electronics

MDPI

Article

Securing a Smart Home with a Transformer-Based IoT Intrusion Detection System

Minxiao Wang [1], Ning Yang [2] and Ning Weng [1,*]

[1] The Computer Engineering Program in the School of Electrical, Computer, and Biomedical Engineering, Southern Illinois University, Carbondale, IL 62901, USA; minxiao.wang@siu.edu
[2] The Information Technology Program in the School of Computing, Southern Illinois University, Carbondale, IL 62901, USA; nyang@siu.edu
* Correspondence: nweng@siu.edu

Abstract: Machine learning (ML)-based Network Intrusion Detection Systems (NIDSs) can classify each network's flow behavior as benign or malicious by detecting heterogeneous features, including both categorical and numerical features. However, the present ML-based NIDSs are deemed insufficient in terms of their ability to generalize, particularly in changing network environments such as the Internet of Things (IoT)-based smart home. Although IoT devices add so much to home comforts, they also introduce potential risks and vulnerabilities. Recently, many NIDS studies on other IoT scenarios, such as the Internet of Vehicles (IoV) and smart cities, focus on utilizing the telemetry data of IoT devices for IoT intrusion detection. Because when IoT devices are under attack, their abnormal telemetry data values can reflect the anomaly state of those devices. Those telemetry data-based IoT NIDS methods detect intrusion events from a different view, focusing on the attack impact, from the traditional network traffic-based NIDS, which focuses on analyzing attack behavior. The telemetry data-based NIDS is more suitable for IoT devices without built-in security mechanisms. Considering the smart home IoT scenario, which has a smaller scope and a limited number of IoT devices compared to other IoT scenarios, both NIDS views can work independently. This motivated us to propose a novel ML-based NIDS to combine the network traffic-based and telemetry data-based NIDS together. In this paper, we propose a Transformer-based IoT NIDS method to learn the behaviors and effects of attacks from different types of data that are generated in the heterogeneous IoT environment. The proposed method utilizes a self-attention mechanism to learn contextual embeddings for input network features. Based on the contextual embeddings, our method can solve the feature set challenge, including both continuous and categorical features. Our method is the first to utilize both network traffic data and IoT sensors' telemetry data at the same time for intrusion detection. Experiments reveal the effectiveness of our method on a realistic network traffic intrusion detection dataset named ToN_IoT, with an accuracy of 97.95% for binary classification and 95.78% for multiple classifications on pure network data. With the extra IoT information, the performance of our method has been improved to 98.39% and 97.06%, respectively. A comparative study with existing works shows that our method can achieve state-of-the-art performance on the ToN_IoT dataset.

Keywords: network intrusion detection systems; deep learning; network security

Citation: Wang, M.; Yang, N.; Weng, N. Securing a Smart Home with a Transformer-Based IoT Intrusion Detection System. *Electronics* **2023**, *12*, 2100. https://doi.org/10.3390/electronics12092100

Academic Editor: Harald Vranken

Received: 11 February 2023
Revised: 3 April 2023
Accepted: 1 May 2023
Published: 4 May 2023

1. Introduction

There has been a growing trend toward smart homes in recent years, driven by the increasing diversity and availability of heterogeneous devices and smart objects [1]. As technology improves, artificial intelligence (AI) and machine learning (ML) are integrated into smart homes to enhance functionality, such as home automation systems [2], energy management [3], and home security [4].

Internet of Things (IoT) is a network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, and network connectivity,

which enables these objects to collect, exchange, and analyze data [5]. The wide-ranging IoT applications have been adopted and deployed in smart homes in the last few years to increase the ability to monitor and control devices remotely, implemente the automation of various household tasks and systems, and provide the residents with a more comfortable, convenient, energy-efficient, and secure living experience [6,7]. The possibilities are endless as new devices and technologies continue to emerge.

However, many IoT devices have limited or no built-in security features and are vulnerable to cyberattacks. Some IoT devices can generate, collect and transmit sensitive data, making them a prime target for cybercriminals. Due to the diversity of devices and technologies involved in IoT, it can be challenging to ensure consistent and effective security across the entire system. Some of the security challenges specific to the heterogeneous nature of IoT include different security standards, incompatibility between devices, vulnerability to attack, and complexity in management. Currently, some existing ML-based IoT IDS approaches [8,9] monitor the IoT telemetry data (such as the GPS location values, temperature, and atmospheric pressure) to detect attacks against IoT devices. Other ML-based IoT IDS studies [10,11] follow the traditional NIDSs way by considering the network traffic only. It is important to note that these two types of NIDS methods are fundamentally different. The network traffic-based NIDS detects patterns in network traffic to determine whether it constitutes an attack. In contrast, the telemetry data-based NIDS monitors the status of potential victim devices by analyzing their telemetry data as input. While the telemetry data is transmitted in the form of network traffic, the telemetry data-based NIDS does not analyze the behavior of the traffic. Instead, it focuses on the content of the traffic, specifically the current telemetry values. The telemetry data-based NIDS determines whether the devices are under attack, even if the telemetry traffic itself is not necessarily malicious. For instance, as a result of an attack, a temperature sensor may show a temperature that is outside the expected range. In summary, the network traffic-based NIDS focuses on the behavior of attack traffic, whereas the telemetry data-based NIDS focuses on the impact of attack traffic on the device's telemetry values.

In this work, we aim to combine network traffic-based NIDS and telemetry data-based NIDS together. Since each type of those two NIDS methods only has a single view of the smart home network, it is hard to learn generalized insight knowledge based on the limited view. Therefore, we propose a transformer-based IoT IDS, which detects each flow by taking both the flow's statistical features and the current IoTs' telemetry values as input. It is important to note that a smart home network can vary depending on the setup. Figure 1 shows an example of a smart home network and illustrates how our NIDS design can be integrated into a smart home network.

As Figure 1 shows, we set up a smart home gateway to provide a connection between the home intranet and the internet. The smart home gateway consists of a NIDS, which is responsible for securing the whole home intranet, and a middleware server, which runs all IoT services and provides interfaces for users to access IoT services. The IoT services indicate that home members and authorized users can connect their smart devices to the home intranet and control home IoT devices, such as making some coffee, controlling the gate, or checking the temperature and air quality. In the intranet of the smart home, the IoT devices publish their telemetry data to the middleware server through the MQTT protocol (a lightweight, publish-subscribe, machine-to-machine network protocol for IoT message queue/message queuing service), and users can subscribe to those data (such as temperature and moisture) on other devices, such as smartphones or PCs, through the HTTP protocol (an application-layer protocol for web services and other purposes). Nowadays, various cyber-security incidents (i.e., DDoS and ransomware, XSS—Cross-Site Scripting, backdoor, and injection) are launched against different IoT sensors [12].

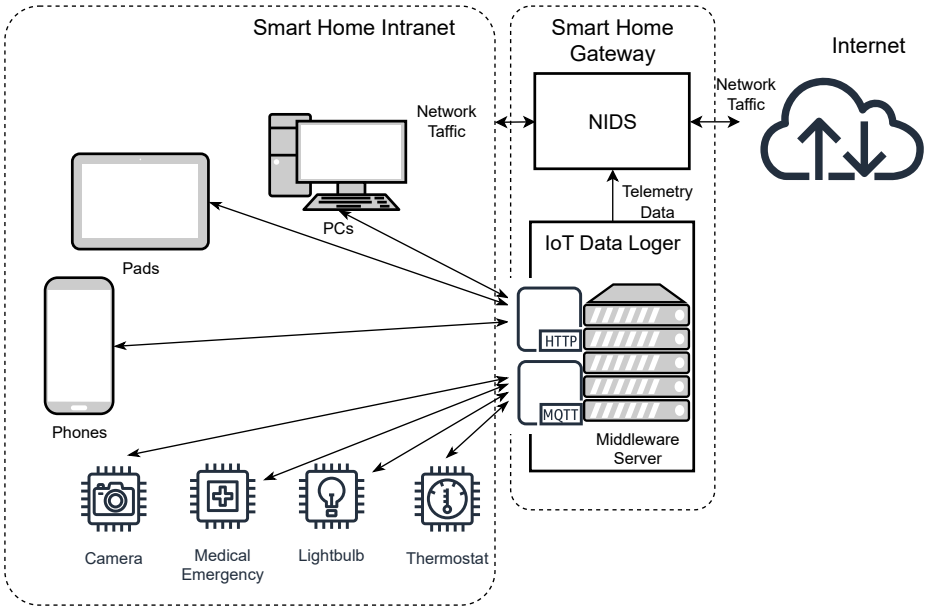

Figure 1. An example smart home scenario. The smart home gateway consists of the functions of both NIDS and the middleware server of IoTs. The smart home intranet block represents the communications of IoT services publish/subscribe through MQTT and HTTP protocols, respectively. The NIDS monitors all network traffic and all IoT devices' telemetry values, which are obtained from the middleware.

