Privacy-Preserving Driver Monitoring on the Edges: Transformer-Based Processing of Secret Shares from Video Streams

TIANYU BAI, University of North Texas, Denton, United States
DANYANG SHAO, Department of Biological Sciences, University of North Texas, Denton, United States
YING HE, Department of Computer Science and Engineering, University of North Texas, Denton, United States
QING YANG, Department of Computer Science and Engineering, University of North Texas, Denton, United States
YUNHE FENG, Department of Computer Science and Engineering, University of North Texas, Denton, United States
SONG FU*, Department of Computer Science and Engineering, University of North Texas, Denton, United States

Modern vehicles increasingly rely on advanced driver monitoring systems (DMS) to ensure safety and enhance the driving experience. These systems assess driver status to prevent accidents caused by fatigue, inattentiveness, or intoxication. While some DMS applications process video data on vehicle, many rely on edge or cloud-based solutions, raising significant privacy concerns due to the storage of sensor data from vehicles. Existing approaches, such as de-identification and homomorphic encryption, either impose heavy computational overhead on vehicles or insufficiently address privacy.

To overcome these limitations, we present the Privacy-preserving Driver Monitoring System (PDMS), a novel framework based on the additive secret sharing theory and privacy-preserving Transformer-based deep learning models. PDMS creates randomized secret shares from driver's facial video data on vehicle, processes them independently through privacy-preserving Transformer models on edges, and securely aggregates partial results on vehicle, ensuring vehicles' sensor data and final results remain protected. This approach reduces the computational load on the vehicle, enabling cost-effective and scalable DMS solutions that protect the privacy of the driver both in transit and in processing.

Our contributions include the design and optimization of the PDMS system, incorporating privacy-preserving DNN layers that are capable of processing randomized secret shares. Furthermore, we present a practical system that utilizes a vision transformer (ViT)-based gaze estimation model, demonstrating the effectiveness of PDMS through comprehensive experiments.

Additional Key Words and Phrases: Vehicle-edge computing, Privacy protection, Driver monitoring systems, Transformers.

Authors' Contact Information: Tianyu Bai, University of North Texas, Denton, Texas, United States; e-mail: tianyubai@my.unt.edu; Danyang Shao, Department of Biological Sciences, University of North Texas, Denton, Texas, United States; e-mail: danyangshao@my.unt.edu; Ying He, Department of Computer Science and Engineering, University of North Texas, Denton, Texas, United States; e-mail: yinghe@my.unt.edu; Qing Yang, Department of Computer Science and Engineering, University of North Texas, Denton, Texas, United States; e-mail: Qing.Yang@unt.edu; Yunhe Feng, Department of Computer Science and Engineering, University of North Texas, Denton, Texas, United States; e-mail: yunhe.feng@unt.edu; Song Fu, Department of Computer Science and Engineering, University of North Texas, Denton, Texas, United States; e-mail: song.fu@unt.edu.

ACM 2577-6207/2025/12-ART
https://doi.org/10.1145/3777384

1 Introduction

Modern vehicles increasingly combine autonomous driving technologies with human interaction, placing a significant emphasis on driver status for both personal and public safety. Monitoring the driver's condition is essential not only to prevent accidents, but also to enhance the overall driving experience. Advanced vehicle applications aim to learn from and adapt to the driver's status during operation, further improving comfort and safety.

The driver's status plays an essential role in ensuring safety during driving. Conditions such as fatigue, intoxication, or inattention can cause serious accidents. To address this, automotive industries have integrated internal cameras or driver recorders into vehicles to capture in-cabin information. Applications like Guardian used by General Motors and Smart Eye's Driver Monitoring System process driver's video frames in a vehicle. Local processing is crucial for preserving privacy, as it avoids transmitting and processing driver's facial video data on third-party servers.

A key challenge for on-vehicle driver monitoring systems (DMS) is the limited computational power and battery capacity available on the vehicle, which is primarily allocated to driving-related operations. As advanced network transmission technologies like 5G [7, 23] continue to evolve, cloud or edge servers gain ever-increasing computational power, and deep learning models grow in both complexity and size. Vehicles can leverage these external resources for driver monitoring tasks while preserving driver privacy.

Many applications, such as Amazon's Delivery Van Monitoring and Lytx DriveCam, rely on cloud and/or edge platforms to analyze driver's video data. These applications/systems upload video data to third-party servers to perform driving habit analysis, which faces significant privacy concerns. Transmitting raw video data to servers risks exposing sensitive personal information during transportation and processing.

Technologies such as de-identification [25] aim to mitigate privacy risks by preprocessing raw video data, replacing personally identifiable information (PII) [32], such as facial features with anonymized values. While effective in theory, the best practice for de-identification requires it to be conducted before transmitting data to servers. This introduces additional computational overhead for vehicles and increases processing latency. Homomorphic encryption [13], a potential solution, enables computations directly on encrypted data without decryption. However, the computational complexity of homomorphic encryption's polynomial arithmetic and the large size of ciphertexts make it impractical for latency-sensitive and resource-limited vehicles.

To address these challenges, in this paper, we present a novel Privacy-preserving Driver Monitoring System (PDMS). PDMS explores the additive secret sharing theory to design privacy-preserving Transformer-based DMS models, which process randomized secret shares on edges. Specially, PDMS generates multiple secret shares for driver's facial video from a vehicle. Each secret share is processed independently by a privacy-preserving Transformer model on an edge. PDMS securely aggregates results from the edges, ensuring that the driver's facial video and the final results are protected throughout the entire process.

PDMS reduces the computational burden on the vehicle, making it scalable and easy to deploy while saving computing and battery resources for mission-critical perception, prediction, and planning tasks. At the same time, it protects the privacy of driver's video data during processing, providing a robust solution to the challenges faced by existing DMS systems. PDMS offers a balance between efficiency, privacy, and security, paving the way for safer and more intelligent vehicle systems.

While the core secure inference protocols in PDMS may be applicable to other IoT scenarios, our system is specifically designed to address the unique constraints and operational demands of vehicle-based driver monitoring. These include the need for timely gaze or fatigue detection, limited computational resources on the vehicle, and variable connectivity with roadside infrastructure. To accommodate these requirements, PDMS integrates privacy-preserving YOLO and FlowNet for efficient video frame filtering, and adopts a lightweight two-server edge architecture to reduce latency and communication overhead.

The main contributions of this paper are as follows.

- We present the Privacy-preserving Driver Monitoring System (PDMS) as a solution for vehicles to safely transmit video data to the edge or cloud, thereby conserving computational power on the vehicle for higher-priority tasks. PDMS ensures the protection of driver's video data during both processing and transmission. In this paper, we detail the design, architecture, and implementation of PDMS.
- We present the design and construction of privacy-preserving Transformers, which serve as the core component of PDMS. This includes a detailed explanation of each privacy-preserving DNN layer within a Transformer model and the secure protocols employed for these layers. The technologies used include the additive secret sharing theory, Beaver's triples, Taylor series, and more. Additionally, optimizations such as enhanced privacy-preserving layer normalization are presented.
- We provide a solid example of a privacy-preserving Vision Transformer (ViT) for gaze estimation and demonstrate its effectiveness through comprehensive experiments. We have evaluated the size of extra data in secret share generation on the vehicle side and the system's processing time and accuracy on the edge side. we have quantified the overall computational time and storage complexity. An optimization approach combining optical flow and image segmentation for continuous video processing is presented.

2 Background

2.1 Transformer for Autonomous Driving

The Vision Transformer (ViT) [1] is a groundbreaking model in computer vision that adapts the Transformer [37] architecture, originally developed for natural language processing, to image analysis tasks. Figure 1 illustrates the architecture of a standard Transformer, which processes embedded token inputs through multiple Transformer blocks. Each block contains key deep learning components such as Layer Normalization, Linear layers, Softmax, and GELU activation functions. The Vision Transformer (ViT) uses a similar encoder architecture; however, unlike traditional Transformers designed for language models that operate on text-based token embeddings, ViT processes embeddings generated from image patches derived directly from pixel values.

Unlike traditional convolutional neural networks (CNNs), ViT divides an image into fixed-size patches, treating each patch as a token similar to words in a sentence. These tokens are embedded into vectors, enriched with positional information, and processed through Transformer encoder layers to capture global relationships across the image. ViT is particularly well-suited for processing vehicle-recorded video frame data, making it an excellent candidate for image processing tasks in this domain. Additionally, ViT can extract rich image features that can be paired with different perception networks to achieve diverse outcomes, such as object detection, segmentation, or classification. In this paper, we leverage ViT to perform gaze estimation [28], showcasing its capability to process video data and extract critical features for driver monitoring applications.

2.2 Object Detection and Optical Flow

You Only Look Once (YOLO) [29] is a single-stage object detection algorithm [43] that treats detection as a unified regression problem, directly predicting bounding boxes and class probabilities from entire images. It divides the input image into a grid and generates predictions for each cell, enabling a balance of high speed and accuracy. Compared to two-stage detectors like Faster R-CNN [14, 15, 30] family and ResNet [34], YOLO offers significantly faster inference by eliminating the separate region proposal step and performing detection in a single pass. This efficiency makes it particularly well-suited for resource-constrained environments such as edge computing, with widespread use in domains like autonomous driving and surveillance.

Optical flow [16] estimates the apparent motion of objects between consecutive video frames based on pixel intensity changes, and is widely used in tasks such as motion detection, object tracking, and video analysis. Traditional methods rely on handcrafted features and iterative optimization, which can be computationally

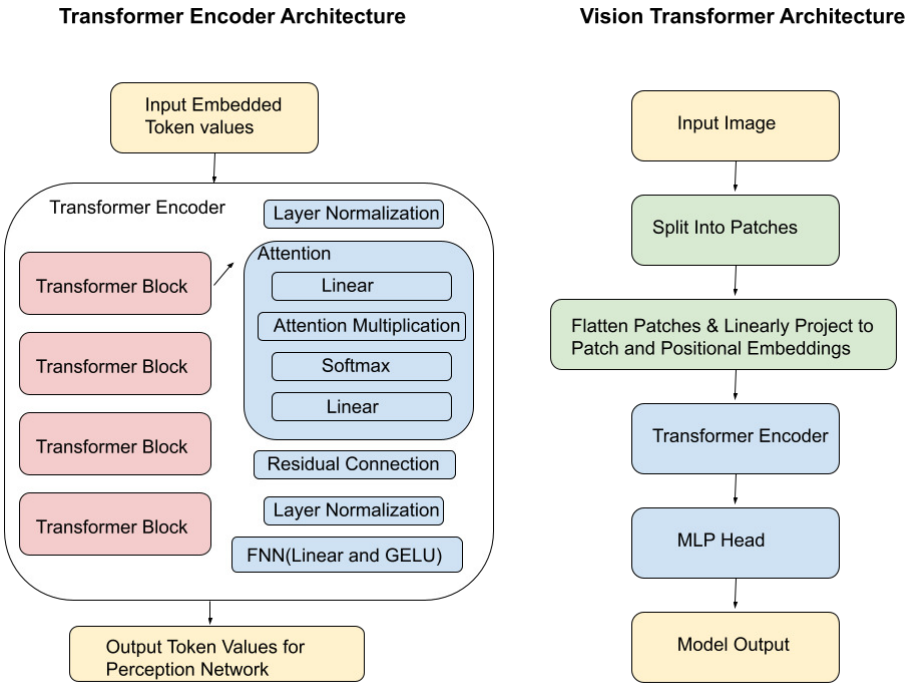

Fig. 1. Transformer Encoder (left) And Vision Transformer Architecture (right)

intensive. Figure 2 illustrates an example where a human body moves between two frames, and the resulting optical flow output is a set of vectors that represent the motion of pixel values. Based on this information, applications can robustly interpret and track the movement of objects of interest across video streams.

FlowNet [11] is a deep learning-based approach that directly learns to predict optical flow from pairs of images using convolutional neural networks, offering a faster and more scalable solution. It marked a shift toward data-driven optical flow estimation, enabling real-time performance in complex visual environments.

