Privacy-Preserving Intrusion Detection System for Internet of Vehicles using Split Learning

Paul Agbaje
University of Texas at Arlington
Texas, USA
pauloluwatowoju.agbaje@uta.edu

Afia Anjum
University of Texas at Arlington
Texas, USA
afia.anjum@uta.edu

Arkajyoti Mitra
University of Texas at Arlington
Texas, USA
arkajyoti.mitra@uta.edu

Sena Hounsinou
Metro State University
Minnesota, USA
sena.houeto@metrostate.edu

Ebelechukwu Nwafor
Villanova University
Pennsylvania, USA
enwafor@villanova.edu

Habeeb Olufowobi
University of Texas at Arlington
Texas, USA
habeeb.olufowobi@uta.edu

ABSTRACT

The Internet of Vehicles (IoV) is envisioned to improve road safety, reduce traffic congestion, and minimize pollution. However, the connectedness of IoV entities increases the risk of cyber attacks, which can have serious consequences. Traditional intrusion detection systems (IDS) transfer large amounts of raw data to central servers, leading to potential privacy concerns. Also, training IDS on resource-constrained IoV devices generally can result in slower training times and poor service quality. To address these issues, we propose a split learning-based privacy-preserving IDS that deploys IDS on edge devices without sharing sensitive raw data. In addition, we propose a regret minimization-based adaptive offloading technique that reduces the training time on resource-constrained devices. Our approach effectively detects anomalous behavior while preserving data privacy and reducing training time, making it a practical solution for IoV. Experimental results show the effectiveness of our approach and its potential to enhance the security of the IoV network.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; • **Computing methodologies → Neural networks**; • **Theory of computation** → *Online learning algorithms.*

KEYWORDS

Intrusion detection, Split learning, Internet of Vehicles, Adaptive offloading, Optimization

ACM Reference Format:
Paul Agbaje, Afia Anjum, Arkajyoti Mitra, Sena Hounsinou, Ebelechukwu Nwafor, and Habeeb Olufowobi. 2018. Privacy-Preserving Intrusion Detection System for Internet of Vehicles using Split Learning. In *Proceedings of The 10th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT'23).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The Internet of Vehicles (IoV) connects autonomous vehicles, sensor networks, embedded devices, and intelligent transportation systems to the Internet to support vehicular applications. These applications generate and share various types of information, including vehicle telemetry, environment, and location data. In doing so, they enable a wide range of new services to improve the safety, efficiency, mobility, and sustainability of vehicles and infrastructures [1]. Increasingly, the interconnectedness of IoV systems to the Internet extends the attack vectors and directions for exploitation to adversaries. IoV has become highly vulnerable to malicious adversaries due to reliance on wireless communication technologies with potential entry points, multiple communication endpoints, and the lack of security standards. An attack on compromised systems has the potential for personal harm or physical damage. Prior works have posited the importance of securing the IoV because a system failure directly affects user safety [15, 16]. A lack of adequate security measures can result in catastrophic consequences, including loss of life. Also, the data generated by IoV devices is often sensitive and requires privacy protection.

Previous works have explored attack mitigation methods in IoV, including encryption, authentication, and authorization protocols that prevent unauthorized access to IoV communication data and network resources [5, 13]; the blockchain technology providing tamper-proof records for transactions to avoid data tampering and identity theft [11]; and intrusion detection systems (IDS) [17]. An IDS offers real-time monitoring and identifies potential threats early, enabling entities in the IoV to respond accordingly and prevent damage to the data and infrastructure. However, traditional IDS require large amounts of data to be transferred to a central server for processing, creating vulnerabilities that attackers can exploit. Split learning (SL) is a privacy-preserving machine learning (ML) technique that allows a model to be trained without sharing the raw data with the central server. The key idea of SL is that the model is split into two parts: the client device, such as the vehicles, that performs local computation on its data, and the server, such as the roadside unit (RSU) and edge devices, that uses the extracted features to update the global model. This approach enables training

ML models on decentralized data without transferring or sharing sensitive data.

This paper proposes a SL-based privacy-preserving intrusion detection system for IoV data communication to detect anomalous network behavior. Our proposed approach uses a client-server architecture, where client-side models are trained on local data sources, and the server-side model is trained on aggregated, encrypted updates from the client-side models. This decentralized training approach enables the IDS to maintain data privacy while still providing high-quality detection of intrusion attempts. To improve the efficiency of the split learning process, we present an adaptive training process that dynamically chooses the optimal strategy that minimizes delay using a regret minimization technique. To evaluate the effectiveness of our approach, we use open-source datasets representing intra- and inter-vehicular networks and assess our technique's accuracy, efficiency, and scalability. Moreover, we compare with baseline methods to determine suitability for practical deployment in real-world IoV environments. Experimental results show that our proposed approach can effectively detect anomalous behavior in the IoV network while preserving the privacy of the data. The main contributions of this paper are:

- We develop and implement a SL-based IDS using different deep neural networks to address concerns around traditional IDS approaches' scalability and efficiency to detect anomalies.
- We use the proposed approach to protect IoV privacy requirements feasibly and its influencing characteristics, including resource utilization and accuracy, to detect various categorical attacks.
- We optimize the SL-based training process by leveraging a regret minimization technique to reduce the overall latency of the training process.
- We prototype and evaluate the performance using open-source datasets and posit that the proposed approach can improve detection experience over time while adapting to the rapidly changing IoV environments.

