Semi-supervised Transformer Network for Anomaly Detection in Cellular Internet of Things

Waleed Abd Elkhalik[1], Ibrahim Elhenawy[2]

[1,2]Faculty of Computers and Informatics, Zagazig University, Zagazig, Sharqiyah, 44519, Egypt
Emails: ielhenawy@zu.edu.eg; waleed.abdlekhalik@zu.edu.eg

Abstract

Because of the lightning-fast expansion of the Internet of Things (IoT) technologies, an enormous amount of data has been produced. This traffic can be mined for information that can be used to identify and avoid intrusions into IoT networks. Despite the significant efforts that have been put into labeling Internet of Things traffic records, the total number of labeled records is still quite low, which makes it more difficult to detect intrusions. This study introduces a semi-supervised deep learning approach for intrusion detection (S2T-Net), in which we propose a temporal transformer module to empower the model to learn valuable interactions in cellular data. An improved spatial transformer is presented to capture local representation in the cellular traffic flow. At the same time, a multilevel semi-supervised training technique is used to account for the consecutive structure of the IoT traffic information. In order to provide effective real-time threat intelligence, the suggested S2T-Net can be tightly coupled into a cellular IoT network. Last but not least, empirical assessments on two current databases (CIC-IDS2017 and CIC-IDS2018) show that S2T-Net boosts intrusion detection accuracy and resilience while retaining resource-efficient computing.

Keywords: Cellular Networks; Internet of Things (IoT); Deep Learning; Semi-supervised Learning; Anomaly Detection; Security**.**

1. Introduction

The rapid integration of cellular technologies into the internet of things (IoT) has allowed for the creation of numerous cutting-edge applications and services such as "smart" versions of traditional industries like manufacturing and hospital, and "dumb" ones like transportation. Lots and lots of cellular IoT traffic are transmitted between existing cellular IoT entities to send data like switch control, smart device management, connectivity information, and maintenance and repair specifics; this knowledge can be damaged by a wide range of problems initiated by manufactured or natural anomalies. As a result, effective intrusion detection has become a crucial feature in cellular IoT[1], [2].

Intrusion Detection Systems, often known as IDS, have been developed in order to identify intrusions that have circumvented traditional security measures. This serves as a very important second line of defense for the purpose of securing cellular IoT. IDS that are based on anomaly detection will establish a profile of typical behavior and will only label activities as intrusions if those behaviors do not fit the usual profile. In light of this, intrusion detection systems (IDSs) can be split into two distinct classifications: anomaly-based IDS and signature-based IDS (A-IDS) [3]. Evaluating the relationship with previously learned signatures of known assaults is the method that the S-IDS employs in an effort to identify malicious interventions. However, as S-IDS are unable to detect new intrusions that have not been observed before, their workload grows as a result of the rise in the count of newly recognized intrusions, which leads to a rise in the number of signatures and, as a result, limits their responsiveness. In addition, S-IDS frequently requires the participation of human specialists in order to inspect

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

and investigate the signatures of innovative attacks. In contrast, A-IDS is able to differentiate between unknown breakouts, which frequently occur in the majority of types of cellular IoT [4].

The problem of recognizing unfamiliar invasions is a specific problem for cellular IoT, which connects a broad variety of gadgets with varying computational power, wireless communications, storage capacity, applications, and operating system configurations. This presents a challenge when it comes to securing the network from unauthorized access. This heterogeneous nature presents a difficulty for the implementation of security mechanisms and raises the threat vectors, both of which contribute to cellular IoT networks being more susceptible to innovative and unexpected incursions. It has been demonstrated that conventional machine learning (ML) algorithms may effectively discern critical anomalies in the cellular network [5]. IoT traffic, and can therefore effectively detect cyberattacks. On the other hand, it has been shown that machine learning is unable to scale to extremely large databases, and it has also been demonstrated it achieves an inadequate level of performance when it came to finding intrusions and cyber threats in situations in which the cellular IoT nodes are extremely dispersed. Alternately, ongoing developments in deep learning (DL) methodologies give rise to different intrusion detection systems that are well-equipped to address and address the necessary incursions and cyberattacks, as well as varying degrees of difficulties and complexity. As a result, this work puts out the idea of an A-IDS based on the revised DL model [6].

Because the contextual knowledge of the cellular IoT is gathered at such a quick rate over time, manually labeling a massive number of cellular IoT information is a task that is becoming increasingly difficult, if not impossible altogether. However, just a tiny part of the cellular IoT traffic could be labeled, while the rest of the data, which constitutes the majority, could be left unnamed. The semi-supervised deep learning methodologies are perfectly adapted for a scenario and were shown to be an appropriate method for analysis and regression. This facilitates the generation of smart systems that can learn from massive volumes of unannotated data utilizing a small number of labeled examples. It is possible to handle cellular IoT records as time-series data due to the fact that they are generated in consecutive order. It has been found that recurrent neural networks, or RNNs, perform exceptionally well when applied to this type of data and have been the subject of research. Among these, the long short-term memory (LSTM) and the gated recurrent unit (GRU) have become known as an improved version of the RNN for a variety of consecutive data functions. Several different investigations on the identification of intrusions and attacks have provided evidence that the RNN is effective. Even though it is not capable of capturing consecutive data representations, more lately, convolution neural networks (CNN) have been utilized for the purpose of anomaly recognition because of the spatial feature extraction ability it possesses. Temporal CNN (TCN) has been developed by changing the theory of CNN to add parallelism methods. TCN has demonstrated significant improvement over CNN due to the fact that it is able to learn long relationships more compellingly. In addition to this, the architectural integrity of the TCN is simpler and more accurate in comparison to that of the RNNs, and the TCN has achieved superior outcomes compared to those of the LSTM in a number of sequencing problems [7], [8].

A. Motivations

This paper addresses a number of the constraints as well as the issues that were discovered after conducting an in-depth analysis of the most recent activities completed for IDS. First, in cellular IoT traffic communications, the examples captured at surrounding sites are connected with others, producing a form of residential dependency. This interdependence is created by associating the observations. Second, the composition of cellular IoT traffic flows is heterogeneous, which means that the involvement of each type of data is not necessarily the same [9]. This can occur in either the spatial domain or the time domain. As an illustration, the spatial information can vary greatly because of the variations in cellular IoT devices, routers, networking protocols, and suppliers; in addition, the historical regularity that corresponds to the spatial information can also vary. Third, mobile IoT traffic examples that were collected at adjacent time frames are particularly tightly related, which means that the temporal context must be taken into consideration when developing new IDS. Fourth, because of the rapid and dramatic increase in the bulk of cellular IoT data, it is challenging to get up-to-date tagged cellular IoT traffic information, particularly to detect intrusions and intrusions. Learning from unlabeled data has consequently become an important research topic in cellular IoT systems [10].