2.3 Additive Secret Sharing

Additive secret sharing [10] is a cryptographic technique where a secret s is divided into multiple shares such that the sum of these shares reconstructs the original secret. Formally, if a secret s is shared among N participants, it can be represented as $s = \sum_{n=1}^{k} s_n$; and $k = N$, where s_i denotes the individual shares. This method ensures that the secret can only be reconstructed when all shares are combined, providing security against partial information leakage. Additive secret sharing allows secure computations on encrypted data. For instance, if secrets s and t are split into shares $(s_1, s_2, ...s_n)$ and $(t_1, t_2, ...t_n)$ respectively, operations like addition can be performed on the shares independently, enabling the computation of $s + t$ without revealing the actual values of s and t. This feature is crucial for various secure multi-party computations, allowing collaborative processing of sensitive data while preserving privacy.

2.4 Beaver's triples and Taylor series

Beaver's triples [6], also known as Beaver's tuples, are a fundamental cryptographic technique used to facilitate secure multi-party computation (MPC). A Beaver's triple consists of three random values (a, b, c) where $c = a \cdot b$.

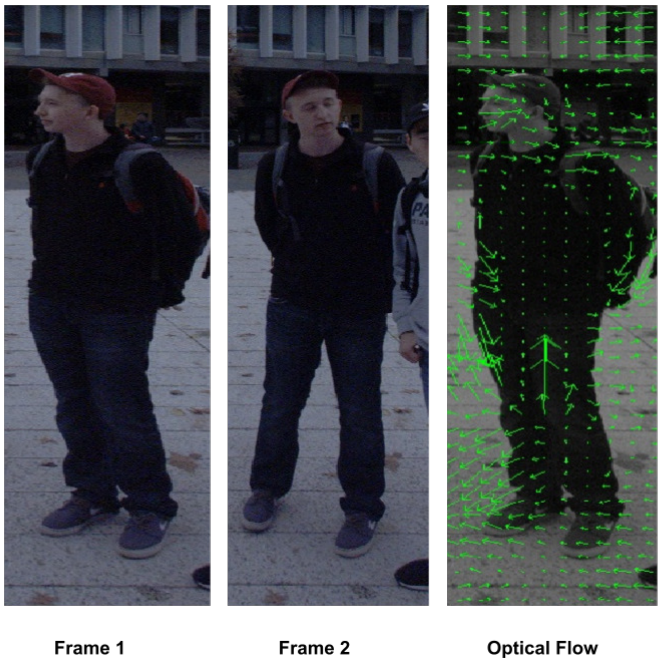

Fig. 2. Example of Optical Flow. The left image represents Frame 1, the middle image shows Frame 2, and the right image depicts the optical flow output between these two consecutive frames. Between Frame 1 and Frame 2, the person undergoes significant body movement, such as a change in facing direction. The optical flow output visualizes this motion as a set of vectors indicating the direction and magnitude of pixel movement. These vectors, applied to Frame 1, estimate how those pixels are expected to move in the next frame.

These tuples allow parties to perform secure multiplications without revealing their inputs. By pre-generating these triples, parties can mask their inputs and securely compute the product of two secret values using a small number of communication rounds. This process significantly enhances the efficiency and security of MPC protocols, making them practical for real-world applications where privacy is paramount.

The Taylor series [35] is a mathematical tool used to approximate functions through an infinite sum of terms calculated from the values of their derivatives at a single point. For a function $f(x)$, the Taylor series around a point a is given by:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots$$

Taylor series expansions are particularly useful in numerical methods and analysis, enabling the approximation of complex functions by polynomials. This is essential for various algorithms in fields such as machine learning, optimization, and scientific computing, where precise computations are needed but direct evaluation of functions might be computationally expensive or infeasible. By truncating the series to a finite number of terms, efficient and accurate approximations can be achieved, facilitating the development of robust computational models and simulations.

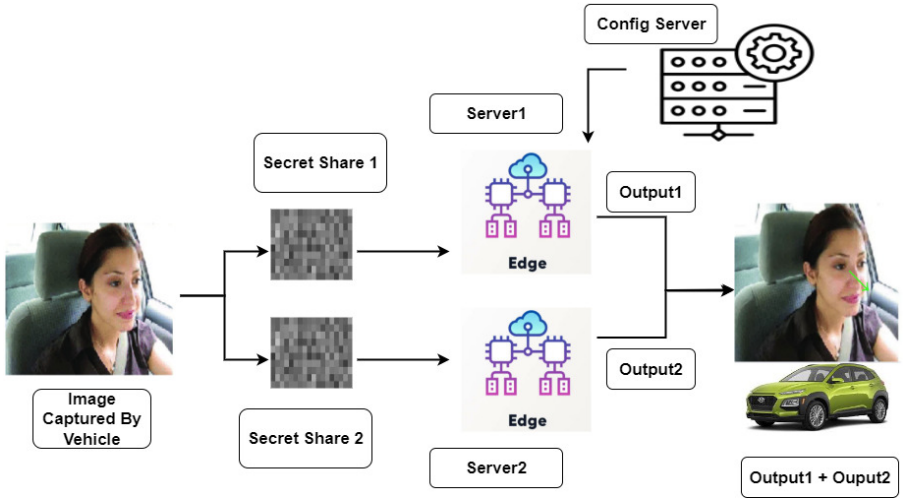

Fig. 3. Privacy Preserving Driver Monitoring Framework Architecture

3 Privacy Preserving Driver Monitoring System Framework Architecture

3.1 PDMS Archetecture and Execution Flow

Figure 3 illustrates the major components of the Secure Driver Monitoring System(PDMS): Vehicles, Edge Servers (S_1, S_2), and a Configuration Server.

A camera-equipped vehicle captures the driver's video data. Each video frame is then pre-processed to generate random secret shares r_1 and r_2, which maintain an additive relationship with the raw data $(r = r_1 + r_2)$. These secret shares are sent to two edge servers, S_1 and S_2, for further processing.

A preprocessing model is employed to capture essential object movement. If no significant motion is detected, the previous detection result is returned; otherwise, the secret-shared data is processed by the Transformer models.

Both servers host identical privacy preserving Vision Transformer (ViT) models. In this design, edge computing platforms are employed to minimize communication costs between the vehicle and the edge servers, but the system can also adapt to normal cloud servers if needed. Each server takes its respective secret share of the video frames as input and produces a partial inference result $(O_1$ and $O_2)$. These partial results are then returned to the data owner, or other trusted party, for final analysis. A comprehensive overview of the Privacy Preserving Transformer model architecture is provided in the following sections.

PDMS requires that each video frame be divided into an equal number of secret shares, which are then sent to the corresponding edge servers for processing. The Config Server is responsible for initializing PDMS with encryption algorithms to secure the transfer of secret shares over public networks and to protect parameters such as random vectors used in privacy preserving multiplication and DNN layers. Additionally, the Config Server facilitates data exchange between edge servers, ensuring adherence to the secure protocol and preventing the exposure of raw user data, thereby protecting user privacy.

Both the Config Server and the Edge Servers reside within a private network domain to enhance protection and reduce latency when transmitting secure parameters between edge servers. While the vehicles and edge

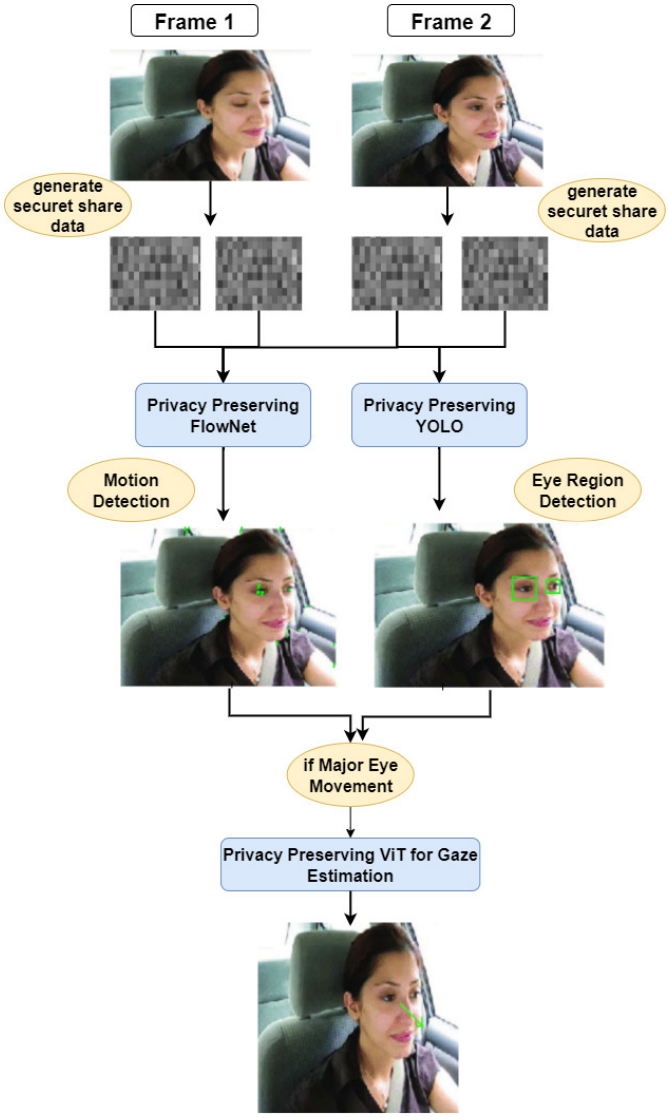

Fig. 4. PDMS Example Execution Flow

servers operate in the public network, the Config Server does not communicate with the vehicles during data processing.

We use a ViT-based gaze estimation transformer as an example to illustrate the PDMS execution flow. To better understand the driver's status and habits, the vehicle records the driver's video using HD camera. The vehicle then leverages the gaze estimation results to customize driving modes and enhance safety.

Initially, the vehicle generates random pixel-value vectors (r_1, r_2) as secret shares for each video frame. These secret shares maintain an additive secret-sharing relationship with each frame. To protect the shares during transmission over public networks, CAV_1 and CAV_2 encrypt them using a predefined AES-512 protocol [33]. The

corresponding encryption and decryption keys are obtained from the Config Server before the inference process begins.

After generating the secret shares, the vehicle sends the encrypted r_1 to S_1 and r_2 to S_2. The edge servers S_1 and S_2 process the received secret shares according to the PDMS protocol, producing partial inference outputs O_1 and O_2. These partial results are then returned to the vehicle, which simply computes $O_1 + O_2$ to reveal the gaze estimation.

The privacy preserving ViT can be readily adapted for different types of computations. For instance, one can replace the gaze estimation network with a classification detection network that leverages the ViT encoder's output—where feature and spatial information of the input data are preserved—to determine whether the driver's eyes are open. Only minimal modifications are required, since the encoder remains unchanged. Furthermore, it is even possible to produce multiple inference results in parallel (e.g., both gaze estimation and eye-state classification) using the same encoder.

Figure 4 illustrates a sample execution flow of the PDMS. Each video frame is divided into secret shares and processed by a preprocessing model that internally uses privacy preserving FlowNet for optical flow analysis and privacy preserving YOLO for eye region detection. The combined results reveal pixel-level motion in the driver's eye region between consecutive frames. Based on this information, the preprocessing model determines whether significant eye movement has occurred. If so, the new frame's secret shares are passed to the privacy preserving Gaze Transformer for gaze estimation; otherwise, the previous frame's result is reused.

The design aims to protect the driver's facial image data within each video frame by leveraging a secret-sharing-based system architecture. A preprocessing model, which combines privacy preserving object detection and optical flow analysis, is executed in parallel to minimize the number of invocations of the privacy preserving Gaze Transformer. Since most video frames do not exhibit significant eye movement, this approach significantly improves overall system efficiency. In our system, the vehicle is only required to generate simple random vectors and perform output addition, keeping its computational burden minimal.

3.2 Threat Model

In our system, we focus on securing the inference process. The edge servers S_1 and S_2 are designated as "Curious but Honest" components. This means they carry out their assigned computational tasks correctly, yet may attempt to glean additional user information out of curiosity. For example, an edge server might analyze incoming video frames to uncover private user details or search for patterns that reveal sensitive information not explicitly shared, thereby infringing upon user privacy and compromising data confidentiality. Although individual edge servers may attempt to infer information about the raw data, within the scope of this paper, we assume that not all computation servers collaborate to retrieve the complete raw data.

