SVMDformer: A Semi-supervised Vehicular Misbehavior Detection Framework based on Transformer in IoV

Zhikang Liu *[1], Hongyun Xu [1], Yong Kuang [1], Feng Li [1]
[1] School of Computer Science & Engineering, South China University Of Technology, Guangzhou, China
202121045561@mail.scut.edu.cn, hongyun@scut.edu.cn, {csdeaisry, 202221044781}@mail.scut.edu.cn

***Abstract*—Vehicle-to-everything(V2X) based technologies have become an important research topic to solve the problem of road congestion and traffic accidents. However, the rapid growth of connected vehicles enables massive messages to flow around in the Internet of Vehicles (IoV), some of which are misbehaving from insider attackers and thus threaten the security of IoV. Recently, some works have tried solving the misbehavior detection problem, but there are problems such as incomplete misbehavior types, inefficient multi-model detection, and insufficient accuracy. In this paper, we propose a Semi-supervised Vehicular Misbehavior Detection framework based on Transformer (SVMDformer), which transforms vehicular message sequences into misbehavior scores and identifies misbehavior by setting a threshold. The SVMDformer is trained in the cloud server and deployed in the edge server to detect misbehavior by incoming vehicular messages in IoV. Based on an open-source dataset, we demonstrate the model's performance in different environments, analyze its sensitivity, and compare it with baselines. The proposed SVMDformer is more versatile, accurate, and comprehensive than prior works. It is a single model that can detect 19 misbehaviors for vehicular message sequences. It achieves an AUC of 99.87%, better accuracy of 99.66%, better precision of 99.68%, better F1-score of 99.66%, and a false detection rate of 0.34% which is over three times lower than the state-of-the-art (SOTA) work.**

***Index Terms*—IoV, Vehicular Misbehavior Detection, Semi-supervised Learning, Transformer**

I. Introduction

According to a recent report [1], the global demand for vehicles has been increasing, and so does traffic congestion and accidents. Internet-of-Vehicles (IoV) is an ideal but not-yet-grounded vehicular network (e.g. Fig.1), which combines with Internet-of-Things (IoT), Vehicle-to-Everything (V2X) [2], and mobile-edge-computing (MEC) [3] technologies. IoV provides vehicle convenience and safety and has become a key part of future cooperative intelligent transportation systems (C-ITS) [4]. In IoV, on-board units (OBUs) are installed in each vehicle to communicate with other infrastructures and be a part of IoV. The connected vehicles exchange basic safety messages (BSMs) that contain position, speed, acceleration, and heading information.

However, V2X systems bring convenience as well as security concerns, including attacks and misbehaviors from outsiders and insiders. Attacks from the outside can be identified by cryptography (e.g., public key infrastructure (PKI) [5]), while insider attackers have legitimate keys to communicate with other infrastructures, but they send misbehaving BSMs to interfere with road safety and traffic efficiency. For example, misbehaving insiders send constant position messages or create fake traffic congestion by forging new IDs (Sybil) so that users believe there is congestion ahead, leading to a waste of road resources. Sending such incorrect messages in IoV is called misbehavior, which is generally classified as malfunction and attack, and detecting them is called misbehavior detection (MD) [6].

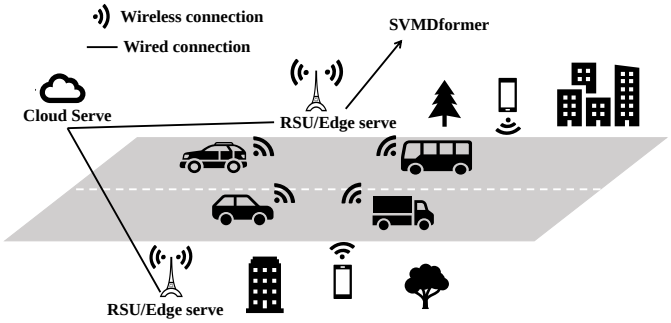


Fig. 1: IoV Model.

Many works [7] have tried machine-learning-based and data-centric approaches to address MD. However, all have problems, such as incomplete misbehavior types, inefficient multi-model detection, and insufficient accuracy. Traditional machine learning (ML) is incapable of handling the massive data scenarios of V2X. However, the development of deep learning provides new ways for MD. Semi-supervised learning is proved to be a feasible solution in the field of Anomaly Detection (AD) [8], which assumes unlabeled data as normal data. It can effectively detect anomalies without much prior knowledge of them by learning a large amount of unlabeled data and a small amount of labeled data. From a Benchmark storage [9], most semi-supervised methods can outperform the best unsupervised method with merely 1% labeled anomalies. So we use semi-supervised learning to train our model, the difference is that our model learns a few anomalous data on top of the unsupervised learning which only learns normal data. Transformer [10], an outstanding figure in deep learning, has achieved state-of-the-art (SOTA) in various natural language processing tasks in [11] and has impressive results in computer

vision, audio, and video community. Data-centric MD is the detection of vehicular messages sent by time, so Transformer is the basic framework in our work. Due to the limited computing and storage capacity of vehicle OBUs and the latency problem of running detection tasks on cloud servers, we use edge computing as the solution for our IoV model (Fig.1).

In this paper, we propose a Semi-supervised Vehicular Misbehavior Detection framework based on Transformer (SVMDformer), which transforms the message sequences into misbehavior scores and identifies misbehavior by a threshold. The main contributions of this paper are highlighted as follows.

- We design a new framework, SVMDformer, based on semi-supervised learning and Transformer to detect misbehavior in IoV and achieve state-of-the-art performance on an open-sourced dataset.
- Due to the flexibility of data preprocessing, our SVMDformer can detect any vehicular sequences or subsequences longer than 10.
- Based on semi-supervised learning, our SVMDformer can detect 19 types of misbehavior even outside the dataset and target training for hard-to-detect misbehavior types.
- In our experiments, we demonstrate our SVMDformer with five key evaluation metrics, evaluate its performance under three supervision methods and single misbehavior and visualize the Scores, analyze the sensitivity of the EventualStop training volume, compare six baseline NN models and prior works, and operate ablation experiments.

The rest of this paper is organized as follows. Section II introduces the related works of misbehavior datasets and detection methods in vehicular networks. Section III introduces the preliminary knowledge of Mutil-head self-attention. In Section IV, we describe the whole SVMDformer framework. Moreover, we evaluate and discuss the performance of our SVMDformer under different environments in Section V. In Section VI, we comprehensively analyze the model and compare it to baseline models and benchmarks. Finally, we summarize our work and suggest improvement points in Section VII.

II. Realated Work

Intrusion-detection-system (IDS) in vehicular networks is a widely studied area. IDS protects vehicle privacy and data security through three main ways: Identification for drivers, Message integrity checking for outsiders, and Message correctness checking for insiders. While studies on Misbehavior-detection-system (MDS) focus more on checking message correctness, i.e., whether insiders send correct BSMs, and is a more data-driven IDS. In the V2X system, misbehavior is generally divided into malfunctions caused by OBUs, sensor and wireless network, and malicious attacks, including Denial of Service (DoS) attacks, Sybil attacks, and Data Replay, etc. Due to the lack of data sets [6], [12] in the early stage, MDS is a late study field. According to the latest survey [7], most of the datasets on the MDS work are not publicly available, and only two datasets are currently open source. Kamel et al. open-sourced the VeReMi_V1 [12] and VeReMi_V2 [13] misbehavior datasets in 2018 and 2020, both of which contain BSMs sent by vehicles. VeReMi_V1 contains only four types of position malfunction and Eventual Stop malfunction. VeReMi_V2 is an improvement and extension of VeReMi_V1 with more fields and 19 types of misbehavior. There have been many works done on both.