2 BACKGROUND

In this section, we provide some background about IoV and the SL approach used for detecting anomalies in IoV.

2.1 Internet of vehicles (IoV)

The IoV is an extension of the Internet of Things (IoT) that enables connected vehicles, road infrastructure, and pedestrians to exchange information to create an intelligent transportation system. An IoV network consists of two sub-networks—an intra-vehicular network and an inter-vehicular network.

2.1.1 Intra-vehicular Network: The intra-vehicle network of modern vehicles consists of a controller area network (CAN) bus and several electronic control units (ECUs). The CAN bus is a serial broadcast communication protocol that provides a medium for the ECUs to exchange messages. Each ECU is linked to a specific in-vehicle component, such as the braking system, engine control module, and body controller. The ECU shares the state of the component it is associated with by transmitting signals related to its functions via CAN frames. Each frame has a unique identifier (ID) and contains a data payload of up to 8 bytes where the signals being transmitted are encoded. The actual size of the payload is indicated in the data length code field. CAN messages with lower IDs are considered to have higher priority and are transmitted first on the bus to ensure timely delivery of the highest priority frames.

2.1.2 Inter-vehicular network: IoV inter-vehicular communication (IVC) involves connecting vehicles to other IoV entities, including other vehicles, RSUs, intelligent devices, and the cloud, to enable more efficient and safer transportation. IVC allows vehicles to exchange information in real-time using several communication protocols such as 4G/LTE, 5G/6G, WiFi, Bluetooth, worldwide interoperability for microwave access (WiMAX), and WAVE. One of the most common protocols is Dedicated Short-Range Communication (DSRC), which is ideal for high-speed communication, making it suitable for emergency situations and collision avoidance. These protocols enable IVC with their unique strengths that allow data transmission between different IoV entities.

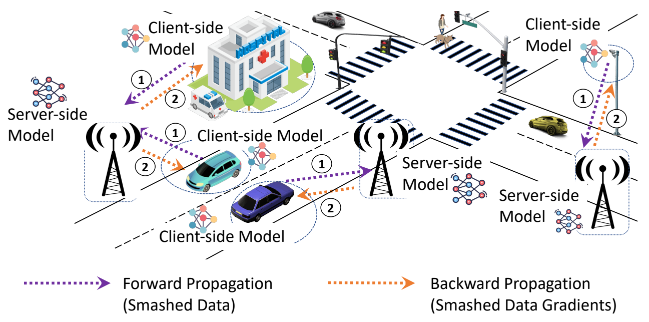
The rapid growth of interconnected smart entities in the IoV ecosystem has significantly increased security risks. The communication network security vulnerabilities, coupled with the lack of security measures in the CAN bus that protects exchanged messages, make it an attractive target for cyber attacks [19]. These attacks can take various forms, such as eavesdropping, injection, denial-of-service (DoS), distributed DoS (DDoS), and spoofing attacks, leading to disruption of the regular operations of safety-critical vehicular components. These security risks pose significant threats to the safety and security of the IoV ecosystem.

2.2 SL Approach

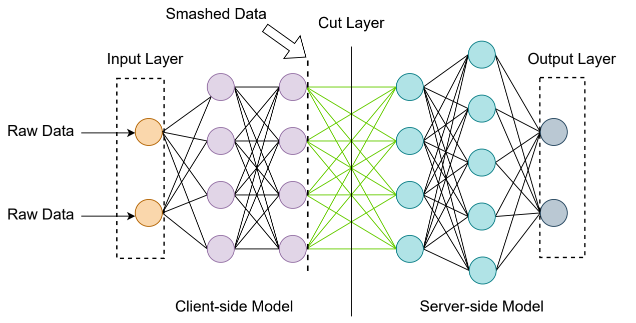
Split learning is a privacy-preserving method for training ML models on distributed data sources. In this approach, a client retains its local dataset and shares only a portion with an edge server, reducing the amount of data transferred and protecting sensitive information. During training, the model goes through forward propagation (FP) and backward propagation (BP) phases, with the client-side model trained locally until a designated cut layer, l_c, is reached. At this point, the client sends the output of the cut layer, known as the *smashed data*, to the server to complete the FP phase, while the server starts the BP phase until the cut layer and sends the server-side *smashed gradients* to the client to complete the BP phase. This split approach reduces the training latency and computational burden while ensuring data privacy and security, making it an attractive solution for IoT and IoV systems.

2.3 Regret Minimization

Regret minimization is a strategic approach aimed at making decisions that minimize the potential for future regrets. *Regrets* refer to the additional costs incurred by a network when processing the data. The core principle behind regret minimization involves carefully considering all potential outcomes and the various decisions that precede any action. By considering these possible outcomes, individuals or systems can make more informed choices that maximize benefits while minimizing negative consequences. Regret minimization takes into consideration the delays that might occur when transferring data across different points within the network



((a)) Resource-constrained devices in IoV begin FP and offload the rest of the FP process to an edge server with more computational resources. The edge server completes the FP and begins the BP. The rest of the BP is completed by the resource-constrained device.



((b)) The client-device starts FP until the cut layer and sends smashed data to the server to complete the FP. The server begins BP until the cut layer, and the client completes the rest of BP with its client-side model.

