mathematics

MDPI

Article

Malicious Traffic Classification via Edge Intelligence in IIoT

Maoli Wang *, Bowen Zhang, Xiaodong Zang *, Kang Wang and Xu Ma

School of Cyber Science and Engineering, Qufu Normal University, Qufu 273165, China; zhangbwqfnu@163.com (B.Z.); wkk981229@163.com (K.W.); xma@qfnu.edu.cn (X.M.)
* Correspondence: wangml@qfnu.edu.cn (M.W.); xdzang@qfnu.edu.cn (X.Z.)

Abstract: The proliferation of smart devices in the 5G era of industrial IoT (IIoT) produces significant traffic data, some of which is encrypted malicious traffic, creating a significant problem for malicious traffic detection. Malicious traffic classification is one of the most efficient techniques for detecting malicious traffic. Although it is a labor-intensive and time-consuming process to gather large labeled datasets, the majority of prior studies on the classification of malicious traffic use supervised learning approaches and provide decent classification results when a substantial quantity of labeled data is available. This paper proposes a semi-supervised learning approach for classifying malicious IIoT traffic. The approach utilizes the encoder–decoder model framework to classify the traffic, even with a limited amount of labeled data available. We sample and normalize the data during the data-processing stage. In the semi-supervised model-building stage, we first pre-train a model on a large unlabeled dataset. Subsequently, we transfer the learned weights to a new model, which is then retrained using a small labeled dataset. We also offer an edge intelligence model that considers aspects such as computation latency, transmission latency, and privacy protection to improve the model's performance. To achieve the lowest total latency and to reduce the risk of privacy leakage, we first create latency and privacy-protection models for each local, edge, and cloud. Then, we optimize the total latency and overall privacy level. In the study of IIoT malicious traffic classification, experimental results demonstrate that our method reduces the model training and classification time with 97.55% accuracy; moreover, our approach boosts the privacy-protection factor.

Keywords: industrial internet of things; encrypted malicious traffic classification; semi-supervised learning; edge intelligence

MSC: 68T07

Citation: Wang, M.; Zhang, B.; Zang, X.; Wang, K.; Ma, X. Malicious Traffic Classification via Edge Intelligence in IIoT. *Mathematics* **2023**, *11*, 3951. https://doi.org/10.3390/math11183951

Academic Editor: Chong-zhi Gao

Received: 21 August 2023
Revised: 10 September 2023
Accepted: 14 September 2023
Published: 17 September 2023

1. Introduction

After the worldwide announcement regarding the fourth industrial revolution, China put forward the 2025 industrial manufacturing strategy [1]. Although the Industrial Internet of Things (IIoT) has been booming in recent years, it has also been subject to more and more cyber attacks and malicious behaviors, which has led to the leakage of sensitive information, damage to industrial infrastructure, and economic losses [2]. Therefore, the IIoT security issue needs to be resolved as soon as possible. Detecting malicious traffic data entering and leaving the IIoT and taking the necessary precautions against malicious attacks is an effective approach. By monitoring and analyzing network traffic, potential security threats, anomalous behavior, and intrusion attempts can be detected.

Previous methods for traffic detection, such as traditional deep packet inspection (DPI) [3,4], have proven effective in identifying malicious unencrypted traffic. However, the majority of traffic data today is encrypted traffic. As the detection of encrypted malicious traffic in IIoT can typically be seen as a traffic classification problem [5], port-based and machine-learning methods are very effective for classifying encrypted malicious traffic [6], but the accuracy cannot be guaranteed in the face of a significant increase in traffic. Deep-learning techniques, which are trained to automatically choose traffic features and have a

larger capacity for learning than that of typical machine-learning techniques, can identify encrypted harmful traffic in the presence of an enormous amount of encrypted traffic classification.

In IIoT malicious traffic classification, traditional deep-learning methods are based on supervised methods [7], which require enough labeled data and sufficient computational power during training. Unfortunately, capturing and labeling large datasets is time-consuming and labor-intensive. In contrast, unlabeled data are abundant and easily available. Therefore, semi-supervised deep-learning methods are more suitable for classifying malicious IIoT traffic [8]. At the same time, semi-supervised learning is better able to handle noise and labeling errors because the presence of unlabeled data can smooth and regularize the model, therefore reducing the risk of overfitting. In addition, unlabeled data can help the model to better adapt to new domains or tasks, therefore improving the generalization ability. This approach combines supervised and unsupervised learning, utilizing a small proportion of labeled data as input.

When using deep-learning models for malicious traffic detection [9], the traditional idea is to transfer large amounts of data from IIoT devices to the cloud for processing [10], which leads to higher cost and transmission latency; additionally, this method may risk privacy leakage issues. However, if the model is put locally, it can lead to higher computational latency due to weak local computing power [11]. Edge servers are introduced to integrate computation latency, transmission latency, and privacy-protection factors across local, edge-side, and cloud environments. This integration improves the model's classification efficiency while maintaining the same accuracy in classifying malicious traffic and reducing privacy leakage concerns.

To enhance the security performance of IIoT and tackle the challenge of detecting malicious traffic, we propose an IIoT malicious traffic classification method. Our method aims to accurately identify malicious traffic within a significant volume of encrypted traffic while minimizing the need for manual feature extraction. To achieve this, we employ a deep-learning model as the classification model. At the same time, to resolve the problem of difficult labeling of real captured traffic, we choose a semi-supervised learning approach. In addition, to address the problems of high latency of model training and classification and the associated privacy leakage during the process, we choose to introduce an edge intelligence model. In summary, our proposed method is an EI-based approach to classify malicious traffic for IIoT. The main contributions of this paper are as follows:

(1) Our proposal involves a semi-supervised deep-learning method capable of classifying malicious traffic in IIoT. This method leverages a significant quantity of unlabeled data alongside a small portion of labeled data.

(2) We propose a method to improve the performance of classification models using edge intelligence. This method considers the computational latency, transmission latency, and privacy-preserving factors during model training and classification.

(3) We model the latency and privacy protection of the edge intelligence model for local, edge, and cloud separately. Then, we optimize them by quantifying the total latency and total privacy level.

(4) Experiments reveal that, compared to that of the previous IIoT malicious traffic classification methods, our method achieves a classification accuracy of 97.55% when using the UNSW-NB15 dataset while minimizing overall latency and the danger of privacy leakage.

2. Related Work

Network traffic is essential for capturing the behavior process of a network [12], containing comprehensive information about the entire communication between source and destination hosts. Analyzing network traffic can not only understand network bandwidth and allocate bandwidth but also evaluate current network capacity utilization. In particular, by analyzing traffic patterns, abnormal behaviors, and malicious traffic characteristics, security issues can be discovered promptly. The traffic data of IIoT exhibit distinct char-

acteristics, including dynamism, volume, and temporal dependence. For the study of IIoT traffic classification, a part of the research approach utilizes the methods of traffic classification in traditional networks, mainly based on machine learning and deep learning.

Although machine learning is effective at classifying malicious traffic, it necessitates the labor and time-intensive human extraction of features. Fu et al. [1] uses clustering to detect anomalous traffic in IIoT, and in the paper, they propose a hierarchical detection method that first statistically analyzes the detected traffic frequencies and then detects the traffic attributes using a clustering algorithm. Niu et al. [13] proposes an adaptive random forest algorithm (IARF) that is capable of adaptively updating parameters when dealing with new types of malicious traffic, and it is also sensitive to traffic information with a few malicious samples. Ikram et al. [14] proposes an MNSWOA IPM RF method, which divides the traffic classification problem into feature selection and classification prediction, and improves the feature selection part using the whale optimization approach and ideal point method. Yan et al. [15] proposes a small-scale learning algorithm HCA-MBGDALRM, which improves the processing speed of the dataset through a parallel framework, while still addressing the problem of data skewing.