NIDS is a beneficial tool for monitoring and securing IoT networks. It can analyze multiple types of data from various sources, such as network packets, system logs, and application data [13]. The use of deep learning for intrusion detection in smart homes and IoT has been a growing research area, as deep learning algorithms can help detect and prevent malicious activities on these devices. However, traditional DL-based NIDS faces some challenges in using the heterogeneous feature tabular data and working with IoT's mixed data.

First, it has been well known that the different scales and value types of heterogeneous features challenge the DL-based models [14]. Heterogeneity is an original property of features in network traffic data for flow-based NIDS. For instance, KDD-99 [15], UNSW data [16], and CICIDS data [17] provide tabular data consisting of different types of features to represent network traffic flow. Those features include both categorical features (such as "Protocols", "Service", and "Flags") and numerical features (such as "Source bytes", "Average Transport Speed", and "Flow duration").

Second, current NIDSs may face compatibility issues in integrating and analyzing data from various data sources in IoT scenarios. Booij et al. [18] demonstrated the relevance of data set heterogeneity for effective intrusion detection in IoT environments. They proposed a heterogeneous IoT network architecture, which consists of three components: "Edge Layer", "Fog Layer", and "Cloud Layer". For the IoT NIDS task, the heterogeneity of IoT brings mixed data sources to NIDS models. Moreover, traditional network data collected in the smart home gateway, and distributed IoT devices will also generate telemetry data, which will be recorded in the IoT middleware server. If IoT devices are being targeted, their telemetry sensor values can indicate an anomaly trend that can be employed for detecting intrusions. For instance, if a smart fridge is being attacked, the temperature sensor readings inside it may display an unusual pattern, either too high or too low, signaling an intrusion. In order to detect IoT attacks, smart home NIDS should monitor all traffic flows and telemetry data, but the network data are collected in terms of traffic flow, and the telemetry data of IoT systems is recorded in terms of time. The NIDS operates on a per-flow basis, while telemetry data are recorded based on time. As a result, longer network flows can have several telemetry data records, whereas shorter ones may not have any, leading to

what is known as a missing sensor record. Therefore, the mismatch of mixed data sources will result in serious partial data missing in the combination of multiple data sources. It is important for ML that the learning model receives a fixed-format input for each detection. In our method, both network flow data and IoT telemetry data are included as parts of the input. Thus, if there is a mismatch between the two, it can result in incorrect input for the ML model.

In order to address the mentioned challenges of heterogeneous features and mixed data sources, we propose a Transformer-based IoT NIDS method. The overview of the proposed method workflow is shown in Figure 2. The multi-head self-attention mechanism [19], which is the core component of Transformer, can learn adaptive embedding for both categorical and numerical features. Compared with traditional ML/DL methods, our method can match the state-of-the-art performance on binary detection tasks. Further, we evaluate our method on the ToN_IoT dataset for multiple-class detection. The results show that our method can efficiently distinguish different types of malicious behaviors. Finally, we test our method on the combination of mixed data sources. We use the network data source and IoT telemetry data source to simulate smart home scenarios. Our method can achieve better performance on mixed data sources than using network data only, even if there is a large amount of "NaN" value in the IoT telemetry part of the combined data. The results prove that our method has good robustness on mixed data in the aspects of both feature heterogeneity and source heterogeneity.

We summarize our main contributions as follows:

- Proposing a Transformer-based IoT NIDS method. The method includes a data processing algorithm for combining IoT data with network traffic data, and an FT-Transformer-based model can adaptively learn from heterogeneous inputs.
- Evaluating our method on the ToN_IoT dataset. The evaluation consists of the binary and multiple classification performance of our method on network traffic data. Further, we evaluate our method with the combination of network and IoT telemetry data. To the best of our knowledge, this paper is the first to use the extra IoT data and network data of the ToN_IoT dataset at the same time.
- The results show that our method can match the performance of the state-of-the-art methods with 97.95% and 95.78% detection accuracy for binary and multiple classifications on pure network data. The additional results show that the extra IoT data can enhance classification performance to 98.39% (+0.44%) and 97.06% (+1.28%), respectively.

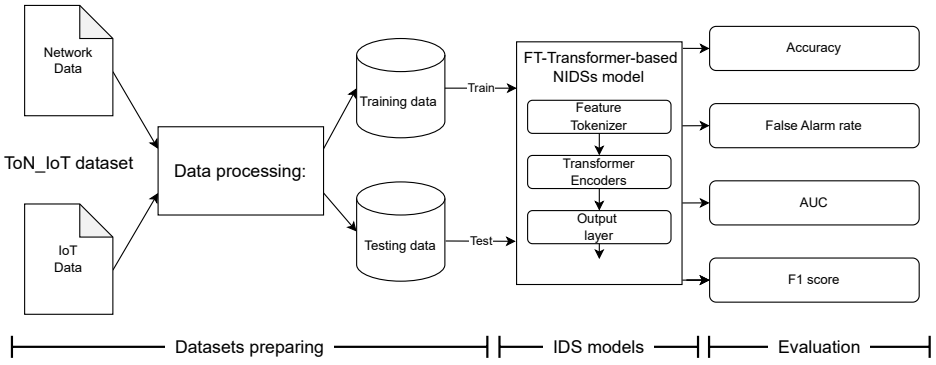

Figure 2. The overview of the proposed workflow for intrusion classification in an IoT network.

The remainder of this paper is organized as follows, background and related work are introduced in Section 2. Section 3 introduces the dataset, and the design of the corresponding data processing method. Section 4 describes the architecture of the transformer-based IoT Intrusion Detection system. The evaluation results and analysis are shown in Section 5. Finally, Section 6 presents our conclusion and future work.

2. Background and Related Work

2.1. IoT Security

The Internet of Things (IoT) has great potential to enable many beneficial applications in smart homes and has rapidly become more important in creating smart home environments. However, there are several characteristics of IoT devices that make them vulnerable to security threats. First, many IoT devices are small and have limited processing power, memory, and storage. This makes it difficult to install and run security software and to properly secure data stored on the device. Generally, IoT devices rely on software updates to fix security vulnerabilities, but many devices do not receive regular updates or are not updated at all, leaving them vulnerable to attack. Second, since IoT devices communicate with each other and with cloud services, there are multiple entry points created, which are the target for attackers. IoT devices often rely on weak authentication mechanisms, such as passwords, making it easier for attackers to gain access to them. Furthermore, IoT devices collect, transmit, and store large amounts of sensitive data, such as personal information, over the network. These data can be vulnerable to interception and theft if the transmission is not properly secured. Typical types of security attacks that can target IoT devices include malware attacks, unauthorized access, and network-based attacks, such as man-in-the-middle attacks, denial-of-service attacks, and packet sniffing [20,21].

2.2. Intrusion Detection Systems

Intrusion detection has a history of more than 25 years, with one of the first workshops dedicated to it, RAID (Recent Advances in Intrusion Detection), held in 1998. NIDS for IoT is a software- or dedicated hardware-based system to identify potential security threats to a network, such as hacking attempts, malware infections, or other types of malicious activity. Machine learning algorithms have been used to enhance the performance of IoT NIDS by automatically learning what normal behavior looks like and detecting any deviations that may indicate an attack [22–24]. Traditional NIDS typically use signature-based detection methods, which detect known attack patterns by comparing incoming data with a predefined set of signatures for known attacks [25]. However, this approach may struggle with detecting unknown or newly evolved attacks. Another NIDS approach is anomaly detection, where the incoming data are compared to a baseline of normal behavior to identify deviations that may indicate an intrusion attempt. This approach may be limited by the quality of the baseline and the accuracy of the deviation detection algorithms [26].