Figure 1: Overview of split learning in IoV

architecture. Regret minimization, having an inherent focus on optimizing choices, offers flexibility that can be particularly beneficial in dynamic contexts. This flexibility is evident when considering data offloading strategies, where the regret minimization approach can dynamically select between processing the data locally or offloading it on the server based on real-time considerations. The primary factor in this determination is the extent of the delay, as it serves as a cornerstone for making well-informed decisions that align with network efficiency and overall performance.

3 THREAT MODEL

We examine the following threat vectors to better understand the landscape of vulnerabilities in IoV.

3.1 Intra-Vehicular Communication

We assume an adversary can access the CAN bus through an unpatched vulnerability that allows the adversary to compromise the ECU modules in the autonomous vehicle. Also, this vulnerability enables the attacker to eavesdrop, analyze the CAN messages, and inject malicious messages to cause malfunction or damage to the vehicular system. To evaluate the effectiveness of the proposed IDS for intra-vehicular communication, we consider the following attack types:

3.1.1 DoS attack: An attacker floods the CAN bus with a high volume of high-priority messages, causing congestion that disrupts the regular communication of legitimate ECUs.

3.1.2 Fuzzy attack: This sophisticated injection attack involves subtle changes to legitimate message content to exploit a network vulnerability. An attacker may use this technique to alter the behavior of the vehicle system to cause malfunction.

3.1.3 Spoofing attack: An attacker sends fabricated messages to the CAN bus, pretending to be a legitimate ECU to inject malicious messages or to manipulate the vehicle system. We investigate two types of spoofing attacks: gear and RPM spoofing. These attacks involve an attacker sending falsified messages to the CAN bus that inaccurately reports the vehicle's current gear or RPM of the vehicle's engine, respectively.

3.2 Inter-Vehicular Communication

We assume an adversary can access the wireless communication channel used by the vehicles to exchange messages, such as DSRC or Vehicle-to-Everything (V2X) protocols. The adversary could use various methods to compromise the communication channel, including jamming, spoofing, and interception. To evaluate the effectiveness of the proposed IDS for inter-vehicular communication, we consider the following attack types:

3.2.1 Portscan attack: A port scanning attack involves an adversary scanning the communication network for open ports or services to identify potential vulnerabilities that could be exploited to gain access to the network.

3.2.2 DoS attack: Similar to CAN DoS, in inter-vehicular communication, a DoS attack can be launched to flood the IoV network with messages.

3.2.3 DDoS attack: The goal of DDoS is the same as DoS, which is to overload the target's resources with traffic and disrupt its availability. However, the difference between DoS and DDoS attacks is the number of sources used to launch the attack. In DoS, a single or a few malicious sources are used to launch the attack, whereas, in DDoS, an attacker uses multiple sources, typically from compromised devices, to launch a coordinated attack.

4 PRIVACY-PRESERVING IDS FOR IOV FRAMEWORK

By providing the ability for nodes to retain their data, SL offers an opportunity to build an IDS for IoV that is both effective and privacy-preserving. We illustrate the SL approach adapted to the IoV environment using the vehicular network shown in Fig. 1. The network consists of vehicles as client devices and RSUs as edge servers with significant computing resources. In contrast to the RSUs, we assume the clients have limited computation capabilities but can access the edge servers via wireless access technologies.

The proposed IDS includes a data preprocessing stage that transforms raw IoV data obtained from realistic network scenarios into usable formats. Following the preprocessing stage, the IDS utilizes a portion of the dataset for training neural network models (Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) Network). Next, the performance of the trained models is evaluated using a separate testing set. In this section, we further explain the details of the proposed IDS.

4.1 Dataset

We utilized two open-source datasets representing intra- and inter-vehicular networks to train and test our model. For intra-vehicular networks, we used a CAN dataset provided by the Hacking and Countermeasure Research Lab [10]. The dataset contains messages logged from the OBD-II port of a vehicle, containing both benign and malicious payloads that compromise the operations of autonomous vehicles via the CAN bus. The attacks include DoS, fuzzy, and spoofing attacks. In addition, the dataset provides information about the timestamp, CAN identifier, data length code, a maximum of 8 bytes of data, and a flag indicating whether the message is benign or malicious. For inter-vehicular networks, we utilized the CICIDS2017 dataset, which is a real-time network dataset for intrusion detection [14]. The CICIDS2017 dataset consists of over 80 network traffic features and provides data samples containing benign network traffic and common network attacks, such as DoS, DDos, and Port scan attacks.

4.2 Data preprocessing

In the preprocessing stage, the CAN and CICIDS2017 datasets undergo a comprehensive filtering process to eliminate any rows containing missing values. In addition, during this step, the datasets are examined for infinite numbers, and any positive infinite values are substituted with a large positive value. Similarly, infinite negative values are replaced with a small negative value. After filtering, a data converter step is applied, which involves normalizing the feature values using a min-max scaler, as the feature values are typically dispersed over a wide range. Furthermore, to reduce the dependence of the model on the order of the data samples and prevent overfitting, we randomly shuffle the training data.