To mitigate these risks, the PDMS framework employs privacy-preserving Transformers based on additive secret sharing, ensuring that video frame data and sensitive information remain protected throughout the computation process. The "Curious but Honest" (HBC) model highlights the importance of strong privacy-preserving techniques, including secure computation and communication protocols, to prevent potential privacy breaches by inquisitive yet compliant edge servers.

The Config Server plays a critical yet narrowly scoped role within PDMS, responsible for generating cryptographic parameters, managing keys, and distributing configuration metadata to the edge servers. Crucially, it never processes, stores, or observes sensor data or the secret shares of driver video frames. This architectural isolation ensures that even if the Config Server were compromised after initialization, it could not reconstruct sensitive inputs, as it lacks access to any data-dependent shares. Its influence is therefore limited to ensuring the correctness and consistency of the secure computation setup.

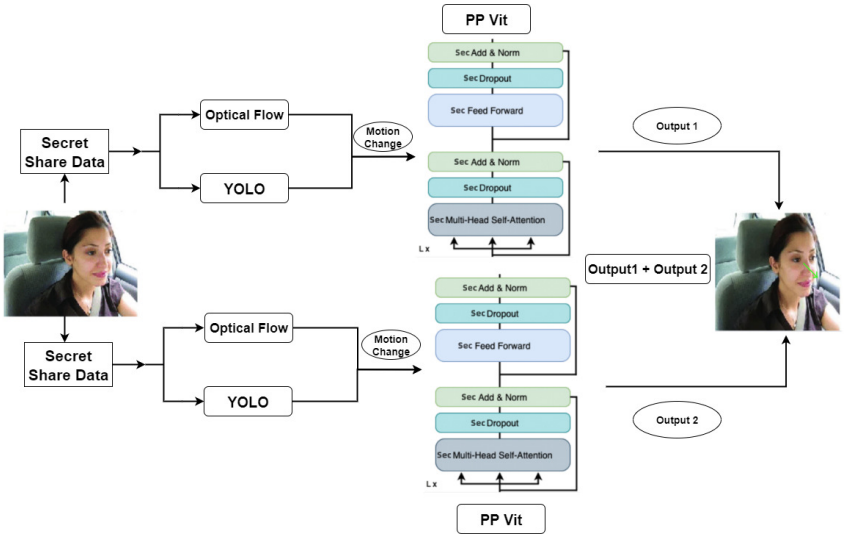

Fig. 5. Privacy Preserving ViT for Gaze Estimation

A malicious Config Server could, in principle, distribute inconsistent parameters that lead to incorrect inference results but would not cause privacy leakage. Such integrity violations can be mitigated through standard security mechanisms, including digital signatures, certificate-based authentication, and verifiable setup protocols—methods already widely adopted in secure edge and vehicular systems. In practice, the trust assumption on the Config Server is readily realizable using established deployment patterns. It can be instantiated as a hardened key management service hosted within a secure enclave or trusted execution environment (TEE) in the cloud or at a central operations center. In automotive contexts, original equipment manufacturers (OEMs) or fleet operators already maintain physically secured infrastructures for vehicle provisioning and software updates, which can naturally host the Config Server. Its role closely parallels that of the dealer or setup authority in secure multi-party computation (SMPC) frameworks, which are trusted for initialization but excluded from online computation.

In a two-server PDMS deployment, the Config Server operates entirely outside the data path and can be placed in a logically and physically isolated management network, further reducing its attack surface. When scaling to configurations with $n > 2$ edge servers, the system's resilience to collusion—tolerating up to $t = n - 1$ compromised nodes—remains unaffected by the Config Server's placement, provided that its initialization parameters are authentic and consistent. Therefore, while PDMS assumes a trusted Config Server for correct setup, this assumption is both minimal and operationally feasible, aligning with established best practices in secure edge computing and automotive system design.

3.3 PDMS Deep Learning Model

In PDMS, each edge server hosts a preprocessing model and a privacy preserving ViT model. Figure 5 illustrates the inference flow of PDMS. In a standard ViT, an input image is divided into fixed-size patches and flattened into a one-dimensional vector called the patch embedding, typically implemented via a convolutional layer. Next, a positional encoding is added, using learnable parameters to capture the spatial relationships and global context among patches. The combined result is then fed into the ViT transformer encoder, which comprises multiple identical transformer blocks. Each block contains layer normalization, dropout layers, feedforward neural network (FNN) layers, and a multi-head self-attention mechanism. The output from these transformer blocks is ultimately passed to a multi-layer perceptron (MLP) network to generate the final inference result.

In the privacy preserving ViT setting, we aim to use a pretrained model for privacy preserving inference. Unlike the traditional ViT, which accepts raw image data, the privacy preserving ViT operates on secret shares of the raw image. Two separate secret shares are input into two corresponding privacy preserving ViT models. The structure remains the same as in the original ViT, including a privacy preserving Patch Embedding, Positional Encoding, a privacy preserving ViT Transformer Encoder, and a privacy preserving Gaze Detection Head (implemented with an MLP). Each ViT component is redesigned as a privacy preserving computation block to comply with PDMS security protocols. All data processed by a single edge server consists solely of intermediate results based on secret shares; neither the raw input data nor the gaze estimation can be revealed until the final receiver combines the outputs O_1 and O_2 from both edge servers.

4 Design and Implementation of PDMS Privacy Preserviong Transformer

In the previous section, we introduced the privacy preserving ViT transformer, which serves as the core component of PDMS. In this section, we delve into its design and implementation in detail.

Based on the execution flow of a traditional ViT, we can represent its behavior with the expression

$$R = F_n \circ \ldots \circ F_2 \circ F_1(I),$$

where I is the input data, each F_i denotes the logical operation of a single DNN layer, and R is the final inference result. In a privacy preserving ViT, our goal is to ensure that for any $I = I_1 + I_2$, the outputs satisfy

$$R_1 = f_n \circ \ldots \circ f_2 \circ f_1(I_1) \quad \text{and} \quad R_2 = f_n \circ \ldots \circ f_2 \circ f_1(I_2),$$

allowing us to recover R by $R_1 + R_2$.

This requirement implies that for each F_i,

$$F_i(x) = f_i(x_1) + f_i(x_2).$$

Let x represent the original input (e.g., a video frame), which is secret-shared between two non-colluding edge servers as x_1 and x_2, such that $x = x_1 + x_2$ under modular arithmetic. Each server S_i holds one share x_i and performs computation on it locally. The final result is reconstructed by summing the individual evaluations $f_i(x_1)$ and $f_i(x_2)$ to obtain $F_i(x) = f_i(x_1) + f_i(x_2)$, without either server learning the complete input x.

When F_i is a linear operation, f_i can match the underlying operation exactly. However, Transformer encoder layers typically include various non-linear operations (e.g., dropout layers, self-attention blocks, and layer normalization). Consequently, we provide secure implementations for each of these non-linear components so that their intermediate outputs also preserve an additive relationship.

4.1 Privacy Preserving Patch Embedding and Positional Positional Encoding

Path embedding requires dividing the image into patches, passing these patches through a convolutional layer, and finally flattening them into a single dimension vector. The patch division is of fixed size, and since the secret shares of the images have an identical shape, this division methodology can be applied directly to both secret shares. As the convolutional layer is a linear operation, it can be applied to the patches of secret shares without modification. Positional encoding adds pretrained learnable parameters to the flattened vector, and since these parameters do not change during inference, positional encoding can also be applied without modification.

4.2 Secure Multiplication

Before delving into the implementation of other secure DNN layers, it is important to note that the fundamental operations on image tokens are addition and multiplication. Since addition is inherently a linear operation, we focus here on introducing secure multiplication within the privacy preserving ViT. Our method employs *Beaver's*

Triple, a well-known technique in secure multiparty computation that efficiently facilitates multiplication without revealing the underlying inputs.

In Algorithm 1, each server S_i holds secret shares (x_i, y_i) of the raw data's feature values, satisfying $x_1 + x_2 = x$ and $y_1 + y_2 = y$. The goal is to produce outputs z_i such that $z_1 + z_2 = x \cdot y$.

To achieve this, the Configuration Server generates random vectors $\{\beta, \theta, \iota_1\}$ following the Beaver's Triple protocol where $\iota = \beta \cdot \theta$. It then calculates ι_2 and distributes $(\beta_1, \theta_1, \iota_1)$ to S_1 and $(\beta_2, \theta_2, \iota_2)$ to S_2. Next, S_1 and S_2 compute intermediate products by combining their secret shares with these random values and exchanging the results. During this exchange, each server calculates M_i based on the transformed shares. Because M_1 and M_2 intertwine secret shares and random vectors—and each server only retains $(\beta_i, \theta_i, \iota_i)$—exchanging M_i does not reveal x or y. Finally, each server computes its output (z_1, z_2), ensuring that $z_1 + z_2 = M_1 \cdot y - M_2 \cdot \beta + \iota_1 + \iota_2 = (x + \beta) \cdot y - (y + \theta) \cdot \beta + \beta \cdot \theta = xy + y\beta - y\beta - \beta\theta + \beta\theta = xy$. This distributive property guarantees the correctness of the final multiplication result while preserving the privacy of the original inputs.

Algorithm 1 Secure Multiplication

On Servers S_1 **and** S_2
Input: Secret shares (x_1, y_1) and (x_2, y_2)
Output: Resultant vector $\{z_i\}$
Trusted Server generates random integer vector: $\beta_1, \beta_2, \theta_1, \theta_2, \iota_1$
Calculate $\iota = (\beta_1 + \beta_2) \cdot (\theta_1 + \theta_2)$ and derive $\iota_2 = \iota - \iota_1$
Trusted Server distributes $(\beta_1, \theta_1, \iota_1)$ to S_1 and $(\beta_2, \theta_2, \iota_2)$ to S_2
S_1, S_2 transform the secret share value by adding the random params
S_1 computes $x_1 + \beta_1$, $y_1 + \theta_1$
S_2 computes $x_2 + \beta_2$, $y_2 + \theta_2$
S_1, S_2 collaborate computes $M_1 = (x_1 + \beta_1) + (x_2 + \beta_2)$ and $M_2 = (y_1 + \theta_1) + (y_2 + \theta_2)$
Each server calculates:
$z_1 = M_1 \cdot y_1 - M_2 \cdot \beta_1 + \iota_1$
$z_2 = M_1 \cdot y_2 - M_2 \cdot \beta_2 + \iota_2$
Return: z_i

4.3 Privacy Preserving Activation Layer

In privacy preserving ViTs, multiple activation layers play a crucial role in enabling the model to learn complex patterns and relationships in the input data. These layers are typically applied after linear transformations or attention mechanisms, enhancing the model's ability to capture non-linear features and distinguish patterns such as edges or textures. While traditional ViTs often use ReLU as the activation layer, Transformer architectures commonly adopt GELU. We introduce privacy preserving versions of both these activation layers.

4.3.1 Privacy Preserving RELU Layer. In the ReLU activation layer, negative feature values are replaced with zero, while positive values remain unchanged. By leveraging the Secure Symbol Extraction (SSE) process, each feature's sign is determined using bitwise operations (e.g., XOR) without revealing the feature's exact magnitude. SSE splits the feature values into shares, allowing each server to securely compute partial sign bits. Once these bits are combined, the final sign information indicates whether the feature is negative or non-negative. Any negative feature value is then masked out (set to zero), while non-negative values are preserved, ensuring privacy throughout the entire computation. The detailed design and implementation can be found in [4].

4.3.2 GELU. The Gaussian Error Linear Unit (GELU) is a widely used non-linear activation function in neural networks, including certain variants of the GPT architecture. Its approximation is given by:

$$\text{GELU}(x) \approx \frac{x}{2}\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3)\right)\right).$$

The GELU function depends on the hyperbolic tangent function, $\tanh(x)$, which is defined as:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

To simplify computations, $\tanh(x)$ can be approximated using its Taylor series expansion. The Taylor series representation of $\tanh(x)$ is given by:

$$\tanh(x) = \sum_{n=0}^{\infty}(-1)^n \frac{2^{2n}(2^{2n}-1)B_{2n}}{(2n)!}x^{2n-1},$$

where B_{2n} are the Bernoulli numbers. Let $P(x)$ represent the polynomial expansion of $\tanh(x)$. By replacing the tanh function with its Taylor series expansion, the GELU formula can be expressed in a polynomial form as $p(x) = \sum_{i=0}^{k} c_i x^i$, where c_i are constant coefficients.

