directly processes on the raw image, whereas PRECISE operates exclusively on cipher text, ensuring enhanced privacy preservation.

The bottom table provides a performance comparison of PRECISE's secure segmentation approach with HSI [7] and BUNET [8]. It is evident that PRECISE exhibits significantly shorter execution times compared to the other methods. While it is important to note that the execution time difference may not solely reflect efficiency, considering that HSI and BUNET are based on more complex CNN segmentation models, PRECISE remains a more practical choice for secure data sharing in CAV edge infrastructure.

One of the costs associated with achieving a high level of security through the adoption of secure protocols in PRECISE is the generation of additional data by secure functions, which must be stored in memory and transmitted over the network. Figure 15 illustrates the amount of extra data in bytes produced by secure SegNet and secure Context Encoder. On average, the generation and encryption of secret shares take 0.66 seconds, and the combining of privacy-removed images from edge servers on the receiver vehicle requires 0.002 seconds. Additionally, the network data transmission times for secure SegNet and secure Context Encoder are measured at 2.4 seconds and 1.7 seconds, respectively.

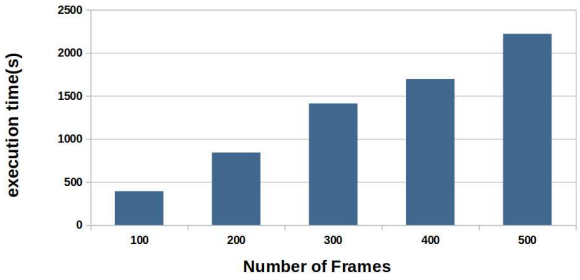

Figure 16: Execution time of Secure SegNet on videos.

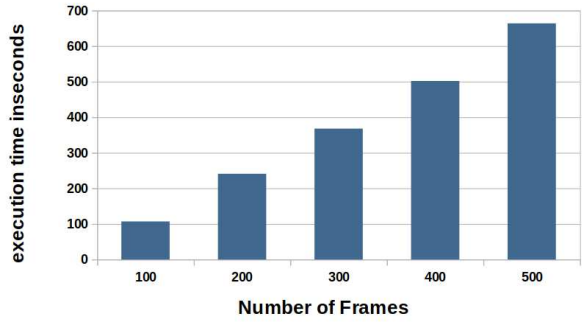

Figure 17: Execution time of Secure Context Encoder on videos.

6.4 Performance Evaluation on Videos

In addition to individual images, PRECISE can be applied to videos. Figures 16 and 17 present the performance of PRECISE processing a stream of video frames (30 FPS). Frames were continually sent to PRECISE and we measured the average execution time in seconds. The figures show that on average the Secure SegNet processed at 4.12 seconds per video frame, which is 16% longer than that for an individual image. Similarly, the average execution time for a video frame by the Secure Context Encoder is 1.21 seconds, i.e., 19% slower than that for an individual image. Figure 18 plots the size of intermediate data generated in processing videos. The extended processing time primarily results from processor and memory contention. This contention arises during the continuous processing of frame data because each frame necessitates transformation into secret shares and ongoing processing by secure DNNs. The additional encoding and decoding of video also contributes to an increase in the overall execution time. It's worth noting that video compression, cannot significantly reduce transmission time, as the data transferred between the CAV and edge servers comprises secret shares of video frames.

6.5 Performance on More Edge Servers

In the preceding discussion, we have explained the use cases of PRECISE on two edge servers. PRECISE can be extended to accommodate additional edge servers, denoted as N (where $N > 2$), to further bolster privacy protection in data sharing among vehicles. However, this may also cause additional processing time and performance latency. For example, secure multiplication (Algorithm 3) needs to calculate bitwise carries from more secret shares. Consequently, this impacts the performance of secure symbol extraction (Algorithm 2) which invokes secure multiplication multiple times.

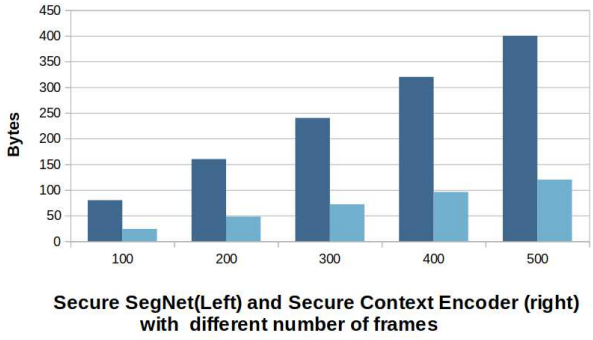

Figure 18: Storage overhead from data produced by secure functions in processing videos.

From our experiments, we observed that as the number of edge servers increased from 2 to 3, the execution time of secure multiplication was extended by an average of 1.43 times and the secure symbol extraction was prolonged by 1.66 times. Additionally, including one more secret share (edge server) resulted in an increase of 78 bytes on average for the generated secret share data. It is critical to find a suitable trade-off between the enhanced level of privacy and the increased latency in real-world scenarios.