B. Primary Contributions

This research offers the following novel contributions to overcome the difficulties that have been listed above:

- We present S2T-Net, a unique semi-supervised transformer network for detecting anomalies in cellular IoT by exploiting the advantages of both labeled and unlabeled data streams throughout training to identify incursions and threats in cellular IoT data.

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

- In S2T-Net, a new temporal transformer module is designed to empower the model to learn valuable interactions in cellular data.
- In S2T-Net, An improved spatial transformer is presented to capture local representation in the cellular traffic flow. At the same time, a multilevel semi-supervised training technique is used to account for the consecutive structure of the IoT traffic information
- For the purpose of training S2T-Net, a novel semi-supervised hierarchical method is presented. In this method, the unlabeled cellular IoT traffic data are divided into a number of different portions that are then ordered sequentially. Then, the training methods are carried out in a gradual manner on each individual component, which allows the successive interdependencies to be maintained throughout the training process.

C. Paper Structure

The work is structured as such. The current research that is pertinent to intrusion detection in cellular IoT is described in Section 2. Section 3 delves into the specifics of the proposed methodology. As for the experimental setups, these are described in Section 4. The findings, evaluation, and related discussion may be found in Section 5. The conclusions are presented in Section 6.

2. Related Work

Studies on the security of cellular IoT communications have been conducted in abundance. Creating trustworthy IDS in a cellular IoT -enabled setting has always been a significant obstacle, and AI-based techniques have been a key component in overcoming this. There are now three main schools of thought when it comes to cellular IoT intrusion detection: supervised, semi-supervised, and unsupervised.

A. Supervised Anomaly Detection in Cellular IoT

Training-supervised DL methods often use labeled cellular IoT data for either binary classification or multi-class classification. A lightweight DL solution (named LNN).for IoT network intrusion detection was developed by the authors of [11]. Due to the high computing cost associated with using high-dimensional raw traffic data, we apply the principal component analysis (PCA) algorithm to complete feature discretization in the information pretreatment step. LNN classifier utilized the expansion and contractions design, the inversion residual construction, and the channel shuffle algorithm to efficiently extract features with low computational overhead. Moreover, the DL model (named Deep-IFS) constructed by the authors of [12] used a local gated recurrent unit (GRU) to acquire localized depiction and multiple attention layers to obtain lengthy interactions.

Although supervised techniques have made considerable strides in anomaly detection, they have not become widely used due to a lack of annotated cellular IoT data, which necessitates extensive work and a lengthy time commitment. Moreover, when information is not distributed uniformly among classes, their performance deteriorates (class imbalance problem). This inspired us to create effective A-IDS using semi-supervised learning.

B. Semi-supervised Anomaly Detection in Cellular IoT

In situations when only a limited number of labeled examples are accessible, semi-supervised methods are useful because they train a specific classifier by making use of both annotated and unannotated samples. Various methods of this kind have been designed with the aim of detecting intrusions in cellular IoT traffic, and many of them have shown that they are effective. The authors of [13] developed a semisupervised hierarchy layering temporal convolutional network (HS-TCN), for detecting intrusion in IoT networks by training on only a tiny number of labeled data. Besides, the authors of [14] developed a key quality indicators-based framework that employs a semi-supervised ML algorithm, namely, iterative positive sample aided one-class support vector machine (IPS-OCSVM) to detect anomalies in cellular IoT. IPS-OCSVM can be implemented in four stages, the most important of which is employing OCSVM to combine ML with the specialized knowledge of the network operator. Through a soft decision, the IPS-OCSVM architecture is able to detect QoE anomalies, and its detection capability can be readily tuned on request. More, the authors of [15], presented a Latent Enhanced regression/classification Deep Generative Model (LEDGM) as a deep generative technique for addressing the high dimensionality anomaly detection challenge. LEDGM moves away from traditional two-stage disconnected models and instead uses a holistic learning approach. Further, the authors of [16] developed a Convolutional Autoencoder (VAE) as a hybrid design for trajectory classification and anomaly detection. They offered a color gradient representation of high-level features for object trajectories of varied lengths. Later, pathways of moving objects recovered with the Temporally Incremental Gravitational Model (TIGM) are annotated using a semi-supervised method for class categorization. The authors of [17], proposed a multiresolution residue temporal convolutional (MS-Res) component for fine-tuning the network's capacity in learning multimodal patterns, as

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

part of a semi-supervised DL model for vulnerability scanning (known as SS-Deep-ID). To help the model zero in on the most relevant data as it learns, they developed a traffic attention (TA) approach for estimating the significance of cellular patterns and a hierarchical learning approach to better account for sequential properties of the IoT traffic data.

C. Unsupervised Anomaly Detection in Cellular IoT

Unsupervised Anomaly Detection refers to methods developed without using clean data or traffic labeling. These methods are preferable since they are inexpensive (because they don't have to pay for traffic labels) and they use the innate properties of cellular IoT traffic samples to identify and prevent various types of assaults. As a result, they are able to reliably detect emerging threats. Using an example, the authors of [18] investigated unsupervised outlier detection on multidimensional time series data in IoT systems and creates a GRU-based Gaussian Mixture VAE method (termed GGM-VAE). specifically, they utilized Gaussian Mixture priors in the higher dimensional space to characterize the different datasets and Gated Recurrent Unit (GRU) cells to uncover the relationships across time series information. Some prior works use overly simplistic distributions for Gaussian Mixture priors, which severely limits their capacity to detect data patterns. More, the authors of [19] presented a stacked autoencoder (AE) for intelligent anomaly detection by narrowing down the feature set by unsupervised learning from the input networking flows' features. The authors of [20] used noisy pseudo-normal data, to develop a unique self-supervised approach for anomaly identification in in-vehicle networks. The generator and detector in the proposed method are both deep-learning models that are responsible for producing noised pseudo-normal data and identifying outliers, respectively. the generator is trained using just "regular" network traffic in order to produce "normal"-looking generated traffic. Then, the abnormality detector was taught to distinguish between authentic traffic and noised faux normal traffic.