Deep learning (DL) has gained significant attention in the field of intrusion detection systems due to its ability to handle large amounts of data and perform complex pattern recognition [27]. DL algorithms can provide better accuracy and improved performance compared to traditional machine learning techniques through learning and improving with more data. This has led to the development of several DL-based NIDS models, which have shown promising results in detecting various types of intrusions, including network attacks, insider threats, and malware. In 2010, Sommer and Paxson [28] summarized their experience of applying ML-based intrusion detection in practice, concluding that data nature is more important than the ML method used. In 2022, Arp et al. [29] gave a general analysis of the dos and do nots of machine learning in computer security at the USENIX Security Symposium. Advanced ML-based methods are used for intrusion detection in commercial IDS systems such as Darktrace [30], as reported. Ashiku et al. [31] used deep learning architectures to develop adaptive and resilient NIDS to detect and classify network attacks. Satam et al. [32] proposed a Wireless Intrusion Detection System (WIDS) with an anomaly behavior analysis for detecting Wi-Fi network attackers with high accuracy and low false alarms.

The heterogeneous nature of the IoT poses more specific challenges and significant security concerns that can compromise the privacy and safety of smart home residents. There are several research approaches that can be used to develop and improve NIDS solutions for heterogeneous data [13]. Mahadik et al. [33] proposed intelligent NIDS to identify and mitigate various DDoS attacks in the heterogeneous IoT infrastructure.

Bertoli et al. [34] described the stacked-unsupervised federated learning (FL) approach to generalize intrusion detection for heterogeneous networks. Almutairi et al. [35] designed intelligent NIDS to protect IoT devices by detecting attacks as close as possible to the corresponding data sources.

2.3. Transformer

In recent years, Transformer-based models have been used in intrusion detection systems to improve the accuracy and efficiency of threat detection. These approaches allow NIDS to learn from previous data and detect new and previously unseen threats, making them a powerful tool in the fight against cyber attacks. Transformer is a deep learning neural network architecture that has been widely used in various natural language processing (NLP) tasks such as language translation and text classification [36], network traffic generation, and classification [37], and it is also gaining popularity in image classification and computer vision [38] with the improvement in accuracy and generalization, and the ability in parallel processing and transfer learning. Wang et al. [39] combined efficient and scalable Transformers and a convolutional neural network (CNN) to detect distributed denial-of-service (DDoS) attacks on Software-Defined Networking (SDN). Wu et al. [40] proposed a robust Transformer-based, all-in-one intrusion detection solution for detecting abnormal activities and applying a self-attention mechanism to facilitate network traffic classifications.

As a typical neural network structure, the Transformer's core layers are self-attention layers, which can dynamically learn from the input context. The basic idea of the self-attention mechanism is to use the input sequence to compute three vectors for each element: the query vector, the key vector, and the value vector. These vectors are then used to compute a weighted sum of the value vectors, where the weights are computed by taking the dot product of the query vector with the key vector and applying a softmax function to the result. The resulting weighted sum is the output of the self-attention layer.

3. Dataset and Data Processing

In this section, we introduce the adopted ToN_IoT dataset and illustrate the data processing modules for pure network data and the combination of network and IoT data.

3.1. Dataset ToN_IoT

The ToN_IoT dataset is the new generation of Industry 4.0/Internet of Things (IoT) dataset [18], which is collected from a systematic testbed in a lab environment. The ToN_IoT includes mixed data sources, such as sensor data, network data, and log data, which are collected from the same realistic and large-scale network environment. Such heterogeneity property of the ToN_IoT dataset can reflect the characteristics of the IoT environment.

In this work, we utilize both network and IoT telemetry data in the ToN_IoT dataset to represent the smart home NIDS scenario. For the pure network traffic data, 10 types of network traffic are included: which are Normal Flow, Scanning Attack, Denial-of-Service (DoS) Attack, DDoS Attack, Ransomware Attack, Backdoor Attack, Injection Attack, Cross-Site Scripting (XSS) Attack, Password Attack, and Man-in-the-Middle (MITM) Attack. In order to classify IoT attacks, 43 features are extracted to illustrate each flow. According to the types of carried information, all features are divided into 6 sub-sets, which are connection activity features, statistical activity features, DNS activity features, SSL activity features, HTTP activity features, and violation activity features. The specific training and testing sets used in the paper are the officially published sub-set of ToN_IoT. This sub-set includes 300,000 normal flows and 20,000 flows for each class of attack, except the XSS attack, which only has 1043 flow records. For the purpose of training and testing the ML model, this sub-set is randomly split into two parts with the ratio 0.6/0.4.

The ToN_IoT dataset also provides IoT sensor data, which is sent from IoT devices, to study the relevance of dataset heterogeneity for effective intrusion detection in IoT [18]. The IoT telemetry data [12] are generated in the same IoT environment by IoT devices, such

as Fridges, GPS Trackers, Motion Light, Garage Door, Modbus, Thermostat, and Weather Monitor. To combine the network and IoT telemetry data, our research involves examining all the data collected by IoT sensors in order to identify data records that are temporally relevant to the network data records being used. The details of the matching method will be described in Algorithm 1.

3.2. Data Processing

In order to prepare the input data to be more suitable for our Transformer-based NIDS model, we design a different data processing module from most NIDS works. Generally, data processing for NIDS includes four steps: data cleaning, data normalization, feature selection, and dataset splitting. In this work, we use different data cleaning methods and different separated data normalizations. Further, we use all of the features instead of only selecting the important ones. Compared to traditional NIDS modules, the differences in our data processing modules are as follows:

- The first difference is that we do not clean data by dropping the invalid data samples. In the ToN_IoT dataset, there is a lot of missing data for some functional sub-sets of features, such as "DNS activity features", "SSL activity features", and "HTTP activity features". Those missing data points are filled by the string "-". Instead of dropping those data records, we transform all invalid values of categorical features into the string "-" and all invalid values of numerical features into the value "-1".
- The second difference is that we separately embed numerical features and categorical features. In many traditional DL-based NIDS methods, the categorical features are transformed into numerical values first; then all features are normalized in the same way. In our method, we encode each categorical feature with a value between 0 and U, where U is the number of unique contents for this categorical feature. The values are further used to calculate embedding vectors for categorical features (see Equation (2)). Then we only normalize numerical features.
- The third difference is that we use all the features instead of only using the more important features.

We make the above changes according to the embedding learning ability of our FT-Transformer-based model. The FT-Transformer model includes a Feature Tokenizer (FT) module for embedding the input features into a vector representation. Therefore, we can make the first difference since those invalid values also can be encoded into representation vectors for FT-Transformer. In the FT module, the embedding process for numerical and categorical features is separated. Therefore, we can make the second difference. Further, the transformer-based method is built upon a self-attention mechanism, which can generate adaptive attention for all features. In other words, the self-attention mechanism adaptively learns from more important features. Hence, it is reasonable to use all features in our method.

Combination of Network data and IoT data. To integrate the network traffic-based NIDS and telemetry data-based NIDS, we need to merge the two types of input data. Our NIDS monitors all network traffic and current telemetry data values of IoT devices. For each traffic flow, the NIDS analyzes its behavior. Simultaneously, the telemetry data values of IoT devices are collected and utilized to aid in traffic behavior analysis. Consequently, our NIDS can detect IoT attacks based on both the attack behavior and its impact.

