Cluster Computing (2021) 24:17–35
https://doi.org/10.1007/s10586-020-03153-8

TempoCode-IoT: temporal codebook-based encoding of flow features for intrusion detection in Internet of Things

Abdul Jabbar Siddiqui[1] · Azzedine Boukerche[1]

Received: 18 March 2020 / Revised: 14 June 2020 / Accepted: 8 July 2020 / Published online: 2 September 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract
In the recent years, the Internet of Things has been becoming a vulnerable target of intrusion attacks. As the academia and industry move towards bringing the Internet of Things (IoT) to every sector of our lives, much attention needs to be given to develop advanced Intrusion Detection Systems (IDS) to detect such attacks. In this work, we propose a novel network-based intrusion detection method which learns patterns of benign flows in a temporal codebook. Based on the temporally learnt codebook, we propose a feature representation method to transform the raw flow-based statistical features into more discriminative representations, called *TempoCode-IoT*. We develop an ensemble of machine learning-based classifiers optimized to discriminate the malicious flows from the benign ones, based on the proposed TempoCode-IoT. The effectiveness of the proposed method is empirically evaluated on a state-of-the-art realistic intrusion detection dataset as well as on a real botnet-infected IoT dataset, achieving high accuracies and low false positive rates across a variety of intrusion attacks. Moreover, the proposed method outperforms several state-of-the-art works based on the used datasets, proving the effectiveness of Tempo-Code-IoT over raw flow features, both in terms of accuracies and processing speeds.

Keywords Intrusion detection systems · Denial of service attacks · Botnet attacks detection · Network Management · Internet of things security

1 Introduction

The unprecedented evolution of networks with a growing plethora of connected devices and things are reshaping the landscape of an Internet-of-Things (IoT). Ranging from devices such as indoor or outdoor surveillance cameras, electrical and mechanical appliances, mobile user-worn devices such as smart watches or health monitors, to connected vehicles and vehicular components, industrial systems, and connected smart cities, the IoT landscape is continuously evolving (see Fig. 1).

Due to the increasing diversity of devices, networks and services in an IoT ecosystem, the vulnerabilities of each constituent technology could be agglomerated, giving rise to novel threats and attack vectors [9, 15, 27, 36]. This poses danger not only to the devices but also to life and property. Consider these recent reports for example. A large pool of internet-connected devices were compromised to conduct distributed denial of service (DDoS) attacks on critical networks [22]. Another serious example is of the Mirai botnet-based attack which exploited IoT devices to attack many popular web-based services and platforms that became inaccessible [16].

The targets of such DDoS attacks could include critical infrastructure, banks, healthcare institutions, smart cities and internet of connected vehicles and things [3, 10, 19, 32]. For example, connected vehicles have been shown to be vulnerable to being controlled by a remote malicious attacker who could shut down a moving car or lock/unlock doors [28]. In another worrisome example, smart (and connected) toys were found to have security and privacy flaws that could be exploited by an adversary to maliciously control the toys or cause serious privacy infringements [35].

✉ Abdul Jabbar Siddiqui
aj.siddiqui@uottawa.ca

Azzedine Boukerche
boukerch@site.uottawa.ca

[1] PARADISE Lab, School of Electrical and Computer Engineering, University of Ottawa, Ottawa, Canada

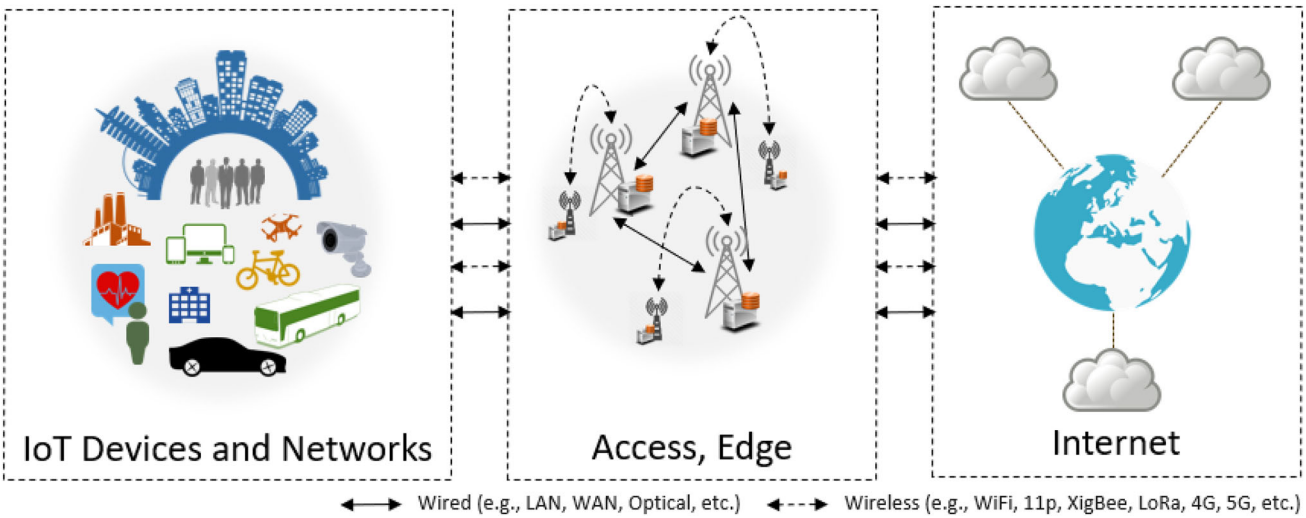


Fig. 1 An architectural overview of the evolving Internet of Things landscape. The diverse range of IoT devices and networks connect to the access points (such as base stations, eNodeBs, RSUs, etc., equipped with edge computing resources) which can communicate amongst themselves and to the larger Internet

In light of these incidents, researchers in academia and industry are gearing efforts to develop novel solutions for intrusion detection in IoT, to secure IoT from different types of intrusions [8, 32, 37, 39]. The various IDSs could be broadly classified in terms of their placement strategy as: centralized, distributed, or hybrid. In the centralized IDS placement strategy, the detection modules or agents are hosted at the network edge devices or the border router (e.g., mobile edge computing servers deployed at eNodeBs, other base stations, or roadside units). On the other hand, in the distributed strategy, the detection modules or agents are hosted at the IoT devices. A mix between the two is the hybrid strategy in which the detection agents or modules are distributed in a hierarchical fashion and hosted at the network edge as well as at the IoT devices. The main advantages of a centralized IDS hosted at the network edge are the availability of richer computing, communication and storage resources, leveraging the advances in edge computing, as well as the ability to efficiently detect attacks originating externally (e.g., via the Internet).

In order to address the problem of intrusion attacks in IoT, this work proposes a novel method to detect intrusions by transforming flow-based features into more discriminative representations and designs an ensemble of classifiers based on these to differentiate between benign and malicious flows. The proposed method is designed to serve in a centralized IDS, leveraging the compute and storage resources therein. The main contributions of this work are summarised as follows:

- We propose a novel flow feature representation, called the *TempoCode-IoT* based on unsupervised learning of a temporal codebook which captures the key patterns in benign traffic over different time windows. The TempoCode-IoT transformation method measures the differences of flow samples from the key patterns in the learnt codebook.
- We study the effect of varying design parameters of TempoCode-IoT on classification scores and processing time, to suggest the optimal set of parameters.
- We develop a machine learning-based ensemble of classifiers and optimize its parameters to learn to discriminate between benign TempoCode-IoT representations and different types of malicious ones.
- We demonstrate the effectiveness of the proposed method in distinguishing benign flows from different types of intrusion attacks through empirical evaluations on two realistic, real-world and recent datasets (CICIDS2017 [34] and NBaIoT [21]).

The remainder of the paper is organised as follows. In Sect. 2, we provide a brief discussion of recent related works, followed by presenting the proposed meth-od in Sect. 3. In Sect. 4, we describe the experimental setup, details and specifications of the datasets used, different attack scenarios considered in this work, impact of design parameters, and the performance metrics to be used for evaluation. After presenting the results and discussions in Sect. 5, we discuss in Sect. 6 some open issues and challenges in the field of IoT security and IDS. The paper finally concludes and outlines future work directions in Sect. 7.

2 Related works

The problem of intrusion detection has been a hot topic over the years, with rising attention due to the evolution of networks into an Internet of Things and the rising threats [2, 5, 17, 40]. In this section, we provide a brief overview of state-of-the-art methods in IDS which are close in concept to our proposed method and highlight the uniqueness in our method.

A multi-layer semi-supervised framework for IDS is proposed in [40] based on pure cluster extraction, pattern discovery, fine-grained classification and model updating. They defined "pure" clusters as those in which a vast majority of their samples belong to the same class. They employed a hierarchical semi-supervised *k*-means algorithm to learn the pure clusters. The samples which did not fall into any pure cluster are then fed into the pattern discovery module in which a clustering-based method is used to find if the patterns are known (normal or intrusions) or unknown patterns. The fine-grained classification module then acts to classify the discovered unknown patterns. However, this step may require manual inspection to determine the class of the unknown patterns. The task of re-training any modules due to changes in traffic distribution is handled by the model updating module. Their experiments were based on the old KDDCUP99 dataset.

A lightweight and rule-based intrusion detection algorithm was proposed in [33] for networks of vehicles. In their method, vehicles aid in evaluation of behavior and reputation of their neighboring vehicles. An ensemble learning-based method is proposed by [24] who proposed statistical flow features and an AdaBoost ensemble comprising of Decision Tree, Naive Bayes, and Artificial Neural Network classifiers.

The authors of [30] proposed a clustered IDS based on Restricted Boltzmann Machine (RBC-IDS) as a deep learning-based solution to monitor critical infrastructures and detect intrusions in wireless sensor networks. Designed to work as a centralised IDS, RBC-IDS groups the sensors into a number of clusters and selects a cluster head in each cluster. The cluster heads are responsible of sending the sensor data (possibly after aggregation) to the central IDS. Their work is based on an old dataset which was not collected from an IoT environment. Moreover, the authors found the Restricted Boltzmann Machine to be slower than traditional machine learning-based methods. We compare (in Sect. 5.4.2) the performance of our method with that of a method which is related (yet more advanced) to [30] and uses deep auto-encoders on a realistic IoT dataset.

Another work addressing the problem of intrusions in a critical infrastructure monitoring application using wireless sensor networks is of [29] in which the authors investigated the development of an IDS based on a Reinforcement Learning (RL) algorithm called Q-Learning. However, their use of Q-learning as the RL algorithm presents some limitations. For example, the Q-tables could grow very large with a large number of states and/or actions. This would imply that with an increase in the number of states, the memory size required to save and update the Q-table would increase. Moreover, an extremely large amount of time would be required to explore each state for building the Q-table.

The work of [3] tackles the issues of intrusions in a connected vehicle cloud environment. It built an IDS comprising of a deep belief network for data reduction and an ID3-based decision tree classifier for classification. In Sect. 5.4 (Table 11) we show that the method proposed in this paper outperforms DBN-based methods and an ID3-based method.