Deep learning has the advantage of automatic feature selection, which can optimize the problem of the manual feature extraction part of machine learning. Moreover, deep learning is capable of handling complex data and exhibits excellent scalability and flexibility. Transfer learning and pre-trained models in deep learning also demonstrate remarkable performance in practical applications. Researchers classify deep-learning-based anomalous traffic detection into three categories: supervised learning, semi-supervised learning, and unsupervised learning [16]. The main difference between the three methods is whether the input dataset is labeled, and recent research on IIoT malicious traffic classification is mainly focused on fully supervised learning.

Wang et al.'s method [17] achieves 86.6% accuracy in classifying 12 traffic types using a one-dimensional convolutional neural network (1D-CNN). Lin et al. [18] presents a cryptographic traffic recognition scheme called TSCRNN. This scheme utilizes CNN to extract abstract spatial features and introduces stacked bidirectional LSTM to learn temporal features. Zainudin et al. [19] proposes a method that can detect DDoS attacks in IIoT well, combining an effective feature selection method XGBoost. Shahin et al. [20] uses the LSTM model when detecting malicious traffic, and enhances the LSTM with CNN and full convolution neural network (FCN).

There is an increasing amount of research on edge intelligence. Edge intelligence technologies are maturing and being applied in a wide range of scenarios. Zhao et al. [21] proposes a model of IoT encrypted flow detection based on edge intelligence, which reduces the time for establishing the model. Zeb et al. [22] proposes a new edge-native framework for intelligently predicting data traffic, and Mohammed et al. [23] also introduces the concept of edge intelligence when classifying IoT traffic. From the above, when edge intelligence is combined with malicious traffic classification of the IIoT, it can also improve the training efficiency of the classification model and reduce the probability of privacy leakage. Qi et al. [24] proposes a blockchain-driven traffic classification method for edge computing in addressing normal traffic classification for IIoT, which effectively reduces time overhead and memory usage. In addition, edge intelligence can also reduce bandwidth requirements, enhance offline functionality, improve system reliability, and save network costs. These advantages can provide technical support for industrial IoT security.

In our approach, we exploit features such as the temporal correlation of IIoT traffic data, classify them using deep learning, an automatic feature extraction method, and semi-supervised learning models to reduce the dependence on labeled datasets, and finally introduce edge intelligence to improve the efficiency of model training and classification and to reduce the probability of privacy leakage. The methods in the above literature are summarized in Table 1.

Table 1. Literature summary.

Paper	Machine Leaning	Deep Leaning	Supervised	Unsupervised	Semi-Supervised	EI
Fu [1]	✓			✓		
Niu [13]	✓		✓			
Ikram [14]	✓		✓			
Yan [15]	✓		✓			
Latif [25]	✓		✓			
Wang [17]		✓	✓			
Shapira [26]		✓	✓			
Lin [18]		✓	✓			
Zainudin [19]		✓	✓			
De [27]		✓	✓			
Shahin [20]		✓	✓			
Zhao [21]	✓		✓			✓
Zeb [22]		✓	✓			✓
Mohammed [23]	✓		✓			✓
Qi [24]	✓		✓			✓
Hu [28]		✓			✓	
Ning [29]		✓			✓	
Our Method		✓			✓	✓

3. Methodology

Overall, our strategy involves classifying malicious traffic for IIoT traffic data, which contains four components. The first part is data processing, where we sample the captured traffic information in three different ways, and then select time series features and basic features in the data for normalization. The second part is the pre-training model in the semi-supervised training model, where we present the encoder–decoder model architecture for encoding in the pre-training phase and use unlabeled data as the input to the pre-training model. The third part is the re-training model in the semi-supervised training model. During re-training with a small quantity of labeled data, this model transfers the parameters and weights from the pre-training model. The re-training step includes decoding; then, the classifier outputs the traffic classes. The fourth part is the edge intelligence model. In this model, we build latency models and privacy-preserving models for cloud, edge, and local areas to improve the training and classification efficiency of the semi-supervised model by optimizing the total latency and privacy level, while reducing the risk of privacy leakage. The overall model architecture is shown in Figure 1.

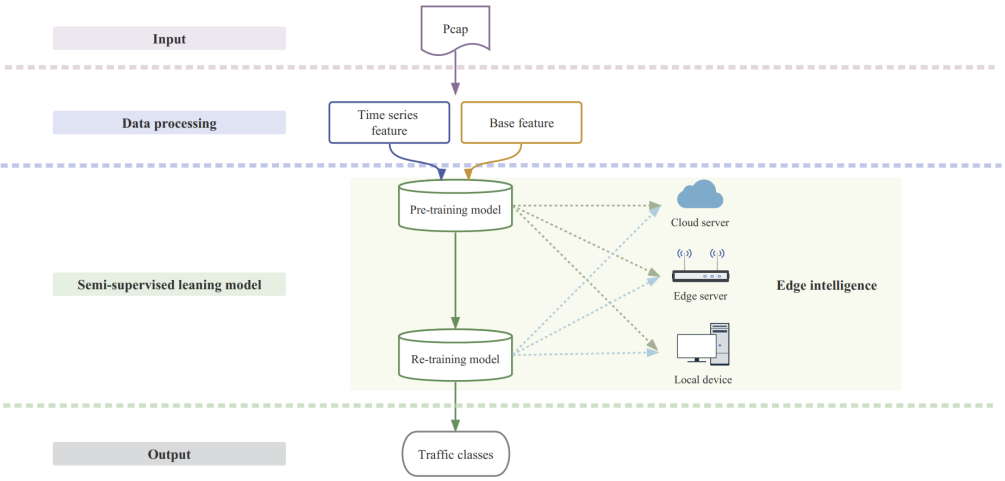


Figure 1. The framework of Malicious Traffic in IIoT.

3.1. Data Processing

Since the traffic density is very high in real scenarios, it is necessary to sample the data. Data sampling can not only save computational resources and accelerate model training but also balance the bias and variance of the model, improving its generalization ability while maintaining data representativeness. Among various sampling methods, we selected three sampling methods that are feasible in practical applications to compare the effects, namely random sampling, systematic sampling, and clustered sampling. Random sampling means the traffic packets are sampled with equal probability. Systematic sampling means that the first traffic packet is selected randomly and that the other traffic packets are selected using a fixed sampling interval. Cluster sampling means we use the sub-packet groups of the overall traffic packet as the sampling unit, and the whole traffic packet is divided into several sub-packet groups, called clusters; then, a complete cluster is randomly selected as the sampling sample. The pattern diagrams of the three sampling methods are shown in Figure 2.