The two main problems with unsupervised methods remain, despite their benefits. To begin, their performance is not as strong as supervised methods, especially when it comes to recognizing previously discovered threats. Secondly, their high processing complexity prevents them from being used in genuine or in applications with limited cellular IoT resources.

In essence, current semi-supervised methods exhibit a significant performance versus computational cost tradeoff. Since spatial abstractions and chronological properties of cellular IoT traffic data are crucial to achieving maximum performance, we propose using TCN instead of recurrent networks. As a further step, we include a learning algorithm to assist the net zero in on relevant features, which in turn reduces the effect of random data and accelerates the learning process. The present semi-supervised techniques also fail to preserve the periodic properties of cellular IoT data while being trained. Therefore, we suggest a multilevel training structure for the compact S2T-Net.

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

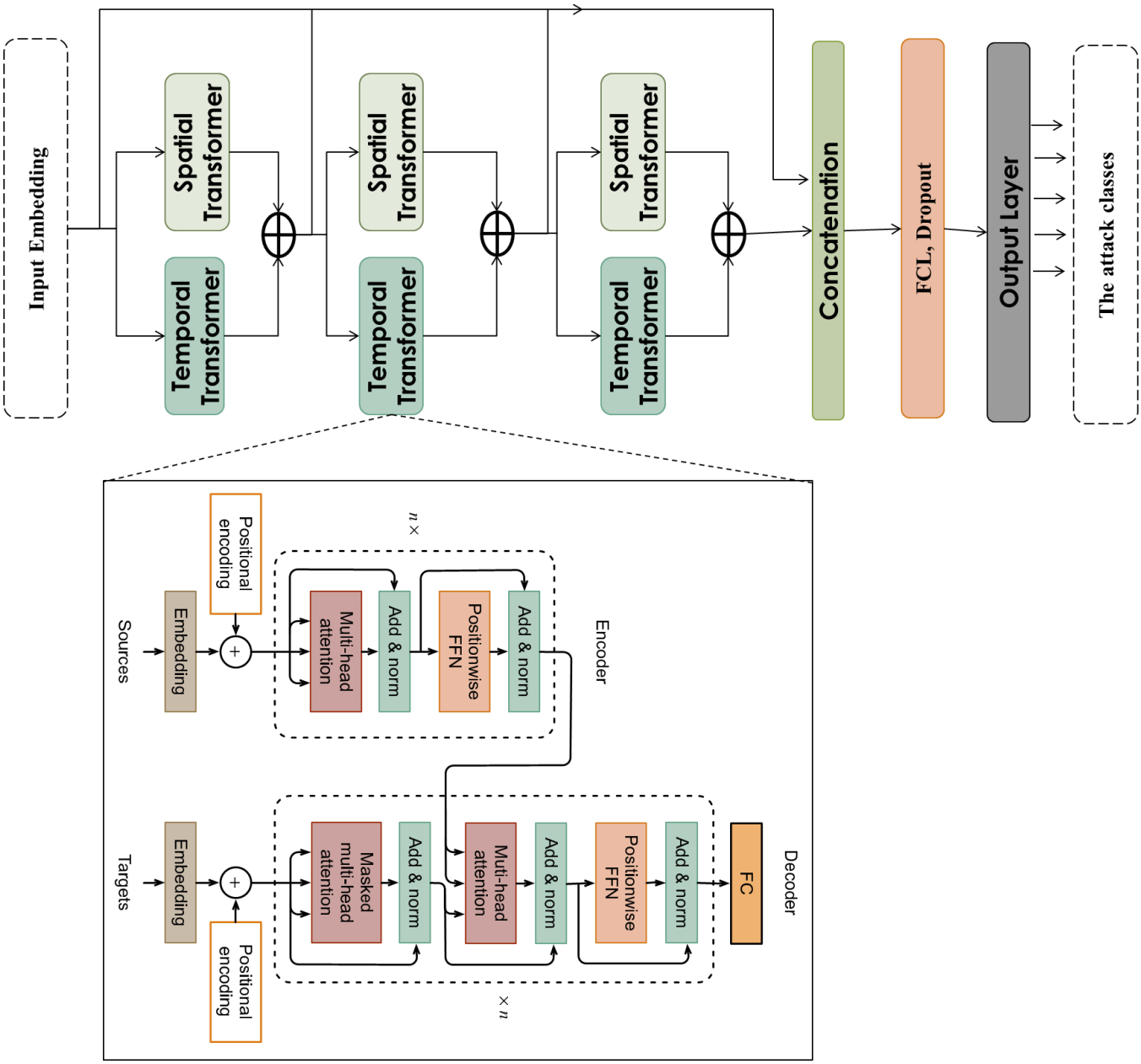

Figure 1: The architecture of the proposed S2T-Net

3. Methodology

In this section, a comprehensive overview of the proposed semi-supervised DL for recognizing attacks in cellular IoT traffic, which is referred to as S2T-Net, is presented. The architecture of S2T-Net, which can be seen depicted in Figure 1, is broken down into three basic modules. To begin, the temporal residual transformer block was suggested as a way to better hone the powers of the net when it came to acquiring the representations. Second, a spatial transformer block that may measure the significance of the input representation in order to direct the network's attention to structures that are reflective of the whole. In the end, the outputs of these modules are combined and then supplied into the linear layer so that the likelihood may be calculated that the base input refers to the particular class.

D. Embedding

To begin building our framework, we use embedded methods to transform the unprocessed data. As a result of the fact that IP packets' origin and destination port numbers and IP addresses are encoded sequentially, network traffic also contains other successive features. To make use of the spatial information conveyed by the input

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

features, our model employs a further positional encoding layer at the base of the encoder stack. The last input of the self-attention layer is the sum of the embedding and positional encoding, as the positional embedding sizes are the same as the input embedding sizes. In addition, sine and cosine functions at varying frequencies are used to adopt positional encoding in the input embedding. In specific, as shown in Equation. (1) and Equation. (2), a cosine function is used to encode odd positions and a sine function is used to encode even positions.

$$PE(pos, 2i) = sin\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right), \quad (1)$$

$$PE(pos, 2i+1) = cos\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right), \quad (2)$$

E. Spatial Transformer

Each spatial transformer in the spatial stack consists of a sparse multi-head self-attention (SMHA) layer followed by point-wise full connected layers (FCLs), for a total of six modules. Training allows for the hyperparameterization of the FCL size. Assigning 1024 neurons to FCLs and making 32 the padding size of embedding yields the best results. In addition, each sub-calculated layer's outcomes are carried onto the following encoder in the stack via leftover connection and layer regularisation (see Figure. 1).