4.3 Neural Network Models

4.3.1 CNN. A CNN [12] is a class of deep learning models designed specifically for processing inputs with a grid-like structure, such as images. CNN comprises three layers: convolutional, pooling, and fully connected. The first two layers are used for feature extraction, whereas the last layer maps the features to the final output. The convolutional layer identifies patterns and features in the input data by performing a mathematical operation called convolution. Next, a rectified linear unit (ReLU) activation function is applied to the output of the convolutional layer to introduce nonlinearity into the model since many real-world problems cannot be accurately modeled solely using linear functions. The pooling layer performs downsampling of the output received from the convolutional layer, which reduces the overfitting of the CNN model. Finally, the fully connected layer transforms the output of the convolutional and pooling layers into a format that can be used for classification. Our proposed architecture consists of two convolutional layers, each followed by a pooling and two fully connected layers.

4.3.2 LSTM. Gradient-based neural network architectures are susceptible to issues such as exploding or vanishing gradients, which can hinder the learning process [8]. To mitigate these problems, LSTM networks incorporate several techniques. Firstly, to address the exploding gradient problem, LSTM uses gradient clipping, which involves constraining the range of gradients during training. In addition, LSTM addresses the vanishing gradient problem by utilizing a memory cell that retains information over multiple time steps and selectively updates or forgets the information as needed. This memory cell is controlled by three gate units—the input gate, i, the forget gate, f, and the output gate, o—which regulate the flow of information into and out of the cell. These techniques make LSTM a powerful tool for processing sequential data. These gates at each time step t, are given by:

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{1}$$

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2}$$

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{3}$$

where W_i, W_f, and W_o are the input, forget, and output gates weights, respectively. b_i, b_f, and b_o are the biases of their respective gates, x_t is the input data, h_{t-1} is the hidden state from a previous time step, and σ is the sigmoid activation. The new hidden state h and cell state C are calculated using:

$$\tilde{C_t} = \tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{4}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C_t} \tag{5}$$

$$h_t = o_t \odot \tanh(C_t) \tag{6}$$

W_c and b_c represent the weight and bias used in the cell state, tanh is the hyperbolic tangent activation function, C_{t-1} is the cell state at the previous time step, and $\odot$ represents the Hadamard product. Our proposed architecture consists of three recurrent LSTM layers with 64 units, a dense layer with 10 units, and a softmax activation function.

4.3.3 GRU. GRU networks reduce the complexity of LSTM using two gates: the update z_t and reset gate r_t, expressed as:

$$z_t = \sigma(W_z.[h_{t-1}, x_t] + b_z) \tag{7}$$

$$r_t = \sigma(W_r.[h_{t-1}, x_t] + b_r) \tag{8}$$

where W_z and W_r are the update and reset gates weights, respectively. b_z and b_r are the biases of their respective gates. The hidden state, h, of the GRU model is given as:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t} \tag{9}$$

$$\tilde{h_t} = \tanh(W_h[r_t \odot h_{t-1}, x_t] + b_h) \tag{10}$$

where W_h and b_h are the weights and biases of the hidden state, respectively. Our proposed architecture consists of three GRU layers with 64 units, followed by a dense layer with 10 units and a softmax activation function.

4.4 Training and Testing

In this stage, the IDS model is trained using the processed dataset, divided into training and testing sets with a ratio of 80:20. The preprocessed training set is used by each vehicle to start the FP phase until the smashed data is obtained and transmitted to the RSU via DSRC or V2X. In return, the RSU completes the FP, starts the BP, and returns the smashed gradients back to the vehicle.

After training, we evaluate the model performance using standard metrics such as recall, precision, F1 score, and accuracy. Recall R, represents the proportion of accurate attack predictions in the attack class and is given as $R = \frac{TP}{TP+FN}$, where TP and FP are the true and false positive values, respectively. The precision, P, is the fraction of correctly identified anomalies, computed as $P = \frac{TP}{TP+FP}$, where FP represents the false positive value. The F1 score is the weighted average of recall and precision, expressed as $F1 = \frac{2\times(Recall\times Precision)}{Recall+Precision}$. Finally, accuracy, A, is the fraction of the correct predictions in the total observations, given as $A = \frac{TP+TN}{TP+FP+TN+FN}$, where TN is the true negative.

5 ADAPTIVE OFFLOADING USING REGRET MINIMIZATION

Choosing the SL approach for training is not always optimal, as the delay incurred due to the data transfer to the server could outweigh the advantages of local processing. Instead of statically defining a training strategy, we can dynamically assess when to utilize the SL approach at specific intervals, considering the observed training time. This dynamic offloading technique effectively minimizes the time it takes to train the neural models. To achieve this, we adapt the PROD regret minimization algorithm [6] for training our IDS algorithms, which aids in selecting the optimal training strategy. The regret calculation is based on the expected delay and the delay due to the observed training time. Mathematically, the regret can be expressed as:

$$R_i = \hat{d} - d_i \tag{11}$$

Here, $\hat{d}$ is the expected delay and d_i is the observed delay due to the i-th strategy of the training device (i.e., train locally or utilize SL). By assigning a certain weight to each strategy, the expected delay can be calculated as:

$$\hat{d} = \sum_{i=1}^{|\mathcal{W}|} (p_i * d_i) \tag{12}$$

Here, p_i is the probability of choosing a strategy based on its current weight w_i and is calculated as:

$$p_i = \frac{\alpha * w_i}{\sum_{i=1}^{|W|} (\alpha * w_i)} \tag{13}$$

Based on the regret calculation, the weights of the strategies are updated with a learning parameter α, where $\alpha > 0$.

$$w_i = w_i(1 + \alpha * R_i) \tag{14}$$

As the delay due to a strategy increases, the weights get readjusted so that the strategy with the lower delay gets preferred, cumulatively reducing the overall training time.