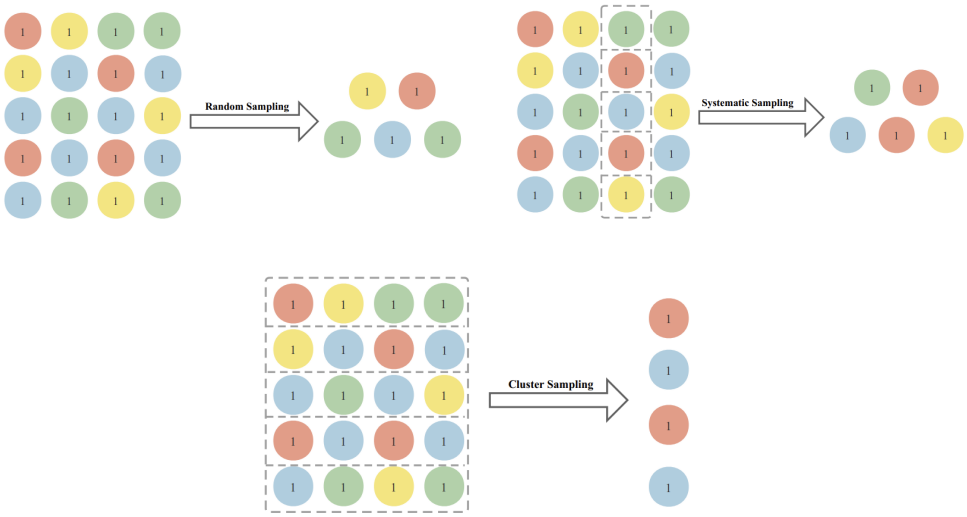


Figure 2. The pattern diagrams of the three sampling methods.

After sampling the traffic data, the time series features in the encrypted traffic are extracted, including the source port, the destination port, the payload size, the window size, and the traffic duration. Then the data are normalized, and the input features are scaled to values in the range $[-1, 1]$.

3.2. Pre-Training Model

CNNs were chosen for both the pre-training and re-training models because of their shift-invariant property, which allows CNNs to capture the output traffic pattern even if it is shifted to another input region. At the same time, CNN reduces the amount of calculation through local connections. Due to weight sharing, the network can use the same weights at different locations for the feature extraction mechanism, thus reducing the number of parameters that need to be trained.

During the pre-training stage of the model, an encoder is employed to convert the input sequence into a fixed-length vector. Subsequently, in the re-training model, a decoder is utilized to transform the vector into an output sequence. The encoder–decoder approach [30], also known as Seq2Seq, is characterized by its end-to-end learning algorithm.

In the encoder–decoder architecture [31] of the Seq2Seq model, given an input sequence $x = \{x_1, x_2, \ldots, x_m\}$ with length m, the model generates a target sequence $y = \{y_1, y_2, \ldots, y_n\}$ with length n. Figure 3 illustrates this architecture, where the encoder hidden states: $\{h_0, h_1, \ldots, h_m\}$, the decoder hidden states: $\{s_0, s_1, \ldots, s_n\}$, the contextual sequence $c = \text{Encoder}(x_1, \ldots, x_m)$.

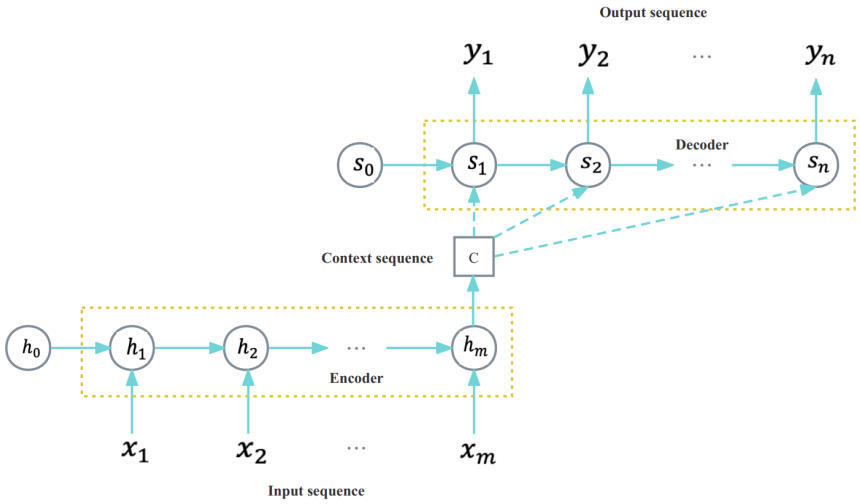


Figure 3. The encoder–decoder architecture of the Seq2Seq model.

To tackle the issue of suboptimal final classification caused by lengthy input sequences, the model [32] incorporates an attention mechanism. Attention mechanism allows the model to allocate different attention weights to different parts of the input data. By doing so, the model can focus more on the information relevant to the current task while ignoring irrelevant information. Additionally, the attention mechanism helps the model suppress noise and interference, which further enhances the model's performance. This updated Seq2Seq model, depicted in Figure 4, addresses the problem effectively.

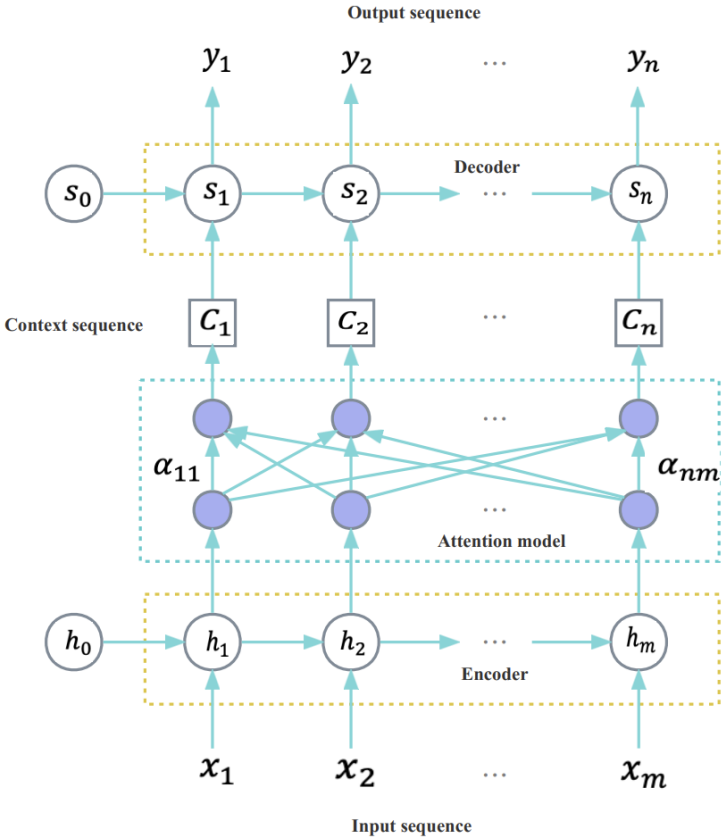


Figure 4. The Seq2Seq-model with attention mechanism.

In the attention model, each context sequence is a weighted sum of all hidden state vectors of the encoder as

$$c_i = \sum_{j=1}^{m} \alpha_{ij} h_j \tag{1}$$

The input sequence is mapped into multiple context sequences $c_1, c_2, c_3, \ldots, c_n$, where c_i is the context information corresponding to the output y_i (where $i = 1, 2, 3, \ldots, n$). When the decoder predicts the output y_i, its result depends on the matching context sequence c_1 and its previous hidden state, i.e.,

$$y_i = \text{Decoder}(c_i, s_1, \ldots, s_{i-1}), i = 1, \ldots, n \tag{2}$$

Figure 5 shows the architecture of the CNNs-based pre-training model, which is encoded in the pre-training phase.

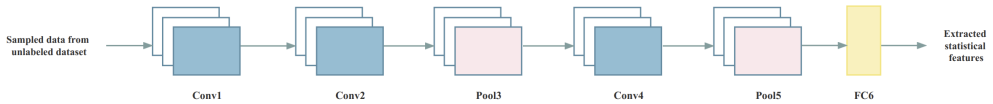


Figure 5. Pre-training model.