In a more general traffic classification context, the work of [41] introduced a Bag-of-flows model which groups together correlated flows. For classification, they aggregate the correlated predictions of Naive Bayes classifiers. Their work differs from ours in various aspects, most important one being that unlike our proposed method, they do not transform the flow features but simply discretize them in an objective to enhance classification accuracies.

In [2], the authors utilize feature quantization based on clusters in a Self-Organised Map (SOM) network. The SOM network is learnt first and then clusters of neurons are constructed through hierarchical agglomerative clustering. They assumed that the largest SOM cluster would represent benign traffic. So, if a test sample falls into this cluster, it is regarded as a benign one. However, unlike them, we don't consider the class of closest cluster to be the class of a sample (benign or malicious), but rather we take into account the similarities or distances to each cluster center (codeword), collecting as features a set of deviations of each training/testing flow from the cluster centers (codewords) which are temporally learnt.

Another work leveraging clustering of flows is that of [5]. However, their method is different from ours in the following aspects. Their clustering is based on destination IPv4 address prefixes, whereas we do not consider IP addresses in our method in order to tackle the attackers' ability to spoof and dynamically change IPs. Moreover, their feature transformation is based on approximating probability densities of individual flow-based statistical features. For this, they use the estimated probability density function of the respective feature in the closest cluster to which a sample flow belongs. In contrast to their approach, our feature transformation is based on capturing the flow features' distances from each of the different cluster centers.

Another clustering-based intrusion detection method called CANN was proposed by [18]. In their work, they transform a flow's features into a sum of distances of the flow features from the cluster centers and from their *k*-nearest neighbors correspondingly, giving a one dimensional feature to be used by a *k*NN classifier. Although CANN results in a very low dimension feature space, its performance in terms of classification scores was limited compared to the performance of non-transformed flow features with other classifiers such as Support Vector Machines (SVM).

A recent study by [17] have tackled the problem of botnet identification through a clustering-based technique. In their method, they dynamically select subset of features expected to be more discriminative for the respective type of botnet. The clustering is performed on benign flows and malicious flows yielding the cluster centers as fingerprints of each application type (benign or botnet). They employ a similarity function that computes a distance-score of a sample flow from the cluster centers. Based on a closeness threshold, the class of the nearest cluster center is decided as a prediction of the sample's class. Their method is very similar to that of [42] who also employed a learnt codebook.

Given a sample flow's features, [42]'s method uses only the deviation from the closest cluster center and classifies it as anomalous if the deviation is greater than a threshold. However, this threshold has to be experimentally determined and may not be applicable to all types of attacks. Moreover, to evade detection, attackers could be mimicking benign flows by dynamically changing their flow features such that these may be within the threshold distance to atleast one of the benign clusters. In these cases, methods such as those of [17, 42] would suffer from decreased True Positive Rates. Hence, in our work, instead of dynamic feature subset selection, we transform the flow-based features into TempoCode-IoT representations which capture the deviation of each flow's features from benign fingerprints' or codewords' features and then leverage a machine learning-based algorithm to discriminate between the benign and malicious TempoCode-IoT representations.

3 Proposed method

In a quest to overcome some of the limitations in prior works discussed above, the focus of this paper is on designing a novel method to transform flow-based features into more discriminative representations. In this regard, this paper proposes the *TempoCode-IoT*, a temporal codebook-based encoding of flow features method which captures the key patterns of benign flow features in an unsupervised, temporally learnt discriminative codebook. The proposed method builds upon enhancing the Bag-of-Features (BoF) model [13] in the context of network flow features-based IDS (BoF is alternatively referred to as Vector Quantization (VQ) in some works). The BoF model has been proven effective in other domains such as image classification, object recognition [26], etc.

Based on the temporally learnt codebook, the flow features are transformed into TempoCode-IoT representations which are then used to train an ensemble of SVM classifiers. Since the next generation IoT will embrace a heterogeneity of devices which will use a diversity of protocols and standards [32], our goal in this work is to build upon flow features that are not dependent on a specific protocol. Moreover, devices in IoT may exhibit time-varying behavior in terms of the traffic flows they generate. For example, at certain times of the day, the sensors or devices may be exchanging data at a higher rate than at other times. Similarly, in certain situations such as upon detection of a specific event, vision or acoustic sensors may be triggered to exchange data at a higher rate. Considering such characteristics of IoT flows, we design a temporally learnt codebook which captures the key patterns in benign traffic over different time windows. The following subsections describe the steps in more detail. An overview of the proposed method's pipeline is depicted in Fig. 2. The symbols and notations used in this paper are summarized in Table 1.

The main steps for intrusion detection based on the proposed method are: (1) Flow-based features extraction, (2) Codebook Learning, (3) TempoCode-IoT Generation, and (4) Classifier Ensemble Training/Testing. The subsections below shall elaborate on these steps.

3.1 Flow-based features extraction

As depicted in Fig. 2, the first step is to extract the flow-based features from the packets. To evaluate the proposed TempoCode-IoT feature transformation method, in this work, we use different sets of features as collected in two recent datasets: CICIDS2017 and NBaIoT. These features include Flow Duration, Number of Packets in forward direction, Flow Inter-arrival time, as well as their statistical summaries such as mean and standard deviation (stdev) over certain intervals of time.

To allow for a fair comparison of our method with related works based on the selected datasets, our experimental evaluations on each dataset utilize the flow features contained in the respective dataset. The selected flow features have been popularly used in intrusion detection works. Using these as raw features, we demonstrate the benefits of the proposed TempoCode-IoT transformed feature representations over the raw features

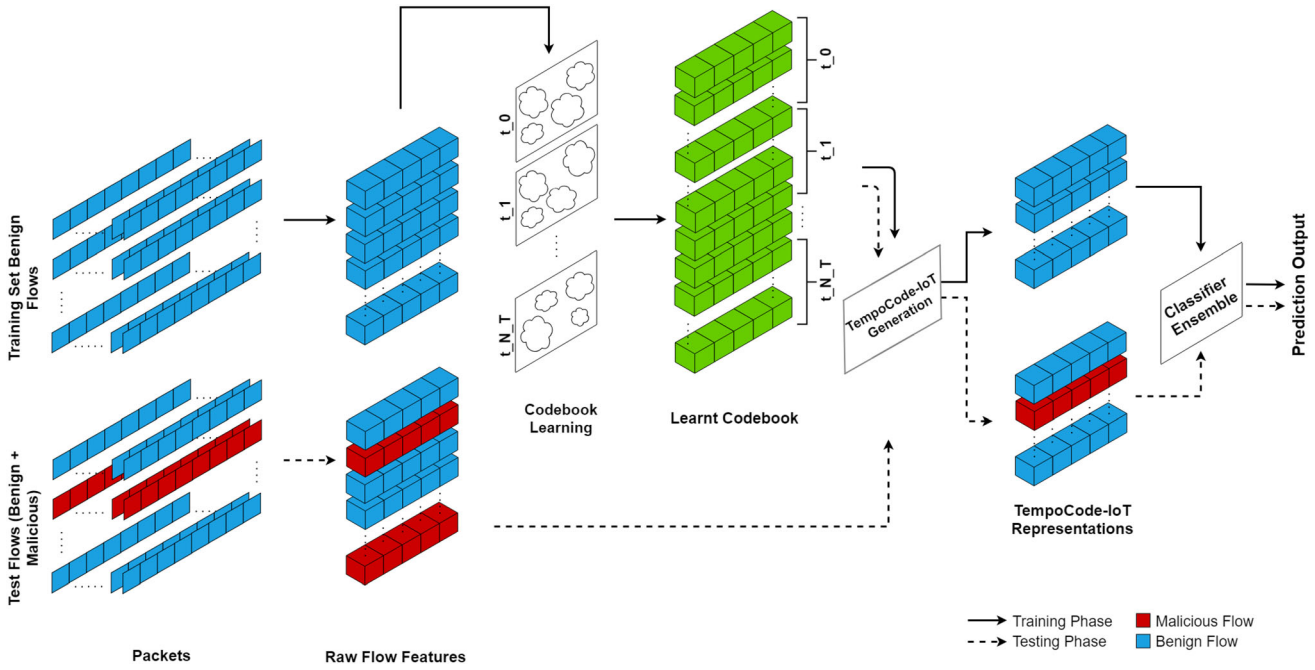


Fig. 2 An overview of the TempoCode-IoT-based intrusion detection pipeline

Table 1 Symbols and notations

Symbol	Description
F_j	Set of features for j-th sample in dataset
N_F	Total number of features, i.e. dimensionality of F_j
f_{nj}	The n-th feature if F_j
t_{dur}	Size of time window for temporal codebook learning
N_T	Total number of time-windows
t_i	Time window i, where $i = 0, \ldots, N_{T-1}$
N_{ct}	Number of benign key patterns to learn from t_i
$\mathbf{C_T}$	The temporal codebook
$cw_{t_i,k}$	k-th key pattern (codeword) in $\mathbf{C_T}$ corresponding to time window t_i
$CSize$	Temporal codebook size $= N_T \cdot N_{ct}$
TC_j	TempoCode-IoT representation of F_j
$q_{t_i,k}$	Distance between $cw_{t_i,k}$ and F_j

The motivation to choose the mentioned datasets with different sets of features stems from the following reasons: (i) recent related works on intrusion detection have utilized these datasets, (ii) to establish the effectiveness of the proposed TempoCode-IoT feature transformation method with different sets of features, (iii) to demonstrate the effectiveness of proposed TempoCode-IoT method for IoT as well as other networking environments. Below, we provide an overview of the features in both datasets, while we give a deeper description of these datasets in Sect. 4.1.

Features in CICIDS2017 Datasets: The following flow-based features of CICIDS2017 datasets were selected and extracted (over every time interval of 1 s) using the CIC-FlowMeter tool [14], based on the findings of [34]:

- Flow Duration
- Packet Length (min, mean, stdev in forward and backward direction)
- Subflow Bytes (in forward direction)
- Flow Inter-Arrival Time (min, mean, stdev in forward and backward direction)
- Active_min, Active_mean

- Init-Win-Bytes (forward and backward),
- Flag Counts (ACK, PSH, SYN)
- Number of packets and bytes per second (forward and backward direction)

Features in NBaIoT Datasets The NBaIoT data-sets [21, 23] contain similar features based on packets and flows (further details can be found in [21]). The below set of features are extracted over different damped time-window sizes, yielding a set of 115 features in total:

- Packet Size (mean and variance; outbound direction)
- Packet Count
- Packet Jitter (mean, variance and count of packet inter-arrival times)
- Packet Size (magnitude, radius, covariance, correlation coefficient; inbound and outbound

From each j-th record in a dataset, the set of features F_j, is represented as in Eq. 1 below (where $n = N_F$ represents number of features). For example, f_{1j} could represent flow duration of j-th flow sample over a time window.

$$F_j = \{f_{1j}, f_{2j}, \ldots, f_{nj}\} \tag{1}$$

3.2 Temporal codebook learning

The proposed method of transforming flow features into TempoCode-IoT representations first involves capturing key patterns of benign flows over different time windows in a temporal codebook which is learnt in an unsupervised manner. Below, we first describe the temporal codebook learning procedure, followed by the TempoCode-IoT transformation method.