As summarized in Algorithm 1, the adaptive offloading technique takes into account various input parameters, such as a weight vector ($\mathcal{W}$) containing initial weights for each training strategy, a learning parameter (α), and the delay (d) associated with each strategy. The weight vector W comprises non-negative values that sum up to 1, i.e., $\sum_{i=1}^{|\mathcal{W}|} w_i = 1$. The algorithm initializes a random vector that stores the values of the expected delay in line 3. Following that, the input delay vector is normalized, as outlined in lines 4-6. Lines 8-11 show the calculation of a probabilistic distribution for selecting each strategy and the associated expected delay based on the current strategy as provided in equations 12 and 13. Subsequently, the expected delay is utilized to calculate regrets for each strategy based on equation 11. Line 14 adjusts the weights as depicted in equation 14 to minimize the overall regrets. Finally, line 17 returns the optimal weight for each task.

Algorithm 1 Adaptive Offloading with Regret Minimization

```
1:  **Input:** weight vector $\mathcal{W} = \{w_1, w_2\}$, learning parameter $\alpha$, delay $d = \{d_1, d_2\}$
2:  **Output:** updated weight vector $\mathcal{W}' = \{w'_1, w'_2\}$
3:  Initialize expected delay $\hat{d}$
4:  for $i \in [1, |\mathcal{W}|]$ do
5:      $d_i = \frac{d_i}{\sum_{i=1}^{|\mathcal{W}|}(d_i)}$
6:  end for
7:  for $t \in [0, T]$ do
8:      for $i \in [1, |\mathcal{W}|]$ do
9:          $p_i = \frac{\alpha * w_i}{\sum_{i=1}^{|\mathcal{W}|}(\alpha * w_i)}$
10:         $\hat{d} = \sum_{i=1}^{|\mathcal{W}|}(p_i * d_i)$
11:     end for
12:     for $i \in [1, |\mathcal{W}|]$ do
13:         $\mathcal{R}_i = \hat{d} - d_i$
14:         Update $w'_i = w'_i(1 + \alpha * \mathcal{R}_i)$
15:     end for
16: end for
17: Return $\mathcal{W}'$
```

6 PERFORMANCE EVALUATION

6.1 Evaluation Settings

We performed the experiments using Pytorch 2.0.0, a Python framework for deep learning. We implemented the split learning architecture with CNN, LSTM, and GRU models and compared our approach to the performance of these models when they are trained fully on the client side. In our experiment, we used the Adam optimizer and set the learning rate to 1e-3. We utilized a batch size of 128 and used the cross-entropy loss function to measure errors in prediction. The client-side models were trained on an Intel(R) Core(TM) i9-10900 CPU @ 2.80GHz with 32.0 GB RAM. The server-side models were trained on an NVIDIA Quadro P400 GPU.

6.2 Evaluation Results

6.2.1 Comparison with baselines: To evaluate the effectiveness of our IDS against attacks in intra- and inter-vehicular networks, we performed experiments using the CAN and CICIDS2017 datasets and set l_c to 2 for all the models.

Table 1 presents the results of our inter-vehicular attack detection. The table indicates that without split learning, the GRU model achieves an accuracy and F1 score of 0.95, while with split learning, the model's performance improves to an accuracy and F1 score of 0.96. The LSTM model also shows similar improvement, increasing accuracy and F1 score from 0.96 to 0.98 when using split learning.

In addition, our result indicates that the CNN model outperforms GRU and LSTM models, achieving an F1 score and accuracy of 0.99, both with and without split learning.

Table 2 shows the result for the intra-vehicular intrusion detection. From the result, the LSTM model achieves 0.98 for both accuracy and F1 score with and without split learning. CNN improves the detection performance with an F1 score and accuracy of 0.99 with and without split learning. GRU performed best compared with CNN and LSTM, achieving an F1 score and accuracy of 1. With split learning, this score remained 1, showing again that using split learning does not degrade the performance of the IDS. Our results highlight the efficiency of split learning in training IDS models for IoV while preserving data privacy during training and maintaining high model performance during inference.

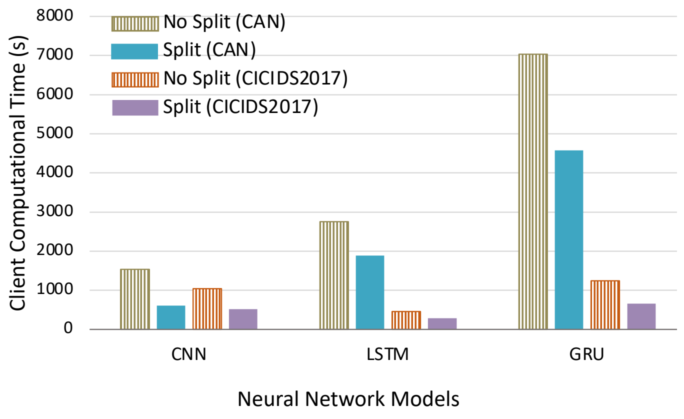


Figure 2: Computation time with and without split learning

6.2.2 Does our approach save computation time on the client?: We assessed the effectiveness of our IDS in reducing client training time by comparing the training time with and without split learning. For each scenario, we measured the duration of both FP and BP on the client device. Our results, shown in Fig. 2, demonstrate that split learning reduces computation time compared to training the CNN model entirely on the client device. We observed similar results with LSTM and GRU, where split learning reduced computation time on the client device. Our approach enables clients to considerably decrease computation time, allowing them to concentrate on other tasks and enhance the quality of service in the IoV.