3.3. Re-Training Model

The weights learned by the pre-trained model are transferred to the retrained model, which is then retrained with a small, labeled dataset and finally decoded for classification. Since many traffic patterns have been observed as part of the pre-trained model, adding a small amount of manually labeled data during re-training can accomplish a fast classification task that makes re-training converge faster.

To avoid experimental chance, we choose a five-fold cross-validation method by dividing the traffic package into five parts, taking one of them as the test set each time and the remaining four as the training set; we cycle through the validation five times.

The final classifier selected the softmax classifier, which solved the multi-classification problem, i.e., it can classify normal traffic and different types of malicious traffic. The softmax function formula is as follows.

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{c=1}^{C} e^{z_c}} \tag{3}$$

where (z_i) is the output value of the i-th node, and C is the number of output nodes, i.e., the number of classified categories.

Recent neural networks commonly employ cross-entropy as the loss function for classification problems [33]. Extensive experiments have confirmed that utilizing cross-entropy as the loss function is indeed a superior choice.

The CNNs-based re-training model architecture is shown in Figure 6. Table 2 shows the model architecture parameters.

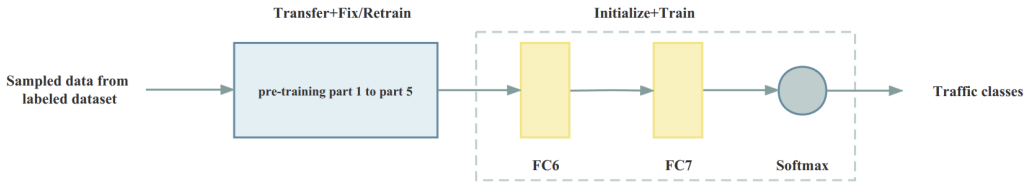


Figure 6. Re-training model.

Table 2. Structure of the CNNs model.

-	Conv1	Conv2	Pool3	Conv4	Pool5	FC6	FC7
Number of filters/neurons	32	32	-	64	-	64	32
Kernel size	5	5	3	3	3	-	-

3.4. Edge Intelligence Framework

Edge intelligence (EI) technology synergistically combines the computing capabilities of end devices and edge servers, harnessing the complementary strengths of local and high computing. As a result, it effectively minimizes latency and reduces energy consumption during the inference of deep-learning models [34]. Ref. [35] Edge intelligence is classified into six levels, from level one to level six, with an increasing percentage of edge intelligence involvement. As the EI level increases, the number of data offloads and path lengths decrease, which then leads to a decrease in transmission waiting for the time for data offloading, an increase in data confidentiality, and a decrease in WAN bandwidth cost; however, computational latency and energy consumption increase [35]. Figure 7 shows the comparison of the six capabilities corresponding to the cloud, edge, and device sides.

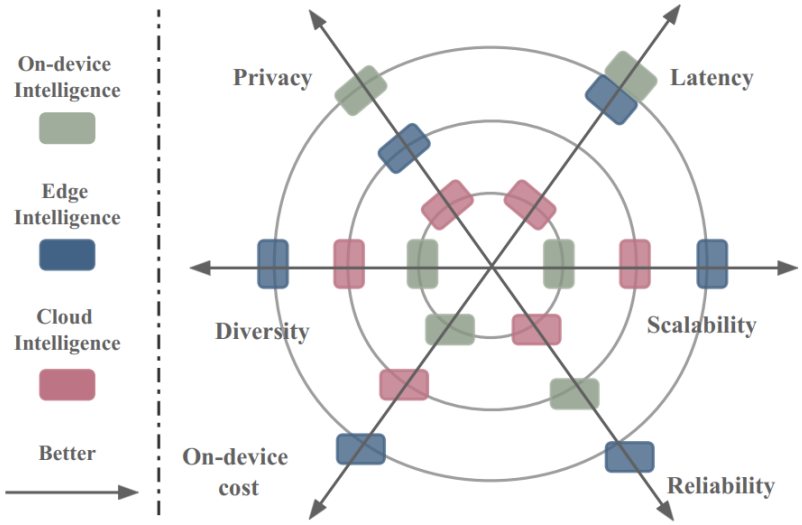


Figure 7. Comparison of cloud, edge, and device-side capabilities.

As shown in Figure 7, the advantage of edge intelligence is that diversity, scalability, reliability, and latency are better. Meanwhile, privacy can be protected; however, although cloud-side execution has strong computational power, the data transmission process increases the transmission latency and raises the risk of privacy leakage. For local device-side execution, although the data transmission process is reduced, which reduces the transmission latency and risk of privacy leakage, the computational power is insufficient and adds considerable computation time. Edge intelligence combines the advantages of cloud and local, which can better serve the model.

Many factors affect the choice of cloud, edge, and local. We consider three main aspects, namely calculation delay, transmission delay, and privacy protection. Among them, the calculation delay and transmission delay are established as delay models, and the privacy-protection model is established at the same time. Finally, we choose the scheme with the better two models.

3.4.1. Latency Model

When the task is executed locally [36], the cause of the delay is mainly the computational delay in executing the task $T_L(t)$, which is highly related to the working frequency

of the local Central Processing Unit (CPU) of user devices [37]. The local computational latency of the device $T_L(t)$ can be represented as

$$T_{L,C}(t) = A(t) \cdot \alpha_L(t) \cdot L f_L^{-1}(t) \tag{4}$$

where $A(t)$ denotes the total task volume, $\alpha_L(t)$ denotes the ratio of tasks processed locally, L refers to the necessary CPU cycles for executing the one-bit task [38], and $f_L(t)$ denotes the corresponding CPU-cycle frequency of the user device [39].

When tasks are offloaded to the edge server [40], the edge server has abundant computational resources to reduce computational latency compared to that of the local device, but it adds additional transmission latency during the offloading process. Therefore the latency of task offloading to the edge server includes both computation latency and transmission latency [41].

The computational latency of the edge server $T_{E,C}(t)$ can be described as

$$T_{E,C}(t) = A(t) \cdot \alpha_E(t) \cdot L f_E^{-1}(t) \tag{5}$$

The transmission latency of the edge server $T_{E,O}(t)$ can be formulated as

$$T_{E,O}(t) = \frac{A(t) \cdot \alpha_E(t)}{\left[\omega \log_2\left(1 + \frac{h(t)p(t)}{N_0 \omega}\right)\right]} \tag{6}$$

where ω is the channel bandwidth of the user device, $h(t)$ denotes the channel gain, $p(t)$ is the transmit power of the user device, and N_0 represents the power spectral density.

In summary, the total latency of the edge server $T_E(t)$ is the accumulation of computation latency and transmission latency [42].

$$T_E(t) = T_{E,C}(t) + T_{E,O}(t) \tag{7}$$

When tasks are offloaded to cloud servers, cloud servers have the most computational power compared to that of local devices and edge servers, but have the longest data transfer distance, incurring greater transfer latency and more noise interference during transmission. The latency of task offloading to cloud servers also includes two components.