Servers S_1 and S_2 compute $p(x_1)$ and $p(x_2)$ locally. The difference between the polynomial approximation of GELU using secret shares and the actual GELU value, $p(x)$, can be expressed as $p(x) - (p(x_1) + p(x_2)) = q(x_1, x_2)$, where $q(x_1, x_2)$ is a polynomial of the form $q(x_1, x_2) = \sum_{i=0}^{k}\sum_{j=0}^{k} c_{i,j}x_1^i x_2^j$. This represents adjustment needed to bridge the gap between the secret share data based output and raw data GELU value: $p(x_1) + p(x_2) + q(x_1, x_2) = p(x) = \text{GELU}(x)$.

To obtain the final result, the polynomial correction $q(x_1, x_2)$ can be applied to the output of either S_1 or S_2. Since $q(x_1, x_2)$ is a polynomial in x_1 and x_2, it can be efficiently computed using a secret multiplication protocol to calculate each term. For example, when $n = 3$, a specific form of $q(x_1, x_2)$ is $q(x_1, x_2) = -0.865855(x_1^2 x_2 + x_1 x_2^2)$. The computed polynomial is then added to the final output to accurately approximate the GELU function. Algorithm 2 the design of Privacy Preserving GELU.

Algorithm 2 Privacy Preserving GELU

On Server S_1 and S_2
Input: secret shares of feature value x_1, x_2
Output: vector g_1, g_2

Let $p(x) = \sum_{i=0}^{k} c_i x^i$ equals the Taylor expansion form of GELU(x)
Compute the compensation polynomial for the secret shares ($q(x_1, x_2) = \sum_{i=0}^{k}\sum_{j=0}^{k} c_{i,j}x_1^i x_2^j$)
for for element with in $q(x_1, x_2)$ **do**
    generates random vector r,$r_1 + r_2 = x_1, r_3 + r_4 = x_2$
    S_1, S_2 collaborate compute let $t = t + SecMult(r_i, r_j)$
end for
S_1 returns $p(x_1) + t$, S_2 returns $p(x_2)$

4.4 Privacy Preserving Layer Normalization

Layer normalization can be simplified to $\text{LayerNorm}(x) = \frac{x-\mu}{\sqrt{\sigma^2}}$. The critical challenge is to compute the variance of the raw data x on edge servers that only hold its secret shares. By expanding the variance formula, $\text{Var}(x) = \frac{1}{n}\sum_{i=1}^{n}(\alpha_i - \mu)^2$, where α_i represents the i-th element of the raw data x.

For each element in the polynomial expansion of variance, the goal is to compute $(\alpha_i - \mu)^2$ and then sum across all elements. Since $\alpha_i = \alpha_{1i} + \alpha_{2i}$, where α_{1i} and α_{2i} are the i-th elements of the secret shares x_1 and x_2, respectively, there is a need to compensate for the difference between $(\alpha_i - \mu)^2$ and $(\alpha_{1i} - \mu)^2 + (\alpha_{2i} - \mu)^2$.

By expanding the polynomial, this difference can be expressed as $q(\alpha_{1i}, \alpha_{2i}) = 2\alpha_{1i} \cdot \alpha_{2i} - \mu^2$. Using a Secure Multiplication protocol, the value of $q(\alpha_{1i}, \alpha_{2i})$ can be computed. The variance of x is then obtained by summing $\text{Var}(x_1) + \text{Var}(x_2) + q(x_1, x_2)$.

To address the complexity and frequent use of the secure multiplication protocol in privacy preserving layer normalization during inference, we propose an optimization to compute $(q(\alpha_{1i}, \alpha_{2i})$ for non zero vectors.

In the optimized protocol, server S_1 computes a ratio vector for neighboring elements in x_1 and sends it to S_2. Server S_2 generates a random vector and modifies each pair of elements in x_2 by adding random values to the first element and a scaled version of these values to the second. The updated vector is then sent back to S_1.

Using this modified vector, S_1 calculates the variance difference without requiring the secure multiplication protocol. The random values cancel out during the computation, enabling S_1 to efficiently compute the variance difference. The process concludes by adding a final adjustment term, providing a secure and computationally efficient solution. Here is an example of the secure protocol.

Example: Optimized Variance Difference Computation Protocol

Consider a dataset with $n = 4$, where $x_1 = [x_{11}, x_{12}, x_{13}, x_{14}]$ and $x_2 = [x_{21}, x_{22}, x_{23}, x_{24}]$ are secret shares of the raw data $x = x_1 + x_2$. Let us compute the variance difference using the proposed protocol.

1. **Step 1: Compute Ratio Vector (R) at** S_1 Server S_1 computes the ratio vector R for neighboring elements in x_1:

$$R_1 = \frac{x_{11}}{x_{12}}, \quad R_2 = \frac{x_{13}}{x_{14}}.$$

The ratio vector $R = [R_1, R_2]$ is sent to S_2.

2. **Step 2: Generate Random Vector (L) at** S_2 Server S_2 generates a random vector $L = [L_1, L_2]$ of length $\frac{n}{2}$. For each pair of elements in x_2, the first element is modified as:

$$\theta_1 = x_{21} + L_1, \quad \theta_3 = x_{23} + L_2,$$

and the second element is modified as:

$$\theta_2 = x_{22} + (-R_1) \cdot L_1, \quad \theta_4 = x_{24} + (-R_2) \cdot L_2.$$

The resulting vector $\theta = [\theta_1, \theta_2, \theta_3, \theta_4]$ is sent back to S_1.

3. **Step 3: Compute Variance Difference at** S_1 Using the received vector θ, S_1 computes:

$$2(x_{11} \cdot \theta_1 + x_{12} \cdot \theta_2 + x_{13} \cdot \theta_3 + x_{14} \cdot \theta_4).$$

Substituting θ_1 and θ_2:

$$2(x_{11} \cdot (x_{21} + L_1) + x_{12} \cdot (x_{22} + (-R_1) \cdot L_1)),$$

the random values L_1 cancel out due to the ratio $R_1 = \frac{x_{11}}{x_{12}}$, leaving:

$$2(x_{11} \cdot x_{21} + x_{12} \cdot x_{22}).$$

Similarly, for the second pair, the result is $2(x_{13} \cdot x_{23} + x_{14} \cdot x_{24})$.

4. **Final Result:** The variance difference is computed as:

$$\text{Variance Difference} = 2(x_{11} \cdot x_{21} + x_{12} \cdot x_{22} + x_{13} \cdot x_{23} + x_{14} \cdot x_{24}) - 4\mu^2.$$

Adding this to the local variances $\mathrm{Var}(x_1)$ and $\mathrm{Var}(x_2)$ gives the total variance of the raw data x.

Algorithm 3 Privacy Preserving Layer Normalization

Input: secret share of Feature vector x_1, x_2
Output: layer normalization partial result of z_1, z_2

S_1, S_2 computes $\bar{x} = \bar{x_1} + \bar{x_2}$
for each α_{1i}, α_{2i} in x_1, x_2 **do**
    generate random vector r, $r_1 + r_2 = \alpha_{1i}$, $r_3 + r_4 = \alpha_{2i}$,
    S_1, S_2 collaborate compute $t_i = SecMult(r_i, r_j)$
end for
$q(x_1, x_2) = \sum_{i=1}^{n} 2t_i - \bar{x}^2$
S_1, S_2 collaborate compute $Var(x) = Var(x_1) + Var(x_2) + q(x_1, x_2)$
S_1, S_2 return $\frac{x_i - \bar{x}}{Var(x)}$

4.5 Privacy Preserving Dropout Layer And Privacy Preserving SoftMax

The softmax layer in a Transformer normalizes attention scores into a probability distribution, ensuring they sum to one. It allows the model to focus on the most relevant parts of the input by assigning higher weights to important tokens. In a privacy preserving setting, the key computation of softmax involves evaluating e^x, which can be approximated using a polynomial expansion of x. Similar to the privacy preserving GELU approach, a secure multiplication protocol is used to compute the difference polynomial $q(x_1, x_2)$ securely.

The dropout layer is a regularization method in Transformers that prevents overfitting by randomly setting a fraction of input units to zero during training. This helps the model learn more generalized features by reducing dependency on specific neurons. In a privacy preserving dropout layer, there is no need to compute secret shares of feature values. Instead, S_1 randomly selects the indices to set to zero and shares this information with S_2, allowing both parties to set the corresponding secret share values to zero.

5 Privacy Preserving Convolution Neural Networks

In the previous section, we introduced the secure protocols and design of the privacy preserving ViT Transformer. In this section, we present the design of two convolutional neural network (CNN)-based models used in the preprocessing stage: the privacy preserving FlowNet and the privacy preserving YOLO.

To align with the design principles of the PDMS and the privacy preserving Transformer, the privacy preserving CNNs in the preprocessing model must also operate on secret-shared representations of raw video frames. Similar to the privacy preserving ViT, we design secure deep neural network (DNN) layers for each of the two CNN models. For linear transformation layers, such as convolutional and fully connected layers, the computation remains unchanged under secret sharing. However, non-linear operations require special handling. In FlowNet and YOLO, the most commonly used non-linear layers are the ReLU activation and max pooling layers. In this section, we present privacy preserving designs for both.

5.1 Privacy Preserving Max Pooling

Max pooling extracts the most prominent features from a feature map based on a given stride. However, the max operation is inherently non-linear, which complicates secure computation—particularly because additive secret sharing does not preserve ordering across shares. To address this, we propose an iterative difference evaluation

approach, outlined in Algorithm 4, where edge servers cooperatively compute the maximum value by exchanging intermediate differences.

Algorithm 4 Privacy Preserving Max Pooling

Input: Secret shares of feature values x_1, x_2
Output: Partial maximum feature value
for each channel j in feature map x_i **do**
    Extract x_{ij}, the j^{th} channel from x_i
    Let w, h be max pooling strides
    for each pooling region R in x_{ij} with strides w, h **do**
        Initialize pooling index: $\alpha = 0$, $\beta = 0$
        Compute local difference: $I_i = R[\alpha][\beta] - R[w][h]$
        Exchange I_i, I_q between edge servers E_i and E_q
        Combine differences: $I = I_i + I_q$
        if $I < 0$ **then**
            Update pooling index: $\alpha = w$, $\beta = h$
        end if
    end for
    return $R[\alpha][\beta]$
end for

5.2 Privacy Preserving ReLU

ReLU introduces non-linearity into neural networks by preserving positive activations and mapping negative values to zero. In the privacy preserving setting, applying ReLU securely requires bit-wise operations to determine the sign of each feature value. Our privacy preserving ReLU protocol leverages secure primitives (Algorithms 6, 7, and 8) to accomplish this. Specifically, the sign of each feature is determined through secure bit addition, and the ReLU operation is applied accordingly.

Leaky ReLU can be implemented with a similar structure, except that negative values are scaled by a pre-defined small factor instead of being set to zero.

Algorithm 5 Privacy Preserving ReLU

Input: Secret shares of feature values x_1, x_2
Output: Partially revealed positive features or zero
for each feature f in x_i **do**
    $\{T_i, r_i\} \leftarrow$ Secure_Bit_Array_Generation(x_i)
    if Secure_Bit_Addition$(T_i, r_i) == 1$ **then**
        Set $f \leftarrow 0$
    end if
    return x_i
end for

5.2.1 Secure Bit Array Generation. In privacy preserving ReLU, the Secure Bit Array Generation protocol (Algorithm 6) transforms each feature share x_i into two bit arrays $\{T_i, r_i\}$, satisfying the condition:

$$\sum_{i=1}^{n} x_i = (T_1 \bigoplus T_2 \bigoplus \cdots \bigoplus T_n) + (r_1 \bigoplus r_2 \bigoplus \cdots \bigoplus r_n) = T + r,$$

where $\bigoplus$ denotes bit-wise XOR and + represents standard numeric addition. The bit arrays $\{r, \tau\}$ are randomly generated by a configuration server for each round, ensuring that identical plaintext inputs yield different ciphertext outputs, thereby enhancing data privacy.