6.2.3 Impact of client-side layer complexity: To further illustrate the impact of our IDS on training time, we varied the complexity of the client-side model by using different values of l_c. We conducted this experiment using the LSTM model with the CAN dataset. Fig. 3 depicts that the split learning approach saves the most computation time for the client when $l_c = 1$. It is worth mentioning that we obtained similar results with CNN and GRU networks. Moreover, as l_c increases, the client's computation time also increases. Our findings suggest that reducing the client's computation time necessitates selecting an appropriate value for l_c that will shift more computation to the powerful device and enhance the overall training performance.

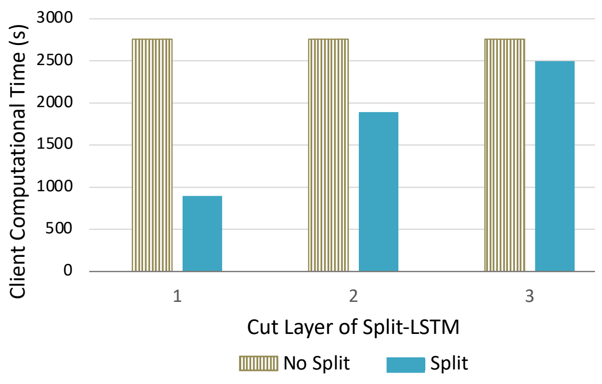


Figure 3: Impact of layer complexity on the client device

6.2.4 Does our approach preserve privacy?: To show that our approach does not transmit raw data between the client and the server, we use color maps to show that the data transmitted to the server differs from the client's input. For this experiment, we used the CICIDS2017 dataset. Fig. 4(a) and Fig. 4(b) show the first epoch's inputs to the client and server-side CNN model, respectively. After the client receives the raw input in Fig. 4(a), the input passes through the convolution, activation, and pooling layers. This process changes the raw data to the smashed data shown in Fig. 4(b) before sending it to the server. Our result shows that training efficiency and data privacy can be preserved by devices taking part in collaborative training in IoV. Additions of perturbations to the client-side model output before being sent to the server can further improve privacy.

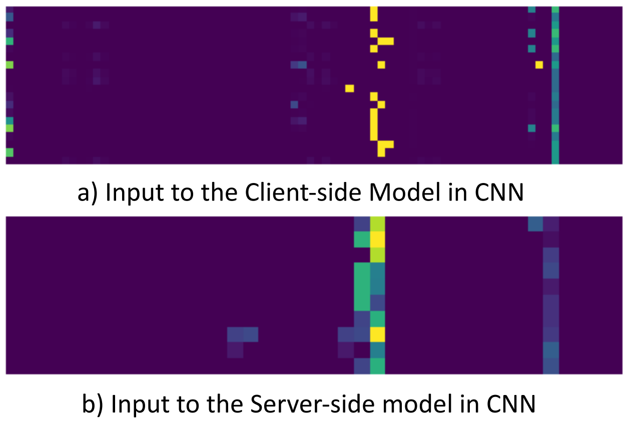


Figure 4: Color map showing client-side and server-side inputs using split-CNN and data from CICIDS2017

6.2.5 Impact of adaptive offloading on training latency: We evaluate the impact of the adaptive offloading strategy on the overall training time of the SL-based training procedure. As illustrated in Figure 5, the client's computation time reduces with split learning compared to the scenario where the client conducts training exclusively on a resource-limited device. Nonetheless, offloading during each epoch might not be optimal for the fluctuating network

Table 1: Summary of split learning-based IDS Evaluation on CICIDS2017 dataset

Attack Type	CNN								LSTM								GRU							
	No Split				Split				No Split				Split				No Split				Split			
	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
No Attack	0.99	0.96	0.99	0.98	0.99	0.98	0.99	0.99	0.99	0.96	0.99	0.97	0.99	0.98	0.99	0.98	0.99	0.94	0.99	0.97	0.99	0.95	0.99	0.97
DDoS	1	0.99	1	1	1	1	1	1	0.99	0.98	0.99	0.98	0.98	0.99	0.98	0.98	0.86	0.99	0.86	0.92	0.84	0.99	0.84	0.91
PortScan	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.92	0.99	0.92	0.95	0.98	0.99	0.98	0.98	0.85	0.98	0.85	0.91	0.93	0.99	0.93	0.96
DoS	0.99	1	0.99	0.99	0.99	1	0.99	1	0.89	0.97	0.89	0.93	0.94	0.97	0.94	0.96	0.94	0.96	0.94	0.95	0.94	0.96	0.94	0.95
Weighted Avg		0.99	0.99	0.99		0.99	0.99	0.99		0.97	0.96	0.96		0.98	0.98	0.98		0.96	0.95	0.95		0.96	0.96	0.96
Network Acc	0.99				0.99				0.96				0.98				0.95				0.96			