Based on VeReMi_V1, The work in [12] used plausibility checks combined with ML (SVM and KNN algorithms) to detect position malfunction and compared and analyzed five attacks by using four detectors with an average Recall of 96.8%. Uprety et al. used federated machine learning in [14], which locally trained vehicles' data and then averaged the model in the cloud, and circumvented the vehicle privacy problem, with an accuracy of 85.71% for detecting position malfunction. Ercan et al. proposed a location forgery detection algorithm with centralized training and distributed detection in [15], which used ensemble methods (KNN and Random Forest) and achieved 85.3% accuracy on average. Sharma et al. integrated plausibility checks with ML models and ensemble-voted six models' results in [16], which proved the plausibility checks are valid and got an AUC of 85.03% and an accuracy of 88.61%. Liu proposed a deep learning of CNN and LSTM method in [17] to detect position malfunction and achieved 90% and 88% accuracy, respectively. However, most of the above methods use traditional machine learning, which is unsuitable for the large amount of data in V2X systems. Moreover, the VeReMi_V1 dataset needs to be revised. There are only five types of misbehavior, no malicious attacks such as DoS and Sybil, and no acceleration and heading fields.

There are more misbehavior types (Increased from 5 to 19) and message fields (added PseudoID, Acceleration, Heading) in VeReMi_V2. Kamel et al. used plausibility and consistency checks in [13] to detect 19 misbehaviors and achieved an accuracy of 92.93%. Mahmoudi et al. in [18] extracted features from the local detector checks, engineered additional features from the raw beacon data, and detected 19 misbehaviors by the LSTM model and finally achieved an accuracy of 97%. Alladi et al. proposed a subsequences-based unsupervised CNN-LSTM architecture to detect 19 misbehaviors in [19], which got an accuracy of 98% by a thresholding algorithm. However, it could only detect fixed-length message sequences. Furthermore, another work [20] by the same authors added more message features and proposed three classification methods, which improved the detection efficiency but only detected 17 misbehaviors. The work in [21] proposed a detection algorithm based on reinforcement learning and LSTM without a score threshold, whose detection accuracy of 19 misbehaviors reached 98.8%. In addition, some works focused on a few specific types of misbehavior, such as position malfunction in [22], Sybil attacks in [23], and DoS attacks in [24]. However, there are many different types of misbehavior in real-world, and detecting only some would increase the risk of vehicles being attacked.

In summary, VeReMi_V1 fills the gap in the public data set

of misbehavior detection and VeReMi_V2 makes up for the deficiency of V1. The detection accuracy is significantly improved based on VeReMi_V2 and deep reinforcement learning methods. However, many works are poor in detection accuracy. Some works [22]–[24] only focus on a few misbehavior types. Some works [19], [20] have limitations on input. Some works [15], [16], [20] use multiple models for ensemble or voting, which will undoubtedly reduce the detection efficiency. Therefore, we need to build a single model to detect all misbehavior efficiently.

III. Preliminary

This section introduces the preliminary knowledge about the Attention mechanism and Mutil-head self-attention, which is the essential part of our SVMDformer.

A. Attention mechanism

The Attention mechanism is based on human visual attention, people tend to focus on some features when observing an object. To implement the Attention mechanism, we treat the input as $< Key, Value >$ pairs and calculate the similarity coefficient between Key and $Query$, which is regarded as the $Value$ weight coefficient, and then weight the sum of $Value$ to obtain the attention output. We use Q, K, V to denote $Query$, Key, and $Value$.

B. Self-attention

the Self-attention mechanism focuses on connections from within, with Q, K, V coming from the same data source. That is, Q, K, V is derived from the same vector by different linear transformations. The Self-attention process is as follows:

1. Suppose the input is $x = \mathbb{R}^{a\times b}$ (in our model, a=200, b=128), three linear matrices are $\boldsymbol{w}^q, \boldsymbol{w}^k \in \mathbb{R}^{b\times d}$, and $\boldsymbol{w}^v \in \mathbb{R}^{b\times c}$.
2. Q, K, V transformations: x is multiplied with each of the three linear matrices to obtain $Q, K \in \mathbb{R}^{a\times d}$, and $V \in \mathbb{R}^{a\times c}$:
$$\begin{cases} Q = x * \boldsymbol{w}^q, \\ K = x * \boldsymbol{w}^k, \\ V = x * \boldsymbol{w}^v. \end{cases} \tag{1}$$
3. Multiplication and Scaling: QK^T and divide by $\sqrt{d_k}$ to get the attention score matrix $G \in \mathbb{R}^{a\times a}$, which denotes the score between each time step:
$$G = (QK^T)/\sqrt{d_k} \tag{2}$$
d_k is the dimension of K, $1/\sqrt{d_k}$ is the scaling factor to prevent the inner product value from being too large.
4. Key-padding-mask (optional): Identify the padding part of the input by the Kpm vector: $[1, 1, ..., 0]$, 1 for non-padding and 0 for padding, allowing the self-attention layer to assign 0 weight to the padding part to ignore it:
$$G = G * Kpm^T. \tag{3}$$

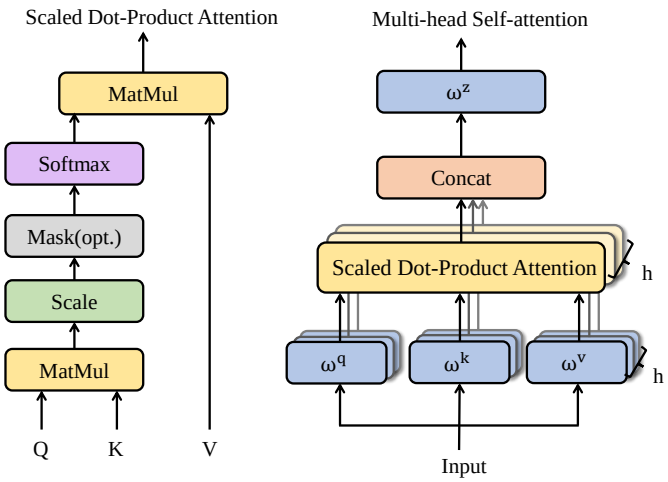


Fig. 2: Scaled dot-product (left) and Multi-head Self-attention (right). [10]

5. Softmax: Use $softmax$ to transform the score G to the attention weight matrix $W \in \mathbb{R}^{a\times a}$, which denotes the importance between each time step:
$$W = softmax(G) = softmax(QK^T/\sqrt{d_k}). \tag{4}$$
6. Dot-product: $W \cdot V$ yields the self-attention output matrix $z \in \mathbb{R}^{a\times c}$, which is an attention sequence incorporating a self-attentive mechanism:
$$\begin{aligned} z =& Attention(Q, K, V) \\ =& softmax(QK^T/\sqrt{d_k}) \cdot V. \end{aligned} \tag{5}$$

C. Mutil-head Self-attention

The Multi-head self-attention divides the Q, K, V into h subparts, thus allowing the model to acquire attention through different representation subspaces at different positions. With a single attention head, averaging inhibits this [10]. Multi-head self-attention with h-group linear matrices, and Perform steps 2 to 6 for h times to get h attention output $(z_1, ..., z_h)$, concatenate $(z_1, ..., z_h)$ together and linearly projected once to obtain the Mutil-head self-attention output:

$$\begin{aligned} &MultiHead(Q, K, V) = Concat(z_1, ..., z_h)W^z, \\ &\text{where } z_i = Attention(x * \boldsymbol{w}_i^q, x * \boldsymbol{w}_i^k, x * \boldsymbol{w}_i^v). \end{aligned} \tag{6}$$

The Scaled dot-product calculation and the Multi-head self-attention mechanism see Fig. 2.

IV. SVMDformer Framework

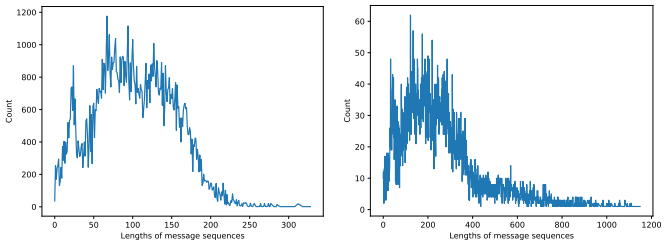
This section introduces the whole SVMDformer framework and presents both the IoV model and the misbehavior models.