The benign flow features are grouped according to successive (non-overlapping) time windows determined by t_{dur} (e.g., if $t_{dur} = 1$, its an hourly window). The total number of time windows is given by N_T which would depend on the training dataset and t_{dur}. For example, in the case of hourly windows, $t_{dur} = 1.0$, and a training dataset of 10 h, then $N_T = 10/1.0 = 10$. The time windows are denoted by indices t_i (where $i = 0, \ldots, N_{T-1}$).

In each time window t_i, a clustering method such as K-Means is applied to learn N_{ct} key patterns as codewords to represent the benign traffic in t_i. In the K-Means clustering method, the initial set of cluster centres is chosen randomly. Nearest neighbours for each cluster centre are found from the data points. In cases where a data point is found to be close to more than one cluster centre, it gets assigned to the closest one only. In each cluster, an average of its member data points is calculated and set as a new cluster centre. With the new cluster centres, each data point is re-grouped to the (new) centre closest to it. This process of updating cluster centres followed by re-grouping of data points based on the updated centres is repeated a number of times to ensure stability of the clusters. The cluster centres (from each time window) are stored as codewords $cw_{t_i,k}$. In this way, we obtain the temporal codebook $\mathbf{C_T}$, as expressed below, where $\mathbf{C_T} \in \mathbb{R}^{N_F}$

$$\mathbf{C_T} = \{cw_{t_i,k} | i = 0, \ldots, N_T; k = 1, \ldots, N_{ct}\} \tag{2}$$

The temporal codebook size $CSize$ can then be given by $N_T \cdot N_{ct}$, considering that we use a fixed N_{ct} in all time windows. The procedure to learn $\mathbf{C_T}$ is given in Algorithm 1.

Algorithm 1 Temporal Codebook Learning

Input: $\mathbf{F} = \{F_j | j = 0, ..., N_s\}$, $CSize$ (Size of $\mathbf{C_T}$), t_{dur} (Time window), N_s is the total # of training samples, N_{s_i} (Number of samples in time window i)
Output: $\mathbf{C_T}$

Initialisation :
$\mathbf{C_T} = \{\}$
Grouping $\mathbf{F}$ *by time-window* t_{dur}
$\mathbf{F_T} = \{\}$
for $i = 0$ to N_T **do**
  $\mathbf{F_i} = \{\}$
  for each F_j in time window t_i, $j = 0$ to N_{s_i} **do**
    $\mathbf{F_i} = \mathbf{F_i} \cup F_j$
  end for
  $\mathbf{F_T} = \mathbf{F_T}.append(i, \mathbf{F_i})$
end for
Learning TempoCode-IoT Codebook
$\mathbf{C_t} = \{\}$
for each $\mathbf{F_i}$ in $\mathbf{F_T}$, $i = 0$ to N_T **do**
  $\mathbf{C_t} = Cluster(\mathbf{F_i}, N_{ct})$
  $\mathbf{C_T} = \mathbf{C_T}.append(i, \mathbf{C_t})$
end for
return $\mathbf{C_T}$

3.3 TempoCode-IoT generation: temporal codebook-based encoding of flow features

Leveraging the temporal codebook $\mathbf{C_T}$ learnt above, the flow features are transformed into TempoCode-IoT representations based on their distances from the learnt benign patterns across different time windows. The procedure to generate TempoCode-IoT representations for training and testing dataset samples is shown in Algorithm 2. The raw features of a flow, F_j are compared to all the codewords of $\mathbf{C_T}$, where the deviations are captured to form the TempoCode-IoT representation, summarised in the equation below:

$$TC_j = \{q_{t_i,k} | i = 0, \ldots, N_T; k = 1, \ldots, N_{ct}\} \tag{3}$$

where a bin $(q_{t_i,k})$ holds the distance of F_j from $cw_{t_i,k}$. In this manner, a TempoCode-IoT representation is a set of distances of a raw feature vector to each of the codewords

in $\mathbf{C_T}$, organised in a time-ordered fashion. The TempoCode-IoT representation is computed as follows, where $dist(.,.)$ is a distance metric such as euclidean distance:

$$q_{t_i,k} = dist(cw_{t_i,k}, F_j) \tag{4}$$

Algorithm 2 TempoCode-IoT Generation

Input: $\mathbf{F} = \{F_j | j = 0, ..., N_s\}$ (Training or testing samples), N_F (Dimensionality of F_j), $\mathbf{C_T}$,N_{ct} , N_T
Output: $\mathbf{F_T}$ (the TempoCode-IoT representations)
Initialisation :
$\mathbf{F_T} = \{\}$
TempoCode-IoT Transformation :
for each flow sample F_j in $\mathbf{F}$ **do**
  $F_{T_j} = []$ *(the TempoCode-IoT representation of* F_j*)*
  for each time window t_i, $i = 0, ..., (N_T - 1)$ **do**
    for each $cw_{t_i,k}$ in $\mathbf{C_T}$, $k = 0, ..., (N_{ct} - 1)$ **do**
      $F_{T_j}[k + i \cdot N_{ct}] = dist(cw_{t_i,k}, F_j)$
    end for
    $\mathbf{F_T} = \mathbf{F_T}.append(F_{T_j})$
  end for
end for
return $\mathbf{F_T}$

3.4 Classifier ensemble training

The TempoCode-IoT representations obtained from previous steps are then used to train an ensemble of machine learning-based classifiers (see Fig. 2) such as Support Vector Machines (SVM) [12, 38] using the *scikit-learn* machine learning library [31]. The codebook, and the respectively generated TempoCode-IoT representations are expected to be discriminative enough in the feature space to aid the classifier in learning the differences.

The SVM is originally a binary classifier that has proved its effectiveness in many classification problems. It learns the support vectors by leveraging kernel functions such as Radial Basis Functions (RBF). The support vectors are basically a subset of the training Tempo-Code-IoT samples that represent the best separation between two classes. A test TempoCode-IoT sample is classified based on its distance from these support vectors. A single multi-class SVM classifier is built by collecting many such binary classifiers, depending on the number of classes in the dataset. We determine the optimal parameters for SVM through extensive cross-validation experiments, owing to the imbalanced nature of the dataset used.

In this work, we designed an ensemble of multi-class SVM classifiers, a method based on Bagging [11]. Each classifier in this ensemble is trained on a random subset of the training dataset. The prediction of each constituent classifier is then combined through voting to produce the overall classification output.

We choose SVMs as the base classifiers motivated by their proven generalization ability, robustness, and success in achieving globally optimal solutions [6]. As such, they have found popular use in diverse applications such as multimedia analysis, object detection and recognition, medical image analysis, etc.

The motivation to adopt ensemble learning stems from the following reasons: (i) Training an ensemble of multiple classifiers on smaller subsets of training data could be done in parallel and faster, (ii) An ensemble has better generalization ability when compared to that of the individual learners (classifiers) [43].

In the testing phase, the TempoCode-IoT representations for test samples (which include both benign and malicious ones) are generated using the codebook $\mathbf{C_T}$ learnt above and passed on to the ensemble of classifiers. Then, each of the classifiers adds a vote to its predicted class. The class with the highest votes is assigned as the predicted class (benign, malicious, or a specific attack class) of the test flow sample.

3.5 Computational complexity

The complexity of the proposed methods can be analysed by looking at the complexity of the main steps involved. The Temporal Codebook Learning step is only carried out in training phase. In this work, the clustering process involved in codebook learning employs the K-Means method which has an average complexity of $O(CSize \cdot n \cdot N_{iter} \cdot N_F)$ and worst case complexity of $O(n^{CSize+2/N_F})$ [4], where n is the number of (training) samples, $CSize$ is the codebook size (i.e., total number of cluster centres) and N_{iter} is the number of iterations of the K-Means method.

The TempoCode-IoT Generation step involves using the learnt temporal codebook to transform raw features into their TempoCode-IoT representations mainly by computing the deviations of raw features from the codewords of the codebook. If the complexity of computing the distance of a N_F-dimensional sample from $CSize$ codewords (each of dimensionality N_F) of the codebook $\mathbf{C_T}$ is given by $O(CSize)$, then the complexity of TempoCode-IoT generation step for n (training or testing) samples would be $O(n \cdot CSize)$.

As for the complexity of Classifier Ensemble Training and Testing step, we look at the complexity of SVM classifiers which we employ as base classifiers in this work. The computational complexity of training SVM depends mainly upon the number of support vectors. It involves a quadratic term and a cubic term because the asymptotic number of support vectors (n_{sv}) grows linearly with the number of samples (n). When the parameter C is small, n_{sv} grows atleast like n^2 and when C becomes large, it grows atleast like n^3 [7]. In testing (prediction), the complexity of

executing each SVM classifier is around $O(n_{sv} \cdot CSize)$ (each support vector has dimensionality $= CSize$).

Thus, the overall complexity of training by the proposed method is non-linear whereas testing (i.e., execution) is linear with respect to the number of (training or testing) samples n.

4 Experimental setup

The diversity of devices in IoT adds to the challenges of developing security solutions. The mix of low-, moderate- and high-traffic generating devices, possibly running on different operating systems and protocols, and the fact that these devices could change their behaviors of exchanging data depending on various conditions, makes it difficult to discriminate between malicious flows from the benign ones.

Moreover, to the best of our knowledge, there is a lack of publicly available datasets for intrusion detection in IoT containing and comprehensively representing real-world traces from a diversity of devices, operating systems, underlying protocols, and diverse attack types [32]. A recent work towards filling this gap is by [21, 23] who collected the NBaIoT datasets out of real IoT devices infected with popular botnets such as Mirai and BASHLITE. Hence, in this work, we use the NBaIoT datasets to evaluate the proposed method. In addition, for further evaluation of our method on a diversity of intrusion attacks, and to compare against state-of-the-art IDS, we have also used the recently published CICIDS2017 realistic intrusion detection data-sets [34].

In what follows, we provide descriptions of the data-sets to elaborate on the reasons of choosing these (Sect. 4.1). Then, we describe the optimal parameters selection for the codebook learning and TempoCode-IoT classification (Sect. 4.2). The performance of the proposed method is evaluated based on the metrics described in Sect. 4.3. The computing platform utilized to evaluate our proposed method is equipped with an Intel i7 CPU, 4 cores, 16 GB RAM, CPU clock rate 1.8 GHz.

4.1 Datasets description

4.1.1 CICIDS2017 datasets

The CICIDS2017 is a recently published collection of realistic intrusion detection datasets that has been collected to overcome the challenges and limitations of other popularly used IDS datasets that have been proposed prior to it. Since our objective is to detect various kinds of intrusion attacks, we were interested in this dataset for having realistic traces representing a variety of attacks. The CICIDS2017 dataset built by researchers from the Canadian Institute of Cybersecurity (University of New Brunswick), has several advantages (over other datasets used in the literature) besides the attack diversity, operating system variety, local and internet communications, etc., as discussed in detail in [34].