F. Temporal Transformer

In contrast to the conventional transformer framework [21], we augment each decoder with a masked SMHA layer. To make our suggested S2T-Net framework more bulletproof, we arbitrarily mask some features and forecast them with other, unmasked features. Likewise, we use a temporal transformer stack with six temporal transformers to preserve our model's hierarchy organization. Rebuilding the encoder and decoder stacks allows us to use the SoftMax function as the final output layer for classification. Layer normalization occurs after an input's self-attention layer has a leftover connection to a point-wise FCLs, as shown in Figure. 1. These methods aim to boost the effectiveness of the model by fixing issues like the vanishing gradient and the potential confounders shift. Every encoder and decoder has a set of linear sublayers they call a point-wise FCL, in addition to the SMHA layer. Refer to Eq. (3) for an illustration; it consists of two mathematical operations triggered by the ReLU function. In addition, the attention matrices are transformed by performing a convolution on each row using the same weights. This process is seen as adding value to the embeddings by providing more specific data for them to work with.

$$FCL(x) = \sigma(max(0, xW_1 + b_1)\, W_2 + b_2) \quad (3)$$

The masked scaled dot-product operation in SMHA for each head h in the Temporal transformer is displayed in Figure.1. this operation receives three sets as input namely queries (Q), keys(K), and values (V). The embedding is transformed into a matrix Y with dimensions of $E \times L$.

$$\boldsymbol{Q}^{(h)} = (\boldsymbol{W}_Q^{(h)})^\top \boldsymbol{Y}, \quad (4)$$

$$\boldsymbol{K}^{(h)} = (\boldsymbol{W}_K^{(h)})^\top \boldsymbol{Y}, \quad (5)$$

$$\boldsymbol{V}^{(h)} = (\boldsymbol{W}_V^{(h)})^\top \boldsymbol{Y}, \quad (6)$$

Then, the sparse attention (SA) is computed via masked scaled dot-product as follows:

$$SA(Q, K, V) = softmax\left(M + \frac{QK^T}{\sqrt{d_k}}\right)V \quad (7)$$

$$\boldsymbol{Z}^{(h)} = [\boldsymbol{z}_0^{(h)} \cdots \boldsymbol{z}_{(L-1)}^{(h)}] = \boldsymbol{V}^{(h)} \cdot softmax(\frac{1}{\sqrt{p}}(\boldsymbol{Q}^{(h)})^\top \boldsymbol{K}^{(h)} + \boldsymbol{M}), \quad (8)$$

whereby the d_k denotes the scaling factor. The training gradients are stabilized by the division by d_k in the above formula. To guarantee that the forecasting for situation i is based only on the known outputs at locations less than i. we then use $M \in R^{L \times L}$ to prevent attending successive positions. Given the mask matrix M with dimensions $L \times L$, the $(i, j) - th$ component, $m_{i,j}$, is characterized as:

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

$$m_{i,j} = \begin{cases} 0 & if j \le i \\ -\infty & if j > i \end{cases} \quad (9)$$

Using this method, the overfitting issue can be mitigated. In addition, the SoftMax function incorporates the masking procedure by adding negative infinity. The SoftMax layer is then applied to each row of the matrix to normalize it into a posterior distribution.

A new input representation is built using the dot product of the normalized matrix and the vector V. With the multi-head mechanism, as shown in Figure. 1, the self-attention layer's effectiveness could be further enhanced. When using SMHA, each attention head keeps its own $Q/K/V$ weight matrix. The following formulas illustrate a calculation procedure analogous to the classic "one-head attention" paradigm. An $i-th$ attention of the sparse self-attention is denoted as the head. The final outcome is a concatenation of the individual results from each head. Because of this method, the framework can switch its attention to various locations, and it can use a variety of encoding threads in its attention layer.

$$SMHA(Q,K,V) = Concat(h_1, \dots, h_h)W^O \quad (10)$$

$$h_i = SA(QW_i^Q, KW_i^K, VW_i^V) \quad (11)$$

Layer normalization is performed on each sample for the sake of steady training and quick convergence.

$$LN(x) = \left(\frac{x-\mu}{\delta}\right) \cdot \alpha + \beta \quad (12)$$

G. Output Module

This section describes the outputs of the output calculation in the proposed S2T-Net by exploiting feature representation obtained from the above modules. in particular, the output module of applied full connected layers (FCLs) encodes the learned features into a linear vector, according to which the probability of each class is computed with the SoftMax function.

$$F_{linear} = f_{fcl}(F) \quad (13)$$

$$S = SoftMax(F_{linear}) = \frac{exp(F_{linear})}{\sum_1^C exp(F_{linear})} \quad (14)$$

$$\tilde{y} = argmax(S) \quad (15)$$

whereas F_{linear} indicates the production of the FCLs, and S signifies the softmax score. The S2T-Net is trained to minimize the categorical cross-entropy formulated as follows.

$$loss = -\sum_{c=1}^{C} y_c \, log \, (\tilde{y}_c) \quad (16)$$

where y_c is the authentic class and $\tilde{y}_c$ is the model-predicted, class.

H. Semi-supervised training

The massive volume of everyday cellular IoT traffic data combined with the time needed to label each data has contributed to the rise in popularity of semi-supervised training as a means to train a DL model with a mixture of these two types of records. Anomaly detection in cellular IoT connectivity is thus better served by a semi-supervised algorithm. Eighty percent of the training data in this research set is unmarked. The rest 20 percent was used as labeled samples. Tagging the previously collected cellular IoT traffic information is generally recommended by the scientific community. This means that the labeled records have an earlier time than the unlabeled ones. Because they are collected over a lengthy time frame, the sequential pattern of the records of cellular IoT traffic is also a thoughtful design. In this case, the unlabeled data set has a greater temporal span than the labeled data set. While learning unlabeled cellular IoT data, the serial association is ignored if there is no corresponding action for the corresponding unenforced. Because of this, we use a hierarchical approach to divide the unlabeled observations into relatively little timespan.