For collecting telemetry data from different IoT devices, it is desirable for NIDS to monitor all telemetry data from all IoT devices at all available time instances for optimal detection accuracy. In order to detect any attacks on IoT devices, the telemetry data values from different IoT devices within the same time instance should be combined into a mixed telemetry vector. This mixed telemetry vector can aid in network traffic analysis by the NIDS. The selected telemetry data should be within the same time window as the traffic-detecting duration, which is typically a very short period. However, to create the mixed telemetry vector quickly, we use a random selection method to choose telemetry data

values within the time period for each IoT device. Because this period is very short, we do not mix flows from different devices in a random way.

In the ToN_IoT dataset, the two originally provided data sources cannot be simply combined. After analyzing the ToN_IoT dataset, we found that the network data records do not match well with IoT data records. The reason is that the network and IoT data are asynchronous. Specifically, network data samples are recorded in terms of traffic flows, which have a duration, but IoT sensor data samples are instantaneously recorded with time stamps. Therefore, we provide a matching algorithm to look for corresponding IoT records for network data. In this work, we use provided training and testing sets of network data the same as [18]. Although the ToN_IoT dataset also provides a training and testing set of IoT data, we use the entire IoT data to match the network records as much as possible. The matching algorithm is shown as Algorithm 1.

Algorithm 1 Matching the network and IoT data

Require: $\{traffic\_flow\}$, $\{IoT\_data_i\}$
  for $n \leftarrow 1$ to $length(traffic\_flow)$ **do**
    $cur\_ts\_1 \leftarrow traffic\_flow^n[ts]$
    $cur\_ts\_2 \leftarrow traffic\_flow^n[ts] + traffic\_flow^n[duration]$
    for $m \in \{Fridge, Weather, GPS, Motionlight, Garagedoor, Modbus, Thermostat\}$ **do**
      $selected\_set_m \leftarrow [\,]$
      while $cur\_ts\_1 \leq IoT\_data_m^j[ts] \leq cur\_ts\_2$ **do**
        add $IoT\_data_m^j$ to $selected\_set_m$
      end while
      add $random(selected\_set_m)$ to $traffic\_flow^n$
      if none of IoT_data_m matched,
        add invalid values '-' or '-1' to $traffic\_flow^n$
    end for
  end for

In Algorithm 1, we first assign the start and end time stamps of a traffic flow as cur_ts_1 and cur_ts_2. Then we search each type of IoT device's data to look for all records whose time stamps are in the range between cur_ts_1 and cur_ts_2. Then we randomly choose one for this traffic flow record. The reason is that if our NIDS needs to collect current telemetry data reflecting the current state of IoT devices (which could be either attacks or normal traffic), the telemetry data may be updated multiple times throughout the duration of the traffic flow. We chose random sampling because, during the traffic flow duration, the telemetry data likely will not change significantly. Additionally, the telemetry data received from different IoT devices inherently have different time instances, making it acceptable to select them randomly within a short time period. While there are other ways to sample telemetry data, random selection has little effect on NIDS detection accuracy, and speed is crucial for a quick response. Therefore, we made a trade-off by utilizing random sampling. If there is no IoT record in that duration, we fill in invalid values "-" or "-1" for the empty categorical and numerical features, respectively.

4. FT-Transformer-Based NIDS Model

In order to solve the heterogeneity problem in IoT NIDS, we design an FT-Transformer-based NIDS model [41]. The unique designs of the FT-Transformer make it extremely suitable for heterogeneous data in aspects of both different feature types and different data sources. The overview of our model is illustrated in Figure 3. The matched IoT data are concatenated with the original network data. Similar to the network data, all columns of IoT telemetry data are split into numerical and categorical parts, which are fed into the model separately. As the model is named, FT-Transformer generally includes a Feature Tokenizer part and a Transformer part followed by an MLP layer for final classification.

Feature Tokenizer. The Feature Tokenizer module is responsible for transforming the input feature values into learnable embeddings. Specifically, the Feature Tokenizer module takes categorical features and numerical features as two parts of input (x_i^{num} and x_i^{cat}) separately. Each part can include network data only or a combination of network and IoT data. The categorical features and numerical features are embedded in different ways, which are explained as Equations (1) and (2) in detail, where W_i^{num} and b_i^{num} are the weights of the numerical embedding layer and bias, and $embed()$ and b_i^{cat} are the categorical embedding function and bias.

$$E_i^{num} = x_i^{num} \cdot W_i^{num} + b_i^{num} \tag{1}$$

$$E_i^{cat} = embed(x_i^{cat}) + b_i^{cat} \tag{2}$$

Besides the categorical and numerical embeddings, a learnable classification embedding E^{cls}, which has the same tensor shape as each feature embedding, is created and combined with all feature embeddings. The classification embedding E^{cls} is used for participating in the further learning processing and extracting valued features in Transformer. Therefore, the output embeddings are a normalized concatenation of categorical features, numerical features, and classification embeddings, which is described in Equation (3).

$$E_{out} = Normalize(concat([\{E_i^{num}\}, \{E_i^{cat}\}, E^{cls}])) \tag{3}$$

Transformer blocks. As shown in Figure 3, the output of Feature Tokenizer, E_{out}, is fed into multiple Transformer encoders, which is a stack of $N = 6$ in this work. Each block [19] has a multi-head attention layer and a fully connected feed-forward network. Each of them is followed by the residual link and normalization layer. In particular, more hyperparameters we adopted in this work are 8 heads of multi-head attention layers, 32 embedding dimensions, a 0.1 post-attention dropout ratio, and a 0.1 feed-forward dropout ratio.

The core component of the Transformer encoder is the multi-head attention (MHA) layer. MHA allows the model to adaptively learn information from different features' embedding representations. MHA consists of multiple (M = 8 in this work) heads of self-attention, which are also named "scaled dot-product attention" [19].

$$MHA(E_{out}) = concat([\left\{ Attention(E_{out}^{head_i}) \right\}]) \tag{4}$$

As shown in Figure 3, the output of Feature Tokenizer, E_{out}, is fed into multiple Transformer encoders, which is a stack of $N = 6$ in this work. Each block [19] has a multi-head attention layer and a fully connected feed-forward network; each of them is followed by the residual link and normalization layer. Equation (5) illustrates the details of calculating the attention. Based on the input embedding $E_{out}^{head_i}$, three learned vectors are calculated: the query (Q) vector, the key (K) vector, and the value (V) vector. Weighted sums of V vectors are computed using them by taking the dot product of the Q vector with the K vector and then applying the softmax function to the result to obtain the weights.

$$Attention(E_{out}^{head_i}) = softmax(\frac{Q(E_{out}^{head_i})K^T(E_{out}^{head_i})}{\sqrt{d}})V(E_{out}^{head_i}) \tag{5}$$

Classification. The output of Transformer encoders is the learned embeddings for all input features, but we do not classify all of them. Instead, we only feed the learned $E^{cls\prime}$ to the final MLP layers because the input E^{cls} has taken advantage of the attention mechanism to extract information from all other feature embeddings. As an independent third-party embedding (not numerical or categorical feature embeddings), it is more suitable for final classification.