The dataset comprises of malicious traffic arising from six diverse attacks types: (i) Brute-force, (ii) Heartbleed, (iii) Botnet, (iv) DoS and DDoS, (vi) Web Attack, and (vii) Infiltration attacks. In Table 2, we provide a brief description of these attacks. As for the benign traffic recorded in the dataset, it is based on a realistic benign profiling system, covering a mix of protocols such as email, SSH, FTP, HTTP, HTTPS over TCP/UDP.

The Table 3 shows the composition of the dataset which was collected over a period of five days (Monday–Friday) during (nine) working hours each day. Different attacks were run on each day, leaving Monday with benign traffic only. As we can see, there is huge imbalance in the dataset due to the low ratio of malicious samples for many attack types. For example, the Friday morning dataset for Bot attacks has only about 0.01% of malicious samples. Hence, to avoid overfitting in training our classifiers, we randomly downsample the benign data so as to retain 50% benign flows and 50% malicious flows in each dataset. Despite downsampling the benign data for training, the results of the proposed methods are encouragingly good in detecting the benign flows (see Sect. 5). In our experiments, we partition the respective datasets into 60% training, 20% validation, and 20% testing sets (represented by #Tr, #Val, and #Te in Table 3, respectively), following a popular approach in classification works [1] We use the traces from Monday (containing benign flows alone) only to learn the codebook $\mathbf{C_T}$.

4.1.2 NBaIoT datasets

The NBaIoT datasets of [21, 23] are the best IoT-related intrusion detection datasets publicly available, to best of our knowledge. The datasets were collected from a real testbed of nine wireless-ly connected IoT devices, setup in a way to resemble an enterprise setting. These devices were infected with two families of real-world IoT-based botnets (Mirai and BASHLITE/Gafgyt). The dataset contains the packet- and flow-based features representing the benign and malicious traffic.

Table 4 provides a summary of the different attacks covered in the NBaIoT dataset under the Mirai and BASHLITE botnet families. The composition of the dataset is given in Table 5. Following the approach of [21], we split the datasets (benign and malicious samples) into equal-sized partitions for training, validation and testing.

Table 2 Attack types in CICIDS2017 datasets

Attack type	Description
Bruteforce	Based on the FTP- and SSH-Patator tools. The attacker tries to gain access to content or documents via a hit and try method
Heartbleed	Targeted against OpenSSL-based Transport Layer Security (TLS) protocol
Botnet	A number of devices are compromised and exploited to carry out different attacks/operations. Ares-based Botnet
DoS/DDoS	Targeted against a network resource or service to make it unavailable for benign users. When many different devices are exploited (e.g. by a botnet), it is called DDoS. Tools used: GoldenEye, Slowloris, Hulk, Slowhttptest, Heartleech, LOIC
Web attack	Attacks like SQL Injection or Cross-Site Scripting (XSS), over the web, exploiting vulnerabilities in code
Infiltration attack	Internally originated attacks. Attacker exploits software vulnerabilities to setup a backdoor on victim devices to carry out various attacks such as portscan or IP sweep, etc. Tool: Metasploit, Nmap, portscan

Table 3 Composition of CICIDS2017 datasets, and our training, validation and testing splits

Dataset	Class	#Samples	Ratio	#Tr	#Val	#Te
Monday	Benign	529919	1.0	529919	–	–
Tuesday	Benign	431813	0.969	259088	86362	86363
	FTP-patator	7935	0.018	4761	1587	1587
	SSH-patator	5897	0.013	3539	1179	1179
Wednesday	Benign	439683	0.636	263810	87937	87936
	DoS GoldenEye	10293	0.015	6176	2059	2058
	DoS Hulk	230124	0.333	138074	46025	46025
	DoS Slowhhtptest	5499	0.008	3299	1100	1100
	DoS Slowloris	5796	0.008	3478	1159	1159
	Heartbleed	11	0.000016	7	2	2
Thursday morning	Benign	168051	0.988	100831	33610	33610
	Brute force	1507	0.009	905	301	301
	SQL Injection	21	0.0001	13	4	4
	XSS	652	0.004	391	131	130
Thursday afternoon	Benign	288359	0.9999	173015	57672	57672
	Infiltration	36	0.0001	22	7	7
Friday morning	Benign	188955	0.99	113372	37791	37792
	Bot	1956	0.01	1174	391	391
Friday afternoon 1	Benign	127292	0.445	7676	2558	2558
	PortScan	158804	0.555	95283	31761	31760
Friday afternoon 2	Benign	97686	0.433	58612	19537	19537
	DDoS	128025	0.567	76815	25605	25605

Table 4 Attack types in NBaIoT dataset

Attack type	Botnet family	Description
Scan	Mirai and BASHLITE (Gafgyt)	Looks for vulnerable devices in the network
Junk	BASHLITE (Gafgyt)	Sends junk/spam data
COMBO	BASHLITE (Gafgyt)	Sends spam data; Opens connection to a given IP address and port
Flooding	Mirai	ACK, SYN, UDP, UDPplain (higher PPS, enabled by fewer options)
	BASHLITE (Gafgyt)	UDP, TCP

Table 5 Composition of NBaIoT datasets: IoT devices, and number of benign and malicious samples

ID	Device	#Benign	#Mirai	#Gafgyt
1	Danmini (doorbell)	40395	652100	316650
2	Ecobee (thermostat)	13111	512133	310630
3	Ennio (doorbell)	34692	N/A	316400
4	Philips B120N10 (baby monitor)	160137	610714	312723
5	Provision PT737E (security camera)	55169	436010	330096
6	Provision PT838 (security camera)	91555	429337	309040
7	Samsung SNH1011N (Webcam)	46817	N/A	323072
8	SimpleHome XC57-1002-WHT (security camera)	42784	513248	303223
9	SimpleHome XC57-1003-WHT (security camera)	17936	514860	316438

The benign traffic data for each IoT device consists of frequent and infrequent actions as well.

4.2 TempoCode-IoT configurations

The main parameters to configure TempoCode-IoT -based intrusion detection are the t_{dur} (length of time windows) and N_{ct} (number of codewords to learn from each time window). In this work, we utilise the CICIDS2017 datasets (due to the wide diversity of attack types it contains) in a binary classification setting (i.e., binary vs malicious) to study the effect of TempoCode-IoT parameters on the intrusion detection performance.

4.2.1 Effect of t_{dur}

We evaluated TempoCode-IoT-based intrusion detection under varying t_{dur} from 0.25 to 1.0, in steps of 0.25. This respectively corresponds to taking time windows of 15 mins., 30 mins., 45 mins., and 1 h, in the codebook learning phase. Figure 3a shows how correct classification rate (or ACC_i) varies with t_{dur}, while Fig. 3b shows the average processing time (ms) per TempoCode-IoT vector during testing phase.

We note that the best ACC_i is obtained when t_{dur} is 0.25. However, this comes at a cost of higher processing time per TempoCode-IoT vector (around 15.57ms) in comparison to the lower processing times at higher t_{dur}. This is expected because with lower t_{dur}, the codebook size increases (given a fixed N_{ct}), and hence the dimensionality of TempoCode-IoT vectors increases.

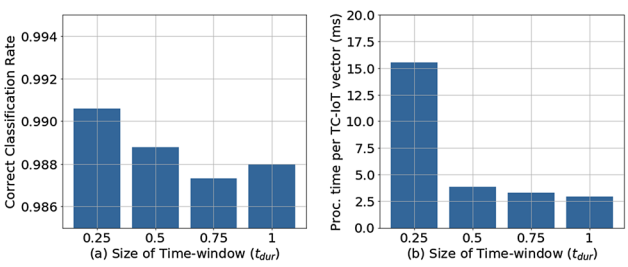


Fig. 3 **a** Effect of t_{dur} on accuracy (correct classification rate of benign and malicious samples) and **b** processing time (per TempoCode-IoT vector)

An interesting observation is when $t_{dur} = 0.75$, the ACC_i is lower than at $t_{dur} = 1.0$, although we expected it to be higher. This indicates that with a time window of 45 mins., for the CICIDS2017 dataset, the learnt temporal codebook doesn't capture the benign patterns as effectively as with a time window of 1 h.

The specific choice of t_{dur} would depend on an application's or system's requirements and constraints, taking into consideration a trade-off between accuracy and processing time. In this work, we choose $t_{dur} = 1$ which yielded the lowest processing time (2.94ms) and a reasonably good ACC_i of around 98.79%.

4.2.2 Effect of N_{ct} and *CSize*

The number of patterns N_{ct} learnt per time window in the codebook C_T plays a role in determining the performance of TempoCode-IoT in terms of accuracy as well as processing time. It is expected that at a higher N_{ct}, since the codebook is more comprehensive than at a lower N_{ct}, a higher accuracy could be achieved.

In Table 6, we present the classification scores of TempoCode-IoT with varying N_{ct} values (i.e., 5, 10, 15, 20, 25). As it can be observed, the precision, recall and F1 scores for both benign and malicious classes increased with increase in N_{ct}. The *CSize* column shows the overall codebook size for each N_{ct}

The best F1 scores for the benign and malicious classes were achieved at $N_{ct} = 15$. Looking at the precision of benign class (Prec_Ben), F1 scores of benign and malicious classes, as well as the recall of malicious class, the scores were better at $N_{ct} = 15$ than at $N_{ct} = 20$.

From Table 6, we also note that a larger codebook (higher *CSize*, as a consequence of a higher N_{ct}) results in higher classification scores. These results are obtained by setting $t_{dur} = 1.0$, which makes $N_T = 9$ non-overlapping

Table 6 Effect of N_{ct} and *CSize* (codebook size) on TempoCode-IoT classification scores

N_{ct}	CSize	Ben-Ben	Ben-Mal	Mal-Ben	Mal-Mal	Prec-Ben	Recall-Ben	F1-Ben	Prec-Mal	Recall-Mal	F1-Mal
5	45	345555	2813	2917	108394	0.9916	0.9920	0.9918	0.9747	0.9738	0.9743
10	90	346495	1873	2577	108734	0.9926	0.9946	0.9936	0.9831	0.9769	0.9800
15	135	346618	1750	**2534**	**108777**	**0.9927**	0.9950	**0.9939**	0.9842	**0.9772**	**0.9807**
20	180	**346762**	**1606**	2794	108517	0.9920	**0.9954**	0.9937	**0.9854**	0.9749	0.9801

The bold values indicate the highest values in the respective columns

time windows (on Monday's benign flows dataset of CICIDS2017), where the $CSize = N_T \cdot N_{ct}$. However, at $N_{ct} = 20$, the F1 scores are lower than those at $N_{ct} = 15$. A reason could be because the codebook becomes redundantly large at $N_{ct} = 20$ for the used dataset, leading to TempoCode-IoT representations which cause classifier confusion or to classifier overfitting, deduced from the observation that at $N_{ct} = 20$, Ben-Ben (*TP*) and Mal-Ben (*FP*) counts are higher than at $N_{ct} = 15$.