In order to train the S2T-Net in a hierarchical manner, we partitioned the unannotated samples into P subsets. The order of parts P-1 and P is substantial. Increasing P allows the network's learning capability to be enhanced with each new component deemed, thereby increasing the network's representational power. Nevertheless, overfitting may occur if P is set too high, and the computational time may increase as a result. Therefore, it is

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

crucial to look into the best possible value of P. The S2T-Net could then be assessed using the labeled data and the unlabeled data from the investigation, with each unlabeled component. A complete set was thus obtained after combining all relevant elements. Therefore, we can express S2T-experiment Net's on the entire collection of unlabeled traffic data as

$$\dot{X}_u^i = S2T - Net_i\left(X_l + \sum_{j=1}^{i} \dot{X}_u^j\right) \quad (17)$$

When the training of S2T-Net's layers is finished, the unlabeled data are evaluated after processing all unlabeled components. The classification of intrusions or cyberattacks is ultimately determined by combining labeled and experimental data and computing the judgment output thus according to:

$$F = S2T - Net\left(X_l + \dot{X}_u\right) \quad (18)$$

The hierarchical structure allows for the unlabeled data to be partitioned into multiple parts within a limited time period, as shown above. All through the unlabeled traffic data experimental tests, the S2T-Net i-th layer incorporates the results of the previous experiments, indicating that the tests are carried out incrementally. This semi-supervised hierarchy learning is thus well suited to the traffic information in cellular IoT, as it incrementally accounts for the temporal interconnectedness in the unlabeled cellular IoT records.

I. Deployment in cellular IoT

This section discusses the deployment and operation of the proposed S2T-Net in a practical cellular setting. Figure. 2 shows a system diagram for the suggested S2T-Net, a semi-supervised anomaly detection framework for cellular IoT that makes use of edge nodes. As a general rule, the system has three main parts, or layers: the cloud layer, the edge node layer, and the edge layer. Models training, which necessitates access to a vast number of cellular IoT traffic data, is carried out from the central cloud due to its well-known high and strong computational resource. The cloud is an ideal location for aggregating and storing data of this magnitude. The model designs, previous pre-trained editions, and more training-transaction-related settings are all kept on the central cloud.

To move processing to the edge nodes of the cellular IoT, the edge node layer often employs a large number of edge nodes. The edge node layer plays an important part in the SS-IDF because that is where intrusions would be detected. Essentially, each edge node has four major parts: the traffic gathering element, the traffic planning element, the traffic defense element, and the traffic analysis component. The edge cellular IoT's linked element generates traffic records, which must be captured and received by the traffic consolidation element, which then forwards the batched examples to the preparation process. The traffic processing part is in charge of standardizing the incoming batches, performing any data cleansing or normalization, and then sending them on to be processed. Next, the cellular IoT traffic data is generated, and the trained S2T-Net is used to classify it without contacting the cloud backbone, avoiding any lag.

In this system, in binary class or multi-class scenarios, the process can be executed. After an action has been flagged as malicious, the relevant data is sent to a server in the cloud for analysis. For the cellular IoT, each edge node is liable for analyzing the local area. The corresponding edge node is always on the lookout for IoT traffic logs, and as a result, it collects all of them. Given significant changes in traffic on a specified zone of the underlying cellular IoT, if these changes are malicious, i.e., denial-of-service (DoS) occurrences, the proposed S2T-Net would recognize this and send the prevention and mitigation elements. To prevent the edge network from becoming overburdened, it may be necessary to switch certain nodes over to a different available edge node even if the change is harmless. The defenses module accepts the S2T-Net-generated decisions and applies any appropriate warnings, blocking, or removal operations from a preset module. After that, the discovered action's details are uploaded to the cloud, where they will be used by the Log section of the final findings [22].

The last layer is called the edge layer, and it is comprised of the network edges and end devices (such as notebooks, cell phones, wearables, and so on). These devices communicate with one another through cellular IoTs using connectivity and trying to switch gadgets, and they are concurrently linked with a particular edge node that acts as an information processor connected to the server.

4. Experiments

J. Dataset Description

There is a vast selection of security databases that are available to the general public. From these, we select two common and large-scale databases to evaluate the effectiveness of the proposed model; these are the CIC-IDS2017 and the CIC-IDS2018 datasets. The CIC-IDS2017 is made up of 2,830,743 IoT traffic examples that

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

were obtained in a miniature, virtual communication network. Within this network, there are six different kinds of anomalies that were started from a distributed network. each sample consists of 78 standard attributes and belongs to one of seven classes, and they include both malicious and non-malicious traffic. Additionally, there are 14 different types of attacks. On the opposite side, the CIC-IDS2018 is a well-known example of real-world heterogeneous anomaly recognition data. These data are often more difficult to work with since they contain null values, unnecessary features, extremes, incorrect occurrences, and significant disparities. Therefore, more comparable to the cellular Internet of Things found in the real world. It is composed of 16,233,002 different IoT traffic samples that were collected over ten days of network activity. Despite this, the samples were collected from a larger network known as victim networks, which consisted of 30 servers and 420 client workstations in total. The dataset is made up of 79 different features, and about 17% of the samples are considered to attack traffic. is broken out into ten separate CSV files that can be accessed and downloaded by the general public.

K. Data preparation

Before beginning the training, the subsequent processes for data preparation were executed on both datasets. First, we got rid of any unnecessary features (such as Fwd Header Length), as well as any illegal IoT flow records. These included records with null or missed values, as well as any fields that contained a character. Because of this, the CIC-IDS2017 [23] dataset was reduced by 2,867 records, while the CIC-IDS2018 dataset was reduced by 782,296 entries. Second, the qualitative characteristics are given a quantitative measure by way of one-hot encoding, which is an operation that is performed on the features. Third, according to the definitions found in [17], both datasets exhibit considerable imbalance, which can on occasion lead to an increased loss. The concept described served as the basis for the mapping of the original labels into a new traffic labeling that mixes labels that are functionally identical. Samples are then normalized by scaling each of the variables in a consistent manner, which was the fourth step. In the fourth step, each dataset was partitioned into three primary groups, and the partitioning was carried out in a stratification manner in order to preserve the initial distribution of observations across the various classes. As a consequence of this, sixty percent of the data was used for training data, twenty percent of the data was utilized for validation, and twenty percent of the data was utilized for the test.