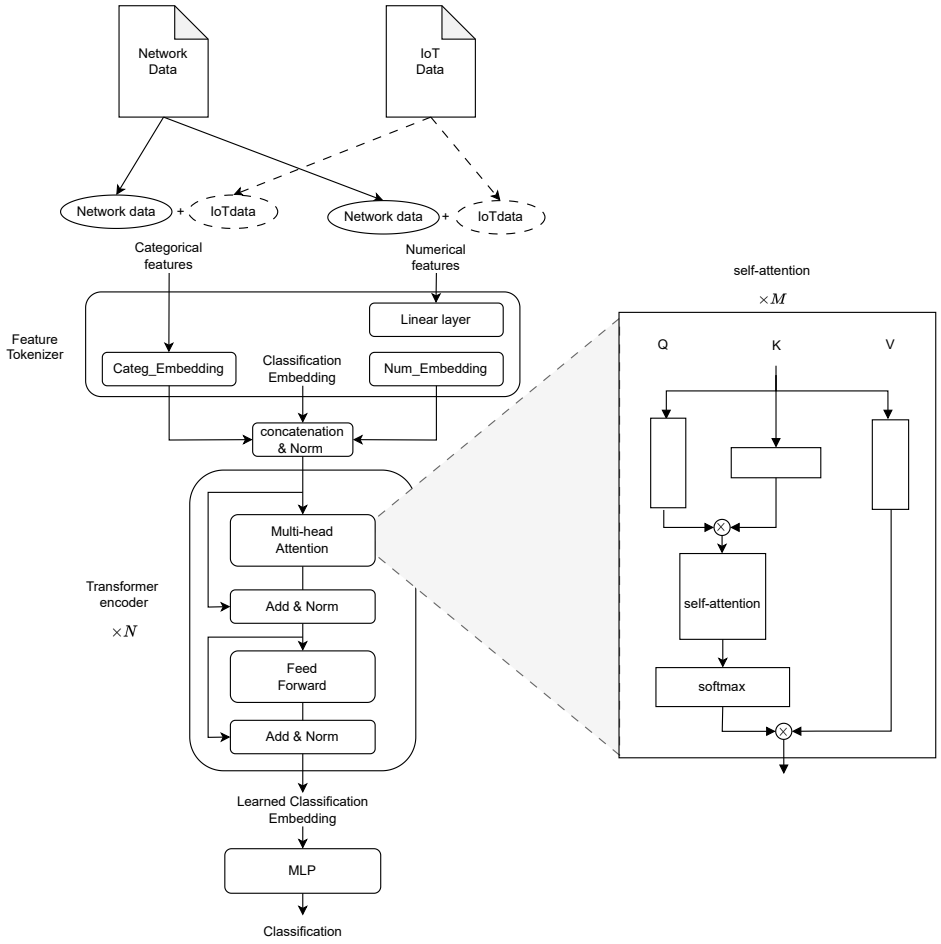

Figure 3. The overview of the proposed FT-Transformer-based NIDS model. The model consists of a Feature Tokenizer, N Transformer encoders, and an MLP layer for classification. The core module of the Transformer is the multiple head attention layers, which are zoomed in.

5. Results and Discussions

In this section, we first introduce the experimental environment, then we discuss the experiments and results to show why our method is suitable for IoT NIDS. In Section 5.2, we evaluate our method on the network data of the ToN_IoT dataset for binary and multi-classes classification. Then we take both the network data and IoT sensor data into consideration in Section 5.3. In addition, we compare our method with classical ML-based methods in Section 5.4. Finally, we present an interpretability analysis by visualizing the attention map in Section 5.5.

5.1. Experimental Environment

To train models, we use a high-performance computer running Ubuntu 18.04 on 3.30GH Intel(R) Core(TM) i9-9820X CPU with 128 GB main memory equipped with two NVIDIA GeForce RTX 2080 Ti GPUs. All models are built with Pytorch 1.11.0 and distributed and trained on two GPUs using the Distributed Data-Parallel module.

5.2. Evaluation Results on Pure Network Data on the ToN_IoT Dataset

In this section, we evaluate our method on pure network data from the ToN_IoT dataset to prove our method can address the problem of heterogeneous features. The training and testing data are the same as the data in [18]. The training and testing data are passed to the data processing module, which is introduced in Section 3.2. During data processing, we first drop three columns for "ts", "src_ip", and "dst_ip", which could work as spurious features to harm IDS models. Then, we separate the other columns by two types of numerical or cat-

egorical features. We select 12 features as numerical type, which are "src_port", "dst_port", "duration", "src_bytes", "dst_bytes", "missed_bytes", "src_pkts", "src_ip_bytes", "dst_pkts", "dst_ip_bytes", "http_request_body_len", and "http response_body_len". The other 28 features are categorical types. As shown in Figure 3, the numerical or categorical features are separately fed into the Feature Tokenizer for generating embeddings, which actually are the instances participating in further learning progress.

Figure 4 shows the binary classification performance by using a confusion matrix, which performs a total accuracy of 97.95%; both true positive and true negative are around 98%, the false positive is 2.1%, and the false negative is 1.9%. More evaluation results for the binary classification on network data are shown in the FT-Transformer1 column in Table 3, particularly the F1 score is 97.09% and the AUC score is 99.82%.

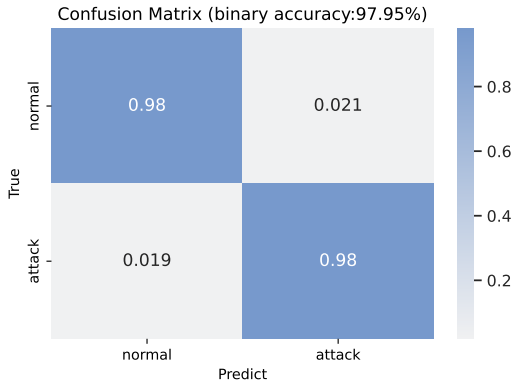

Figure 4. Confusion matrix for binary classification performance on pure network data.

The multiple classification performance on pure network data is shown in Figure 5 and Table 1; our method can achieve 95.78% overall accuracy. Based on the result in Figure 5, we observe that the three attacks with the worst detection accuracy are "mitm (Man In The Middle)", "injection", and "password" attacks with accuracies of around 65%, 73%, and 85%, respectively. The confusion matrix presents that the poor performance of "injection" and "password" are caused by the difficulty in distinguishing them from each other. Around 28% of "mitm" attacks are classified in normal traffic.

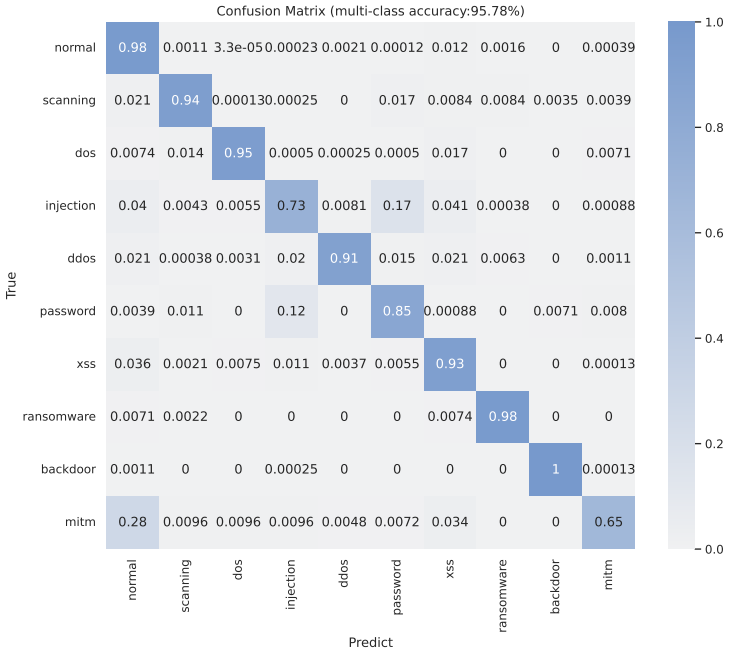

Figure 5. Confusion matrix for multiple classes on pure network data.

We also evaluate our method in terms of precision, recall, false positive rate (FAR), and F1 score. More details about multiple classification performance are presented in Table 1. Although the confusion matrix shows that our model has a high recall on "XSS" attack, the high FAR 22.48% problem is shadowed. However, the table also illustrates that our method can successfully detect "normal", "scanning", "dos", "ddos", "ransomwave", and "backdoor" traffic flows.

Table 1. Performance on pure network data of the ToN_IoT dataset for 10 classifications.