7 RELATED WORK

Blind Medical Image Segmentation Based on Secure UNET (BUNET) [8] presents a novel framework for preserving privacy during medical image segmentation. The framework utilizes a secure variant of the UNET architecture to safeguard sensitive medical data. By leveraging secure multi-party computation techniques, BUNET ensures that the image data remains encrypted throughout the segmentation process, preventing unauthorized access to confidential medical information. The proposed approach enables accurate segmentation results while maintaining the privacy and confidentiality of the underlying data.

HSI [7] also achieves secure segmentation but it utilizes a hybrid trusted execution environment (TEE). The TEE combines the benefits of hardware-based and software-based approaches, offering a secure and confidential processing environment for sensitive medical data. Both papers focus on privacy-preserving medical image segmentation; however, the execution times reported in the studies raise concerns regarding their feasibility in CAV edge infrastructure. BUNET demonstrates a prolonged execution time of 1078 seconds, while HSI takes 46 seconds to complete. These long execution times are impractical for real-time applications. In contrast, PRECISE offers a lightweight solution with significantly faster performance. It achieves a speedup of 310 times compared to BUNET and 13.2 times compared to HSI in single-frame segmentation inference, making it highly suitable for real-time deployment in CAV edge environments.

EdgeMask [38] is a system that enables privacy-preserving video data sharing using edge computing. It uses edge devices to process and analyze video locally, detecting and masking sensitive information like faces and license plates. The system distributes the workload among edge devices to ensure efficient processing and low latency. Users can define privacy preferences and specify regions of interest or objects to be preserved or masked. EdgeMask is a system that excels in fast real-time privacy-preserving video processing. Its architecture allows for efficient inference performance, outperforming many existing solutions. However, PRECISE stands out in two key aspects. Firstly, EdgeMask exposes raw data on edge servers, whereas PRECISE only exposes encrypted data, enhancing data security. Secondly, PRECISE enables data owners to define privacy object classifications, catering to diverse real-world applications where different users may have specific privacy requirements in various scenarios.

Visor [33] addressed the privacy issue in video analytics by leveraging trusted execution environments (TEE) and employing data-oblivious primitives and communication protocols. Osia et al. [31] introduced a hybrid user-cloud framework using a feature extractor to remove sensitive object features. They explored the Siamese architecture and privacy measures. PRECISE complements these works by supporting secret shares for secure segmentation on the edges and removing sensitive objects defined by users. Close to the approach in [19], PRECISE allows a vehicle to offload the processing workload, including resource-intensive convolution layers, to the edge.

8 CONCLUSION

This paper presents a Customized Secure Image Segmentation and Inpainting (PRECISE) framework, which addresses the privacy challenges associated with data sharing in connected autonomous vehicles (CAVs). The primary goal of this research is to facilitate data sharing among vehicles and protect data privacy, with the goal of extending their sensing capabilities and improving the accuracy of their perception systems. Once sensitive objects are removed from the sender vehicle's sensor data, sharing the segmentation results becomes crucial for the receiver vehicle to accurately detect objects for safety. While the segmentation results do not contain the pixel values from the original images, thereby enhancing privacy, it is important to note that they may still retain specific information, such as the shapes of background and objects. This particular type of information is essential for autonomous driving purposes. PRECISE enables users to define the types of objects that should be excluded prior to data sharing with other vehicles. Presently, the object specification operates at the class level, rather than on an individual object basis. For example, a user can designate the "houses" class as sensitive, and PRECISE will then remove all instances of houses while retaining other object types in an image. Our future plans for PRECISE involve extending its capability to identify and protect individual objects within a class for enhanced privacy protection at the object level. It is worth noting that the design of secure functions and layers in DNNs within PRECISE is generic and can be applied to other network architectures that handle text or audio data processing.

ACKNOWLEDGEMENT

This work has been supported in part by the U.S. NSF grants CNS-2231519, CNS-2113805, CNS-1852134, OAC-2017564, ECCS-2010332, CNS-2037982, DUE-2225229, and CNS-1828105. We thank the reviewers for their constructive comments and our shepherd for the valuable feedback on this paper.

REFERENCES

[1] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. 2017. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE transactions on information forensics and security* 13, 5 (2017), 1333–1345.

[2] Karim Armanious, Youssef Mecky, Sergios Gatidis, and Bin Yang. 2019. Adversarial inpainting of medical image modalities. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 3267–3271.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence* 39, 12 (2017), 2481–2495.

[4] Tianyu Bai, Song Fu, and Qing Yang. 2022. Privacy-Preserving Object Detection with Secure Convolutional Neural Networks for Vehicular Edge Computing. *Future Internet* 14, 11 (2022), 316.

[5] Tianyu Bai, Danyang Shao, Ying He, Song Fu, and Qing Yang. 2023. P3: A Privacy-Preserving Perception Framework for Building Vehicle-Edge Perception Networks Protecting Data Privacy. In *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 1–10.

[6] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. 2000. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 417–424.