4.2.3 Codebook learning time

The time consumed in learning the codebook is an important factor in assessing the adaptability of the proposed method. Application scenarios in which the benign network traffic behavior may evolve or change could require re-training the codebook. Hence, having lower codebook learning times are beneficial for such cases. However, as we have shown above, smaller codebooks could fall short of capturing the benign flow patterns thereby reducing accuracies. In Fig. 4, we show the codebook learning times of TempoCode-IoT for increasing *CSize*. The TempoCode-IoT's codebook size depends on N_{ct}. In this figure, we observed the codebook learning times for $N_{ct} = 1, 4, 8, 10, 15, 20$ corresponding to $CSize = 9, 36, 72, 90, 135, 180$, respectively (Note that the time-window $t_{dur} = 1$ h and the CICIDS2017 dataset has $N_T = 9$ h of data).

In learning the temporal codebook, for each time-window, the clustering process is applied only on the benign samples from that time-window, and not on the entire set of samples over the whole duration of the dataset. This allows for parallel codebook learning, where each time-window's codebook could be learnt in parallel.

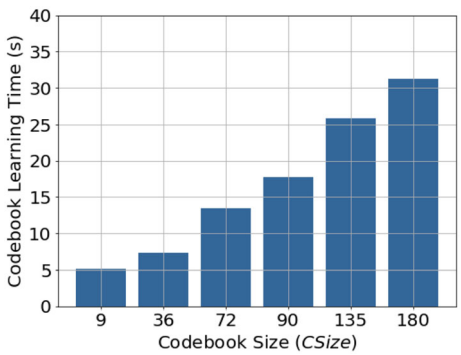


Fig. 4 Codebook learning time for TempoCode-IoT, with varying *CSize*

4.2.4 Classifier configurations

In this work, for TempoCode-IoT-based intrusion detection, we used the following parameters for the ensemble of SVM classifiers. The ensemble size was 10, using a majority voting-based classification output. Based on empirical evaluations, we found that the constituent SVM classifiers performed best with $C = 2000$, $\gamma = 500$, for the CICIDS2017 datasets.

4.3 Performance metrics

We assess the performance of the proposed TempoCode-IoT-based intrusion detection based on the following metrics: Precision, Recall, F1-score, and Class-wise accuracy. In addition, we utilise the confusion matrices as a tool to interpret and understand the performance in terms of these scores.

- Precision: For a class i, its Precision score P_i is:

$$P_i = \frac{TP_i}{TP_i + FP_i} \tag{5}$$

- Recall: For a class i, its Recall score R_i is:

$$R_i = \frac{TP_i}{TP_i + FN_i} \tag{6}$$

- F1-Score: a harmonic average of Precision and Recall scores

$$F1 - Score_i = \frac{2 \cdot P_i \cdot R_i}{P_i + R_i} \tag{7}$$

- Class-wise Accuracy (or Correct Classification Rate): A measure of how many samples were correctly classified (TP_i) as benign or corresponding attack class respectively, with respect to the total number of test samples of the class.

$$ACC_i = \frac{TP_i}{\#TestSamples_i} \quad (8)$$

– Confusion Matrix: Depicts the percentage of test samples of each ground-truth class (represented by rows) classified to each class (represented by columns). So, the main diagonal values are the ACC_i and R_i, while the other values indicate the False Positives or False Negatives. In a row i, the value at (i, i) is the ACC_i and R_i; the values in (i, j), where $j \neq i$, show the False Negatives (FN_i), i.e., the percentage of samples of class-i that were mis-classified to class-j.

The definitions of TP, FP, FN are based on the class being considered and the granularity of classification (binary or multi-class). While TP_i refers to the True Positives, FP_i refers to the False Positives, and FN_i refers to the False Negatives with respect to class-i. In a binary classification setting (Sect. 5.2), the two classes are 'Benign' ($i = 0$) and 'Malicious' ($i = 1$). The TPs with respect to benign class (TP_0) are the benign samples which were classified as benign, while the FPs with respect to benign class (FP_0) are the malicious samples which were classified as benign. Similarly, TP_1 and FP_1 refer to the TPs and FPs with respect to the malicious class. In multi-class classification (Sect. 5.1), we consider TP, FP, FN with respect to each class. So, TP_i would be the samples of class-i classified as class-i while TP_j would be samples of class-j classified as class-j. In calculating the ACC_i, P_i, R_i, $F1\text{-}Score_i$ for a class-i, we consider the TP, FP, FN defined with respect to class-i.

5 Results and discussions

To evaluate the performance of the proposed TempoCode-IoT representations for intrusion detection, we use the CICIDS2017 and NBaIoT datasets (described in Sect. 4.1). With the CICIDS2017 datasets, we conduct two sets of experiments. First, in Sect. 5.1, we evaluate TempoCode-IoT representations for detection of individual attack types in the different datasets of CICIDS2017 (as listed in Table 3). Second, in Sect. 5.2, we investigate the effectiveness of TempoCode-IoT representations in a binary classification setting (to differentiate between benign and malicious flows). On the NBaIoT datasets, we evaluate the performance of Tempo-Code-IoT representations in differentiating the benign flows from the malicious flows arising from the compromised IoT devices. Finally, in Sect. 5.4, we compare the performance of our proposed method with results of prior works on the CICIDS2017 and NBaIoT datasets, demonstrating the superiority of our method over state-of-the-art.

5.1 On the CICIDS2017 datasets: attack type classification

In these set of experiments, we evaluate the performance of TempoCode-IoT-based IDS on each dataset of CICIDS2017 [34], i.e., each attack type's dataset. Figures 5, 6, 7, 8 show the confusion matrices for each attack type. The results are summarised in Table 7 which presents the average scores along with the respective 95% confidence intervals from 10 test runs (i.e., ten randomly split test subsets). The temporal codebook $\mathbf{C_T}$ is learnt using Monday's benign training data samples, with size $N_{ct} = 15$, $T_{dur} = 1$, $N_T = 9$, i.e., $CSize = 135$ used in these experiments, based on the findings in Sect. 4.2.

In the case of DoS attacks dataset, looking at Fig. 5 and Table 7, we observe the following. On average, 99.34% of benign samples were correctly classified as benign, with the benign class achieving an average f1-score of 0.9989 ± 0.0005. The four variants of DoS attacks, namely the GoldenEye, Hulk, Slowhttptest, and Slowloris also achieved high ACC_i and recall scores. However, the Heartbleed samples were mis-classified as benign, and hence the poor performance scores. The total number of samples in the dataset for the Heartbleed class were only 11, out of which 60% (i.e., 7) were taken for training, 20% (i.e., 2) for validation, and 20% (i.e., 2) for testing. Due to the large imbalance between the number of samples of benign, other DoS variants, and Heartbleed classes, the classifier wasn't able to perform well on the Heartbleed samples. Another observation to make here are the confusions between the DoS variants. For example, 0.91% of Slowhttptest samples were confused to be of Slowloris. This indicates some degree of similarity in the feature space, between the samples from DoS attack variants.

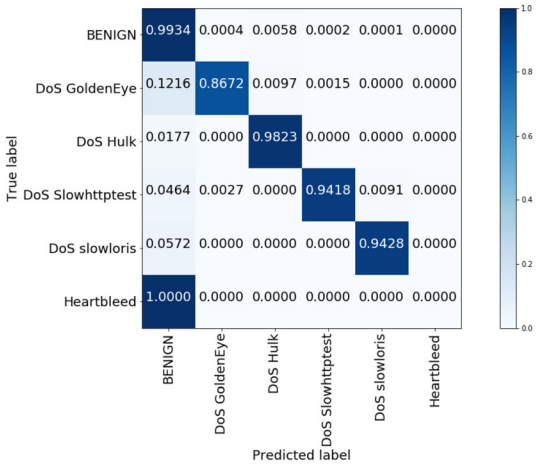


Fig. 5 TempoCode-IoT: Benign vs DoS (GoldenEye, Hulk, Slowhttptest, Slowloris, Heartbleed)

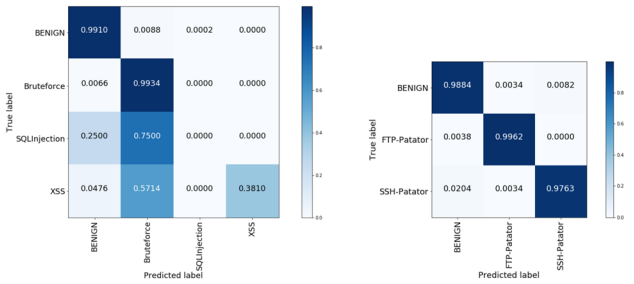


Fig. 6 TempoCode-IoT: [Left] Benign vs Web Attacks (BruteForce, SQL Injection, XSS); [Right] Benign vs Patator-based Bruteforce attacks over FTP/SSH

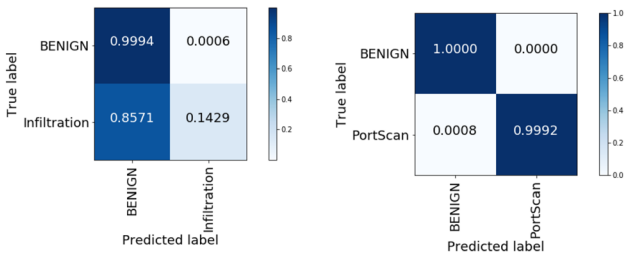


Fig. 7 TempoCode-IoT: Benign vs Infiltration, and Benign vs PortScan

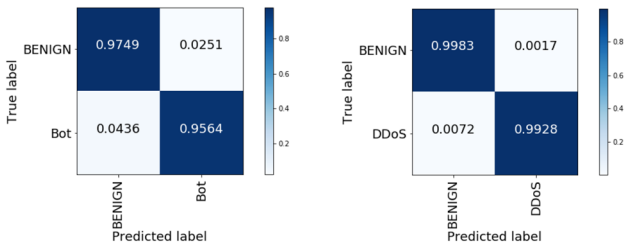


Fig. 8 TempoCode-IoT: Benign vs Bot, and Benign vs DDoS

On the web attacks dataset, we note that the benign and bruteforce web attack samples were very accurately classified, at average recall scores of 0.9911 ± 0.0010 and 0.9928 ± 0.0109, respectively (see Table 7). However, the SQL injection and XSS variants of web attacks suffered from very low classification scores. Observing the confusion matrix (Fig. 6 (Left)) gives an insight to the cause. As one can see, 75% of the SQL injection attack samples were mis-classified as bruteforce web attack, and 57.14% of XSS attack samples were also mis-classified as bruteforce web attacks. Although these samples were not classified to the correct attack variant, they were correctly identified as non-benign, thereby maintaining a high intrusion detection performance.