Class Type	Precision	Recall	False Alarm	F1 Score
normal	98.98%	98.29%	1.02%	0.99
scanning	94.91%	93.66%	5.09%	0.94
dos	98.22%	95.33%	1.78%	0.97
injection	82.58%	73.29%	17.42%	0.78
ddos	95.37%	91.21%	4.63%	0.93
password	80.33%	85.10%	19.67%	0.83
xss	77.52%	93.38%	22.48%	0.85
ransomware	96.23%	98.33%	3.77%	0.97
backdoor	98.95%	99.85%	1.05%	0.99
mitm	55.53%	64.99%	44.47%	0.60

5.3. Evaluation Results on Both Network Data and IoT Sensor Data

In this section, we show that our method is capable of handling the data from heterogeneous sources. First, the entire IoT data source is matched with the same training and testing network data, which have been mentioned in Section 5.2, by using Algorithm 1. The network data and matched IoT data are combined, as shown in Figure 3. For the seven provided types of IoT device data, there are 11 numerical features, which are "fridge_temperature", "temperature", "pressure", "humidity", "latitude", "longitude", "FC1_Read_Input_Register", "FC2_Read_Discrete_Value", "FC3_Read_Holding_Register", "FC4_Read_Coil" (which are the "input register", "discrete value", "holding register", and "coil" of Modbus), and "current_temperature", and six categorical features, which are "temp_condition", "motion_status", "light_status", "door_state", "sphone_signal", and "thermostat_status". For the reason of asynchronous data collection, we cannot find the matching IoT data records for many flows' feature records in network data. In order to maintain the same training and testing sets of network data as the previous experiment in Section 5.2, we fill invalid values '-' or '−1' in the empty positions of categorical and numerical features, respectively.

The binary classification performance on the combination of network and IoT data is shown as the confusion matrix in Figure 6. With the extra IoT sensor data, our method achieves 98.39% binary accuracy, which increases by 0.44% compared with pure network data performance. The true positive is around 99% and the true negative is around 98%, the false positive is 1.9% and the false negative is 1.0%. More evaluation results for binary classification on the combined data are shown in the FT-Transformer2 column of Table 3. Specifically, the F1 score is 97.72% and the AUC score is 99.88%.

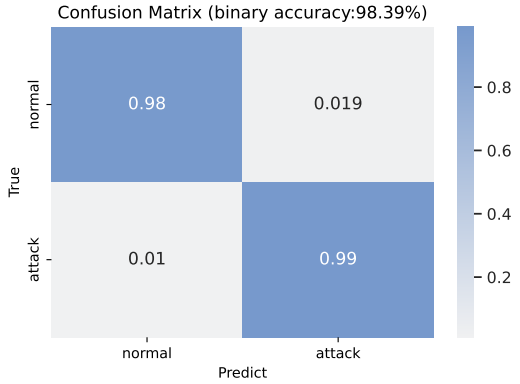

Figure 6. Confusion matrix for binary classification performance on the combination of network and IoT data.

Figure 7 reports the confusion matrix of multiple classification performances on the combined data. Compared with Figure 5, the performance has improved overall. In particular, the extra IoT data enhanced the detection performance by around 10%, 16%, and 4% in terms of true positive ratio on the classes of "mitm", "injection", and "password" attacks, which are hard to detect with only network data. An interesting finding is that the entire IoT data does not include any records about "mitm" attacks, but the detection performance on "mitm" is improved after using extra IoT sensor data. We believe there are two reasons: (1) Only the "mitm" class has data on all IoT-related features that are invalid values and that can also be recognized as a discriminate character. Given that features belonging to other types of traffic consistently comprise some valid values in the IoT-related part, the presence of invalid values indicates a significant likelihood of being a 'mitm' attack. (2) The extra IoT data can improve the FT-Transformer model's ability to distinguish normal traffic and attacks. Based on Figure 7, we can see that fewer "mitm" flows are recognized as a "normal" flow.

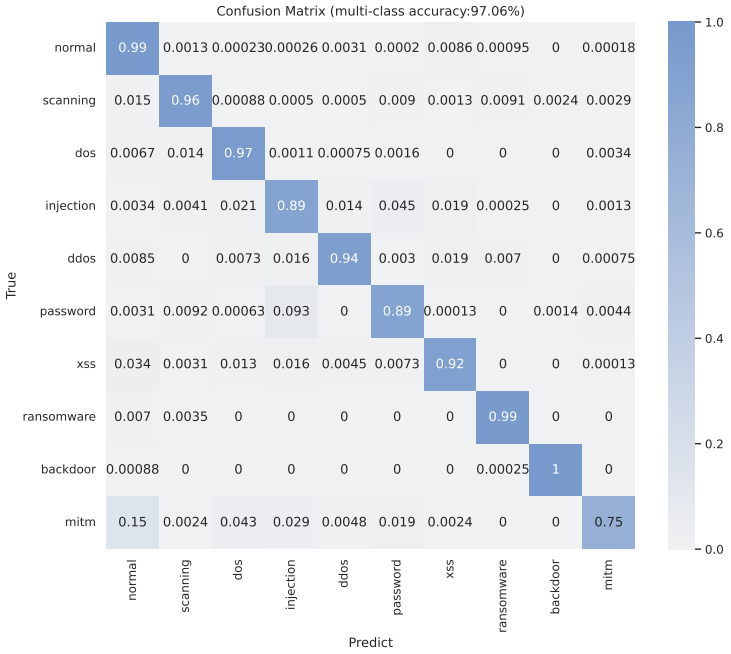

Figure 7. Confusion matrix for multiple classes on network and IoT data.

Table 2 reports more evaluation results about the multiple classification performance on the combined data by using Precision, Recall, False Positive Rate (FAR), and F1 score. Compared to Tables 1 and 2, the attack classes with high FAR in Table 1, such as "mitm",

"xss", "password", and "injection", have all been improved across all evaluation metrics, specifically, their FARs are decreased by 16.16%, 6.96%, 12.41%, and 4.53%, respectively.

Table 2. Performance on network and IoT data of ToN_IoT dataset for 10 classifications.

Class Type	Precision	Recall	False Alarm	F1 Score
normal	99.42%	98.51%	0.58%	0.99
scanning	94.69%	95.84%	5.31%	0.96
dos	95.25%	97.23%	4.75%	0.96
injection	87.11%	89.24%	12.89%	0.87
ddos	93.34%	93.85%	6.66%	0.93
password	92.74%	88.81%	7.26%	0.89
xss	84.46%	92.13%	15.54%	0.88
ransomware	96.97%	98.95%	3.03%	0.96
backdoor	99.63%	99.89%	0.37%	0.99
mitm	71.69%	75.30%	28.31%	0.76

The performance on combined data shows that even though the IoT data are asynchronous with network data, our method can still extract valuable information to enhance the classification performance. Those results prove that our method can achieve the matched performance with the state-of-the-art on pure network data of ToN_IoT. For the combination of network and IoT data, our method outperforms the state-of-the-art ML method on all metrics of accuracy, F1 score, and AUC.

5.4. Comparison with Classical ML-Based Methods

We compare the binary classification performance of our FT-Transformer-based NIDS model with three classical methods: Gradient Boosting Machine (GBM), Random Forest (RF), and Multi-Layer Perceptron Neural Network (MLP), which have been investigated in [18]. Meanwhile, we also compare the performance of our FT-Transformer on pure network data and the combination of network and IoT data.

The evaluation metrics of accuracy, F1 score, and AUC score are presented in Table 3. In this table, the FT-Transformer1 represents the model, which is trained and tested on pure network data of the ToN_IoT dataset, and the FT-Transformer2 works on the combination of network and IoT data. If only considering the first three columns of this table, RF has the best performance of all classical methods with 98.08% accuracy, 97.26% F1 score, and 99.69% AUC score. As known, the tree-based ML methods normally can achieve better performance on tabular data than other NN-based methods, no matter whether deep or not [14]. However, our FT-Transformer-based method can achieve 97.95% (−0.13%) accuracy, 97.09% (−0.17%) F1 score, and 99.82% (+0.13%) AUC score, a matched performance with RF on pure network data. Our method outperforms the MLP method, which is also a neural network method.

Table 3. Performance on the ToN_IoT dataset for binary classification compared with [18].

	GBM	RF	MLP	FT-Transformer1	FT-Transformer2
Accuracy	94.64%	98.08%	97.82%	97.95%	**98.39%**
F1	92.58%	97.26%	92.12%	97.09%	**97.72%**
AUC	98.71%	99.69%	97.85%	99.82%	**99.88%**

With extra IoT data, the column of FT-Transformer2 outperforms all of the other methods with 98.39% accuracy, 97.72% F1 score, and 99.88% AUC score. This result further proves that the FT-Transformer-based method cannot only address the problem of heterogeneous features (including both numerical and categorical features) but also improve performance by combining mixed data sources.