A. IoV Model

In the IoV we design, RSUs are deployed on both sides of the road, and edge servers equipped with the SVMDformer framework are deployed in RSUs. The edge servers are connected to the cloud server through a high-speed wired link.

TABLE I: Misbehavior Model

ID	Type	ID	Type
0	Genuine	1	ConstPos
2	ConstPosOffset	3	RandomPos
4	RandomPosOffset	5	ConstSpeed
6	ConstSpeedOffset	7	RandomSpeed
8	RandomSpeedOffset	9	EventualStop
10	Disruptive	11	DataReply
12	DelayedMessages	13	Dos
14	DosRandom	15	DoSDisruptive
16	GridSybil	17	DataReplySybil
18	DoSRandomSybil	19	DoSDisruptiveSybil



(a) Dos not contained (b) Dos contained

Fig. 3: Statistics on the number of vehicular messages

Vehicles send BSMs with other vehicles, RSUs, pedestrians, and servers via wireless. After the SVMDformer model is trained on the cloud server, misbehavior detection can run on the edge server. Whenever a vehicle broadcasts over 10 messages, the nearest RSU will receive them and detect if it is a misbehaving vehicle through its edge server. Misbehaving vehicles will be reported to the cloud server and nearby vehicles so that users can take action. The IoV model is shown in Fig. 1.

B. Misbehavior Model

Misbehavior includes malfunctions and attacks. The former is a non-malicious behavior caused by the failure of OBUs, network, or vehicle sensors, while the latter is a malicious behavior of sending wrong information, mainly including:
malfunctions:

1) Position malfunctions: OBUs or sensor faults cause the vehicle to send wrong position information, including constant, random, constant offset, and random offset.
2) Speed malfunctions: OBUs or sensor faults cause the vehicle to send wrong speed information with the same error type as above.
3) Delayed Messages: Network problem causing the vehicle to send correct and complete messages, but from reality with delay δt.

Attacks:

4) DoS: The attacker sends messages more frequently than the limit set by the corresponding IEEE or ETSI standard.
5) DataReply: The attacker replayed the message previously received from specific neighbors.
6) Disruptive: The attacker replayed the message previously received from random neighbors.
7) Eventual Stop: The attacker simulates a sudden stop of the vehicle by freezing the position and setting the speed to null.
8) GridSybil: The attacker creates fake traffic congestion by forging a new ID and keeping the correct message frequency.

The 19 misbehaviors contained in VeReMi_V2 are shown in Table I

C. Dataset and preprocessing

We used the VeReMi_V2 [13] dataset to evaluate the model performance, which contains 19 misbehavior types as shown in Table I. The data fields include send time, sender ID, sender pseudo ID, message ID, position, speed, acceleration, and heading. For each misbehavior scenario, a GroundTruth (GT) file is generated, which contains unmodified genuine BSMs sent by all vehicles; meanwhile, each vehicle generates a trace file containing the received BSMs from all neighboring vehicles, where the messages of the Genuine vehicles are the same as those within the GT file, while the misbehaving vehicles modify the messages, vehicles of genuine: misbehaving= 7:3.

We generated the training and test sets by the BSMs of the vehicles; one training/test data corresponds to one vehicle. The number of messages for vehicles without Dos attacks varies from 0 to 328 (Fig. 3(a)), 101.7 on average, and more than 97.7% are less than 200. In contrast, the Dos-contained vehicles can reach up to 1000 (Fig. 3(b)) or more because of their shorter sending interval. To fit the input to the neural network, we fix the number of messages for a single vehicle to 200, truncating the long and making up the short with zeros. Due to their distinct features, the Dos contained vehicles are not affected by truncating in our experiments. Moreover, we considered that vehicles with less than ten messages contain too little information that should be discarded. Therefore, our model can detect any sequences with lengths > 10. In our IoV world, the detection starts after the vehicle enters the RSU range for 10 seconds. In addition, it is necessary to traverse the trace file names to generate the labels for the vehicles, 0 for Genuine and 1 for Misbehaving.

We selected six message features: sendTime, PseudoID, The (x, y) of pos, spd, acl, hed. Standardize pos, spd, acl, and hed with z-score before inputting to the model. After preprocessing, the original sequence of vehicle i at time step t is:

$$Seq_t^i = \{\Delta T_t^i, SumP^i, pos_{tx}^i, pos_{ty}^i, spd_{tx}^i, spd_{ty}^i, acl_{tx}^i, acl_{ty}^i, hed_{tx}^i, hed_{ty}^i\}, \tag{7}$$

where ΔT_t^i denotes the message sending interval:

$$\Delta T_t^i = sendtime_t^i - sendtime_{t-1}^i, \tag{8}$$

and $SumP^i$ denotes the sum of PseudoIDs that vehicle i has in the whole sequence. $i, t \in \mathbb{N}^+, t \in [0, 199]$. Thus, the size of the original sequence (Seq^i) is 200×10.

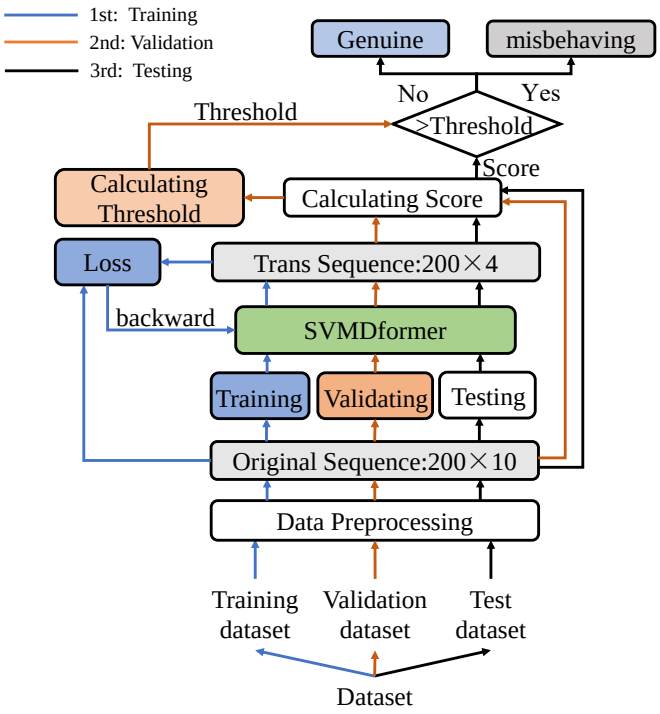


Fig. 4: SVMDformer Framework.

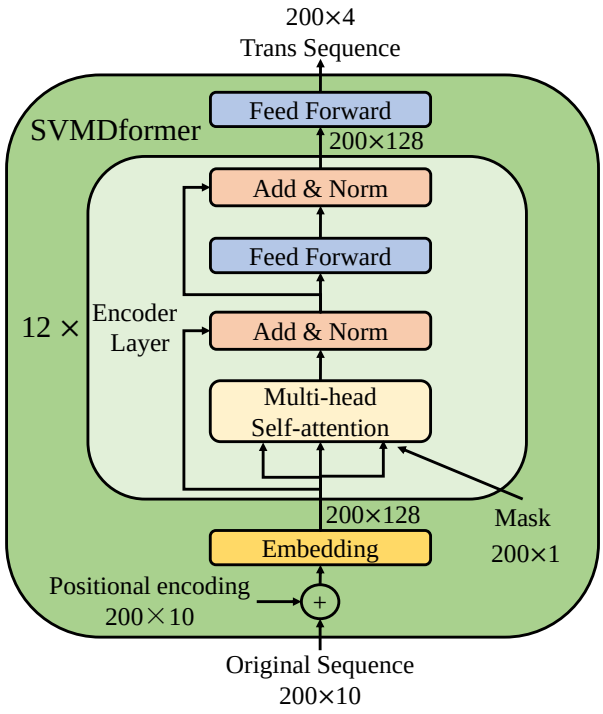


Fig. 5: SVMDformer Model Architecture.