L. Evaluation measures

The following sets of assessment measures will be used in this investigation so that the performance of S2T-Net could be estimated.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \quad (19)$$

$$Precision = \frac{TP}{TP + FP} \times 100 \quad (20)$$

$$Recall = \frac{TP}{TP + FN} \times 100 \quad (21)$$

$$F1 - measure = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (22)$$

where the amount of positive traffic examples that were incorrectly identified as negative is denoted by the False Negative (FN) value. This value is sometimes referred to as the Type II error. The proportion of negative traffic observations that are incorrectly detected as positive is referred to as the False Positive (FP), which is also referred to as the Type I error. The count of positive traffic observations that have been accurately identified as positive is referred to as the True Positive (TP). The amount of traffic observations that were correctly identified as having a negative outcome is referred to as the True Negative (TN).

M. Working environment

The experiment of this work is implemented in a Python 3.9 environment, in which the model building is coded with TensorFlow API. This environment is installed on a computer workstation operated by Windows 10 OS tailored to 64- the bus system. The workstation comes with, 32 gigabytes of RAM (32 core) Intel Xeon E5-2667 CPU@ 2.9 GHz, and NVIDIA RTX 2080 GPU.

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

5. Experimental Analysis

N. Quantitative Results

In the following subsection, we will explain the results that the suggested S2T-Net was able to achieve in two different scenarios, namely the binary-class situation and the multiclass situation. IoT traffic is categorized as either normal or abnormal in the classification algorithm, whereas in the latter scenario, the S2T-Net classifies IoT traffic observations into seven different classes. Specifically, in binary classification, the IoT traffic is categorized as either normal or abnormal.

Tables I and II, correspondingly, demonstrate the confusion matrices generated by the S2T-Net when applied to the CIC-IDS2017 and CIC-IDS2018 datasets, in both, for the binary class situation. Even though there was a significant disparity between the number of benign examples and the number of attack examples, the S2T-Net was still able to achieve a high efficiency of greater than 99%. This was something that was observable.

Table III contains a tabular version of the confusion matrix that S2T-Net generated using the CIC-IDS2017 for the multi-class situation. In addition to this, the confusion matrix of S2T-Net as measured by CIC-IDS2018 can be seen in Table IV. The proposed S2T-Net is still demonstrating excellent results, in spite of the large disparities in the total amount of samples for each of these categories. In addition to this, it is obvious that there is a pretty high level of misunderstanding between DoS data and DDoS data, which results in a remarkably low F1 measure.

O. Comparative Analysis

Table 1: Binary confusion matrix of S2T-Net over a test set of CIC-IDS2017.

		Estimated			
Actual		Normal	Abnormal	Recall (%)	F1-score (%)
	Normal	453126	1009	99.77	99.7
	Abnormal	1633	109807	98.53	98.81
	Precision (%)	99.64	99.08	-	-

Table 2: Binary confusion matrix of S2T-Net over a test set of CIC-IDS2018.

		Estimated			
Actual		Normal	Abnormal	Recall (%)	F1-score (%)
	Normal	2538929	614	99.97	99.97
	Abnormal	536	550059	99.90	99.89
	Precision (%)	99.97	99.88	-	-

Table 3: Multi-class confusion matrix of S2T-Net over a test set of CIC-IDS2017.

		Estimated								
Actual		Benign	Bot	DoS	Infiltration	PortScan	Web Attack	Brute Force	Recall (%)	F1-score (%)
	Benign	453485	3	249	0	328	2	68	99.86	99.85
	Bot	2	227	51	1	67	8	29	58.96	71.38
	DoS	395	9	75407	3	153	19	118	99.08	99.24
	Infiltration	0	0	0	199	1	0	0	99.50	98.51
	PortScan	290	5	150	1	31250	25	35	98.41	98.33
	Web Attack	3	3	1	0	3	423	0	97.69	92.26
	Brute Force	10	4	10	0	3	7	2718	98.76	95.03
	Precision (%)	99.85	90.44	99.39	97.55	98.25	87.40	91.58	-	-

Table 4: Multi-class confusion matrix of S2T-Net over a test set of CIC-IDS2018.

		Estimated								
Actual		Benign	Bot	DoS	Infiltration	DDoS	Web Attack	Brute Force	Recall (%)	F1-score (%)
	Benign	2528086	1288	2727	903	2906	1761	1872	99.55	99.55
	Bot	2111	54262	137	198	333	158	39	94.80	95.54
	DoS	1964	198	115926	135	5002	145	138	93.86	93.82
	Infiltration	1812	89	108	30009	162	2	204	92.66	93.22
	DDoS	1807	229	4213	167	254477	93	102	97.47	97.04
	Web Attack	1834	157	307	254	278	35260	599	91.14	92.33
	Brute Force	1901	134	194	334	214	273	34636	91.91	92.02
	Precision (%)	99.55	96.28	93.78	93.78	96.62	93.55	92.14	-	-

In this section, we evaluate the proposed S2T-Net against recent state-of-the-art methods (namely SS-Deep-ID [17], Deep-IFS [12], HS-TCN [13], LNN [11], LCVAE[24]), under binary classification, on the two datasets as detailed in Table V and Table VI, respectively. The results can be easily demonstrated that all methods are achieving robust performance in classifying normal traffic from abnormal ones. Table VII and Table VIIII

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

tabulate comparable results of the multi-class scenario on both CIC-IDS2017 and CIC-IDS2018 respectively. The results demonstrate the ability of the proposed framework to model and discriminate between different attacks. Results from previous deep networks used as benchmarks were consistent with those from the corresponding studies. This means that the experiments prove that our implementation is correct and that the ML algorithms have been adequately trained. The P-values for the accuracy metrics on both datasets for semi-supervised techniques following a Paired Student's t-test are displayed in Table I*x* to help us understand the statistical significance of the observed inconsistencies between the evaluation metrics. P-values were calculated with the help of SciPy Library [25], a free and open-source set of scientific tools written in Python. When the P-value is less than 0.05, there is a statistically significant variation in the findings between the models; else, there is no difference. The fact that the P-values for the competing methods are less than 0.05 demonstrates that the suggested S2T-Net significantly surpasses the other competitive methodologies of accuracy. A further t-test, a paired t-test, was run to evaluate the significance of the F1-measure findings on the two datasets. P-values are calculated to show how the proposed model compares statistically to other approaches.