Table 2: Summary of split learning-based IDS Evaluation on CAN dataset

Attack Type	CNN								LSTM								GRU							
	No Split				Split				No Split				Split				No Split				Split			
	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
No Attack	0.98	0.96	0.98	0.97	0.98	0.96	0.98	0.97	1	0.99	1	1	1	0.99	1	1	1	1	1	1	1	1	1	1
DoS	1	0.99	1	0.99	1	0.99	1	0.99	0.94	0.99	0.94	0.97	0.94	1	0.94	0.97	0.99	1	1	1	1	1	1	1
Fuzzy	0.99	1	0.99	0.99	0.98	1	0.98	0.99	0.81	1	0.81	0.89	0.59	1	0.59	0.74	0.96	0.99	0.96	0.97	0.98	0.99	0.98	0.99
Gear	1	1	1	1	1	1	1	1	0.79	1	0.79	0.88	0.97	1	0.97	0.98	1	1	1	1	1	1	1	1
RPM	1	1	1	1	1	1	1	1	1	0.83	1	0.91	1	0.83	1	0.91	1	1	1	1	1	1	1	1
Weighted Avg		0.99	0.99	0.99		0.99	0.99	0.99		0.99	0.98	0.98		0.99	0.98	0.98		1	1	1		1	1	1
Network Acc	0.99				0.99				0.98				0.98				1				1			

conditions prevalent in the IoV, which can degrade over time. Our result shows that the regret minimization-based adaptive offloading technique can significantly reduce the client's computation time compared to only SL, reducing the overall latency by 23.81%, as opposed to the 14.79% decrease achieved by the SL-based method alone.

6.2.6 Impact of adaptive offloading on training strategy: To illustrate the impact of the adaptive offloading strategy on the decision made by the device to offload or train locally, we perform the training of the CNN model for IoV IDS over 20 epochs. Fig. 6 shows the variation of the offloading probabilities at each epoch for the adaptive offloading strategy compared to a static approach of local computation or offloading to an edge server. As our result shows, the adaptive strategy changes its decision depending on the observed network condition to minimize the overall latency of the training process.

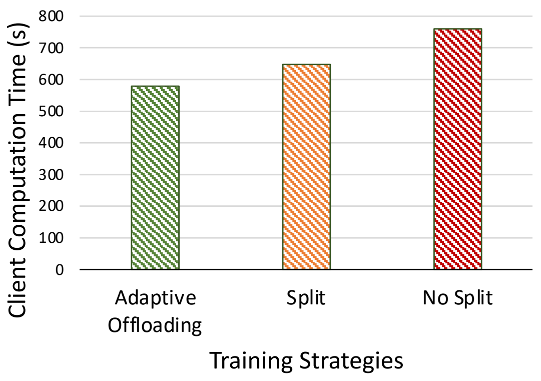


Figure 5: Impact of adaptive offloading strategy on the overall training time

7 RELATED WORK

Several works have been proposed for anomaly detection in IoV. Some approaches have used ML techniques for intrusion detection in intra-vehicular networks, such as CAN [2, 4], while others have focused on detecting anomalies in inter-vehicular networks [9, 18].

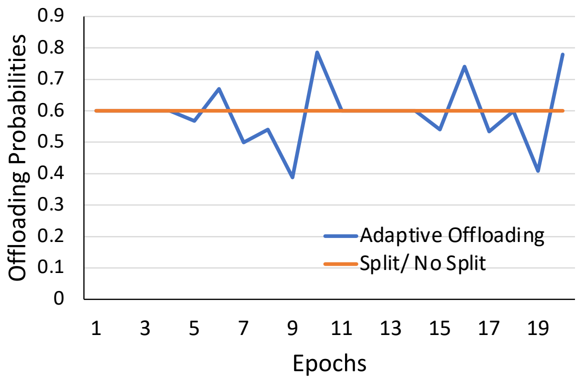


Figure 6: Impact of the adaptive strategy on offloading decision

Alkhatib et al. [2] proposed a deep learning-based multi-agent IDS for CAN anomaly detection, using an attention-based self-learning technique that relies on asynchronous signals and CAN IDs to reduce the need for long-term monitoring. Anjum et al. [4] proposed using an extreme gradient boosting machine (XGBoost) to detect anomalies in CAN data and employed a data filter module to extract useful features for the XGBoost algorithm. Although these methods enhance anomaly detection in the in-vehicle network, they do not address inter-vehicular network communication issues.

Karthiga et al. [9] employed known IDS (KIDS) and unknown IDS (UIDS) modules to detect anomalies in inter-vehicular networks. The KIDS used an adaptive neuro-fuzzy inference system to identify known attacks in vehicular ad-hoc networks, while the UIDS utilized a modified LeeNet network, a deep-learning architecture with few internal layers, to detect unknown attacks. Almutlaq et al. [3] proposed a two-stage attack detection approach for the IoV network. In the first stage, they classified traffic as either an attack or a normal message, followed by the identification of the attack type using rule extraction methods in the second stage. Yang et al. [18] proposed a multi-tiered IDS architecture for intra- and inter-vehicular networks using an ensemble of machine learning models, including decision trees, random forests, extra trees, and XGBoost. They also utilized cluster labeling k-means to detect zero-day attacks. However, central training of IDSs creates data privacy issues, despite its consideration of wider IoV networks. Hbaieb et al. [7] studied a privacy-preserving IDS based on federated learning in the IoV, including trust metrics for secure communication. However, their approach assumed that local devices can handle the entire training process, which may be challenging for resource-limited IoV devices. Our method differs by enabling these devices to offload part of their training to an edge server with greater computing power while preserving data privacy and improving training efficiency.