Algorithm 6 Secure Bit Array Generation

1: **Executed on:** Edge server S_i, where $i \in \{1, \ldots, n\}$
2: **Input:** Secret share of feature value x_i
3: **Output:** Bit arrays $\{T_i, r_i\}$
4: Configuration server TS generates random bit arrays $\{r, \tau\}$ such that:

$$r_1 \bigoplus r_2 \bigoplus \cdots \bigoplus r_n = \tau_1 + \tau_2 + \cdots + \tau_n$$

and securely distributes $\{r_i, \tau_i\}$ to each edge server S_i
5: Each server S_i computes $d_i = x_i - \tau_i$ and locally generates a random bit array T_i
6: Note that:

$$T = T_1 \bigoplus T_2 \bigoplus \cdots \bigoplus T_n = x - \tau$$

7: Servers S_1 through S_{n-1} send d_i and T_i to server S_n
8: Server S_n computes:

$$d = \sum_{i=1}^{n} d_i, \quad T_n = d \bigoplus T_1 \bigoplus T_2 \bigoplus \cdots \bigoplus T_{n-1}$$

9: Each S_i returns the generated pair $\{T_i, r_i\}$

Algorithm 6 generates randomized bit arrays based on secret shares, preparing them for secure bit-wise operations. The resulting T_i and r_i arrays are passed to protocols like secure bit addition and multiplication, which exchange minimal intermediate values between two servers to protect the confidentiality of the original data.

5.2.2 Secure Bit Addition. Given bit arrays $\{T_i, r_i\}$ produced from Secure Bit Array Generation, the Secure Bit Addition protocol computes a boolean value π that indicates whether the underlying secret value x is positive. Specifically, the combined value is computed as:

$$x = \sum_{i=1}^{n} h_i = (T_1 \bigoplus T_2 \bigoplus \cdots \bigoplus T_n) + (r_1 \bigoplus r_2 \bigoplus \cdots \bigoplus r_n),$$

where $\bigoplus$ represents bit-wise XOR and + represents numeric addition.

To preserve privacy, edge servers must avoid directly exchanging the original bit arrays. Instead, the protocol uses bit-wise additions, relying only on the exchange of intermediate values. It performs correctly under the condition that no two bits at the same index in T and r are simultaneously 1—i.e., when no carry is needed.

As an example, if $(T_1 \bigoplus r_1) \bigoplus (T_2 \bigoplus r_2) = T \bigoplus r = L$, then $L = T + r$ only if the carry is zero. In cases where both T and r contain a 1 in the same position, a carry to the next significant bit is required. To handle this, we calculate the carry bits using the Secure Bit Multiplication protocol (Algorithm 8), which returns partial carry arrays $\{p_1, p_2\}$. These are combined into $p = p_1 \bigoplus p_2$, indicating the positions requiring carry propagation.

Algorithm 7 Secure Bit Addition

1: **Input:** Bit arrays $\{T_i, r_i\}$
2: **Output:** Boolean π indicating whether $(T + r)$ is positive
3: $p_i \leftarrow$ Secure_Bit_Multiplication(T_i, r_i)
4: $z_i \leftarrow T_i \bigoplus r_i$
5: Left-shift p_i by one bit
6: Servers exchange p_i
7: **while** $p_1 \bigoplus p_2 \bigoplus \cdots \bigoplus p_n \neq 0$ **do**
8: $z_i \leftarrow z_i \bigoplus p_i$
9: $p_i \leftarrow$ Secure_Bit_Multiplication(z_i, p_i)
10: Left-shift p_i and forward to other servers
11: **end while**
12: **if** $z_i < 0$ **then**
13: $\pi_i \leftarrow 1$
14: **else**
15: $\pi_i \leftarrow 0$
16: **end if**
17: Servers exchange π_i and compute $\pi = \pi_1 \bigoplus \pi_2 \bigoplus \cdots \bigoplus \pi_n$
18: **return** π

Line 12 of Algorithm 7 performs a left-shift operation on p_i to ensure that the carry bit is correctly propagated to the next more significant position. In bitwise representation, the most significant bit (MSB) denotes the sign: a value of 0 indicates a positive number, while 1 indicates a negative number.

Rather than exchanging z_i directly, each server uses the local boolean π_i to represent its partial contribution to the overall sign bit. The global boolean π thus reflects the bitwise sign of the reconstructed secret value z, which is consistent with the sign of the original secret x. This protocol enables all parties to infer the sign of the secret value without ever reconstructing or revealing the full value itself.

5.2.3 Secure Bit Multiplication. The Secure Bit Multiplication protocol receives bit arrays $\{T_i, r_i\}$ and computes a partial carry array p_i for each edge server S_i, where $i \in \{1, \ldots, n\}$. When the resulting values p_i are aggregated using XOR, the output corresponds to the bit-wise multiplication of the original arrays T and r—a key step required for secure carry computation in Secure Bit Addition.

To ensure privacy, the protocol leverages additional random parameters generated by a trusted configuration server: bit arrays n, β, h, and a random selection index χ. The correctness of the protocol relies on the distributive properties of bit-wise logical operations, particularly conjunction (AND) and disjunction (OR).

As detailed in Algorithm 8, the value $\chi \in \{1, \ldots, n\}$ is randomly chosen to determine which party includes an additional term $\theta = A \bigotimes B$ in its output. Since θ is XORed across the shares, it cancels out during aggregation, meaning its inclusion on a single server does not affect the correctness of the overall result. Here, $\bigotimes$ denotes bit-wise multiplication.

6 Performance Evaluation

To evaluate the performance of the Privacy Preserving Driver Monitoring System (PDMS), we conducted experiments on a vehicle-edge testbed. The testbed includes a Polaris GEM e4 electric vehicle equipped with Sekonix cameras for face monitoring and the on-vehicle processing unit is the NVIDIA Jetson AGX Xavier. The edge

Algorithm 8 Secure Bit Multiplication

Input: Bit arrays $\{T_i, r_i\}$
Output: Partial carry $p_i = T \bigotimes r$
Configuration server TS generates random bit arrays:
- $n = n_1 \bigoplus n_2 \bigoplus \cdots \bigoplus n_n$
- $\beta = \beta_1 \bigoplus \beta_2 \bigoplus \cdots \bigoplus \beta_n$
- $h = h_1 \bigoplus h_2 \bigoplus \cdots \bigoplus h_n$, where $h = \beta \bigotimes n$

Randomly sample $\chi \in \{1, \ldots, n\}$
Each server S_i computes:

$$A_i = T_i \bigoplus \beta_i, \quad B_i = r_i \bigoplus n_i$$

Servers exchange A_i and B_i to reconstruct:

$$A = T \bigoplus \beta, \quad B = r \bigoplus n$$

Compute local terms:

$$Q_i = \beta_i \bigotimes A, \quad W_i = n_i \bigotimes B, \quad \theta = A \bigotimes B$$

if $i = \chi$ **then**
    $p_i \leftarrow h_i \bigoplus Q_i \bigoplus W_i \bigoplus \theta$
else
    $p_i \leftarrow h_i \bigoplus Q_i \bigoplus W_i$
end if
return p_i

server used in the experiment is powered by an AMD Ryzen 7 processor with 6 cores running at 3.2 GHz, 16 GB of DRAM, and an NVIDIA GeForce GTX 1660 Super GPU, operating on Ubuntu Linux v20.04 and Python v3.8.

For the experiments, we adopted Gaze360 datasets [17] and LISA Gaze datasets [38]. The Gaze360 Dataset is a large-scale gaze estimation dataset designed for unconstrained environments, containing over 170,000 images from diverse subjects with annotations for 3D gaze vectors across a 360-degree field of view. The LISA Gaze Dataset comprises recordings from drivers in real-world driving scenarios.

In the experiments, we implemented a privacy preserving gaze transformer as a proof of concept. The model architecture consists of two ViT encoders combined with a gaze estimation detection network comprising fully connected layers. The input data is the secret share of an RGB video frame image with shape of (224,224,3) and patch size (16,16). The output is a 2D vector representing normalized coordinates of the estimated gaze location.

The model parameters were initialized using a pre-trained model with four transformer blocks, trained on 8,000 frames from the training dataset and validated on 2,000 frames. These frames were carefully selected to ensure diversity in age, gender, ethnicity, and gaze directions, providing a robust evaluation across various real-world conditions. The experiments aimed to validate the accuracy, security, and efficiency of the privacy preserving gaze transformer, demonstrating that the PDMS is a practical and effective framework.

6.1 Security Analysis

PDMS employs additive secret sharing to preserve the confidentiality of driver data during the inference process. We consider a semi-honest (honest-but-curious) adversary model, where each server S_i follows the protocol correctly but may attempt to infer private information from its local data view.

Fig. 6. The testbed includes a Polaris GEM e4 electric vehicle equipped with Sekonix cameras for face monitoring. The on-vehicle processing unit is the NVIDIA Jetson AGX Xavier.

Let $x \in \mathbb{R}^d$ represent a driver-related input (e.g., a face frame), which is secret-shared as $x = x_1 + x_2$, with x_1 and x_2 distributed to servers S_1 and S_2, respectively. Additive secret sharing ensures that each share is uniformly random and independently distributed, satisfying:

$$\Pr[X_i = x_i] = \text{Uniform}(\mathbb{R}^d), \quad \text{for } i \in \{1, 2\}$$

This guarantees that no single server can reconstruct or infer any information about x without access to the full set of shares.

In the PDMS architecture, linear operations (e.g., matrix multiplications and convolutions) are performed using efficient secure protocols with low communication overhead. These operations can also be accelerated using GPU parallelism without compromising privacy, as no data leakage occurs through local processing. For non-linear operations (such as GELU and Softmax), we employ Taylor series approximations in combination with Beaver's triples, which involve interaction between servers. Despite their higher communication cost, these secure protocols ensure that intermediate values remain hidden throughout the computation.

Simulation-based security is achieved: for each server S_i, there exists a simulator $\mathcal{S}_i$ that can reproduce its view using only the public output $f(x)$, such that:

$$\text{View}_{S_i}(x_i, f_i(x_i)) \approx_c \mathcal{S}_i(f(x))$$

where $\approx_c$ denotes computational indistinguishability under the standard cryptographic security parameter.

The system is also resistant to feature inversion attacks, where an adversary attempts to reconstruct input data by analyzing intermediate representations or gradients. Formally, such an attack seeks to find an approximation $\hat{x}$ of the true input x by solving:

$$\hat{x} = \arg\min_{x'} \|f(x') - f(x)\|^2$$

where $f(x)$ denotes the model's observable output (e.g., logits or features). In traditional (non-private) inference settings, attackers can use optimization-based inversion (e.g., gradient descent) or direct linear reconstruction, which typically incur time complexity of $O(n \cdot d)$ to $O(n \cdot d^2)$, depending on model depth n and input dimension d. However, in PDMS, all intermediate computations are performed over secret shares, and no single server observes $f(x)$ or any partial activations in plaintext. This eliminates the attacker's ability to evaluate the loss or perform updates, making inversion attacks computationally infeasible under our threat model.

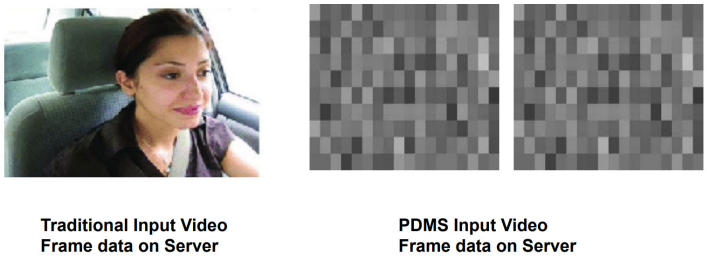

Fig. 7. Traditional Input Video Frame on Server vs PDMS Input Video Frame on Server

Table 1. Comparison of Gaze Estimation Results

	White	Black	Asian	Hispanic
Young	3.75e-06	9.51e-07	7.32e-06	5.99e-07
Mid	1.56e-07	1.56e-06	5.81e-07	8.66e-07
Old	6.01e-06	7.08e-07	2.06e-07	9.70e-06

While the current implementation uses two servers, the PDMS design and underlying protocols naturally support n-party configurations using threshold or replicated secret sharing. This allows tolerance against up to $t - 1$ colluding servers. A discussion of the trade-offs between communication cost and security level in multi-party extensions is included in Section 8.