P. Computational analysis

In this part, we contrast the model's learning raining time and inference time because of the time-sensitive nature of cellular IoT applications and the limited resources seen in IoT contexts (for a batch of 1,000,000 records). Due to the S2T-placement Nets in the edge node of cellular IoT, we test its test time on a Raspberry Pi 4 as an edge node. Table *xi* depicts the timings of the various methods with respect to both data sets. On both datasets, ANN has the shortest training time among supervised methods. On the contrary, the competing methods all have extremely lengthy training times. Due to the high cost of computing incurred during the unsupervised pre-training phase, semi-supervised techniques require more time to finish the training task. The suggested S2T-Net trained in significantly less time than the competition. It turns out that inference time exhibits the same sort of behavior. More so, running the same trials again on the edge node reveals significant improvements for all models, but the suggested S2T-Net consistently achieves the lowest calculation time on both datasets. This can be explained by a combination of variables, the first of which is that the stacked Transformer modules allow for a larger receptive field to be achieved with a relatively small number of layers, without sacrificing computing efficiency. Consequently, this leads to a reduction in the total number of parameters that can be trained, resulting in an architecture that is both simple and capable of rapid improvement. Training is made easier by the gradient flow, and computation is sped up thanks to BN layers and residual connection. Additionally, the training process can be sped up by dividing the unlabeled data into multiple

Table 5: The comparative results on binary-class CIC-IDS2017

Study	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
LNN [11]	96.57	98.01	94.93	96.45
SS-Deep-ID [17]	97.46	97.51	95.92	96.71
LCVAE[24]	96.80	99.10	98.48	98.79
Deep-IFS [12]	98.32	98.29	95.07	96.65
S2T-Net	99.53	99.16	99.37	99.26

Table 6: The comparative results on binary-class CIC-IDS2018

Study	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
LNN [11]	99.54	98.34	98.90	98.62
SS-Deep-ID [17]	96.90	95.33	98.36	96.82
LCVAE[24]	98.66	98.39	99.86	99.12
Deep-IFS [12]	96.39	96.82	95.90	96.36
S2T-Net	99.96	99.94	99.93	99.94

Table 7: The comparative results on multi-class CIC-IDS2017

Study	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
LNN [11]	96.95	90.51	90.84	90.67
SS-Deep-ID [17]	96.63	91.45	92.27	91.85
LCVAE[24]	95.74	94.30	88.65	91.39
Deep-IFS [12]	97.98	92.18	91.94	92.06
S2T-Net	99.64	94.92	93.18	93.52

Table 8: The comparative results on multi-class CIC-IDS2018

Study	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
LNN [11]	94.49	91.29	90.16	90.72
SS-Deep-ID [17]	94.15	91.40	92.37	91.88
LCVAE[24]	94.20	93.76	90.49	92.10
Deep-IFS [12]	93.80	90.46	91.82	91.14
S2T-Net	98.79	95.10	94.48	94.79

Table 9: numerical results obtained from the significance test.

	CIC-IDS2017		CIC-IDS2018	
Study	**Accuracy (%)**	**F1-score (%)**	**Accuracy (%)**	**F1-score (%)**
LNN [11]	8.23E-02	3.82E-03	7.93E-03	6.34E-04
SS-Deep-ID [17]	3.76E-02	5.23E-05	6.07E-03	8.91E-03
LCVAE[24]	8.69E-03	7.04E-03	8.05E-03	4.66E-07
Deep-IFS [12]	7.48E-03	1.10E-09	9.13E-03	8.49E-04

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

portions. After trying out a variety of training epoch counts, we found that the suggested model converges quickly when the training epoch count is 10, while other semi-supervised methods require up to 50 epochs. Following the preceding sections, it can be concluded that the suggested S2T-Net is fast and accurate enough to be used in an IoT setting. It might be readily taught on a cloud server that aggregates and prepares for training a huge volume of tagged and unlabeled IoT traffic records. If you want to boost real-time intrusion detections in various cellular IoT parties, you can confidently deploy the trained variant of S2T-Net on cloud or edge nodes.

6. Conclusion and Future Directions

In this work, we introduce S2T-Net, a deep-learning framework for anomaly detection in cellular IoT data logs. To increase the network's capacity to detect outliers in cellular traffic flows, S2T-Net incorporates a transformer module to compensate for the shortcomings of traditional convolutions. To further aid the suggested S2T-Net in its training, a sophisticated attention module is presented. In addition, the S2T-Net model is trained using both labeled and non-labeled IoT traffic data in a semi-supervised fashion. The operational efficiency and general efficiency of the suggested S2T-Net have been confirmed by extensive evaluation. With these enhancements, the S2T-Net is ready for deployment in a wide range of cellular IoT scenarios.

This research paves the path for a number of developments down the road. First, an adaptable DL technique for identifying intrusions is needed because of the randomness of IoT time-series traffic. As a result, while offline methods may be employed for initial model installation, there need to be methods for the DL model to adapt over time to accommodate for both anticipated and unanticipated shifts in the dispersion of IoT traffic without requiring full retraining. Second, true detection of anomalies is crucial since providing instantaneous decisions is a need for the majority of IoT systems (like smart transportation or smart industry). If the time spent processing a series of IoT traffic is greater than the time between incoming traffics, the intrusion detectors risk causing a catastrophic failure in the operating platform. It is important to explain and confirm the judgments made by ML and DL techniques, which brings us to our third point: research into the interpretability of these conclusions. However, more research into the understandability of DL-based IDS is still warranted. Finally, given that no single approach is guaranteed to succeed in every situation, enhancing the performance of DL-based anomaly detection solutions by comparing them to multiple datasets will provide useful insights for enhanced decision-making in a multi-domain setting, boost model recyclability, and simplify mobilization across a wide range of use cases.

Table 9: The computational time for both datasets

	CIC-IDS2017			**CIC-IDS2018**		
	Cloud		Edge	**Cloud**		Edge
Study	Train time (s)	Test time (s)	Test time (s)	Train time (s)	Test time (s)	Test time (s)
LNN [11]	165.4	23.6	51.4	334.5	26.1	51.3
SS-Deep-ID [17]	74.4	17.4	67.3	218.5	11.1	56.6
LCVAE[24]	30.7	1.4	1.1	296.6	11.8	7.8
Deep-IFS [12]	97.8	3.4	5.0	378.6	5.3	4.5
S2T-Net	341.2	2.6	6.5	1487.7	15.3	10.2

References

[1] D. Breitenbacher, I. Homoliak, Y. L. Aung, Y. Elovici, and N. O. Tippenhauer, "HADES-IoT: A Practical and Effective Host-Based Anomaly Detection System for IoT Devices (Extended Version)," *IEEE Internet Things J.*, 2022, doi: 10.1109/JIOT.2021.3135789.