The computational latency of a cloud server $T_{C,C}(t)$ can be formulated as

$$T_{C,C}(t) = A(t) \cdot \alpha_C(t) \cdot L f_C^{-1}(t) \tag{8}$$

The transmission latency of the cloud server $T_{C,O}(t)$ can be defined as

$$T_{C,O}(t) = \frac{A(t) \cdot \alpha_C(t)}{\left[\omega \log_2\left(1 + \frac{h(t)p(t)}{N_0 \omega}\right)\right]} \tag{9}$$

In summary, the total latency of the cloud server $T_C(t)$ is the accumulation of computation latency and transmission latency

$$T_C(t) = T_{C,C}(t) + T_{C,O}(t) \tag{10}$$

3.4.2. Privacy Preservation Model

As the data transition is from local execution to edge server offloading, and eventually to cloud server offloading, the distance of data transmission increases significantly, resulting in a reduced level of data privacy and confidentiality. Therefore, the following privacy-protection model is established, which is different from the encryption-based approach in cryptography. The overall privacy level P is represented by the following formula, The smaller the value of P, the higher the privacy level.

$$P = (P_L \alpha_L + P_E \alpha_E + P_C \alpha_C) * 10 \tag{11}$$

where P_L denotes the privacy level when the task is executed locally and takes values in the range [0, 0.1], P_E denotes the privacy level when the task is offloaded to the edge server, and the values range from [0.1, 0.5], and P_C denotes the privacy level when the task is offloaded to the cloud server, and the values range from [0.5, 1]. A higher privacy level indicates a higher chance of privacy leakage, and conversely, a lower privacy level indicates a lower chance of privacy leakage.

4. Experiments

4.1. Experimental Setup

We implemented the classification model using GPU-accelerated Python 3.7.0 and PyTorch, with CPU frequencies of 2.4 GHz (local device), 2.7 GHz (edge server), and 3.3 GHz (cloud server), corresponding to the experimental hardware platforms. Additionally, we assume that the local, edge, and cloud environments have the same task scheduling priorities when running the classification model and that the resources in each environment are capable of meeting the corresponding requirements.

We completed most of our experiments based on the UNSW-NB15 dataset which was created by the Cyber Range Laboratory of the Australian Cyber Security Center [43]. The dataset consists of 1,776,851 training data points and 761,508 test data points, which include nine families of attacks.

To verify the generality of our approach, we also conducted experiments on other datasets. One is the CTU-13 dataset, which is an IIoT network traffic dataset. Another dataset is the BoT-IoT dataset of IoT network traffic captured in recent years, which contains both normal traffic and botnet traffic [44]. Three different datasets, the UNSW-NB15 dataset, CTU-13 dataset, and BoT-IoT dataset, are denoted by D1, D2 and D3, respectively. The data volume and its proportion of attack types of the three datasets are shown in Table 3.

Table 3. Samples of different families in three data sets.

Data Set	Total	Family	Sessions	Percentage
D1	2,538,395	Normal	2,218,761	87.41%
		Generic	215,481	8.49%
		Exploits	44,525	1.75%
		Fuzzers	24,246	0.95%
		DoS	16,353	0.64%
		Reconnaissance	13,987	0.55%
		Analysis	2677	0.11%
		Backdoors	2329	0.09%
D2	2,824,636	Background Flows	2,753,290	97.47%
		Botnet Flows	39,933	1.41%
		Normal Flows	30,387	1.07%
		C&C Flows	1026	0.03%
D3	2,739,000	DDoS	1,954,760	71.37%
		DoS	519,706	18.77%
		Service Scanning	231,627	8.66%
		OS Fingerprinting	23,364	0.85%
		Normal	9543	0.35%

We used four metrics to evaluate the classification performances of the various methods, including accuracy (AC), precision (PR), recall (RC), and F1 [18].

$$\text{Accuracy (AC)} = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

$$\text{Precision (PR)} = \frac{TP}{TP + FP} \tag{13}$$

$$\text{Recall(RC)} = \frac{TP}{TP + FN} \tag{14}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{15}$$

where TP, FP, TN, and FN refer to true positives, false positives, true negatives, and false negatives, respectively.

There are also two evaluation metrics for latency and privacy level, which are defined from Formulas (6)–(13).

4.2. Experimental Results and Analysis

We compare the total latency and the privacy level of model training and classification under six scenarios during the experiment to make the classification model lower the total latency and minimize the risk of privacy leakage after incorporating edge intelligence. The classification model is divided into two parts, executed on local devices, edge servers, and cloud servers, respectively. The ratio parameters of the six scenarios are shown in Table 4, and the changes in the two evaluation metrics corresponding to different scenarios are shown in Figure 8.

Table 4. Parameter values.

Scenario	$\alpha_L(t)$	$\alpha_E(t)$	$\alpha_C(t)$
1	0	0	1
2	0	0.5	0.5
3	0.5	0	0.5
4	0	1	0
5	0.5	0.5	0
6	1	0	0

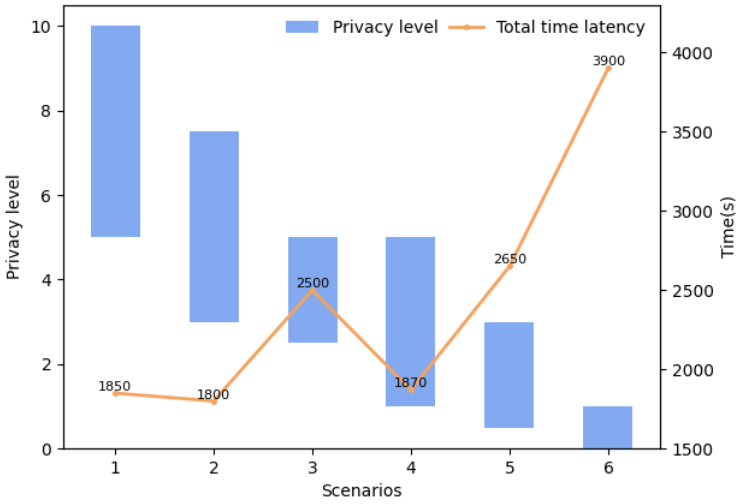
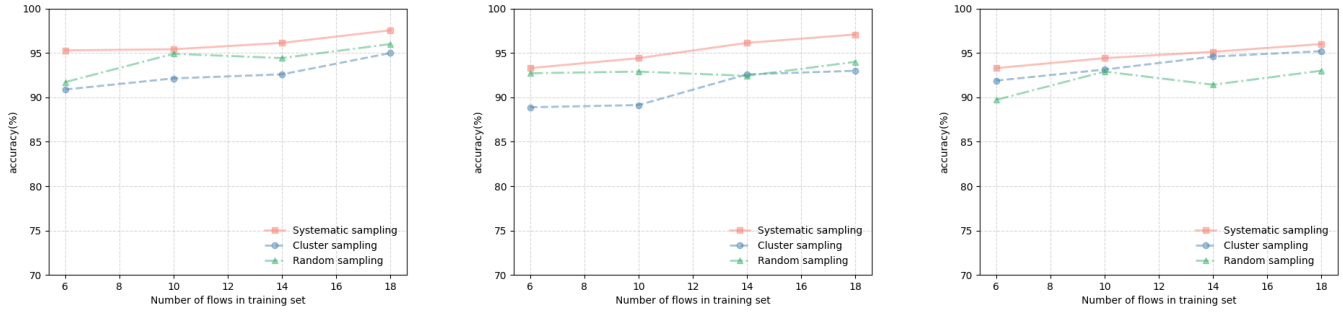


Figure 8. Variation of total latency and privacy level in different scenarios.

The definitions of total latency and privacy level are given in 3.4. Figure 8, shows that these six scenarios have advantages and disadvantages in terms of total latency and privacy-protection issues. The total latency of scenarios 1, 2, and 4 is lower and not much different, and the threshold of privacy level is set to 5. When the privacy level is lower than 5, privacy confidentiality is better, and the possibility of privacy leakage is lower. Therefore, scenarios 2–6 have a lower likelihood of privacy leakage.