In conclusion, PDMS provides strong confidentiality guarantees against passive adversaries, protects both linear and non-linear operations through secure computation, and defends against advanced attacks such as input reconstruction and feature inversion.

6.2 Accuracy

We evaluated the performance of the privacy preserving gaze transformer by comparing the Euclidean distance between the estimated gaze and ground truth with that of the baseline model. The experimental results demonstrate that the average difference is negligible, on the order of e^{-6}. Figure 8 demonstrates a sample output, while Table 1 provides a detailed comparison of the results.

Given that gaze estimation is sensitive to a driver's facial features, we further grouped the data by race and age and compared the average Euclidean distance across these groups. The results confirm that, in all cases, the privacy preserving gaze transformer achieves a comparable level of accuracy to the baseline model. This effectiveness is due to the additive secret sharing mechanism, which randomly partitions the raw data's features, ensuring that variations in facial attributes such as skin color and age do not impact the model's output.

To further evaluate the fidelity of PDMS compared to the non-private baseline, we report the difference in two key metrics: the mean Euclidean distance and standard deviation of the gaze prediction error. PDMS exhibits only a marginal increase of 0.07 pixels in average Euclidean distance and 0.04 pixels in standard deviation compared to the base model. Furthermore, we report the Cumulative Angular Accuracy (CAA), a common metric in gaze estimation that measures the percentage of predictions within a given angular threshold. PDMS demonstrated a 2.4% lower CAA within 5 degrees and a 1.1% lower CAA within 15 degrees compared to the baseline. This demonstrates that the privacy-preserving design of PDMS introduces negligible accuracy degradation, while maintaining high consistency in prediction quality.

Fig. 8. Gaze Estimation Result with PDMS

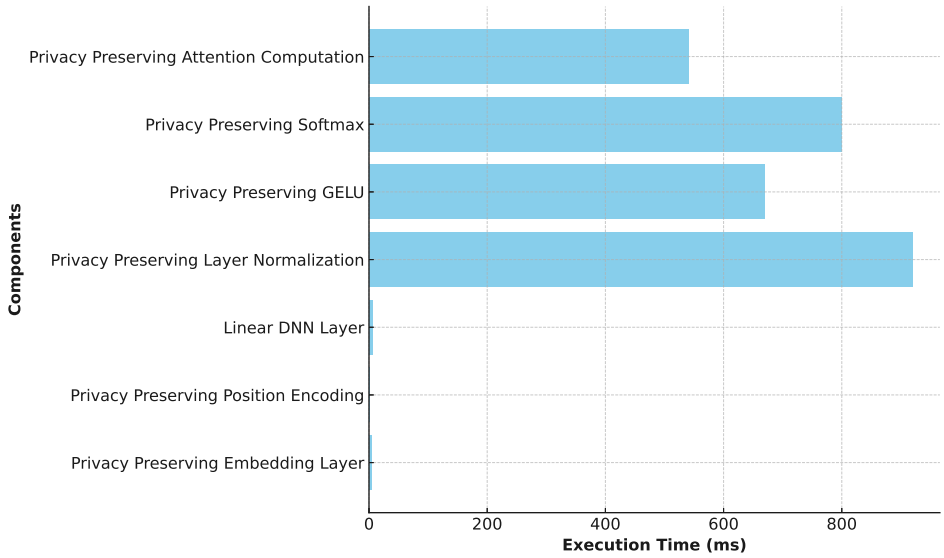

Fig. 9. Execution Time breakdown in Privacy Preserving Gaze Transformer

6.3 Execution Time

Figure 9 demonstrates the inference time of 3.472 seconds for processing a single frame within a 20 FPS video, based on average results. The primary source of latency arises from the non-linear privacy preserving DNN layers, which heavily rely on secure multiplication computations. Examples include the Privacy Preserving GELU and Privacy Preserving Softmax layers. Figure 10 illustrates the execution time of Secure Multiplication as the number of invocations increases.

Figure 11, 12 present the performance of the Privacy Preserving GELU and Privacy Preserving Softmax layers. Notably, the Privacy Preserving Softmax layer exhibits longer execution times, primarily due to its additional iterations over the dataset.

Figure 13 and 14 compares the performance of two approaches for Privacy Preserving Layer Normalization, as discussed in Section V. The optimized version outperforms the default implementation due to a reduced number of invocations on the Secure Multiplication protocol.

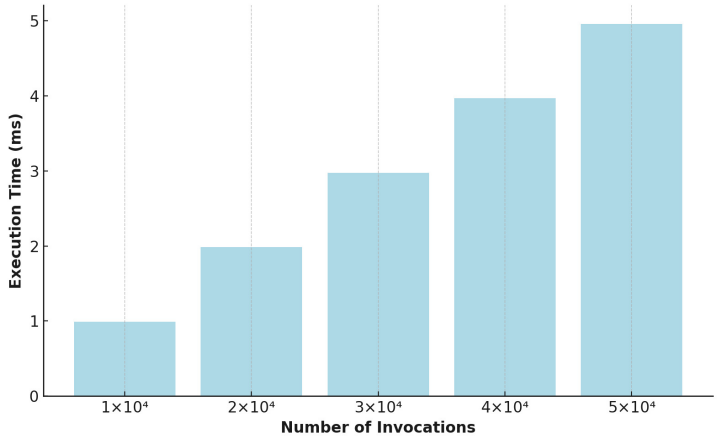

Fig. 10. Performance of Secure Multiplication

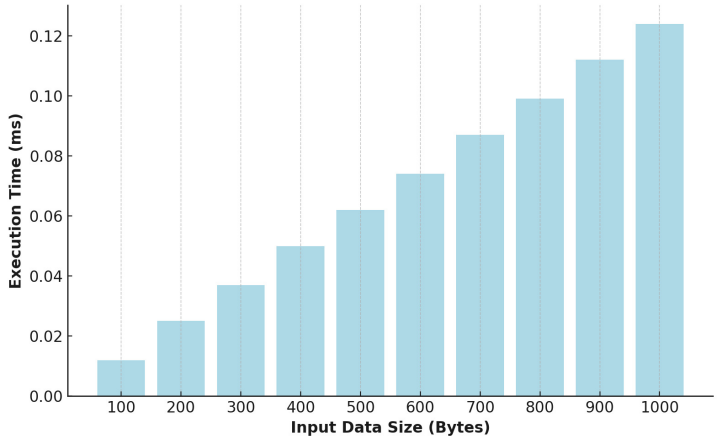

Fig. 11. Performance of Privacy Preserving GELU

Table 2 presents the execution time breakdown for the privacy preserving FlowNet, which consists of 10 privacy preserving convolutional layers for feature extraction, 3 privacy preserving deconvolutional (transposed convolution) layers for upsampling, and 9 privacy preserving GELU activation layers.

Compared to the base model, there is no difference in the convolutional and deconvolutional layers, as both use the standard Conv2d and ConvTranspose2d implementations in PyTorch, which are linear transformations. Therefore, the privacy preserving FlowNet and the base model perform similarly in these layers. However, the privacy preserving GELU layers, which internally utilize a secure multiplication protocol, introduce a significant delay in the inference process of the privacy preserving FlowNet.

Table 3 presents the execution time breakdown for both the privacy preserving YOLO and the base YOLO model. The privacy preserving YOLO architecture consists of 9 privacy preserving convolutional layers, 9 privacy preserving GELU activation layers, 5 privacy preserving max pooling layers, and a secure detection layer. Compared to the base model, the primary source of overhead comes from the privacy preserving GELU layers,

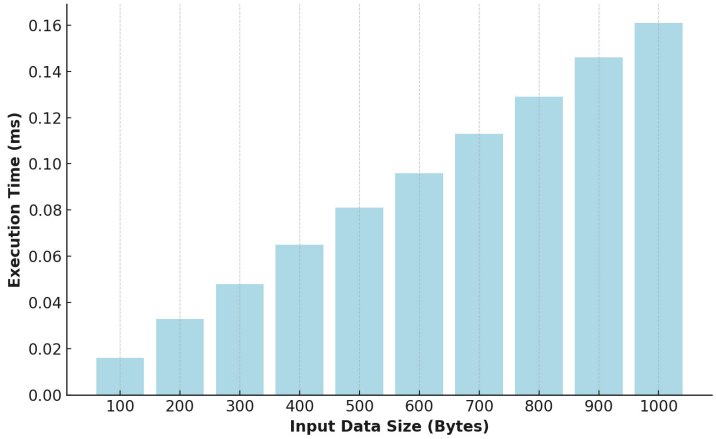

Fig. 12. Performance of Privacy Preserving Softmax

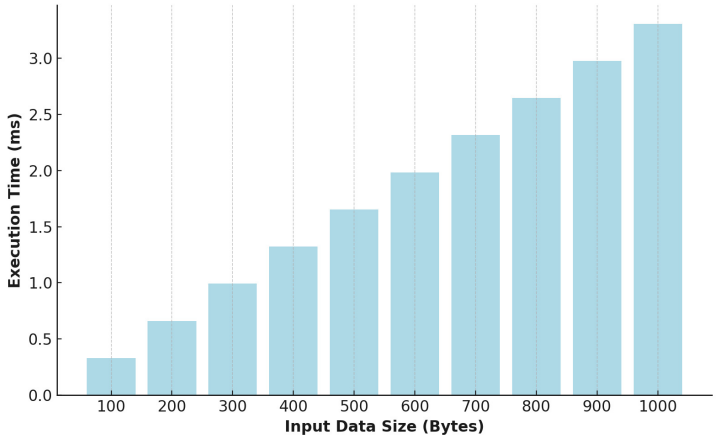

Fig. 13. Performance of Privacy Preserving Layer Normalization Layer

which is consistent with the slowdown observed in the privacy preserving FlowNet. However, an additional significant slowdown arises from the privacy preserving max pooling layers. This is due to the secure protocol used in max pooling, which requires computing and comparing differences between secret share data within the pooling window.

Notably, by replacing ReLU with GELU in both privacy preserving YOLO and privacy preserving FlowNet, we achieve a considerable speedup. The privacy preserving GELU implementation is approximately six times faster privacy preserving ReLU. This improvement stems from the fact that privacy preserving ReLU requires complex bitwise operations, including Secure Bit Array Generation, Secure Bit Addition, and nested Secure Bit Multiplication. These nested bitwise computations significantly increase the computational complexity of privacy preserving ReLU, whereas privacy preserving GELU avoids such heavy bit-level protocols, resulting in a more efficient implementation.