Looking at the results on Bruteforce attacks dataset based on the FTP- and SSH-Patator tools, the average f1-scores achieved for the benign, FTP-Patator, and SSH-Patator classes were 0.9989 ± 0.0005, 0.9127 ± 0.0134, and 0.7574 ± 0.0184, respectively (see Table 7). The confusions between these classes were also very low (see Fig. 6 [Right])

In the case of Infiltration attacks, TempoCode-IoT's performance was not as high as expected (see Fig. 7 [Left] and Table 7). Although the benign samples were correctly classified at an average recall of 0.9994 ± 0.0002, a high percentage of infiltration attack samples were mis-classified as benign. This could be attributed to the insufficient number of training samples for the Infiltration classes, as mentioned in Table 3.

On the Portscan attacks dataset, the achieved average f1-scores were 0.9992 ± 0.0003 and 0.9993 ± 0.0002 for the Benign and Port-scan classes, respectively. Moreover, the false positives count (i.e., malicious samples mis-classified as benign samples) were very low, around 0.08% (see Fig. 7 [Right]) Similarly encouraging results were obtained on the Botnet and DDoS attacks dataset (see Fig. 8), where the Botnet attacks were detected at an average recall of 0.9600 ± 0.0192, and the DDoS attacks at an average recall of 0.9928 ± 0.0013.

5.2 On the CICIDS2017 datasets: binary classification (Benign vs Malicious)

In these set of experiments, we investigate how well can TempoCode-IoT representations discriminate between benign samples and the diversity of malicious samples wherein the samples from various attack types from the individual datasets were combined together and labelled as "malicious".

We utilise the codebook $\mathbf{C_T}$ (which was learnt from Monday's benign training data, with parameters described in Sect. 4.2) to generate TempoCode-IoT representations respectively. The samples from rest of the days were combined together in one dataset and subsequently partitioned into training, validation and testing splits following the 60–20–20% ratio. The best performing $CSize = 135$ is taken for the codebook (see Sect. 4.2). Consequently, the proposed scheme is referred to as TempoCode-IoT-135. In binary classification, the objective is to detect anomalous (malicious) samples which may belong to different attack types and hence different characteristics.

Figure 9 shows the confusion matrix between benign and malicious classes for the Tempo-Code-IoT-based scheme. On average, around 99.50% of benign samples and 97.72% of malicious samples were correctly classified. The mean False Positive Rate was 2.28% while the mean False Negative Rate was 0.5%. The precision, recall and f1 scores are provided in Table 8, along with the 95% confidence intervals obtained from tests on 10 randomly partitioned subsets of the test data.

Table 7 Performance evaluation of TempoCode-IoT-based intrusion detection on CICIDS2017 datasets

Dataset	Class	Precision	Recall	F1-Score
Tuesday	Benign	0.9996±0.0001	0.9989±0.001	0.9989±0.0005
	FTP-Patator	0.8427±0.0220	0.9961±0.0034	0.9127±0.0134
	SSH-Patator	0.6194±0.0246	0.9763±0.0094	0.7574±0.0184
Wednesday	Benign	0.9866±0.0007	0.9934±0.0007	0.9900±0.0006
	DoS GoldenEye	0.978±0.0064	0.866±0.0104	0.9184±0.0060
	DoS Hulk	0.9884±0.001	0.9823±0.001	0.9853±0.001
	DoS Slowhhtptest	0.9717±0.012	0.9429±0.018	0.9567±0.008
	DoS Slowloris	0.9832±0.012	0.9374±0.009	0.9596±0.005
	Heartbleed	0.0	0.0	0.0
Thursday morning	Benign	0.9996±0.0002	0.9911±0.0010	0.9953±0.0005
	Brute force	0.4174±0.0184	0.9928±0.0109	0.5873±0.0109
	SQL injection	0.0	0.0	0.0
	XSS	0.0	0.0	0.0
Thursday afternoon	Benign	0.9999±0.0001	0.9994±0.0002	0.9997±0.0001
	Infiltration	0.0333±0.0533	0.0667±0.0107	0.0444±0.0711
Friday morning	Benign	0.9585±0.0197	0.9742±0.0191	0.9659±0.0129
	Bot	0.9736±0.0205	0.9600±0.0192	0.9663±0.0124
Friday afternoon 1	Benign	0.9985±0.0005	0.9998±0.0002	0.9992±0.0003
	PortScan	0.9999±0.0001	0.9988±0.0005	0.9993±0.0002
Friday afternoon 2	Benign	0.9907±0.0017	0.9983±0.0005	0.9945±0.0008
	DDoS	0.9987±0.0004	0.9928±0.0013	0.9958±0.0007

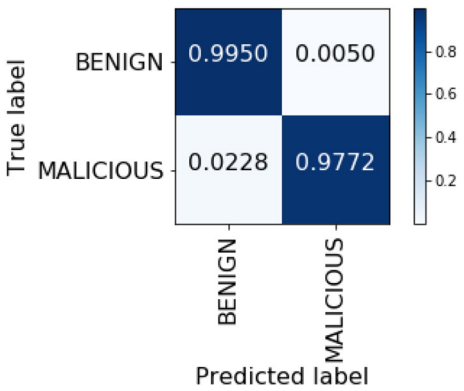


Fig. 9 Confusion matrix for TempoCode-IoT-based binary classification

Table 8 Binary classification scores with TempoCode-IoT on CICIDS2017 datasets

Class	Precision	Recall	F1-Score
Benign	0.9927±0.0004	0.9950±0.0003	0.9939±0.0003
Malicious	0.9842±0.0010	0.9772±0.0014	0.9807±0.0011

5.3 On the NBaIoT datasets: botnet attacks detection

In our empirical evaluations of TempoCode-IoT on the NBaIoT datasets, we adopt the following approach. Considering the diversity of device types, we propose to learn a temporal codebook for each device, capturing its key benign flow patterns. Each device's dataset is split into three equal-sized partitions for training, validation and testing (similar to the approach of [21]). The features (as described in Sect. 3.1) in each device's dataset were normalized to be in the range [0, 1], and only the benign traffic's features from the training dataset were used in codebook learning for the device.

Due to the lack of timestamp data in the datasets, we adaptively set the size of time-window (t_{dur}) depending upon the number of benign training samples available for a device. For each device, t_{dur} is taken as a number of samples given by $\frac{N_{tr-ben}}{(CSize/N_{ct})}$, where N_{tr-ben} is the number of benign samples in training dataset of this device, $CSize = 140$ (codebook size), and $N_{ct} = 10$ (number of codewords to be learnt from each time-window). For example, the first t_{dur} number of samples could have arrived in say 1 h while the next t_{dur} number of samples could have arrived in 0.5 h. Our empirical evaluations have proven the strength of TempoCode-IoT representations even with such an adaptive t_{dur}. The values of $CSize$ and N_{ct} have been chosen

such that the learnt codebook is of $CSize = 140$ for all devices, to be close to the best codebook size found in TempoCode-IoT evaluations on CICIDS2017 datasets (as described in Sect. 4.2).

Consequently, the TempoCode-IoT representations of the benign and malicious flows from each device are generated using the respective device's temporal codebook. Based on these TempoCode-IoT representations, the ensemble of SVM classifiers is trained using the training dataset and optimized using the validation dataset, in a supervised manner. Because each device has different behaviors and actions resulting in different traffic behaviors, a separate ensemble of SVM classifiers is learnt for each device (see Table 9). The testing datasets which contain a mix of benign and malicious samples that have not been seen in the training phase are used to evaluate the performance of TempoCode-IoT representations through the ensemble of SVMs. Each test set is further split into 10 random partitions to run 10 tests for each device and obtain average scores with 95% confidence intervals.

The experimental evaluations of the proposed TempoCode-IoT-based intrusion detection yielded very encouraging results for all the devices (see Fig. 10). The proposed method was able to differentiate between benign and malicious traffic from each compromised IoT device with very high recall scores (0.9940–0.9991), and very low false positive rates (0.02–0.71%), as observed from Table 10 which presents the scores averaged over ten test runs along with the respective 95% confidence intervals.

These results indicate the effectiveness of the proposed temporal codebook in learning key benign traffic patterns of each device and the high discriminative capability exhibited by the TempoCode-IoT representations through the ensemble of SVM classifiers. Moreover, since our approach builds a temporal codebook and ensemble of classifiers for each device (using the flows from the respective device only), the addition of new IoT devices would only require learning of a codebook and classifier ensembles for those specific devices alone. In this way, retraining costs are avoided as the system doesn't require retraining the codebooks and classifiers previously learnt for other devices. Furthermore, such an approach yields tolerance to a growing heterogeneity of IoT devices.

5.4 Comparison with related works

In this section, we present a comparison of TempoCode-IoT's performance against other methods proposed recently on the CICIDS2017 and NBaIoT datasets. Table 11 provides the precision, recall and f1 scores (for detection of malicious flows) of our work and those reported by the respective related works on the CICIDS2017 datasets. Additionally, Table 12 compares our results with those reported in [21] on the NBaIoT datasets.

5.4.1 On the CICIDS2017 dataset

Marir et al. [20] proposed a deep belief network (DBN) and a multi-layer ensemble of SVMs (MLE-SVM) for detection of malicious flows. They studied the performance of DBN and MLE-SVM individually, as well as in a coupled fashion. The proposed TempoCode-IoT method of this paper outperforms the three methods of [20].

To evaluate the performance of the proposed TempoCode-IoT method against the raw statistical flow features (such as those mentioned in Sect. 3.1), we compare with the top three scores reported in [34]'s study. In their work, they found the following three machine learning algorithms performed best in detecting the malicious flows: *k*-Nearest Neighbors (*k*NN), Random Forest, and ID3, based on the raw statistical flow features. From Table 11, we can see that TempoCode-IoT performed better than their *k*NN and RF methods and slightly better than the ID3-based method. The results of [34] were after a feature selection process by which only the most important features were retained. If a similar procedure is applied to TempoCode-IoT features, it could result in even better performance. However, we dedicate this to be investigated in a future work.