Edge intelligence is introduced to reduce the total latency while reducing the possibility of privacy leakage. Combining these two factors, scenario 2 and scenario 4 are more suitable for the model in this paper. Additional metrics to evaluate our approach under scenarios 2 and 4.

When sampling the flow samples, the three sampling methods, random sampling, systematic sampling, and whole-group sampling, are compared; the accuracy after systematic sampling is found to increase as the training sample grows larger, but random sampling and whole-group sampling do not. We speculate that the increased randomness of random sampling loses some of the key information, making it more difficult to fit the model to the true distribution, and whole-group sampling can only observe the local distribution. Whole-group sampling can observe only local traffic patterns, which has certain limitations. The final accuracy of the three sampling methods is shown in Figure 9.

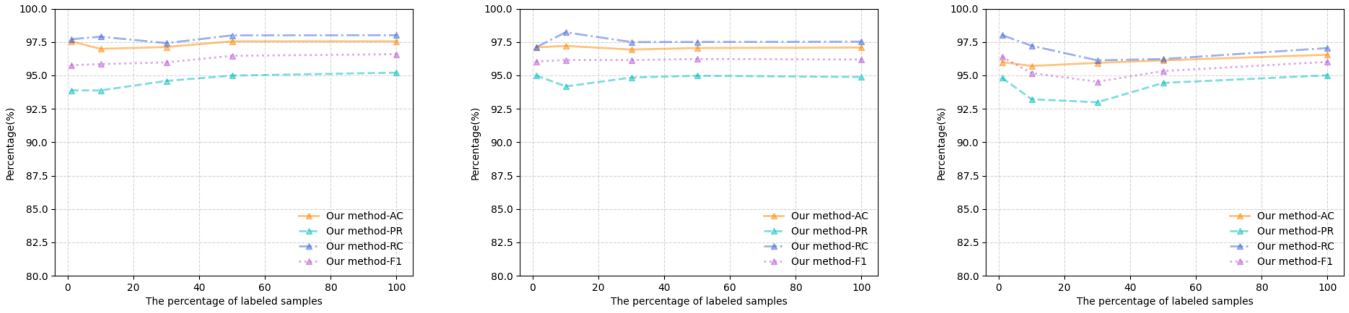


(**a**) D1 (**b**) D2 (**c**) D3

Figure 9. Classification accuracy corresponding to the three sampling methods.

As depicted in Figure 9, the final accuracy for the three datasets tends to stabilize around 97% as the number of training samples increases when utilizing systematic sampling. Among the datasets, D1 achieves the highest classification accuracy of 97.55%. The other two sampling methods do not learn the traffic pattern comprehensively when the training set is small, resulting in low classification accuracy. However, with the increase in the training set, the accuracy of D1 and D3 increased more obviously, and the increase in D2 was less significant because the distribution of key information in D2 was more concentrated. In summary, our model can determine the use of systematic sampling.

Figure 10 displays the results of experiments that were conducted on three datasets using systematic sampling and altering the number of labeled samples. The four evaluation metrics (AC, PR, RC, and F1) are shown to be stable at high levels regardless of the percentage of labeled samples, which suggests that our technique has some degree of generalizability. Specifically, at a labeled sample proportion of 1%, D1 and D3 demonstrate the highest classification accuracy, while at a labeled sample proportion of 5%, D2 exhibits the highest classification accuracy.



(**a**) D1 (**b**) D2 (**c**) D3

Figure 10. Trends of AC, PR, RC, and F1 with increasing proportion of marker samples.

4.3. Comparison

For the identical dataset, we compare our method with the fully supervised CNN (sCNN) for malicious traffic classification. In scenarios with a low proportion of labeled

samples, our method achieves high levels of accuracy (AC), precision (PR), recall (RC), and F1 score. As the proportion of labeled samples continues to increase, the accuracy of the fully supervised CNN approaches that of our method. Figure 11 illustrates the trend of the four evaluation metrics as the proportion of labeled samples increases.

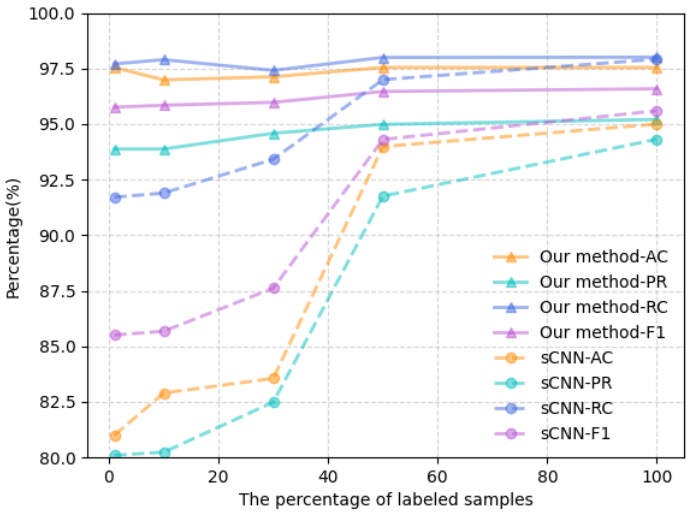


Figure 11. Trends of AC, PR, RC, and F1 with increasing proportion of marker samples.

From Figure 11, our method produces the best classification when the proportion of labeled data is between 1% and 5%, but the fully supervised CNN cannot reach the classification requirement at this time, knowing that the classification accuracy of the fully supervised CNN can be maintained at approximately 97% when the proportion of labeled data is above 50%, which illustrates the necessity of using unlabeled data for pre-training to learn traffic patterns. Additionally, our approach achieves the desired goal that malicious traffic can be classified even using a small number of labeled datasets.

At the same time, we compare our method with four other recent industrial IoT malicious traffic classification methods, which are KTDA-ConvLaddernet Ning et al. [29], IAPF Niu et al. [13], TSCRNN Lin et al. [18], and method of Zhao et al. [21]. These methods are introduced in related work.

The AC, PR, RC, and F1 pairs of the above four methods are shown in Table 5, and the comparison of total delay and total privacy level is shown in Figure 12.

Table 5. Comparison between our method and other methods.

Algorithms	AC	PR	RC	F1
Our Method	97.55%	95.21%	98.01%	96.59%
KTDA-ConvLaddernet [29]	97.19%	95.13%	95.99%	95.56%
IAPF [13]	97.04%	94.23%	97.71%	95.94%
TSCRNN [18]	94.15%	90.77%	95.01%	92.84%
Method of [21]	90.28%	88.47%	85.98%	87.21%

The following expands the comparison between the above four recent malicious traffic classification methods of the industrial Internet of Things and our methods:

KTDA-ConvLaddernet is similar to our method in the use of datasets, and it is also a method to train the model using a small proportion of labeled data, so we also compare the accuracy of the two methods when the labeled dataset accounts for 1–5% of the total dataset. As a result, the accuracy of KTDA-ConvLaddernet is 0.2–0.7% lower than that of our method. Thus, our method has a better classification effect. Simultaneously, compared with the total delay of model establishment and privacy level, the total delay of KTDA-ConvLaddernet is approximately 2.4 times that of our method, and it needs long-distance data transmission when using cloud computing, which has a great risk of privacy leakage.