[2] M. Savic *et al.*, "Deep Learning Anomaly Detection for Cellular IoT with Applications in Smart Logistics," *IEEE Access*, 2021, doi: 10.1109/ACCESS.2021.3072916.

[3] A. A. Cook, G. Misirli, and Z. Fan, "Anomaly Detection for IoT Time-Series Data: A Survey," *IEEE Internet of Things Journal*. 2020, doi: 10.1109/JIOT.2019.2958185.

[4] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, "Anomaly Detection Based on Convolutional Recurrent Autoencoder for IoT Time Series," *IEEE Trans. Syst. Man, Cybern. Syst.*, 2022, doi: 10.1109/TSMC.2020.2968516.

[5] L. Cui *et al.*, "Security and Privacy-Enhanced Federated Learning for Anomaly Detection in IoT

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022

Infrastructures," *IEEE Trans. Ind. Informatics*, 2022, doi: 10.1109/TII.2021.3107783.
[6] R. Li, Q. Li, J. Zhou, and Y. Jiang, "ADRIoT: An Edge-Assisted Anomaly Detection Framework Against IoT-Based Network Attacks," *IEEE Internet Things J.*, 2022, doi: 10.1109/JIOT.2021.3122148.
[7] Y. An, F. R. Yu, J. Li, J. Chen, and V. C. M. Leung, "Edge Intelligence (EI)-Enabled HTTP Anomaly Detection Framework for the Internet of Things (IoT)," *IEEE Internet Things J.*, 2021, doi: 10.1109/JIOT.2020.3024645.
[8] T. B. Dang, D. T. Le, T. D. Nguyen, M. Kim, and H. Choo, "Monotone Split and Conquer for Anomaly Detection in IoT Sensory Data," *IEEE Internet Things J.*, 2021, doi: 10.1109/JIOT.2021.3073705.
[9] Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng, "Learning Graph Structures With Transformer for Multivariate Time-Series Anomaly Detection in IoT," *IEEE Internet Things J.*, 2022, doi: 10.1109/JIOT.2021.3100509.
[10] M. O. Osifeko, G. P. Hancke, and A. M. Abu-Mahfouz, "SurveilNet: A Lightweight Anomaly Detection System for Cooperative IoT Surveillance Networks," *IEEE Sens. J.*, 2021, doi: 10.1109/JSEN.2021.3103016.
[11] R. Zhao *et al.*, "A Novel Intrusion Detection Method Based on Lightweight Neural Network for Internet of Things," *IEEE Internet Things J.*, 2022, doi: 10.1109/JIOT.2021.3119055.
[12] M. Abdel-Basset, V. Chang, H. Hawash, R. K. Chakrabortty, and M. Ryan, "Deep-IFS: Intrusion Detection Approach for Industrial Internet of Things Traffic in Fog Environment," *IEEE Trans. Ind. Informatics*, 2021, doi: 10.1109/TII.2020.3025755.
[13] Y. Cheng, Y. Xu, H. Zhong, and Y. Liu, "Leveraging Semisupervised Hierarchical Stacking Temporal Convolutional Network for Anomaly Detection in IoT Communication," 2021, doi: 10.1109/JIOT.2020.3000771.
[14] Y. Lu *et al.*, "Semi-Supervised Machine Learning Aided Anomaly Detection Method in Cellular Networks," *IEEE Trans. Veh. Technol.*, 2020, doi: 10.1109/TVT.2020.2995160.
[15] Q. Xie, P. Zhang, B. Yu, and J. Choi, "Semisupervised Training of Deep Generative Models for High-Dimensional Anomaly Detection," *IEEE Trans. Neural Networks Learn. Syst.*, 2022, doi: 10.1109/TNNLS.2021.3095150.
[16] K. Kumaran Santhosh, D. P. Dogra, P. P. Roy, and A. Mitra, "Vehicular Trajectory Classification and Traffic Anomaly Detection in Videos Using a Hybrid CNN-VAE Architecture," *IEEE Trans. Intell. Transp. Syst.*, 2022, doi: 10.1109/TITS.2021.3108504.
[17] M. Abdel-Basset, H. Hawash, R. K. Chakrabortty, and M. J. Ryan, "Semi-Supervised Spatiotemporal Deep Learning for Intrusions Detection in IoT Networks," *IEEE Internet Things J.*, 2021, doi: 10.1109/JIOT.2021.3060878.
[18] Y. Guo, T. Ji, Q. Wang, L. Yu, G. Min, and P. Li, "Unsupervised Anomaly Detection in IoT Systems for Smart Cities," *IEEE Trans. Netw. Sci. Eng.*, 2020, doi: 10.1109/TNSE.2020.3027543.
[19] G. Muhammad, M. S. Hossain, and S. Garg, "Stacked Autoencoder-based Intrusion Detection System to Combat Financial Fraudulent," *IEEE Internet Things J.*, 2020, doi: 10.1109/JIOT.2020.3041184.
[20] H. M. Song and H. K. Kim, "Self-Supervised Anomaly Detection for In-Vehicle Network Using Noised Pseudo Normal Data," *IEEE Trans. Veh. Technol.*, 2021, doi: 10.1109/TVT.2021.3051026.
[21] A. Vaswani *et al.*, "Attention is all you need," 2017.
[22] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, "IoT-KEEPER: Detecting Malicious IoT Network Activity Using Online Traffic Analysis at the Edge," *IEEE Trans. Netw. Serv. Manag.*, 2020, doi: 10.1109/TNSM.2020.2966951.
[23] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," 2018, doi: 10.5220/0006639801080116.
[24] X. Xu, J. Li, Y. Yang, and F. Shen, "Toward Effective Intrusion Detection Using Log-Cosh Conditional Variational Autoencoder," *IEEE Internet Things J.*, 2021, doi: 10.1109/JIOT.2020.3034621.
[25] P. Virtanen *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nat. Methods*, 2020, doi: 10.1038/s41592-019-0686-2.

Doi: https://doi.org/10.54216/IJWAC.040106
Received: December 12, 2021 Accepted: January 10, 2022