8 CONCLUSION

In this paper, we presented a privacy-preserving split learning-based IDS to address the security concerns of traditional IDSs in the IoV by avoiding the need to share raw data. Our proposed method enables resource-constrained IoV devices to offload a significant part of their training to an edge server with more computational resources without sharing sensitive data. In addition, our approach further optimizes the training process by using an adaptive offloading technique that minimizes overall latency. We evaluated the effectiveness of the proposed model using open-source real IoV datasets, and the results show that the approach maintains data privacy and detects anomalies with an average accuracy of 99%. Also, our proposed adaptive offloading approach reduces the training time by up to 23.81%. In future work, we plan to fine-tune the model by predicting the most efficient cut layer that optimizes time and energy, and consider adding perturbations to the client-side model output to improve privacy.

REFERENCES

[1] Paul Agbaje, Afia Anjum, Arkajyoti Mitra, Emmanuel Oseghale, Gedare Bloom, and Habeeb Olufowobi. 2022. Survey of Interoperability Challenges in the Internet of Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 23, 12 (2022), 22838–22861. https://doi.org/10.1109/TITS.2022.3194413

[2] Natasha Alkhatib, Lina Achaji, Maria Mushtaq, Hadi Ghauch, and Jean-Luc Danger. [n. d.]. WIP: AMICA: Attention-based Multi-Identifier model for asynchronous intrusion detection on Controller Area networks. ([n. d.]).

[3] Samah Almutlaq, Abdelouahid Derhab, Mohammad Mehedi Hassan, and Kuljeet Kaur. 2022. Two-stage intrusion detection system in intelligent transportation systems using rule extraction methods from deep neural networks. *IEEE Transactions on Intelligent Transportation Systems* (2022).

[4] Afia Anjum, Paul Agbaje, Sena Hounsinou, and Habeeb Olufowobi. 2022. In-Vehicle Network Anomaly Detection Using Extreme Gradient Boosting Machine. In *2022 11th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 1–6.

[5] Chien-Ming Chen, Bin Xiang, Yining Liu, and King-Hang Wang. 2019. A secure authentication protocol for internet of vehicles. *Ieee Access* 7 (2019).

[6] Pierre Gaillard, Gilles Stoltz, and Tim Van Erven. 2014. A second-order bound with excess losses. In *Conference on Learning Theory*. PMLR, 176–196.

[7] Amal Hbaieb, Samiha Ayed, and Lamia Chaari. 2022. Federated learning based IDS approach for the IoV. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*. 1–6.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[9] B Karthiga, Danalakshmi Durairaj, Nishad Nawaz, Thiruppathy Kesavan Venkatasamy, Gopi Ramasamy, A Hariharasudan, et al. 2022. Intelligent Intrusion Detection System for VANET Using Machine Learning and Deep Learning Approaches. *Wireless Communications and Mobile Computing* (2022).

[10] H. Lee, S. H. Jeong, and H. K. Kim. 2017. OTIDS: A Novel Intrusion Detection System for In-vehicle Network by Using Remote Frame. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, Vol. 00. 57–5709. https://doi.org/10.1109/PST.2017.00017

[11] Muhammad Baqer Mollah, Jun Zhao, Dusit Niyato, Yong Liang Guan, Chau Yuen, Sumei Sun, Kwok-Yan Lam, and Leong Hai Koh. 2020. Blockchain for the internet of vehicles towards intelligent transportation systems: A survey. *IEEE Internet of Things Journal* 8, 6 (2020), 4157–4185.

[12] Keiron O'Shea and Ryan Nash. 2015. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* (2015).

[13] Manjari Singh Rathore, M Poongodi, Praneet Saurabh, Umesh Kumar Lilhore, Sami Bourouis, Wajdi Alhakami, Jude Osamor, and Mounir Hamdi. 2022. A novel trust-based security and privacy model for internet of vehicles using encryption and steganography. *Computers and Electrical Engineering* 102 (2022).

[14] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* 1 (2018), 108–116.

[15] Nishant Sharma, Naveen Chauhan, and Narottam Chand. 2018. Security challenges in Internet of Vehicles (IoV) environment. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*. https://doi.org/10.1109/ICSCCC.2018.8703272

[16] Surbhi Sharma and Baijnath Kaushik. 2019. A survey on internet of vehicles: Applications, security issues & solutions. *Vehicular Communications* (2019).

[17] Li Yang, Abdallah Moubayed, Ismail Hamieh, and Abdallah Shami. 2019. Tree-based intelligent intrusion detection system in internet of vehicles. In *2019 IEEE global communications conference (GLOBECOM)*. IEEE, 1–6.

[18] Li Yang, Abdallah Moubayed, and Abdallah Shami. 2021. MTH-IDS: A multitiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet of Things Journal* 9, 1 (2021), 616–632.

[19] Clinton Young, Joseph Zambreno, Habeeb Olufowobi, and Gedare Bloom. 2019. Survey of automotive controller area network intrusion detection systems. *IEEE Design & Test* 36, 6 (2019), 48–55.