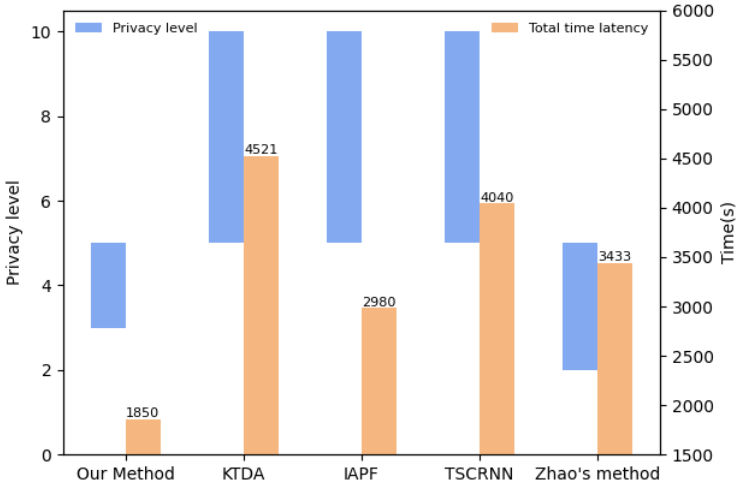


Figure 12. Comparison between our method and other methods.

IARF uses the random forest model in machine learning to improve the output. It has a good classification effect when the dataset is small, and the total delay in establishing the model is approximately 1.6 times that of our method. However, the model needs a large amount of labeled data, and features need to be extracted manually. At the same time, like KTDA-ConvLaddernet, it has a great risk of privacy leakage. TSCERNN's performance on the UNSW-NB15 dataset is not as good as the first two methods, and its accuracy is 3.4% different from our method. The total delay is approximately 2.2 times that of our method, and the risk of privacy leakage is also great. The method of Zhao et al. [21] introduces edge intelligence to accelerate the model and to reduce the risk of privacy leakage simultaneously, but it does not consider the edge server and other cooperation, which is approximately 1.8 times our method in total delay and 7.25% different from our accuracy method. Thus, it cannot classify malicious traffic in the UNSW-NB15 dataset well.

5. Conclusions

Aiming to classify malicious traffic within a mixture of encrypted traffic data from the Industrial Internet of Things, we propose a semi-supervised deep-learning method that achieves more accurate classification. The method achieves a classification accuracy of 97.55%, precision of 95.21%, recall rate of 98.01%, and F1 rate of 96.59%. Compared to fully supervised classification methods, our approach improves accuracy by 2.55% while using a lower proportion of labeled data, aligning better with the characteristics of real-world IIoT traffic data.

During model optimization, we propose a cloud-side collaboration scheme considering factors such as computing delay, transmission delay, and privacy protection. Through comparison with six different training scenarios, we calculate the total delay and privacy level associated with model training and classification. The results indicate that the introduction of edge intelligence reduces the total delay and the risk of privacy leakage.

Moving forward, we plan to further optimize our proposed model. This includes more accurately determining the proportion of offloading to local-edge-cloud, as well as incorporating the energy consumption of all three parties in the comprehensive performance index. Additionally, we will continue to investigate methods for implementing active defense to protect the security of IIoT after classifying malicious traffic.

Author Contributions: Conceptualization, M.W., X.Z., K.W. and X.M.; Methodology, B.Z.; Validation, B.Z.; Formal analysis, M.W.; Investigation, B.Z.; Writing—original draft, B.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Shandong Provincial Natural Science Foundation of China under Grant (No. ZR202111260301) and the Shandong Province Agricultural Major Application Technology Innovation Project of China under Grant (No. SD2019NJ007).

Data Availability Statement: Data will be made available on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

IIoT	Industrial Internet of Things
EI	Edge Intelligence
CNN/CNNs	Convolutional Neural Network
LSTM	Long Short-Term Memory
DDoS	Distributed Denial of Service
CPU	Central Processing Unit
GPU	Graphics Processing Unit
D1	UNSW-NB15 dataset
D2	CTU-13 dataset
D3	BoT-IoT dataset
$\alpha_L(t)$	Ratio of tasks offloaded to local
$\alpha_E(t)$	Ratio of tasks offloaded to edge servers
$\alpha_C(t)$	Ratio of tasks offloaded to cloud servers

References

1. Fu, L.; Zhang, W.; Tan, X.; Zhu, H. An algorithm for detection of traffic attribute exceptions based on cluster algorithm in industrial internet of things. *IEEE Access* **2021**, *9*, 53370–53378. [CrossRef]
2. Liu, Y.; Tong, K.; Mao, F.; Yang, J. Research on digital production technology for traditional manufacturing enterprises based on industrial Internet of Things in 5G era. *Int. J. Adv. Manuf. Technol.* **2020**, *107*, 1101–1114. [CrossRef]
3. Rezaei, S.; Liu, X. Deep learning for encrypted traffic classification: An overview. *IEEE Commun. Mag.* **2019**, *57*, 76–81. [CrossRef]
4. Cui, S.; Liu, J.; Dong, C.; Lu, Z.; Du, D. Only Header: A reliable encrypted traffic classification framework without privacy risk. *Soft Comput.* **2022**, *26*, 13391–13403. [CrossRef]
5. Wang, T.; Li, W.; Rong, H.; Yue, Z.; Zhou, J. Abnormal traffic detection-based on memory augmented generative adversarial IIoT-assisted network. *Wirel. Netw.* **2022**, *28*, 2579–2595. [CrossRef]
6. Chuanxia, S.; Han, Z.; Peixuan, Y. Machine learning and IoTs for forecasting prediction of smart road traffic flow. *Soft Comput.* **2023**, *27*, 323–335. [CrossRef]
7. Wang, P.; Chen, X.; Ye, F.; Sun, Z. A survey of techniques for mobile service encrypted traffic classification using deep learning. *IEEE Access* **2019**, *7*, 54024–54033. [CrossRef]
8. He, M.; Wang, X.; Zhou, J.; Xi, Y.; Jin, L.; Wang, X. Deep-feature-based autoencoder network for few-shot malicious traffic detection. *Secur. Commun. Netw.* **2021**, *2021*, 6659022. [CrossRef]
9. Jin, M.; Sun, C.; Hu, Y. An intelligent traffic detection approach for vehicles on highway using pattern recognition and deep learning. *Soft Comput.* **2022**, *27*, 5041–5052. [CrossRef]
10. Al-Qerem, A.; Alauthman, M.; Almomani, A.; Gupta, B.B. IoT transaction processing through cooperative concurrency control on fog–cloud computing environment. *Soft Comput.* **2020**, *24*, 5695–5711. [CrossRef]
11. Xie, F.; Xu, A.; Jiang, Y.; Chen, S.; Liao, R.; Wen, H. Edge intelligence based co-training of cnn. In Proceedings of the 2019 14th International Conference on Computer Science & Education (ICCSE), Toronto, ON, Canada, 19–21 August 2019; pp. 830–834.
12. Pekar, A.; Mocnej, J.; Seah, W.K.; Zolotova, I. Application domain-based overview of IoT network traffic characteristics. *ACM Comput. Surv. CSUR* **2020**, *53*, 1–33. [CrossRef]
13. Niu, Z.; Xue, J.; Qu, D.; Wang, Y.; Zheng, J.; Zhu, H. A novel approach based on adaptive online analysis of encrypted traffic for identifying Malware in IIoT. *Inf. Sci.* **2022**, *601*, 162–174. [CrossRef]
14. Ikram, S.T.; Priya, V.; Anbarasu, B.; Cheng, X.; Ghalib, M.R.; Shankar, A. Prediction of IIoT traffic using a modified whale optimization approach integrated with random forest classifier. *J. Supercomput.* **2022**, *78*, 10725–10756. [CrossRef]
15. Yan, X.; Xu, Y.; Xing, X.; Cui, B.; Guo, Z.; Guo, T. Trustworthy network anomaly detection based on an adaptive learning rate and momentum in IIoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6182–6192. [CrossRef]
16. Nagaraja, A.; Boregowda, U.; Khatatneh, K.; Vangipuram, R.; Nuvvusetty, R.; Kiran, V.S. Similarity based feature transformation for network anomaly detection. *IEEE Access* **2020**, *8*, 39184–39196. [CrossRef]
17. Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China, 22–24 July 2017; pp. 43–48.