Table 9 TempoCode-IoT evaluations on the NBaIoT datasets: SVM parameters, number of training/testing samples ($N_{tr}, N_{tr-ben}, N_{te}$), codebook learning time (CLT), time consumed in training/testing (T_{tr}, T_{te})

Dev	SVM params	$\mathbf{N_{tr-ben}}$	CLT (s)	$\mathbf{N_{tr}}$	$\mathbf{T_{tr}}$ (s)	$\mathbf{N_{te}}$	$\mathbf{T_{te}}$ (s)
1	C1000, G0.001	13465	4.24	26930	2.31	336389	14.1
2	C1000, G0.001	4370	2.15	8740	5.29	278619	10.5s
3	C1000, G0.001	11564	2.21	23128	1.5	117034	4.4
4	C0.1, G0.001	53379	9.03	106758	10.5	361198	114
5	C1, G0.001	18389	2.34	36778	2.67	273769	35.8
6	C10, G0.001	30518	3.38	61036	3.1	276652	19.5
7	C10, G0.001	15605	3.21	31210	2.08	123301	5.7
8	C10, G0.001	14261	4.81	28522	2.34	286423	27.6
9	C100, G0.01	5978	1.32	11956	1.4	283086	10.7

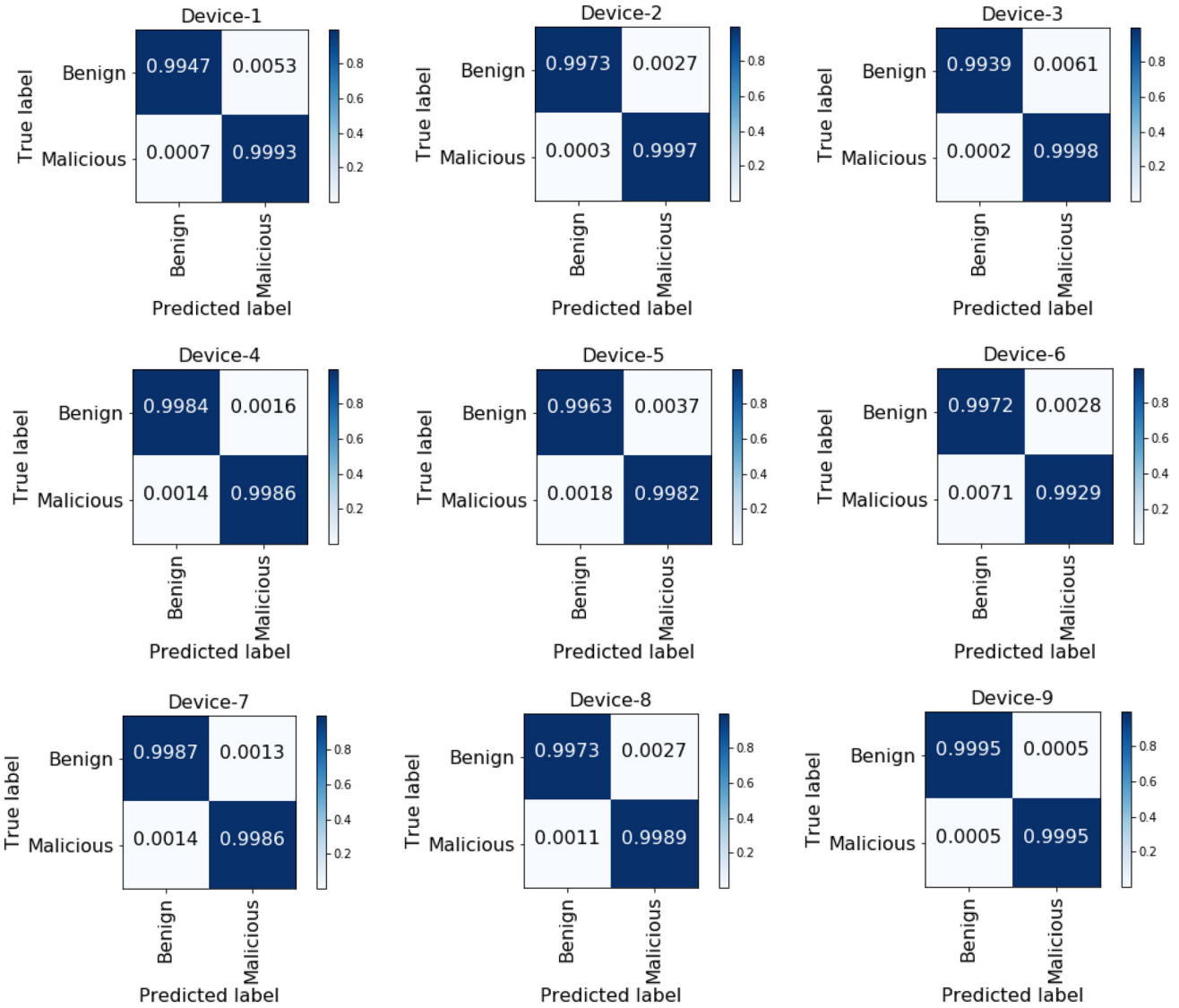


Fig. 10 Evaluating TempoCode-IoT on NBaIoT datasets: confusion matrices for each of the nine devices (the values are averaged across test test runs)

Table 10 TempoCode-IoT results on the NBaIoT datasets: precision, recall, F1 scores for benign and malicious classes, and false positive rate (FPR)

Device	Prec_Ben	Rec_Ben	F1_Ben	Prec_Mal	Rec_Mal	F1_Mal	FPR (%)
1	0.9845±0.0016	0.9948±0.0011	0.9896±0.0010	0.9998±0.0	0.9993±0.0001	.9996±0.0001	0.07±0.01
2	0.9829±0.0027	0.9973±0.0010	0.9900±0.0013	1.0±0.0	0.9997±0.0001	0.9998±0.0	0.03±0.0
3	0.9983±0.0009	0.9940±0.0019	0.9961±0.0011	0.9993±0.0002	0.9998±0.0001	0.9996±0.0001	0.02±0.01
4	0.9919±0.0007	0.9984±0.0004	0.9951±0.0004	0.9997±0.0001	0.9986±0.0001	0.9992±0.0001	0.14±0.01
5	0.9754±0.0025	0.9963±0.0015	0.9858±0.0015	0.9997±0.0001	0.9982±0.0002	0.9990±0.0001	0.18±0.02
6	0.9455±0.0027	0.9972±0.0006	0.9706±0.0013	0.9997±0.0001	0.9929±0.0003	0.9963±0.0001	0.71±0.03
7	0.9903±0.0021	0.9987±0.0006	0.9945±0.0010	0.9998±0.0001	0.9986±0.0003	0.9992±0.0002	0.14±0.03
8	0.9800±0.0026	0.9973±0.0010	0.9886±0.0013	0.9999±0.0001	0.9989±0.0001	0.9994±0.0001	0.11±0.01
9	0.9779±0.0036	0.9991±0.0009	0.9889±0.0019	1.0±0.0	0.9995±0.0001	0.9997±0.0001	0.05±0.01

Table 11 Performance comparison of TempoCode-IoT with related works on CICIDS2017 datasets

Method	Precision	Recall	F1-Score
DBN [20]	0.9006	0.9540	0.9265
MLE-SVM [20]	0.9056	0.9494	0.9269
DBN with LE-SVM [20]	0.9040	0.9565	0.9295
*k*NN [34]	0.9600	0.9600	0.9600
RF [34]	0.9800	0.9700	0.9700
ID3 [34]	0.9800	**0.9800**	0.9800
TempoCode-IoT (Our)	**0.9842**	0.9772	**0.9807**

The bold values indicate the highest values in the respective columns

5.4.2 On the NBaIoT datasets

We compare the performance of TempoCode-IoT against the results of two recently published works on NBaIoT data-sets: [25] and [21]. In comparison to the results of deep auto-encoders-based intrusion detection of [21] who ach-ieved a mean FPR of 0.7%, the proposed Tempo-Code-IoT yielded a slightly higher mean FPR of 0.16% (still below 1%). In addition, their work tested other methods such as Isolation Forest and Local Outlier Factor (LOF) which also yielded higher mean FPRs of 2.7% and 8.6% respectively, as summarized in Table 13. Moreover, the mean detection time reported by [21] of 174ms is significantly slower than that of the proposed Tempo-Code-IoT-based method which required an average of 0.110ms per Tempo-Code-IoT vector.

Another work based on the NBaIoT dataset is of [25] who investigated SVMs and Isolation Forests trained over a subset of the statistical features provided by the datasets. The features were selected based on metrics such as entropy, variance and Hopkins statistic. As shown in Table 12, the proposed TempoCode-IoT-based method outperforms the best models of [25] for all devices. Their paper did not report results for devices 3 and 7.

The works of [21, 25] are unsupervised learning-based approaches where the malicious traffic samples are detected as anomalies from the learnt benign traffic patterns. And although this paper proposed unsupervised learning of the temporal codebook to capture benign traffic patterns, the ensemble of classifiers is trained in a supervised fashion. The higher FPRs and lower precisions of [21, 25] in comparison to our work indicate the benefits of supervised learning. However, considering the envisioned dynamic IoT ecosystem of tomorrow and evolving techniques of malicious attackers, we believe that unsupervised or possibly hybrid approaches may be a better choice. Hence, as part of our future work, we shall investigate TempoCode-IoT-based unsupervised anomaly detection approaches to identify intrusion attacks.

Table 13 Performance comparison of TempoCode-IoT with results reported by [21] on NBaIoT datasets, in terms of mean false positive rates (FPR)

Method	Mean FPR (%)
Deep-autoencoders	0.7
Isolation forest	2.7
LOF	8.6
TempoCode-IoT (Our)	0.16

6 Open issues and challenges

Although considerable efforts have been made in the field of IoT security, achieving fully robust, secure and resilient heterogeneous IoT environments is an ongoing pursuit. In this section, we briefly discuss some open issues and challenges associated with securing the evolving IoT landscape against intrusions.

Although IDSs may be very accurate, there may be certain situations in which an IDS or the device hosting an IDS may itself become a target of attacks. The vulnerability increases if the IDS is hosted on the edge or on the end device. Hence, further studies are needed to investigate defense mechanisms that are resilient to such attacks.

In order to foster research in developing IDSs for heterogeneous IoT, efforts are needed to build publicly available large-scale datasets using real devices and realistic scenarios. Currently, most works use simulations or small-scale datasets.

Despite the success of IDSs based on supervised learning, the problem of detecting unknown, advanced attacks in realistic heterogeneous IoT environments remains an open

Table 12 Performance comparison of TempoCode-IoT with results reported by [25] on NBaIoT datasets, in terms of precision scores (malicious class)

Work	Device								
	1	2	3	4	5	6	7	8	9
[25]	0.9952	0.9981	N/A	0.9474	0.9914	0.9860	N/A	0.9938	0.9976
TempoCode-IoT (Our)	0.9998	1.0	0.9993	0.9997	0.9997	0.9997	0.9998	0.9999	1.0

issue. The latest advances in unsupervised learning could be promising in this regard.

7 Conclusions and future work

In this work, we proposed *TempoCode-IoT*, a temporal codebook-based encoding of flow features, as a novel feature transformation of network flow features based on capturing the key patterns of benign traffic in a learnt temporal codebook. Using the codebook, distances of (benign and malicious) traffic flows from those key patterns are measured and recorded as the transformed features to train an ensemble of SVM classifiers. The experimental evaluations on recent realistic datasets (CICIDS2017 and NBaIoT) proved the effectiveness of TempoCode-IoT representations in detecting intrusions of various types and for different devices used in the datasets. On the NBaIoT datasets, TempoCode-IoT achieved high mean precision scores (0.9993–1.0), low mean False Positive Rates (0.02–0.71%), and low average detection time of 0.110ms per TempoCode-IoT sample. Similarly, on the CICIDS2017 datasets, TempoCode-IoT achieved high precision and F1-scores of 0.9842 and 0.9807 respectively.

For future work, we plan to investigate the performance of TempoCode-IoT representations over larger and combined datasets of different attack types, in an unsupervised fashion, which effectively turns into a more challenging multi-class anomaly detection problem.