D. SVMDformer Framework

The SVMDformer framework (Fig. 4) proposed in our work has three stages: Training, Validation, and Test. Training and Test are alternating processes, the model will be trained and validated at each epoch after convergence, and we save the best epoch of the model on the validation set during the training.

1) Stage 1: Training, optimize the model parameters by using loss back-propagation (backward) algorithm on the training set.
2) Stage 2: Validation, calculate the Scores of the validation set data by the trained model, and compute the AUC and a Threshold for misbehavior classification by the Scores and labels.
3) Stage 3: Testing, measure the final performance of the model. Calculate AUC by scores; calculate the Accuracy, F1, Recall, and Precision of the test set by the Threshold from stage 2.

After data preprocessing, the original sequences are inputted into the model and transformed into trans sequences containing position and speed features. Then we calculate the similarities of the two sequences to construct the Loss function in training and represent the misbehavior scores in Validation and Testing. In training, the model learns a large number of genuine data and a small number of difficult-to-identify misbehaving data (EventualStop); the training objective is to make the trans sequences of unlabeled and genuine vehicles fit their original sequences while misbehaving vehicles deviate from the original sequences. In this way, since the model has learned lots of genuine behavioral patterns, the misbehaving vehicles, including EventualStop, are bound to unfit the data distribution features learned by the model, thus causing the trans sequences to deviate from the original sequences. Therefore, misbehaving vehicles will also have a higher misbehavior score during the Validation and Testing than unlabeled and genuine vehicles so that a Threshold can be set for misbehavior classification.

E. SVMDformer Model Architecture

The SVMDformer proposed in our work is fine-tuned based on the Transformer Encoder. SVMDformer Model transforms the 200×10 original sequence into 200×4 trans sequence. The model structure is as Fig. 5. SVMDformer consists of an input layer with Positional Encoding and Embedding layer, an Encoder with 12 identical layers, and an output layer. We retain the structure of the vanilla encoder layer of Transformer and fine-tune the input and output layers. Each encoder layer has three layers: Multi-head self-attention (Attention), Add&Norm (ResNet [25]), and FeedForward (FF). Each layer of SVMDformer is as follows:

1) Positional Encoding: The Transformer is internally composed of linear layers, which are parallel computational structures without sequential information. To extract the sequential features of the input, we add vanilla sin/cos encodings [10] to the original sequences, which are calculated as:

$$\begin{cases} PE_{(pos,2i)} = sin(pos/10000^{2i/d_{msg}}), \\ PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{msg}}), \end{cases} \tag{9}$$

where i is the position, pos is the value at position i, and d_{msg} is the dimension of the original sequence. After

adding 200×10 positional encodings to the 200×10 original sequence, we get the 200×10 positional sequence.

2) Embedding layer: We add an Embedding layer before the encoder, aiming to expand the positional sequence to the 200×128 embedding sequence. More neurons reinforce the encoder layer learning the data distribution of the sequence.
3) Attention layer: We input the embedding sequence (or encoder sequence after the first encoder layer) to the Mutil-head self-attention layer and output the 200×128 attention sequence. See Section III-B and III-C for details of Mutil-head self-attention.
4) FeedForward layer: After the Attention layer is a FF layer consisting of two linear transformations, with a Relu activation in the middle to increase its nonlinear expressiveness. The middle layer is 512, and the FF layer transforms the attention sequence to 200×128 FeedForward sequence.
5) ResNet: is expressed as $LayerNorm(x + Sublayer(x))$. In the encoder, $Sublayers$ are Multi-head self-attention and FeedForward, which do not change the input dimension, the key to executing the add operation. ResNet can keep input and output data features of $Sublayer$, effectively avoiding the forgetting problem caused by a deep network. After adding the residuals, LayerNorm is used for normalization to avoid gradient disappearance and explosion problems.
6) Output layer: After the 12-layer encoder, we add an extra Feed Forward layer to convert the 200×128 encoder sequence into the 200×4 trans sequence. So we can compare the position and speed difference between the trans and original sequence.

F. Loss Function

We optimize the model parameters using the Loss backpropagation (backward) algorithm, so we need to set a training objective ($\min Loss$). First, we use the mean absolute error (MAE) of the trans and original sequence to represent the similarity between the sequences. The main malfunctions of vehicles are position and speed, and we obtain the best detection performance by comparing only position and speed features. So, the trans sequence has only four features indicating the position (x, y) and speed (x, y), and the original sequence only need to extract position and speed fields, so:

$$MAE = \frac{1}{l * w} \sum_{i}^{l} \sum_{j}^{w} |y_i^j - \hat{y}_i^j|, \tag{10}$$

where $l = 200, w = 4, y_i^j$ and $\hat{y}_i^j$ denote the values in the original sequence and trans sequence.

A semi-supervised method is used to fit trans sequences and original sequences of unlabeled and genuine vehicles, while a small number of labeled misbehaving vehicles are away from the original sequence, and the $Loss$ function is:

$$Loss = \frac{1}{n * m} \sum_{i}^{n} MAE_i + \frac{\lambda}{n * m} \sum_{j}^{m} (MAE_j)^{y_j}, \tag{11}$$

Where n is the number of unlabeled vehicles, m is the number of labeled vehicles, y_j is the semi-supervised label of vehicles, genuine is 1, and misbehaving is -1. It is assumed that all unlabeled vehicles are genuine. λ is the discount factor of labeled vehicles, if $\lambda > 1$ then the importance of labeled vehicles is emphasized, and $\lambda < 1$ then the importance of unlabeled vehicles is emphasized.

Since the vehicle labels are all known in the VeReMi dataset, we treat some genuine vehicles as unlabeled in order to distinguish labeled vehicles from unlabeled vehicles and highlight the role of λ. However, we found in our experiments that the model achieves the best results when $\lambda = 1$, so when $\lambda = 1$, the $Loss$ function is:

$$Loss = \frac{1}{n * m} (\sum_{i}^{n} MAE_i + \sum_{j}^{m} \frac{1}{MAE_j}), \tag{12}$$

where n is the number of genuine vehicles, m is the number of misbehaving vehicles.

G. Score and Threshold

In the Validation and Testing stage, instead of computing $Loss$, we only calculate the MAE between the original and trans sequences as the misbehaving $Score$ (S), abbreviated as:

$$S = MAE = \frac{1}{l * w} \sum |y - \hat{y}|, \tag{13}$$

Then we calculate the AUC to evaluate the detection effectiveness of the model. AUC (Area Under the roc Curve) is an important indicator in anomaly detection. Suppose there are n Genuine (G) vehicles and m Misbehaving (M) vehicles, so there are $n \times m$ pairs of opposite samples. The misbehavior score of each sample pair is expressed as $< S_G, S_M >$, and the AUC is equal to the probability that $S_G > S_M$ in these sample pairs, with the formula:

$$\begin{aligned} AUC &= \frac{1}{n * m} \sum f(S_G, S_M), \\ \text{where } f(S_G, S_M) &= \begin{cases} 1, & \text{if } S_G > S_M \\ 0.5, & \text{if } S_G = S_M \\ 0. & \text{if } S_G < S_M \end{cases} \end{aligned} \tag{14}$$

And then, We use an iterative algorithm (Algorithm 1) to find the best classification Threshold to classify the vehicles' behavior by validation set. Since S is high for misbehaving vehicles and low for genuine, the vehicle is considered as misbehaving if $S > Threshold$, and genuine if vice versa.

Finally, We calculate **Accuracy, F1-Score, Recall, Precision, and AUC (Eq** (14)**)** to evaluate the detection performance of the model in the Testing as well as Validation stage.

V. Model Performance Evaluation

This section evaluates the detection performance of the model, including detection under three types of supervision, detection of single misbehavior as well as score visualization of single misbehavior.