18. Lin, K.; Xu, X.; Gao, H. TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of IIoT. *Comput. Netw.* **2021**, *190*, 107974. [CrossRef]
19. Zainudin, A.; Ahakonye, L.A.C.; Akter, R.; Kim, D.S.; Lee, J.M. An efficient hybrid-dnn for ddos detection and classification in software-defined iiot networks. *IEEE Internet Things J.* **2022**, *10*, 8491–8504. [CrossRef]
20. Shahin, M.; Chen, F.F.; Bouzary, H.; Hosseinzadeh, A.; Rashidifar, R. A novel fully convolutional neural network approach for detection and classification of attacks on industrial IoT devices in smart manufacturing systems. *Int. J. Adv. Manuf. Technol.* **2022**, *123*, 2017–2029. [CrossRef]
21. Zhao, Y.; Yang, Y.; Tian, B.; Yang, J.; Zhang, T.; Hu, N. Edge Intelligence Based Identification and Classification of Encrypted Traffic of Internet of Things. *IEEE Access* **2021**, *9*, 21895–21903. [CrossRef]
22. Zeb, S.; Rathore, M.A.; Mahmood, A.; Hassan, S.A.; Kim, J.; Gidlund, M. Edge intelligence in softwarized 6G: Deep learning-enabled network traffic predictions. In Proceedings of the 2021 IEEE Globecom Workshops (GC Wkshps), Madrid, Spain, 7–11 December 2021; pp. 1–6.
23. Mohammed, B.; Hamdan, M.; Bassi, J.S.; Jamil, H.A.; Khan, S.; Elhigazi, A.; Rawat, D.B.; Ismail, I.B.; Marsono, M.N. Edge computing intelligence using robust feature selection for network traffic classification in internet-of-things. *IEEE Access* **2020**, *8*, 224059–224070. [CrossRef]
24. Qi, H.; Wang, J.; Li, W.; Wang, Y.; Qiu, T. A blockchain-driven IIoT traffic classification service for edge computing. *IEEE Internet Things J.* **2020**, *8*, 2124–2134. [CrossRef]
25. Latif, S.; Zou, Z.; Idrees, Z.; Ahmad, J. A novel attack detection scheme for the industrial internet of things using a lightweight random neural network. *IEEE Access* **2020**, *8*, 89337–89350. [CrossRef]
26. Shapira, T.; Shavitt, Y. Flowpic: Encrypted internet traffic classification is as easy as image recognition. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April–2 May 2019; pp. 680–687.
27. de Elias, E.M.; Carriel, V.S.; De Oliveira, G.W.; Dos Santos, A.L.; Nogueira, M.; Junior, R.H.; Batista, D.M. A Hybrid CNN-LSTM Model for IIoT Edge Privacy-Aware Intrusion Detection. In Proceedings of the 2022 IEEE Latin-American Conference on Communications (LATINCOM), Rio de Janeiro, Brazil, 30 November–2 December 2022; pp. 1–6.
28. Hu, X.; Ning, J.; Yin, J.; Yang, J.; Adebisi, B.; Gacanin, H. Efficient Malicious Traffic Classification Methods based on Semi-supervised Learning. In Proceedings of the 2022 9th International Conference on Dependable Systems and Their Applications (DSA), Wulumuqi, China, 4–5 August 2022; pp. 230–235.
29. Ning, J.; Gui, G.; Wang, Y.; Yang, J.; Adebisi, B.; Ci, S.; Gacanin, H.; Adachi, F. Malware traffic classification using domain adaptation and ladder network for secure industrial internet of things. *IEEE Internet Things J.* **2021**, *9*, 17058–17069. [CrossRef]
30. Lyu, P.; Zhang, C.; Liu, S.; Qiao, M.; Xu, Y.; Wu, L.; Yao, K.; Han, J.; Ding, E.; Wang, J. Maskocr: Text recognition with masked encoder-decoder pretraining. *arXiv* **2022**, arXiv:2206.00311.
31. Gong, Q.; Wang, P.; Cheng, Z. An encoder-decoder model based on deep learning for state of health estimation of lithium-ion battery. *J. Energy Storage* **2022**, *46*, 103804. [CrossRef]
32. Zhou, Q.; Wang, Q.; Bao, Y.; Kong, L.; Jin, X.; Ou, W. LAEDNet: A lightweight attention encoder–decoder network for ultrasound medical image segmentation. *Comput. Electr. Eng.* **2022**, *99*, 107777. [CrossRef]
33. Li, Y.; Guo, H.; Hou, J.; Zhang, Z.; Jiang, T.; Liu, Z. A Survey of Encrypted Malicious Traffic Detection. In Proceedings of the 2021 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), Beijing, China, 15–17 October 2021; pp. 1–7.
34. Zhang, W.; Wang, N.; Li, L.; Wei, T. Joint compressing and partitioning of CNNs for fast edge-cloud collaborative intelligence for IoT. *J. Syst. Archit.* **2022**, *125*, 102461. [CrossRef]
35. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proc. IEEE* **2019**, *107*, 1738–1762. [CrossRef]
36. Zhang, G.; Ni, S.; Zhao, P. Learning-based joint optimization of energy delay and privacy in multiple-user edge-cloud collaboration MEC systems. *IEEE Internet Things J.* **2021**, *9*, 1491–1502. [CrossRef]
37. Yu, Y.; Zhang, J.; Letaief, K.B. Joint subcarrier and CPU time allocation for mobile edge computing. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6.
38. Liu, D.; Kong, H.; Luo, X.; Liu, W.; Subramaniam, R. Bringing AI to edge: From deep learning's perspective. *Neurocomputing* **2022**, *485*, 297–320. [CrossRef]
39. Gong, Y.; Yao, H.; Wang, J.; Li, M.; Guo, S. Edge intelligence-driven joint offloading and resource allocation for future 6G industrial internet of things. *IEEE Trans. Netw. Sci. Eng.* **2022**. [CrossRef]
40. Sun, Y.; Li, N.; Tao, X. Privacy preserved secure offloading in the multi-access edge computing network. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Nanjing, China, 29 March 2021; pp. 1–6.
41. Xiao, H.; Xu, C.; Ma, Y.; Yang, S.; Zhong, L.; Muntean, G.M. Edge intelligence: A computational task offloading scheme for dependent IoT application. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 7222–7237. [CrossRef]
42. Van Huynh, D.; Khosravirad, S.R.; Masaracchia, A.; Dobre, O.A.; Duong, T.Q. Edge intelligence-based ultra-reliable and low-latency communications for digital twin-enabled metaverse. *IEEE Wirel. Commun. Lett.* **2022**, *11*, 1733–1737. [CrossRef]

43. Yang, Z.; Liu, X.; Li, T.; Wu, D.; Wang, J.; Zhao, Y.; Han, H. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Comput. Secur.* **2022**, *116*, 102675. [CrossRef]
44. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]