[7] Song Bian, Weiwen Jiang, and Takashi Sato. 2021. Privacy-Preserving Medical Image Segmentation via Hybrid Trusted Execution Environment. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1347–1350.

[8] Song Bian, Xiaowei Xu, Weiwen Jiang, Yiyu Shi, and Takashi Sato. 2020. BUNET: blind medical image segmentation based on secure UNET. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part II 23*. Springer, 612–622.

[9] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2017), 834–848.

[10] Guy Barrett Coleman and Harry C Andrews. 1979. Image segmentation by clustering. *Proc. IEEE* 67, 5 (1979), 773–785.

[11] Peter Constantin and Ciprian Foias. 2020. *Navier-stokes equations*. University of Chicago Press.

[12] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. 2006. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Theory of Cryptography Conference*. Springer, 285–304.

[13] Ugur Demir and Gozde Unal. 2018. Patch-based image inpainting with generative adversarial networks. *arXiv preprint arXiv:1803.07422* (2018).

[14] Haokun Fang and Quan Qian. 2021. Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet* 13, 4 (2021), 94.

[15] Arturo Flores and Serge Belongie. 2010. Removing pedestrians from google street view images. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. IEEE, 53–58.

[16] David Froelicher, Juan R Troncoso-Pastoriza, Apostolos Pyrgelis, Sinem Sav, Joao Sa Sousa, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. 2020. Scalable privacy-preserving distributed learning. *arXiv preprint arXiv:2005.09532* (2020).

[17] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*. PMLR, 201–210.

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.

[19] Zecheng He, Tianwei Zhang, and Ruby B Lee. 2020. Attacking and protecting data privacy in edge–cloud collaborative inference systems. *IEEE Internet of Things Journal* 8, 12 (2020), 9706–9716.

[20] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. pmlr, 448–456.

[21] Sungho Lee, Seoung Wug Oh, DaeYeun Won, and Seon Joo Kim. 2019. Copy-and-paste networks for deep video inpainting. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4413–4421.

[22] Tsung-Yi Lin, Michael Maire, et al. 2014. Microsoft COCO: Common objects in context. In *ECCV*.

[23] Yehida Lindell. 2005. Secure multiparty computation for privacy preserving data mining. In *Encyclopedia of Data Warehousing and Mining*. IGI global, 1005–1009.

[24] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. 2018. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European conference on computer vision (ECCV)*. 85–100.

[25] Guilin Liu, Kevin J Shih, Ting-Chun Wang, Fitsum A Reda, Karan Sapra, Zhiding Yu, Andrew Tao, and Bryan Catanzaro. 2018. Partial convolution based padding. *arXiv preprint arXiv:1811.11718* (2018).

[26] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. 2019. Edge computing for autonomous driving: Opportunities and challenges. *Proc. IEEE* 107, 8 (2019), 1697–1716.

[27] Lingjuan Lyu, Xuanli He, Yee Wei Law, and Marimuthu Palaniswami. 2017. Privacy-preserving collaborative deep learning with application to human activity recognition. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1219–1228.

[28] Payman Mohassel and Peter Rindal. 2018. ABY3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 35–52.

[29] Payman Mohassel and Yupeng Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE SP*.

[30] Angelo Nodari, Marco Vanetti, and Ignazio Gallo. 2012. Digital privacy: Replacing pedestrians from google street view images. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2889–2893.

[31] Seyed Ali Osia, Ali Shahin Shamsabadi, Sina Sajadmanesh, Ali Taheri, Kleomenis Katevas, Hamid R Rabiee, Nicholas D Lane, and Hamed Haddadi. 2020. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet of Things Journal* 7, 5 (2020), 4505–4518.

[32] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. 2016. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2536–2544.

[33] Rishabh Poddar, Ganesh Ananthanarayanan, Srinath Setty, Stavros Volos, and Raluca Ada Popa. 2020. Visor: Privacy-Preserving Video Analytics as a Cloud Service. In *Proceedings of USENIX Security Symposium (USENIX Security)*.

[34] Manoj M Prabhakaran and Amit Sahai. 2013. *Secure multi-party computation*. Vol. 10. IOS press.

[35] Ronald L Rivest, Adi Shamir, and Leonard Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 234–241.

[37] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1310–1321.

[38] Samira Taghavi and Weisong Shi. 2020. EdgeMask: An edge-based privacy preserving service for video data sharing. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 382–387.

[39] Alexandru Telea. 2004. An image inpainting technique based on the fast marching method. *Journal of graphics tools* 9, 1 (2004), 23–34.

[40] Guotai Wang, Maria A Zuluaga, Wenqi Li, Rosalind Pratt, Premal A Patel, Michael Aertsen, Tom Doel, Anna L David, Jan Deprest, Sébastien Ourselin, et al. 2018. DeepIGeoS: a deep interactive geodesic framework for medical image segmentation. *IEEE transactions on pattern analysis and machine intelligence* 41, 7 (2018), 1559–1572.

[41] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. 2018. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*. 325–341.

[42] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2018. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5505–5514.

[43] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2019. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4471–4480.