Algorithm 1: Threshold Algorithm

Input: $Scores, labels, Epochs$
Output: $Threshold$

$precision \leftarrow 0.1$
$Threshold \leftarrow 1$
$iteration \leftarrow 18$
foreach *Epochs* **do**
  $accuracy \leftarrow [0] * iteration$
  $threshold \leftarrow [0] * iteration$
  foreach i *in iteration* **do**
    $threshold[i] \leftarrow BestThreshold + precision * (i - iteration/2)$
    Calculating $accuracy[i]$ by $threshold[i]$
  end
  $maxAcc\_idx \leftarrow$ argmax($accuracy$)
  $Threshold \leftarrow threshold[maxAcc\_idx]$
  $precision \leftarrow precision * 0.1$
end
return

TABLE II: Hyperparameters of the SVMDformer

Name	Value	Explanation
epochs	160	Number of model iterations
seed	1	Random seeds
learning rate	1e-4	OneCycle [26] strategy, min_lr= 1e-4
activation	Relu	Activation in the feedforward layer
dropout rate	0.0	Neuron dropout rate
optimizer	AdamW	Adam + Weight decade [27]
n_layers	12	Number of encoder layers
n_es	250	Number of EventualStop training volume
batch size	256	Number of mini-batch samples
d_seq	200	Length of origin sequence
d_msg	10	Dimension of message features
d_emb	128	Dimension of embedding sequence
d_ff	512	Dimension between feedforward
d_out	4	Dimension of trans sequence
h	4	Number of Self-attention heads
eta	1	Factor of misbehaving data in the loss

A. Experimental environment

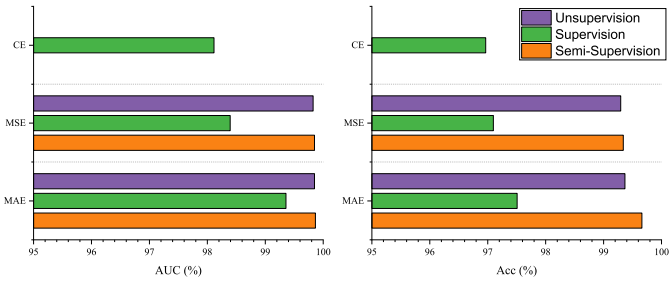
The experiments are based on the deep learning platform Pytorch 1.11.0, Python 3.8, and Cuda 11.3; the hardware environment relied on a cloud server, Graphics card: RTX A5000 * 1, Memory:24GB. CPU: Intel(R) Xeon(R) Platinum 8358P, Memory:80GB.

The training, validation, and test set are randomly divided into three independent subsets and their genuine data: misbehaving data = 62295: 500*19, 5700: 300*19, and 24700: 1300*19, where 19 misbehaviors are considered the same amount of genuine data. In the training set, genuine data will be learned entirely, while misbehaving data depends on the experiments. And the main hyperparameters of the SVMDformer are shown in Table II.

We use AdamW [27] as the optimizer of our model, which adds weight decay on top of Adam and is one of the best optimizers available. The learning rate adjustment strategy is the Onecycle [26]: The learning rate firstly warms-up to max_lr=2e-3 (10 epochs) from an initial_lr=1e-7, then falls to min_lr=1e-4 (40 epochs) and slowly decreases to 2e-6 in the last 110 epochs. The dropout rate is 0.0 because overfitting favors our model, which aims to deviate unlearned misbehaving vehicles from the model.

TABLE III: Detection under three supervision Methods

Method	Train set	Loss	AUC	Acc	F1	Rec	Pre
Unsupervised	62295:0	MSE	99.83	99.30	99.29	99.30	99.36
		MAE	99.85	99.37	99.36	99.37	99.45
supervised	62295:500*19	CE	98.12	96.96	96.81	96.96	97.72
		$MSE^{\tilde{y}}$	98.39	97.10	96.97	97.10	97.71
		$MAE^{\tilde{y}}$	99.36	97.51	97.38	97.51	98.06
Semi-supervised	62295:250	$MSE^{\tilde{y}}$	99.85	99.34	99.34	99.34	99.36
		$MAE^{\tilde{y}}$	**99.87**	**99.66**	**99.66**	**99.66**	**99.68**



(a) AUC (b) Acc

Fig. 6: AUC and Accuracy of each supervision method.

B. Detection under three types of supervision

This subsection compares the three supervision methods by learning different volumes of misbehaving vehicles. The train sets and results are shown in Table III, $\tilde{y}$ is the pseudo label in the Loss function, genuine is 1 while misbehaving is -1. In supervised learning, we also compare the binary classification methods using the cross-entropy (CE) loss function. The model's output is no longer a trans sequence but a 1×2 vector, representing the probability of genuine and misbehaving. However, the difference with the original sequence cannot be reflected in this way, and the ability to represent the data distribution is reduced in the downscaling process. As seen from Table III, it has the worst performance.

As shown in Table III and Fig. 6, among the three supervision methods, the similarity function using MAE is better than MSE. Comparing the MAE, the unsupervised model only learns 62,295 genuine vehicles with good results, the AUC reaches 99.85%, and the Accuracy is 99.37%. The supervised model learns an additional 500 vehicles of each of the 19 misbehaviors, but all indicators are the lowest. The semi-supervised model is the most accurate, with an AUC of 99.87% and an Accuracy of 99.66%, and it only learns **250 EventualStop vehicles**. Comparing AUC and Accuracy in Fig. 6, it is clear that the semi-supervised method is slightly better than the unsupervised method and much better than the supervised method.

C. Detection of Single Misbehavior

For each detection of single misbehavior, the test set with only one type of misbehavior, where genuine data: misbehaving data = 1300:1300. Table IV objectively shows the detection performance of the model for each misbehavior, where the Genuine results are the weighted average of the Genuine class for 19 simulations, and the MixAll results are the weighted average of 20 behaviors.

As can be seen, with unsupervised learning, the model has achieved good results except for the EventualStop (ES) type. The recall on ES type is only 80.7%, reacting that the model misses lots of this type of misbehavior, which indicates that the high similarity between EventualStop and Genuine causes the model to mistake ES for Genuine. The semi-supervised model only learns an additional 250 misbehaving vehicles of the ES type, and the recall of EventualStop improves to 92.34%. The recalls of 14 misbehaviors, such as ConstPos, and RandomPos, are all as high as 100%. The model has no omission for these misbehaviors, and the Accuracy of MixAll is as high as 99.66%, which can be said to be a near-perfect detector.

In contrast, the results of the supervised model are decreased in almost all misbehavior environments, especially RandomPosOffset. After learning the misbehavior of 19 types, the model starts to learn toward strange reverse. The detection for RandomPosOffset becomes unpredictable, with a recall of only 42.95%. It suggests that supervised learning is not a good choice for misbehavior detection because of the diversity of misbehavior types.

D. Score Visualization

In addition, we discuss our SVMDformer performance of single misbehavior scenarios by the Score scatter plot and Threshold line, which is shown in Fig. 7. For presentation purposes, we select the Top 150 data in the Genuine class and the 150 lowest data in the misbehaving class, i.e., the 300 data around the Threshold in each detection.

Overall (Fig. 7(a)), types 3, 7, 12, 14, 18, and 19 have an average score of more than 100, even 1000 due to distinct misbehaving features. It is a large gap from genuine data with an average score of lower than 0.004, which also has a tiny number of outliers. Near the Threshold of about 0.02 (Fig. 7(b)), although a small amount of misbehaving data was incorrectly identified, most of them are above the Threshold and. Type 9 (EventualStop), however, has a large number of occurrences below the Threshold (7.66%*1300=100 in fact), which is not identified by the model, causing the recall of this type only 92.34% (Table IV). In addition, it can be seen that type 2 (ConstPosOffset) is very close to the Threshold, which is also a Genuine-like misbehavior.

VI. Analysis and Discussion

In this section, we analyze the sensitivity of EventualStop (ES) training volume, compare SVMDformer with six baseline models and prior works, and perform ablation experiments.

A. Sensitivity analysis of EventualStop

We fix the number of genuine data as 62,295 and explore the effect of learning 0 to 500 EventualStop data on the SVMDformer in increments of 50.