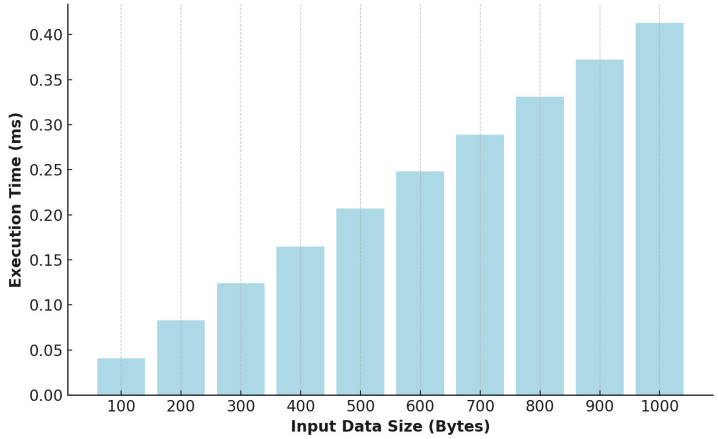

Fig. 14. Performance of Optimized Privacy Preserving Layer Normalization Layer

Table 2. Execution time comparison between the Privacy Preserving FlowNet and Base Model

Model	Component	Count	Time (ms)
Privacy Preserving FlowNet	Privacy Preserving Convolutional Layers	10	70
	Privacy Preserving Deconvolutional Layers	3	36
	Privacy Preserving GELU	9	1830
	Total	–	**1936**
FlowNet(Base Model)	Convolutional Layers	10	64
	Deconvolutional Layers	3	32
	GELU Activations (Default)	9	2
	Total	–	**108**

Table 3. Execution time comparison between the Privacy Preserving YOLO and Base Model

Model	Component	Count	Time (ms)
Privacy Preserving YOLO	Privacy Preserving Convolutional Layers	9	62
	Privacy Preserving GELU	9	230
	Privacy Preserving Max Pooling Layers	5	125
	Privacy Preserving Detection Layer	1	3
	Total	–	**420**
YOLO (Base Model)	Convolutional Layers	9	54
	GELU Layers	9	2
	Max Pooling Layers	5	5
	Detection Layer	1	3
	Total	–	**80**

Privacy Preserving FlowNet and Privacy Preserving YOLO serve as key components in the PDMS preprocessing module.Although the use of these components is optional, since privacy preserving gaze estimation can be applied directly to each frame, their integration leads to a significant improvement in overall performance. Compared to

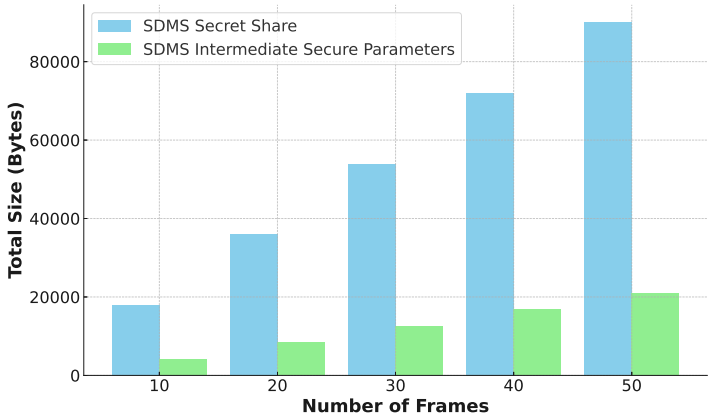

Fig. 15. Storage Overhead Of Privacy Preserving Gaze Transformer

the frame-by-frame approach, the average processing time for a 10-minute video at 20 FPS is improved by 42.03%. This demonstrates that the design effectively boosts system efficiency, particularly for video streams where the driver's movement is infrequent.

Finally, Figure 15 highlights the average storage overhead incurred from the generated secret shares of raw data and the additional intermediate secure parameters, analyzed across varying numbers of frames.

7 Discussion

7.1 Comparision between PDMS and TEE

Trusted Execution Environments (TEEs) [31] such as Intel SGX and ARM TrustZone, provide hardware-level isolation to securely process sensitive data. While TEEs offer security guarantees, they face practical challenges in real-world deployment. First, TEEs are vulnerable to a range of side-channel attacks, including cache timing and speculative execution exploits, which may compromise the confidentiality of processed data. Second, TEEs often have limited memory and are not optimized for compute-intensive tasks like deep learning inference, especially in latency-sensitive vehicle scenarios. In comparison, our PDMS framework leverages additive secret sharing to distribute computation on edge nodes without reliance on specialized hardware. It provides a readily deployable and scalable solution.

7.2 Network Connectivity

PDMS is designed to offload computationally intensive and privacy-preserving inference tasks to edge servers. This approach significantly reduces the processing burden on the vehicle. It requires network connectivity for transmitting secret-shared representations of video frames. In locations where connectivity is weak or intermittent—such as remote or rural driving environments—PDMS will adapt by reducing the resolution of input frames, the overall input image size, and/or the frame sampling rate. These adaptations will lower communication overhead and enable baseline functionality under constrained conditions without compromising privacy. Exploring adaptive streaming strategies and bandwidth-aware optimization techniques will be further studied to enhance PDMS's robustness in low-connectivity settings.

7.3 Scalable Secure Inference Frameworks for Vehicle Applications

While Vision Transformers (ViTs) are general-purpose vision models not inherently designed for automotive use, our objective in PDMS is to establish a modular, privacy-preserving framework that can be adapted to driver monitoring systems (DMS). ViT offers a well-understood, scalable architecture suitable for integration with secure computation protocols. By applying our privacy-preserving ViT-based model to gaze estimation—a critical DMS task—we show that this general architecture can be effectively adapted to the vehicular context. This choice also lays the groundwork for incorporating alternative or vehicle-specific models in future iterations of PDMS without altering the secure design principles.

7.4 Inference Speed Optimization

The previous evaluation was conducted using the default configuration of the privacy preserving gaze transformer model. However, in certain application scenarios, faster inference speed can be prioritized over slight reductions in accuracy. To explore this trade-off, we applied two key modifications to the original model configuration::

- **Reduced Input Image Size:**
  The original input image size of 224×224 was reduced to 128×128. This significantly lowers the number of image patches (tokens) processed by the transformer, leading to a notable reduction in computation. To accommodate the new input shape, the positional embedding layer was resized accordingly, without modifying the internal transformer architecture.
- **GPU Acceleration of Secure Layers:**
  While the linear components of the model, such as convolutional and fully connected layers, were already running on GPU, the secure layers—including GELU and Softmax functions implemented via cryptographic protocols like Beaver's triples and Taylor series approximations—were previously executed on CPU. We migrated the computation-heavy portions of these secure layers to GPU while keeping communication routines on CPU.

Table 4. Inference Speed and Accuracy Trade-offs for Privacy-Preserving Gaze Transformer

Configuration	Input Size	Transformer Blocks	Speed-up (%)	Accuracy Drop (%)
Base model (Default)	224×224	4	0%	-
GPU Acceleration on Secure DNN Layers	224×224	4	21.3%	0
Reduced Input Size	128×128	4	40.9%	8
Reduced Input + GPU Acceleration	128×128	4	51.4%	8

By reducing the input image size from 224×224 to 128×128 and enabling GPU acceleration for secure layers, we observed a significant improvement in inference speed (Table 4). The average per-frame inference time was reduced from 3.47 seconds to 1.69 seconds, representing an overall speed-up of approximately 51.4%.

While a moderate drop in accuracy of approximately 8% was observed due to the reduced input resolution, this trade-off is reasonable in latency-sensitive applications. Importantly, these optimizations preserve the privacy guarantees of PDMS, as the cryptographic protocols and secure computation mechanisms remain intact. This demonstrates the system's flexibility to adapt to various deployment constraints by balancing inference latency and predictive accuracy.

In addition to reducing the input image size and accelerating secure layers via GPU, another approach to improve inference speed is to reduce the number of transformer blocks in the Vision Transformer model. Lightweight ViT architectures—such as TinyViT [40]—commonly use between 1 and 3 transformer blocks to achieve efficient inference on resource-constrained devices. Following this design principle, we experimented with reducing the number of transformer blocks from 4 to 2. This configuration led to an inference speed-up of approximately

31.4%, decreasing the per-frame latency from 3.47 seconds to 2.38 seconds. However, this change also introduced a more pronounced accuracy drop of around 13%, which was larger than the drop observed from reducing input size alone. These results suggest that while transformer block reduction can improve performance, it requires careful consideration of the trade-off between latency and model accuracy.

We also explored reducing the number of transformer blocks from 4 to 2, following the design of lightweight ViT models such as TinyViT [40], which often use 1–3 blocks. This modification reduced inference time by 31.4% but resulted in a larger accuracy drop of about 13%. While effective for acceleration, this approach introduces a steeper trade-off and is not used in the final configuration, though it remains a viable option depending on application needs.

In addition, our previously introduced privacy-preserving FlowNet and YOLO components help reduce redundant frame processing by detecting and skipping frames with minimal visual change. This preprocessing pipeline contributed an additional 42.03% speed-up in continuous video stream scenarios, further enhancing the overall system efficiency during real-world deployment.

7.5 Inference Efficiency and Applicability

Although the current performance of PDMS may not be sufficient for mission-critical applications like autonomous driving, it is well suited for non-mission-critical applications, such as driver behavior analysis. Tasks such as post-trip evaluation, fatigue trend monitoring, and long-term driving habit profiling prioritize privacy over immediate response, making PDMS relevant even in scenarios with limited connectivity or higher latency. To further enhance efficiency, we have integrated a preprocessing pipeline using privacy-preserving YOLO and FlowNet to detect and filter frames with minimal change, which reduces redundant processing and achieves an additional 42.03% speed-up in video stream analysis. Moreover, we have evaluated a lightweight configuration of the PDMS model by reducing the input image size from 224×224 to 128×128 and enabling GPU acceleration for secure layers. This combination yielded a 51.40% reduction in inference time (from 3.47 seconds to 1.69 seconds per frame). These optimizations preserve the underlying secure computation protocols and demonstrate that PDMS can be optimized for scenarios where faster inference is desired.

7.6 Extending PDMS to Multiple Edge Servers

While our current implementation of PDMS employs a two-server model, the underlying secure protocols are naturally extensible to n-server configurations. Adopting multiple edge servers increases resilience against collusion and enhances the overall security of the system—as long as not all servers are compromised simultaneously. For instance, threshold secret sharing and replicated secret sharing schemes can tolerate up to t colluding servers out of n, offering a stronger security guarantee.

An increase in the number of edge servers (n) can enhance system security and resilience against collusion. For instance, a configuration with $n = 5$ servers can tolerate up to $t = 4$ colluding servers without compromising privacy. However, this improvement in security comes at a cost. A larger n introduces greater communication overhead during the execution of secure operations, additional secret shares, and increased complexity in generating and managing intermediate secure parameters, all of which contribute to higher inference latency.

Considering that real-time driver monitoring applications demand timely feedback, the two-server setting ($n = 2, t = 1$) represents a practical balance between privacy and efficiency. It prevents single-point privacy breaches while maintaining low coordination and communication overhead. Furthermore, in typical vehicular deployments, the probability of coordinated compromise across multiple geographically distributed edge nodes (such as roadside units or MEC servers) is relatively low, making $t = 1$ a reasonable and effective threat model for most operational scenarios.

The optimal number of servers (n) is context dependent. In high-security environments, such as government or commercial fleets with elevated insider-threat risks, a larger n may be justified despite increased latency. To systematically examine this trade-off, we are extending our evaluation to quantify the relationship between privacy resilience (as a function of t) and system latency for configurations where $n \in \{2, 3, 4, 5\}$. This analysis incorporates realistic models of network heterogeneity, server reliability, and application-specific timing constraints, and will guide adaptive deployment strategies where n can be tuned at provisioning time based on risk tolerance and Quality-of-Service (QoS) requirements. Such flexibility allows PDMS to achieve a balanced integration of enhanced privacy and real-time operational feasibility.

7.7 Limitations

The current end-to-end inference latency of PDMS is approximately 1 second per frame after applying system-level optimizations such as GPU acceleration, input resolution reduction, and selective frame processing. Although this represents a substantial improvement over naive secure inference baselines, it remains above the threshold typically required for responsive in-vehicle DMS applications.

A detailed profiling of our PDMS prototype indicates that latency arises from three interconnected sources, with cryptographic operations playing a central but not exclusive role. First, the privacy-preserving Vision Transformer relies on secure protocols for nonlinear operations such as GELU, softmax, and layer normalization. These non-polynomial functions require interactive cryptographic primitives, including Beaver triples and secure comparison, which introduce multiple rounds of inter-server communication and local computation. Second, secure multiparty operations require precise coordination between edge servers. Even with low-latency edge networks (10–30 ms RTT), repeated synchronization across many rounds of a deep ViT model introduces pipeline stalls and reduces throughput. Third, the lack of native hardware acceleration for secret-sharing primitives limits efficiency. Unlike TEE-based solutions that execute standard neural networks within trusted enclaves at near-native speed, PDMS operates over secret shares that cannot directly leverage optimized deep learning libraries, leading to suboptimal GPU utilization even after linear-layer parallelization.