Additionally, we compare the multi-classes classification performance of our proposed method with that of four existing methods. Two NIDS methods, named CNN-IDS [11] and FED-IDS [10], are proposed for IoT scenarios. Another two methods, named ResNet-50 [8] and P-ResNet [9], are proposed to detect intrusion through learning on sensors' telemetry data. Based on the different "data usage" in the sixth column of Table 4, the CNN-IDS and FED-IDS only utilize the network data in the ToN_IoT dataset, but ResNet-50 and P-ResNet only utilize the IoT sensors' telemetry data. Specifically, the result of ResNet-50 [8] is evaluated on a smaller sub-dataset, which consists of 8 classes, but other methods all use datasets with 10 classes. Table 4 reports that the FT-Transformer1 outperforms other network data-based methods. Further, FT-Transformer2, which uses both network and IoT data, achieves the best performance.

Table 4. Performance on the ToN_IoT dataset for multi-classes classification comparison.

	Accuracy	F1	Recall	Precision	Data Usage
CNN-IDS [11]	90.55%	90.22%	90.55%	90.75%	network
FED-IDS [10]	94.85%	93.13%	93.09%	93.17%	network
ResNet-50 [8]	95.84%	95.78%	94.76%	96.86%	IoT sensor
P-ResNet [9]	87.0%	86.0%	86.0%	88.0%	IoT sensor
FT-Transformer1	95.78%	95.93%	95.71%	96.16%	network
FT-Transformer2	97.06%	96.94%	96.67%	97.23%	both

5.5. Interpretability Analysis

Our FT-Transformer-based method is built upon the self-attention mechanism, which can relate different elements of input and compute representation embeddings for each element. In the self-attention mechanism, the attention matrix, which is described in Equation (5), is used by the neural architecture to emphasize the relevant element embeddings.

In our case, the attention matrix can show the relations between input features. A very important characteristic of FT-Transformer is that the input does not only include the heterogeneous features but also a learnable classification embedding E^{cls}, which also participates in the self-attention calculations and is further used for final classification. Hence, we can claim that the learned classification embedding E^{cls} is the element that is directly relevant to our final task—intrusion detection. Therefore, the attention weights of the classification embedding E^{cls} with other features can show the relevance degree of all input features with E^{cls}, which can be further considered as the feature importance.

In order to visualize the attention-based feature importance, we extract the attention matrix of the last Transformer layer. We only select the row of attention values belonging to E^{cls}. Before visualizing, we also need a summation of the head dimension for the reason for our model uses eight heads for MHA, as mentioned in Section 4. We collect the attention-based feature importance for all testing data and calculate the global average.

The attention-based feature importance figures on pure network data and the combination of network and IoT data are shown in Figures 8 and 9 separately. In Figure 8, the horizontal axis represents all input features of network data, and the vertical axis is a different type of flow. For normal and attack flow, a one-dimension heat map of attention weights is used to show the contribution of each feature. Based on the lightest parts, we notice that the "duration" makes the most contribution to identifying a normal flow and

the "ssl_established" feature is the most important one for attack flow. Similarly, Figure 9 shows the attention-based feature importance for all features from both network and IoT sensor data. Comparing these two figures, we find that the "duration" is always a light region for both images. Additionally, "src_bytes" and "dst_bytes" are important to detect an attack for both situations. However, the "ssl_established" feature, which has a large attention weight in Figure 8, becomes less contributed in Figure 9. Therefore, we claim that the attention weights have limited interpretability and further study and more experiments are required. Interpretability is important for ML-based NIDS for the reason that users need to understand why the decision was made.

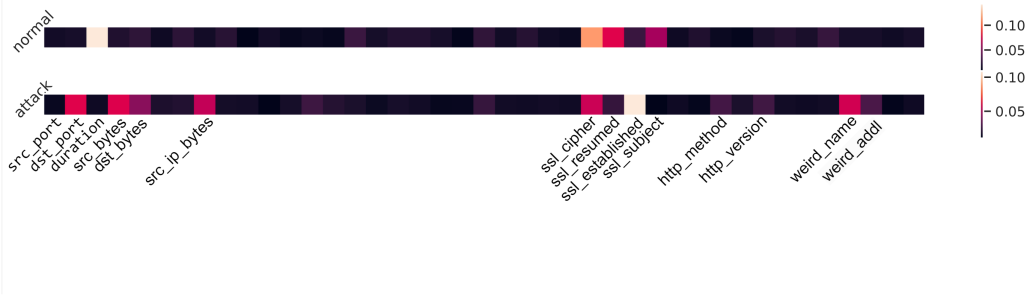

Figure 8. Attention-based feature importance of binary classification on pure network data.

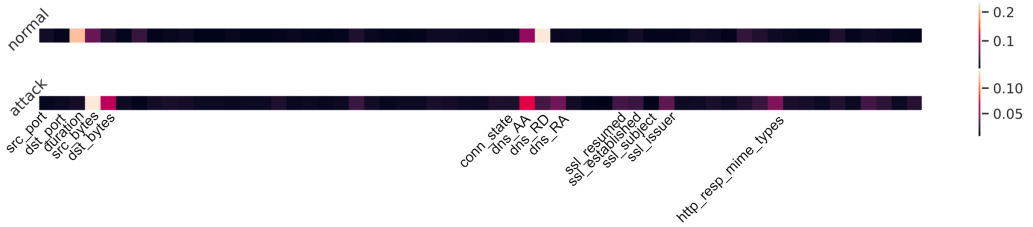

Figure 9. Attention-based feature importance of binary classification on the combined data.

5.6. Reproducibility Details

We use PyTorch to implement the FT-Transformer-based NIDS model on a server with GPU machines. The dropout ratio is set to 0.1 for each Feed Forward layer, and for each attention map in Equation (5), layer normalization is performed before each layer in the model. The embedding dimension is set to 32 for each input feature. The depth of the model is set to 6, which is also the number of Transformer encoders. For each MHA layer in Equation (4), the number of heads is set to eight. The output dimension of the final MLP layer is set to 10, which is the number of traffic types. During the training process, we used Adam optimizer [42] with a 0.001 learning rate and a 4096 training batch size. We also evaluate the efficiency and complexity of our model in our experimental environment by using the PyTorch profiler. During the execution of the model's operators for one input, the model's total CPU time is 952.775 ms, total CUDA (GPU) time is 367.000 us, total CPU memory is 372 bits, and total CUDA memory is 2.03 Mb.

6. Conclusions and Future Works

In this paper, we propose an FT-Transformer-based NIDS method for learning the behaviors of attacks from different types of data, which are generated by heterogeneous IoT devices in a smart home environment. The Feature Tokenizer module of the proposed method can separately learn embeddings for numerical and categorical features, reducing the negative effect of the heterogeneous features. The transformer blocks can use adaptive attention to emphasize the relevant information and filter the irrelevant, such as the invalid values, which are caused by asynchronous mixed data sources. The experiment results

on ToN_IoT show that the proposed method can match the performance of the state-of-the-art on pure network data. Additionally, our method can take advantage of extra IoT sensor data to further improve intrusion detection performance. Finally, we also explore the interpretability of the FT-Transformer model by visualizing the attention-based feature importance.

There are two major limitations of this work. First, our NIDS model is not trained using a real smart home. Second, the obtained NIDS model is not tested on a real practical testbed. Instead, we use the real Ton_IoT dataset [18] as a realistically representative dataset to train and test our NIDS model. Our future work will focus on how to reduce the FAR by using the decoder structure of the Transformer. Furthermore, we will consider how to make the explanation of the attention-based feature importance to be more clear and more reasonable.

Author Contributions: Methodology, software, validation, formal analysis, investigation, resources, data processing, writing—original draft preparation, visualization, M.W.; methodology, supervision, writing—original draft preparation, review and editing, project administration, funding acquisition, N.Y.; writing—review, supervision, N.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by the US National Science Foundation under Grant CC-2018919. Minxiao Wang is supported by Dr. Yang's SIU startup fund.