As seen from Table V and Fig. 8(b), as the training volume of ES increases, the recall of ConstPosOffset, RandomPosOffset, DataReply, and Dos fluctuate but are not correlated with ES. As seen from Fig. 8(a), the recall of ES increases with the training volume, which indicates that the more training, the more can be recognized. However, some genuine vehicles move slowly in the last few steps, which is quite similar to ES misbehavior. Thus the model misclassifies some genuine data as the ES type, causing the Genuine results to decrease (the pink curve of Fig. 8(b)). However, when the training volume exceeds 250, the upward trend of EventualStop and the downward trend of Genuine start to slow down. The model achieves the best result at 250, and the Recall of MixAll is 99.66%.

In summary, when the training volume of EventualStop is 250, the detection between genuine and misbehaving vehicles is balanced, and the results reach the best.

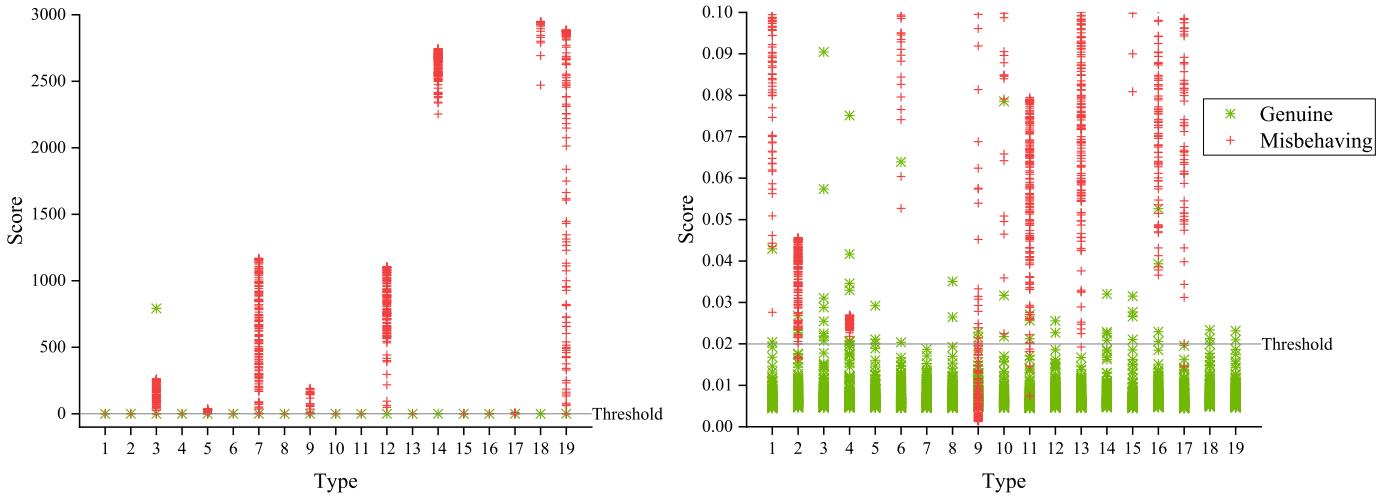
B. Comparison of Neural Networks Models

This subsection compares six baseline neural network models by replacing the model without changing other modules of the SVMDformer framework (Fig. 4). Each model and results (See Table VI) are as follows.

1) CNN: 2 × (Convolution + Batchnorm + Relu activation), then max-pooling. From Table VI, we know that the CNN is not good at the time-series data, the AUC is only 97.77%, and the Accuracy is only 94.35%.
2) Bi-LSTM: Bi-LSTM with two hidden layers of 512 dimensions, LayerNorm, and Feed Forward layer. The AUC is as high as the SVMDformer, 99.87%, confirming the advantage of LSTM for time series. However, other indicators are not good enough.
3) CNN-BiLSTM: Use model 1 to extract the local features and then model 2 to extract the temporal features, averaging the three feature vectors. The AUC 99.92% is among the best results, but the Accuracy is 99.1%, lower than the BiLSTM.
4) ResNet-18 [25]: Four residual blocks, each with four 1-dimensional convolution layers, plus the first convolution layer and the last linear layer, for a total of 18 layers. The AUC is 99.1% much better than CNN, proving its superiority.
5) Feed Forward: Two-layer linear with positional encodings added, middle dimension is 512. The AUC is only 93.94%, and the Accuracy is only 89.23%. Since the two-layer linear is too simple, the results are also terrible even after adding positional encodings.
6) Attention: Replace the 12-layer encoder with a single Mutil-head self-attention layer. While the AUC and Acc are 97.55% and 94.97% better than CNN and ResNet, proving the superiority of Mutil-head self-attention.

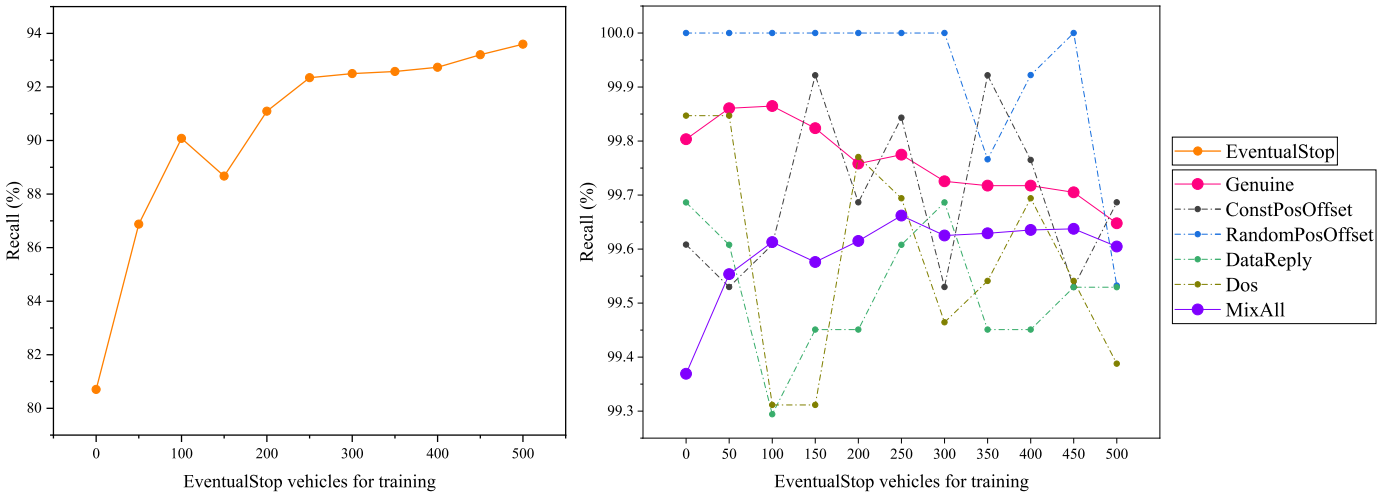
TABLE IV: Detection of single misbehavior