Compared with TEE-based methods, PDMS highlights an intrinsic trade-off between performance and privacy. While TEEs achieve lower latency, they provide only computational privacy and remain susceptible to side-channel attacks, physical probing, and firmware-level exploits. In contrast, PDMS provides information-theoretic protection of input data, guaranteeing privacy even if every edge server except one becomes compromised. This level of assurance is particularly important in open, multi-tenant edge environments where the integrity of TEE attestation cannot always be verified or trusted.

To bridge the performance gap while maintaining this strong privacy model, we are exploring several optimization directions. Preprocessing and pipelining of correlated randomness, such as Beaver triples, can decouple setup from online inference by pre-generating and caching secure parameters during idle periods. Layer fusion and approximation, for instance replacing exact secure softmax with low-degree polynomial approximations derived from Chebyshev interpolation, can reduce communication rounds with minimal impact on gaze estimation accuracy. Finally, multi-frame pipeline parallelism, where consecutive frames are processed across disjoint server groups, can scale throughput and amortize coordination overhead.

Second, PDMS relies on stable network connectivity to transmit secret-shared video data from the vehicle to edge servers. In remote or low-bandwidth environments, this can limit the system's responsiveness. To address this, PDMS supports adaptive adjustments such as reducing image resolution, input size, and frame sampling rate. Future work includes developing bandwidth-aware streaming and fallback mechanisms for minimal on-vehicle inference.

Third, the system operates under the Honest-But-Curious (HBC) model, where edge servers follow protocols but may attempt to infer private information. Although this model aligns with many secure multi-party computation

settings, stronger adversarial models (e.g., malicious servers) will require more robust protocols such as zero-knowledge proofs or verifiable computation—currently beyond the scope of this work.

Finally, secure operations—particularly nonlinear layers like GELU and Softmax—introduce computational overhead due to their reliance on Beaver's triples and polynomial approximations. This remains a key bottleneck. We are exploring approximation techniques and hybrid secure execution models to reduce this cost.

To overcome these challenges, future work will investigate enhanced secure protocols, parallel multi-server processing, and adaptive frame selection strategies. Together, these directions aim to strengthen PDMS's scalability, security, and suitability for real-time deployments.

8 Related Work

In recent years, several innovative approaches have been proposed to enable privacy preserving inference [2–5] in transformer models using homomorphic encryption [24, 27]. One notable contribution is the paper titled "THE-X: privacy preserving Transformer Inference with Homomorphic Encryption," [9] which introduces a framework that leverages fully homomorphic encryption (FHE) [13] to ensure the confidentiality of data during inference. THE-X effectively approximates non-polynomial functions within transformers to accommodate FHE, allowing secure computations without exposing sensitive information. This approach, however, faces computational overhead, with the approximate inference time being around 5.14 seconds per sample, reflecting the challenges of balancing privacy and efficiency in encrypted environments

Another significant work in this domain is "BumbleBee: Secure Two-party Inference Framework for Large Transformers" [21]. This framework focuses on optimizing secure protocols to enable efficient and private inference on large transformer models. BumbleBee employs advanced protocols for matrix multiplication and non-linear activation functions, achieving substantial improvements in communication cost and processing speed compared to previous methods. Notably, BumbleBee demonstrates an ability to generate one token from the LLaMA-7B [36] model in approximately 14 minutes on CPUs, showcasing its practicality and efficiency in privacy preserving applications

Both THE-X and BumbleBee mark significant progress in secure inference for Transformer models, effectively addressing the critical demand for privacy in machine learning applications. However, both approaches suffer from prolonged execution times, making them unsuitable for processing continuous video frame data captured from vehicles. They also underscore the persistent trade-offs between ensuring security, maintaining efficiency, and managing computational complexity.

Several studies have explored de-identification techniques to preserve privacy in video data [18, 20, 22, 25]. For example, the paper "Face De-Identification with Generative Adversarial Networks" [20] introduces a framework that uses GANs to modify facial images by replacing sensitive features with synthesized, non-identifiable ones, while preserving their usability for downstream tasks. Similarly, "A De-Identification Face Recognition Using Extracted Thermal Features Based on Deep Learning" [19] proposes a method to anonymize facial data by leveraging thermal imaging and deep learning to extract and process non-identifiable features. However, both methods require deploying additional deep learning models on the vehicle to process and replace sensitive information. This imposes significant computational overhead and latency, making these approaches unsuitable for resource-constrained environments such as vehicle-based driver monitoring systems.

Additional Recent studies have explored various driver monitoring systems (DMS) to assess driver attention, fatigue, and gaze direction. For instance, traditional DMS solutions often rely on camera-based systems that detect eye closure or head pose using machine learning techniques [39]. Others have explored deep learning models for gaze estimation in the wild [42] or built-in commercial platforms for real-time driver safety monitoring [8].

However, these solutions often process sensitive visual data in the cloud [12, 26, 41] or onboard without incorporating privacy-preserving mechanisms. Our work complements these efforts by introducing privacy-aware gaze inference using secure multi-party computation.

9 Conclusion

In this work, we propose the Privacy Preserving Driver Monitoring System (PDMS) to address privacy concerns associated with sending drivers' video data to remote servers for processing. PDMS also alleviates the burden on vehicles that must perform heavy deep learning tasks locally to avoid leaking sensitive information. The detailed architecture and design of privacy preserving deep neural network (DNN) layers are introduced. Experiments conducted using an PDMS implementation for gaze estimation demonstrate the system's effectiveness and practicality.

10 Acknowledgment

We thank the reviewers for their insightful and constructive feedback, which has substantially strengthened the paper. This work is supported in part by the U.S. National Science Foundation (NSF) under grants CNS-2231519, CNS-2113805, OAC-2017564, CNS-2037982, DUE-2225229 and CCF-2447834.

References

[1] Dosovitskiy Alexey. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929* (2020).
[2] Tianyu Bai, Yunhe Feng, and Song Fu. 2025. Safeguarding user data privacy in online Large Language Model services. *Journal of Systems Architecture* (2025), 103555.
[3] 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.
[4] Tianyu Bai, Danyang Shao, Ying He, Song Fu, and Qing Yang. 2023. 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.
[5] Tianyu Bai, Qing Yang, and Song Fu. 2023. User-Defined Privacy Preserving Data Sharing for Connected Autonomous Vehicles Utilizing Edge Computing. (2023), 145–157.
[6] Donald Beaver. 1992. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology—CRYPTO'91: Proceedings 11*. Springer, 420–432.
[7] Anushka Biswas and Hwang-Cheng Wang. 2023. Autonomous vehicles enabled by the integration of IoT, edge intelligence, 5G, and blockchain. *Sensors* 23, 4 (2023), 1963.
[8] Christian Braunagel, Enkelejda Kasneci, Wolfgang Stolzmann, and Wolfgang Rosenstiel. 2017. Driver-activity recognition in the context of conditionally autonomous driving. In *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1287–1292.
[9] Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. The-x: Privacy-preserving transformer inference with homomorphic encryption. *arXiv preprint arXiv:2206.00216* (2022).
[10] 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.
[11] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. 2015. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 2758–2766.
[12] Song Fu and Cheng-Zhong Xu. 2005. Service migration in distributed virtual machines for adaptive grid computing. In *IEEE International Conference on Parallel Processing*.
[13] Craig Gentry. 2009. *A fully homomorphic encryption scheme*. Stanford University.
[14] Ross Girshick. 2015. Fast r-cnn. In *IEEE ICCV*.
[15] 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.
[16] Berthold KP Horn and Brian G Schunck. 1981. Determining optical flow. *Artificial intelligence* 17, 1-3 (1981), 185–203.
[17] Petr Kellnhofer, Adrià Recasens, Simon Stent, Wojciech Matusik, and Antonio Torralba. 2019. Gaze360: Physically Unconstrained Gaze Estimation in the Wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 6912–6921.

[18] Jinsu Kim and Namje Park. 2022. De-identification mechanism of user data in video systems according to risk level for preventing leakage of personal healthcare information. *Sensors* 22, 7 (2022), 2589.
[19] Chih-Hsueh Lin, Zhi-Hao Wang, and Gwo-Jia Jong. 2020. A de-identification face recognition using extracted thermal features based on deep learning. *IEEE Sensors Journal* 20, 16 (2020), 9510–9517.
[20] Jiacheng Lin, Yang Li, and Guanci Yang. 2021. FPGAN: Face de-identification method with generative adversarial networks for social robots. *Neural Networks* 133 (2021), 132–147.
[21] Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. 2023. Bumblebee: Secure two-party inference framework for large transformers. *Cryptology ePrint Archive* (2023).
[22] Amit Maity, Rishi More, Gitesh Kambli, and Sarita Ambadekar. 2023. Preserving privacy in video analytics: A comprehensive review of face de-identification and background blurring techniques. *Authorea Preprints* (2023).
[23] Ioannis Mavromatis, Andrea Tassi, Giovanni Rigazzi, Robert J Piechocki, and Andrew Nix. 2018. Multi-radio 5G architecture for connected and autonomous vehicles: Application and design insights. *arXiv preprint arXiv:1801.09510* (2018).
[24] Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. 2024. THOR: Secure Transformer Inference with Homomorphic Encryption. *Cryptology ePrint Archive* (2024).
[25] Elaine M Newton, Latanya Sweeney, and Bradley Malin. 2005. Preserving privacy by de-identifying face images. *IEEE transactions on Knowledge and Data Engineering* 17, 2 (2005), 232–243.
[26] Husanbir S. Pannu, Jianguo Liu, and Song Fu. 2012. A self-evolving anomaly detection framework for developing highly dependable utility clouds. In *IEEE Global Communications Conference*.
[27] Prajwal Panzade, Javad Rafiei Asl, Daniel Takabi, and Zhipeng Cai. 2025. BlindTuner: On Enhancement of Privacy-Preserving Fine-tuning of Transformers based on Homomorphic Encryption. *IEEE Internet of Things Journal* (2025).
[28] Primesh Pathirana, Shashimal Senarath, Dulani Meedeniya, and Sampath Jayarathna. 2022. Eye gaze estimation: A survey on deep learning-based approaches. *Expert Systems with Applications* 199 (2022), 116894.
[29] Joseph Redmon, Santosh Divvala, et al. 2016. You only look once: Unified, real-time object detection. In *IEEE CVPR*.
[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* 28 (2015), 91–99.
[31] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. 2015. Trusted execution environment: What it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/Ispa*, Vol. 1. IEEE, 57–64.
[32] Paul M Schwartz and Daniel J Solove. 2011. The PII problem: Privacy and a new concept of personally identifiable information. *NYUL rev.* 86 (2011), 1814.
[33] Douglas Selent. 2010. Advanced encryption standard. *Rivier Academic Journal* 6, 2 (2010), 1–14.
[34] Sasha Targ, Diogo Almeida, and Kevin Lyman. 2016. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029* (2016).
[35] Brook Taylor. 1717. *Methodus incrementorum directa & inversa.* Inny.
[36] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[38] Sourabh Vora, Akshay Rangesh, and Mohan M. Trivedi. 2018. Driver Gaze Zone Estimation using Convolutional Neural Networks: A General Framework and Ablative Analysis. *IEEE Transactions on Intelligent Vehicles* 3, 3 (2018), 254–265.
[39] Siyu Weng and Shu Chien. 2021. Driver drowsiness detection via multi-stream convolutional neural networks. *Sensors* 21, 5 (2021), 1613.
[40] Tianlong Wu, Zhuangwei Wang, Xing Zhang, Junnan Tang, Xiaojun Huang, Richard Socher, and Saining Xie. 2023. TinyViT: Fast and Accurate ViT for Tiny Devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 14385–14395.
[41] Feng Xiao, Ze Xu, Yi Han, Yi Zhong, Yi Zheng, Mingxi Liao, Poshi Qin, and Yuan Wan. 2024. Machine Learning Based Driver Emotion Monitoring for Vehicular IoT. In *IEEE Vehicular Technology Conference*.
[42] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. 2017. MPIIGaze: Real-World Dataset and Deep Appearance-Based Gaze Estimation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 41. 162–175.
[43] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems* 30, 11 (2019), 3212–3232.

Received 31 March 2025; revised 10 October 2025; accepted 20 October 2025