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

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
NIDS	Network Intrusion Detection Systems
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
FT	Feature Tokenizer

References

1. Waleed, J.; Abduldaim, A.M.; Hasan, T.M.; Mohaisin, Q.S. Smart home as a new trend, a simplicity led to revolution. In Proceedings of the 2018 1st International Scientific Conference of Engineering Sciences—3rd Scientific Conference of Engineering Science (ISCES), Diyala, Iraq, 23–24 April 2018; pp. 30–33.
2. Kundu, D.; Khallil, M.; Das, T.; Mamun, A.; Musha, A. Smart Home Automation System Using on IoT. *Int. J. Sci. Eng. Res.* **2020**, *11*, 697–701. [CrossRef]
3. El-Azab, R. Smart homes: Potentials and challenges. *Clean Energy* **2021**, *5*, 302–315. [CrossRef]
4. Touqeer, H.; Zaman, S.; Amin, R.; Hussain, M.; Al-Turjman, F.; Bilal, M. Smart home security: Challenges, issues and solutions at different IoT layers. *J. Supercomput.* **2021**, *77*, 14053–14089. [CrossRef]
5. Abdul-Qawy, A.; Magesh, E.; Tadisetty, S. The Internet of Things (IoT): An Overview. *Int. J. Eng. Res. Appl.* **2015**. *5*, 71–82.
6. Stojkoska, B.L.R.; Trivodaliev, K.V. A review of Internet of Things for smart home: Challenges and solutions. *J. Clean. Prod.* **2017**, *140*, 1454–1464. [CrossRef]
7. Balo, F.; Torğul, B. Internet of Things: A Survey. *Int. J. Appl. Math. Electron. Comput.* **2016**, 104–110. [CrossRef]
8. Kodyš, M.; Lu, Z.; Fok, K.W.; Thing, V.L.L. Intrusion Detection in Internet of Things using Convolutional Neural Networks. In Proceedings of the 2021 18th International Conference on Privacy, Security and Trust (PST), Auckland, New Zealand, 13–15 December 2021; pp. 1–10. [CrossRef]
9. Mehedi, S.T.; Anwar, A.; Rahman, Z.; Ahmed, K.; Islam, R. Dependable Intrusion Detection System for IoT: A Deep Transfer Learning Based Approach. *IEEE Trans. Ind. Inform.* **2023**, *19*, 1006–1017. [CrossRef]
10. Abdel-Basset, M.; Moustafa, N.; Hawash, H.; Razzak, I.; Sallam, K.M.; Elkomy, O.M. Federated Intrusion Detection in Blockchain-Based Smart Transportation Systems. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 2523–2537. [CrossRef]
11. Oseni, A.; Moustafa, N.; Creech, G.; Sohrabi, N.; Strelzoff, A.; Tari, Z.; Linkov, I. An Explainable Deep Learning Framework for Resilient Intrusion Detection in IoT-Enabled Transportation Networks. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 1000–1014. [CrossRef]

12. Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access* **2020**, *8*, 165130–165150. [CrossRef]
13. Zuech, R.; Khoshgoftaar, T.M.; Wald, R. Intrusion detection and Big Heterogeneous Data: A Survey. *J. Big Data* **2015**, *2*, 3. [CrossRef]
14. Huang, X.; Khetan, A.; Cvitkovic, M.; Karnin, Z. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv* **2020**, arXiv:2012.06678.
15. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
16. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6.
17. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.
18. Booij, T.M.; Chiscop, I.; Meeuwissen, E.; Moustafa, N.; Hartog, F.T.H.D. ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets. *IEEE Internet Things J.* **2022**, *9*, 485–496. [CrossRef]
19. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*. Available online: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (accessed on 30 April 2023).
20. Hameed, A.; Alomary, A. Security Issues in IoT: A Survey. In Proceedings of the 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Padova, Italy, 13–15 February 2019; pp. 1–5. [CrossRef]
21. Abdul-Ghani, H.A.; Konstantas, D.; Mahyoub, M. A comprehensive IoT attacks survey based on a building-blocked reference model. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*. [CrossRef]
22. Tahsien, S.; Karimipour, H.; Spachos, P. Machine learning based solutions for security of Internet of Things (IoT): A survey. *J. Netw. Comput. Appl.* **2020**, *161*, 102630. [CrossRef]
23. Xiao, L.; Wan, X.; Lu, X.; Zhang, Y.; Wu, D. IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security? *IEEE Signal Process. Mag.* **2018**, *35*, 41–49. [CrossRef]
24. Hasan, M.; Islam, M.; Islam, I.; Hashem, M. Attack and Anomaly Detection in IoT Sensors in IoT Sites Using Machine Learning Approaches. *Internet Things* **2019**, *7*, 100059. [CrossRef]
25. Masdari, M.; Khezri, H. A survey and taxonomy of the fuzzy signature-based Intrusion Detection Systems. *Appl. Soft Comput.* **2020**, *92*, 106301. [CrossRef]
26. Alsoufi, M.A.; Razak, S.; Siraj, M.M.; Nafea, I.; Ghaleb, F.A.; Saeed, F.; Nasser, M. Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review. *Appl. Sci.* **2021**, *11*, 8383. [CrossRef]
27. Ferrag, M.A.; Maglaras, L.; Moschoyiannis, S.; Janicke, H. Deep Learning for Cyber Security Intrusion Detection: Approaches, Datasets, and Comparative Study. *J. Inf. Secur. Appl.* **2020**, *50*, 102419. [CrossRef]
28. Sommer, R.; Paxson, V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010; pp. 305–316. [CrossRef]
29. Arp, D.; Quiring, E.; Pendlebury, F.; Warnecke, A.; Pierazzi, F.; Wressnegger, C.; Cavallaro, L.; Rieck, K. Dos and don'ts of machine learning in computer security. *arXiv* **2020**, arXiv:2010.09470.
30. Darktrace. Avalable online: https://darktrace.com/ (accessed on 3 March 2022).
31. Ashiku, L.; Dagli, C. Network Intrusion Detection System using Deep Learning. *Procedia Comput. Sci.* **2021**, *185*, 239–247. [CrossRef]
32. Satam, P.; Hariri, S. WIDS: An Anomaly Based Intrusion Detection System for Wi-Fi (IEEE 802.11) Protocol. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 1077–1091. [CrossRef]
33. Mahadik, S.; Pawar, P.M.; Muthalagu, R. Efficient Intelligent Intrusion Detection System for Heterogeneous Internet of Things (HetIoT). *J. Netw. Syst. Manag.* **2022**, *31*, 2. [CrossRef]
34. Bertoli, G.D.C.; Junior, L.A.P.; Saotome, O.; dos Santos, A.L. Generalizing intrusion detection for heterogeneous networks: A stacked-unsupervised federated learning approach. *Comput. Secur.* **2023**, *127*, 103106. [CrossRef]
35. Almutairi, A.H.; Abdelmajeed, N.T. Innovative signature based intrusion detection system: Parallel processing and minimized database. In Proceedings of the 2017 International Conference on the Frontiers and Advances in Data Science (FADS), IEEE, Xian, China, 23–25 October 2017; pp. 114–119.
36. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
37. Bikmukhamedov, R.F.; Nadeev, A.F. Generative transformer framework for network traffic generation and classification. *T-Comm-Telekommunikacii i Transport* **2020**, *14*, 64–71.
38. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv* **2020**, arXiv:2010.11929.

39. Wang, H.; Li, W. DDosTC: A Transformer-Based Network Attack Detection Hybrid Mechanism in SDN. *Sensors* **2021**, *21*, 5047. [CrossRef]
40. Wu, Z.; Zhang, H.; Wang, P.; Sun, Z. RTIDS: A Robust Transformer-Based Approach for Intrusion Detection System. *IEEE Access* **2022**, *10*, 64375–64387. [CrossRef]
41. Gorishniy, Y.; Rubachev, I.; Khrulkov, V.; Babenko, A. Revisiting deep learning models for tabular data. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 18932–18943.
42. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.