		Semi-supervised					Unsupervised					supervised				
ID	**Type**	**AUC**	**Acc**	**F1**	**Rec**	**Pre**	**AUC**	**Acc**	**F1**	**Rec**	**Pre**	**AUC**	**Acc**	**F1**	**Rec**	**Pre**
0	**Genuine**	99.87	99.66	99.67	99.77	99.58	99.85	99.37	99.41	99.80	99.10	99.36	97.51	97.87	99.14	97.05
1	**ConstPos**	100.00	99.88	99.88	100.00	99.77	100.00	99.96	99.96	100.00	99.92	99.95	99.49	99.49	99.84	99.14
2	**ConstPosOffset**	100.00	99.84	99.84	99.84	99.84	100.00	99.69	99.69	99.61	99.76	99.57	92.59	92.08	86.13	98.92
3	**RandomPos**	99.90	99.57	99.57	100.00	99.15	100.00	99.77	99.77	100.00	99.53	100.00	99.41	99.42	100.00	98.84
4	**RandomPosOffset**	99.86	99.77	99.77	100.00	99.53	99.93	99.77	99.77	100.00	99.53	90.58	71.01	59.70	42.95	97.87
5	**ConstSpeed**	100.00	99.92	99.92	100.00	99.84	100.00	99.88	99.88	100.00	99.76	100.00	99.49	99.49	100.00	98.99
6	**ConstSpeedOffset**	100.00	99.92	99.92	100.00	99.84	100.00	99.88	99.88	100.00	99.77	100.00	99.76	99.77	100.00	99.53
7	**RandomSpeed**	100.00	99.96	99.96	100.00	99.92	100.00	100.00	100.00	100.00	100.00	100.00	99.45	99.45	100.00	98.91
8	**RandomSpeedOffset**	100.00	99.88	99.88	100.00	99.77	100.00	99.96	99.96	100.00	99.92	100.00	99.65	99.65	100.00	99.30
9	**EventualStop**	97.72	96.09	95.94	92.34	99.83	97.22	90.27	89.24	80.70	99.81	97.84	96.64	96.57	94.45	98.77
10	**Disruptive**	99.95	99.81	99.81	100.00	99.61	100.00	99.81	99.81	99.92	99.69	99.97	99.57	99.57	99.84	99.30
11	**DataReply**	100.00	99.76	99.76	99.61	99.92	100.00	99.76	99.76	99.69	99.84	99.87	98.78	98.78	98.35	99.21
12	**DelayedMessages**	100.00	99.92	99.92	100.00	99.84	100.00	99.96	99.96	100.00	99.92	100.00	99.37	99.38	100.00	98.76
13	**Dos**	100.00	99.77	99.77	99.69	99.85	100.00	99.89	99.89	99.85	99.92	100.00	99.62	99.62	100.00	99.24
14	**DosRandom**	100.00	99.81	99.81	100.00	99.62	100.00	99.85	99.85	100.00	99.69	100.00	99.58	99.58	100.00	99.16
15	**DosDisruptive**	100.00	99.89	99.89	100.00	99.77	100.00	99.89	99.89	100.00	99.77	100.00	99.69	99.69	100.00	99.39
16	**GridSybil**	100.00	99.92	99.92	100.00	99.85	100.00	99.96	99.96	100.00	99.92	100.00	99.43	99.43	100.00	98.86
17	**DataReplySybil**	100.00	99.96	99.96	99.92	100.00	100.00	99.96	99.96	99.92	100.00	99.97	99.61	99.61	99.92	99.30
18	**DosRandomSybil**	100.00	99.96	99.96	100.00	99.92	100.00	99.88	99.88	100.00	99.77	100.00	99.73	99.73	100.00	99.46
19	**DosDisruptiveSybil**	100.00	99.92	99.92	100.00	99.85	100.00	99.85	99.85	100.00	99.69	100.00	99.65	99.65	100.00	99.31
20	**MixAll**	**99.87**	**99.66**	**99.66**	**99.66**	**99.68**	99.85	99.37	99.36	99.37	99.45	99.36	97.51	97.38	97.51	98.06



(a) Overall (b) Near the Threshold

Fig. 7: Threshold and Scores of single misbehavior scenarios.



(a) EventualStop (b) Five behaviors and MixAll

Fig. 8: Recall of six behaviors and MixAll in different ES training volumes.

TABLE V: Recalls under different ES training volume

Type	0	50	100	150	200	250	300	350	400	450	500
0	99.80	99.86	99.86	99.82	99.76	99.77	99.73	99.72	99.72	99.71	99.65
1	100	100	100	100	100	100	100	100	100	100	100
2	99.61	99.53	99.61	99.92	99.69	99.84	99.53	99.92	99.76	99.53	99.69
3	100	100	100	100	100	100	100	100	100	100	100
4	100	100	100	100	100	100	100	99.77	99.92	100	99.53
5	100	100	100	100	100	100	100	100	100	100	100
6	100	100	100	100	100	100	100	100	100	100	100
7	100	100	100	100	100	100	100	100	100	100	100
8	100	100	100	100	100	100	100	100	100	100	100
9	80.70	86.88	90.08	88.67	91.09	92.34	92.50	92.58	92.73	93.20	93.59
10	99.92	99.84	99.84	99.92	100	100	99.92	100	99.92	100	100
11	99.69	99.61	99.29	99.45	99.45	99.61	99.69	99.45	99.45	99.53	99.53
12	100	100	100	100	100	100	100	100	100	100	100
13	99.85	99.85	99.31	99.31	99.77	99.69	99.46	99.54	99.69	99.54	99.39
14	100	100	100	100	100	100	100	100	100	100	100
15	100	100	100	100	100	100	100	100	100	100	100
16	100	100	99.85	100	100	100	99.92	100	100	100	100
17	99.92	99.92	99.84	99.92	99.92	99.92	99.92	100	100	100	99.92
18	100	100	100	100	100	100	100	100	100	100	100
19	100	100	100	100	100	100	100	100	100	100	100
20	99.37	99.55	99.61	99.58	99.61	99.66	99.63	99.63	99.64	99.64	99.60

TABLE VI: Comparision of NN Models

Model	Class	F1	Rec	Pre	AUC	Acc
SVMDformer	Genuine	**99.67**	**99.77**	**99.58**	99.87	**99.66**
	Misbehaving	**99.65**	**99.55**	**99.77**		
CNN	Genuine	94.90	96.20	94.32	97.77	94.35
	Misbehaving	93.50	92.50	95.89		
BiLSTM	Genuine	99.33	99.42	99.26	99.87	99.32
	Misbehaving	99.31	99.22	99.42		
CNN-BiLSTM	Genuine	99.12	98.71	99.54	**99.92**	99.12
	Misbehaving	99.12	99.52	98.72		
ResNet	Genuine	95.16	95.95	95.12	99.11	94.54
	Misbehaving	93.49	93.14	95.51		
FeedForward	Genuine	90.91	97.16	86.61	93.94	89.23
	Misbehaving	86.45	81.30	96.21		
Attention	Genuine	95.84	98.66	94.04	97.55	94.97
	Misbehaving	93.35	91.27	98.40		

In summary, ResNet is superior to the CNN without residuals, Attention layer is simple but effective, and the temporal model LSTM performs well in the AUC, but all other metrics are inferior to our SVMDformer, which proves the stability and robustness of the SVMDformer.

C. Ablation

We remove each module on the proposed SVMDformer to judge each module's importance without changing the basic structure of the encoder layer. The main modules of SVMDformer include Positional Encoding (PE) and Embedding (Emb) in the input layer, Key-padding-mask (Kpm) in the Attention layer, and FeedForward (FF) in the output layer. In turn, we remove these four modules from SVMDformer, where the FF layer is replaced by a linear layer for converting to the trans sequence. The results in Table VII show that:

TABLE VII: Ablation

Model	Class	F1	Rec	Pre	AUC	Acc
SVMDformer	Genuine	**99.67**	**99.77**	**99.58**	**99.87**	**99.66**
	Misbehaving	**99.65**	**99.55**	**99.77**		
Without PE	Genuine	97.28	99.21	95.93	99.36	96.85
	Misbehaving	96.20	94.50	99.13		
Without Emb	Genuine	96.67	98.75	95.30	98.77	96.08
	Misbehaving	95.00	93.41	98.56		
Without Kpm	Genuine	99.54	99.72	99.37	99.85	99.53
	Misbehaving	99.52	99.34	99.72		
Without FF	Genuine	99.59	99.65	99.55	99.86	99.59
	Misbehaving	99.58	99.53	99.65		

- Without PE, the AUC and the Accuracy decrease by 0.5% and 2.8%, proves the importance of position information for the parallel computational linear transformations, which is the composition of our SVMDformer;
- Without Emb, the AUC decreases by 1.1% and the Accuracy even by 3.6%, which indicates that extending the feature dimension of the data can significantly improve the learning ability of the model;
- Without Kpm, the impact is minor, which means letting the Attention layer learn empty messages will not significantly impact learning normal messages;
- Without FF, replacing with a linear transformation also has little impact on the model, only slightly reducing its feature extraction ability when downscaling.