Acknowledgements This work is partially supported by NSERC CREATE TRANSIT, NSERC DIVA Strategic Research Network and Canada Research Chairs Program.

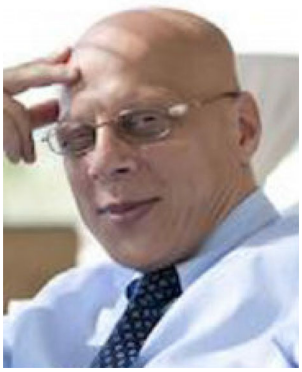
References

1. Aldwairi, T., Perera, D., Novotny, M.A.: An evaluation of the performance of restricted boltzmann machines as a model for anomaly network intrusion detection. Comput. Netw. **144**, 111–119 (2018)
2. Almi'ani, M., Ghazleh, A.A., Al-Rahayfeh, A., Razaque, A.: Intelligent intrusion detection system using clustered self organized map. In: 2018 Fifth international conference on software defined systems (SDS), pp. 138–144 (2018)
3. Aloqaily, M., Otoum, S., Ridhawi, I.A., Jararweh, Y.: An intrusion detection system for connected vehicles in smart cities. Ad Hoc Networks **90**, 101842 (2019). Recent advances on security and privacy in Intelligent Transportation Systems
4. Arthur, D., Vassilvitskii, S.: How slow is the k-means method? In: Proceedings of the twenty-second annual symposium on computational geometry, SCG '06, p. 144-153. Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1137856.1137880
5. Atli, B.G., Miche, Y., Jung, A.: Network intrusion detection using flow statistics. In: 2018 IEEE Statistical Signal Processing Workshop (SSP), pp. 70–74 (2018)
6. Awad, M., Khanna, R.: Support Vector Machines for Classification, pp. 39–66. Apress, Berkeley, CA (2015)
7. Bottou, L., Chapelle, O., DeCoste, D., Weston, J.: Support Vector Machine Solvers, pp. 1–27 (2007)
8. Boukerche, A., Jucá, K.R.L., Notare, M.S.M.A., Sobral, J.B.M.: Biological inspired based intrusion detection models for mobile telecommunication systems. In: Olariu, S., Zomaya, A.Y. (eds.) Handbook of Bioinspired Algorithms and Applications. Chapman and Hall/CRC, New York (2005)
9. Boukerche, A., Jucá, K.R.L., Sobral, JaB, Annoni Notare, M.S.M.: An artificial immune based intrusion detection model for computer and telecommunication systems. Parallel Comput **30**(5–6), 629–646 (2004)
10. Boukerche, A., Machado, R.B., Jucá, K.R.L., Sobral, JaBM, Notare, M.S.M.A.: An agent based and biological inspired real-time intrusion detection and security model for computer network operations. Comput. Commun. **30**(13), 2649–2660 (2007)
11. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
12. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. **2**(2), 121–167 (1998)
13. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual Categorization with bags of keypoints. In: Workshop on statistical learning in computer vision, ECCV, pp. 1–22 (2004)
14. Gil, G.D., Lashkari, A.H., Mamun, M., Ghorbani, A.A.: Characterization of encrypted and vpn traffic using time-related features. In: 2nd International conference on information systems security and privacy (ICISSP 2016), pp. 407–414 (2016)
15. Ioannou, C., Vassiliou, V.: An intrusion detection system for constrained wsn and iot nodes based on binary logistic regression. In: Proceedings of the 21st ACM international conference on modeling, analysis and simulation of wireless and mobile systems, MSWIM '18, p. 259-263. Association for Computing Machinery, New York, NY, USA (2018)
16. Kaspersky: Kaspersky lab ddos intelligence quarterly report: amplification attacks and old botnets make a comeback (2018). "https://www.kaspersky.com/about/press-releases/2018-amplification-attacks-and-old-botnets". Accessed 29 October 2018
17. Lee, W., Rezapour, A., Tzeng, W.: Monsieur poirot: detecting botnets using re-identification algorithm and nontrivial feature selection technique. In: 2018 IEEE international conference on communications (ICC), pp. 1–6 (2018)
18. Lin, W.C., Ke, S.W., Tsai, C.F.: Cann: an intrusion detection system based on combining cluster centers and nearest neighbors. Knowl.-Based Syst. **78**, 13–21 (2015)
19. Machado, R.B., Boukerche, A., Sobral, J.B.M., Jucá, K.R.L., Notare, M.S.M.A.: A hybrid artificial immune and mobile agent intrusion detection based model for computer network operations. In: 19th International parallel and distributed processing symposium (IPDPS 2005), CD-ROM / Abstracts Proceedings, 4-8 April 2005, Denver, CO, USA. IEEE Computer Society (2005)
20. Marir, N., Wang, H., Feng, G., Li, B., Jia, M.: Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. IEEE Access **6**, 59657–59671 (2018)
21. Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., Elovici, Y.: N-baiot: network-based detection of iot botnet attacks using deep autoencoders. IEEE Pervasive Comput. **17**(3), 12–22 (2018)
22. Micro, T.: Ddos—security news—trend micro usa. https://www.trendmicro.com/vinfo/us/security/news/ddos
23. Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A.: Kitsune: an ensemble of autoencoders for online network intrusion detection. In: 25th Annual network and distributed system security symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018 (2018)

24. Moustafa, N., Turnbull, B., Choo, K.R.: An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. IEEE Internet Things J. (2018). https://doi.org/10.1109/JIOT.2018.2871719
25. Nõmm, S., Bahsi, H.: Unsupervised anomaly based botnet detection in iot networks. In: 2018 17th IEEE international conference on machine learning and applications (ICMLA), pp. 1048–1053 (2018)
26. Nanni, L., Lumini, A.: Heterogeneous bag-of-features for object/scene recognition. Appl. Soft Comput. **13**(4), 2171–2178 (2013)
27. Nofal, R.A., Tran, N., Garcia, C., Liu, Y., Dezfouli, B.: A comprehensive empirical analysis of tls handshake and record layer on iot platforms. In: Proceedings of the 22nd international ACM conference on modeling, analysis and simulation of wireless and mobile systems, MSWIM '19, p. 61-70. Association for Computing Machinery, New York, NY, USA (2019)
28. Osborne, C., Day, Z.: The most interesting internet-connected vehicle hacks on record. https://www.zdnet.com/article/these-are-the-most-interesting-ways-to-hack-internet-connected-vehicles/
29. Otoum, S., Kantarci, B., Mouftah, H.: Empowering reinforcement learning on big sensed data for intrusion detection. In: ICC 2019 - 2019 IEEE international conference on communications (ICC), pp. 1–7 (2019)
30. Otoum, S., Kantarci, B., Mouftah, H.T.: On the feasibility of deep learning in sensor network intrusion detection. IEEE Netw. Lett. **1**(2), 68–71 (2019)
31. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
32. Restuccia, F., D'Oro, S., Melodia, T.: Securing the internet of things in the age of machine learning and software-defined networking. IEEE Internet Things J. **5**(6), 4829–4842 (2018)
33. Sedjelmaci, H., Senouci, S.M., Abu-Rgheff, M.A.: An efficient and lightweight intrusion detection mechanism for service-oriented vehicular networks. IEEE Internet Things J. **1**(6), 570–577 (2014)
34. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: 4th International conference on information systems security and privacy (ICISSP) (2018)
35. Shasha, S., Mahmoud, M., Mannan, M., Youssef, A.: Playing with danger: A taxonomy and evaluation of threats to smart toys. IEEE Internet Things J. (2018). https://doi.org/10.1109/JIOT.2018.2877749
36. Siddiqui, A.J., Boukerche, A.: Encoded flow features for network intrusion detection in internet of things. In: 2020 IEEE 17th annual consumer communications networking conference (CCNC), pp. 1–6 (2020)
37. Soundar Raja James, R.J.P., Albasir, A.A., Naik, K., Zaman, M., Goel, N.: A power signal based dynamic approach to detecting anomalous behavior in wireless devices. In: Proceedings of the 16th ACM international symposium on mobility management and wireless access, MobiWac'18, p. 9-18. Association for Computing Machinery, New York, NY, USA (2018)
38. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
39. Venkata Abhishek, N., Tandon, A., Lim, T.J., Sikdar, B.: Detecting forwarding misbehavior in clustered iot networks. In: Proceedings of the 14th ACM international symposium on QoS and security for wireless and mobile networks, Q2SWinet'18, p. 1-6. Association for Computing Machinery, New York, NY, USA (2018)
40. Yao, H., Fu, D., Zhang, P., Li, M., Liu, Y.: Msml: a novel multilevel semi-supervised machine learning framework for intrusion detection system. IEEE Internet Things J. (2018). https://doi.org/10.1109/JIOT.2018.2873125
41. Zhang, J., Chen, C., Xiang, Y., Zhou, W., Xiang, Y.: Internet traffic classification by aggregating correlated naive bayes predictions. IEEE Trans. Inform. Forensics Sec. **8**(1), 5–15 (2013)
42. Zheng, J., Hu, M.: An anomaly intrusion detection system based on vector quantization. IEICE Trans. Inf. Syst. **E89–D**(1), 201–210 (2006)
43. Zhou, Z.H.: Ensemble Learning, pp. 270–273. Springer US, Boston, MA (2009)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Abdul Jabbar Siddiqui is a PhD candidate in the Electrical and Computer Engineering program at University of Ottawa, Canada. He obtained his MASc in Electrical and Computer Engineering from University of Ottawa in 2015. His research interests include AI/Machine-Learning for Cyber Security, Internet of Things, Internet of Vehicles, Intelligent Transportation Systems, and Biometrics.



Azzedine Boukerche (FIEEE, FEIC, FCAE, FAAS) is a Distinguished University Professor and holds a Canada Research Chair Tier-1 position with the University of Ottawa. He is founding director of the PARADISE Research Laboratory and the DIVA Strategic Research Center, University of Ottawa. He has received the C. Gotlieb Computer Medal Award, Ontario Distinguished Researcher Award, Premier of Ontario Research Excellence Award, G. S. Glinski Award for Excellence in Research, IEEE Computer Society Golden Core Award, IEEE CS-Meritorious Award, IEEE TCPP Leaderships Award, IEEE ComSoc ASHN Leaderships and Contribution Award, and University of Ottawa Award for Excellence in Research. He serves as an Editor-in-Chief for ACM ICPS and associate editor for several IEEE transactions and ACM journals, and is also a Steering Committee Chair for several IEEE and ACM international conferences. His current research interests include wireless ad hoc and sensor networks, wireless networking and mobile computing, wireless multimedia, QoS service provisioning, performance evaluation and modeling of large-scale distributed and mobile systems, and large scale distributed and parallel discrete event simulation. He has published extensively in these areas and received several best research paper awards for his work. He is a fellow of the Engineering Institute of Canada, the Canadian Academy of Engineering, the American Association for the Advancement of Science, and the IEEE.