In summary, each SVMDformer module positively affects the model, especially the Positional encoding and the Embedding layer.

D. Comparison with previous work

TABLE VIII: SVMDformer and previous work

Detection Model	AUC	Acc	Pre	Rec	F1
SVMDformer	**99.87**	**99.66**	**99.68**	99.66	**99.66**
RL & LSTM [21]	-	98.82	97.24	**99.70**	98.45
DeepADV [19]	-	98.0	99.60	95.60	97.60
DLCE [20]	-	99.63	96.86	96.74	96.75
	-	97.09	95.81	95.59	95.58
	-	98.24	96.57	96.49	96.49
VeReMi_V2 [13]	-	92.93	99.12	82.28	89.92
LSTM [18]	-	97	94.5	91.77	93

Note: DLCE only detects 17 misbehaviors. Other works are 19.

Recently, there have been many works (Table VIII) on VeReMi_V2 about misbehavior detection.

Alladi et al. achieved 98% accuracy for 19 types of misbehaviors in DeepADV [19], but they have a limitation on input (fixed 20×10). Moreover, they made improvements on this basis in DLCE [20], and the detection performance was improved, but they eliminated the detection of EventualStop and DelayedMessages and proposed the labels as vehicle features for training. In comparison with the SOTA work [21], except for the slightly lower recall, our work has considerably improved in other indicators. The AUC is 99.87%, whereas other works lack this important indicator. The precision and F1-score are improved to 99.68% and 99.66%, especially the detection Accuracy improved from 98.82% to 99.66%. The false detection rate reduced from 1.18% to 0.34%, a reduction of more than three times.

VII. Conclusion

In this work, we propose a data-centric vehicular misbehavior detection framework SVMDformer based on semi-

supervised learning and Transformer. We train the SVMDformer model in the cloud server and perform misbehavior detection in the edge server for the vehicle BSMs in IoV. In our framework, we transform the vehicular message sequences into misbehavior scores and classifier the misbehavior a Threshold. In our experiments, we comprehensively demonstrate the detection performance of the model. We compare three supervision methods and find that semi-supervised is the best. We detect and discuss each misbehavior individually by visualizing the Scores. We target the difficult-to-identify type (EventualStop) for learning and find the optimal ES training volume of 250. We compare the baseline models and prove the superiority of our SVMDformer. We perform ablation experiments on SVMDformer and prove that all modules have positive effects, especially Positional Encoding and Embedding module. Compared to the SOTA work [21], we excel in AUC, Accuracy, Precision, and F1 metrics, reducing the false detection rate by more than three times! Moreover, The proposed SVMDformer framework can detect 19 misbehavior for vehicular sequences longer than 10. We can also detect misbehavior that is not learned or beyond the dataset based on semi-supervision. Thus, our SVMDformer is the most versatile, accurate, and comprehensive misbehavior detection framework available.

However, the SVMDformer can only identify whether a vehicle has misbehavior, but not the exact type. Therefore, we will dedicate how to identify the exact misbehavior type and efficiently deploy our SVMDformer framework to an IoV simulation system for misbehavior detection in the future.

REFERENCES

[1] Bloomberg. Number of cars sold worldwide from 2010 to 2022, with a 2023 forecast. (Oct 25, 2022).

[2] Rim Gasmi and Makhlouf Aliouat. Vehicular ad hoc networks versus internet of vehicles - a comparative view. In *2019 International Conference on Networking and Advanced Systems (ICNAS)*, pages 1–6, 2019.

[3] Yushan Siriwardhana, Pawani Porambage, Madhusanka Liyanage, and Mika Ylianttila. A survey on mobile augmented reality with 5g mobile edge computing: Architectures, applications, and technical aspects. *IEEE Communications Surveys Tutorials*, 23(2):1160–1192, 2021.

[4] Seng W. Loke. Cooperative automated vehicles: A review of opportunities and challenges in socially intelligent vehicles beyond networking. *IEEE Transactions on Intelligent Vehicles*, 4(4):509–518, 2019.

[5] Salabat Khan, Fei Luo, Zijian Zhang, Mussadiq Abdul Rahim, Mubashir Ahmad, and Kaishun Wu. Survey on issues and recent advances in vehicular public-key infrastructure (vpki). *IEEE Communications Surveys Tutorials*, 24(3):1574–1601, 2022.

[6] Rens Wouter van der Heijden, Stefan Dietzel, Tim Leinmüller, and Frank Kargl. Survey on misbehavior detection in cooperative intelligent transportation systems. *IEEE Communications Surveys Tutorials*, 21(1):779–811, 2019.

[7] Abdelwahab Boualouache and Thomas Engel. A survey on machine learning-based misbehavior detection systems for 5g and beyond vehicular networks. *IEEE Communications Surveys Tutorials*, pages 1–1, 2023.

[8] Lukas Ruff, Robert A Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*, 2019.

[9] Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. In *Neural Information Processing Systems (NeurIPS)*.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Rens W van der Heijden, Thomas Lukaseder, and Frank Kargl. Veremi: A dataset for comparable evaluation of misbehavior detection in vanets. In *International conference on security and privacy in communication systems*, pages 318–337. Springer, 2018.

[13] Joseph Kamel, Michael Wolf, Rens W. van der Hei, Arnaud Kaiser, Pascal Urien, and Frank Kargl. Veremi extension: A dataset for comparable evaluation of misbehavior detection in vanets. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[14] Aashma Uprety, Danda B. Rawat, and Jiang Li. Privacy preserving misbehavior detection in iov using federated machine learning. In *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6, 2021.

[15] Secil Ercan, Marwane Ayaida, and Nadhir Messai. Misbehavior detection for position falsification attacks in vanets using machine learning. *IEEE Access*, 10:1893–1904, 2022.

[16] Prinkle Sharma and Hong Liu. A machine-learning-based data-centric misbehavior detection model for internet of vehicles. *IEEE Internet of Things Journal*, 8(6):4991–4999, 2020.

[17] Xiangyu Liu. Misbehavior detection based on deep learning for vanets. In *2022 International Conference on Networks, Communications and Information Technology (CNCIT)*, pages 122–128. IEEE, 2022.

[18] Issam Mahmoudi, Joseph Kamel, Ines Ben-Jemaa, Arnaud Kaiser, and Pascal Urien. Towards a reliable machine learning-based global misbehavior detection in c–its: Model evaluation approach. In *Vehicular Ad-hoc Networks for Smart Cities*, pages 73–86. Springer, 2020.

[19] Tejasvi Alladi, Bhavya Gera, Ayush Agrawal, Vinay Chamola, and Fei Richard Yu. Deepadv: A deep neural network framework for anomaly detection in vanets. *IEEE Transactions on Vehicular Technology*, 70(11):12013–12023, 2021.

[20] Tejasvi Alladi, Varun Kohli, Vinay Chamola, and F Richard Yu. A deep learning based misbehavior classification scheme for intrusion detection in cooperative intelligent transportation systems. *Digital Communications and Networks*, 2022.

[21] Roshan Sedar, Charalampos Kalalas, Francisco Vázquez-Gallego, and Jesus Alonso-Zarate. Reinforcement learning based misbehavior detection in vehicular networks. In *ICC 2022 - IEEE International Conference on Communications*, pages 3550–3555, 2022.

[22] Goodness Oluchi Anyanwu, Cosmas Ifeanyi Nwakanma, Jae-Min Lee, and Dong-Seong Kim. Real-time position falsification attack detection system for internet of vehicles. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2021.

[23] Jing Zhao and Ruwu Wang. Fedmix: A sybil attack detection system considering cross-layer information fusion and privacy protection. In *2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 199–207, 2022.

[24] Roshan Sedar, Charalampos Kalalas, Jesus Alonso-Zarate, and Francisco Vázquez-Gallego. Multi-domain denial-of-service attacks in internet-of-vehicles: Vulnerability insights and detection performance. In *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, pages 438–443, 2022.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.