FS3: Few-Shot and Self-Supervised Framework for Efficient Intrusion Detection in Internet of Things Networks

Ayesha Siddiqua Dina
adina@floridapoly.edu
Florida Polytechnic University
Lakeland, FL, USA

A.B. Siddique
siddique@cs.uky.edu
University of Kentucky
Lexington, KY, USA

D. Manivannan
mani@cs.uky.edu
University of Kentucky
Lexington, KY, USA

ABSTRACT

Securing the Internet of Things is critical for its successful deployment in various industries. While Machine Learning techniques have shown promise for intrusion detection in the Internet of Things, existing methods require large amounts of labeled training data; moreover, they encounter challenges with the presence of extreme class imbalance, i.e., some classes are underrepresented in the datasets used. Supervised methods rely on extensive labeled data, which can be costly and time-consuming to obtain. Class imbalance in datasets further exacerbates the challenge by skewing the model's learning process toward the majority classes, leading to poor detection of attacks belonging to minority classes. This issue is particularly pronounced in the Internet of Things environments due to diverse devices and the varying frequency of intrusions targeting them. To overcome these challenges, we present a **F**ew-**S**hot and **S**elf-**S**upervised framework, called FS3, for detecting intrusions in IoT networks. FS3 works in three phases. The first phase employs self-supervised learning to learn latent patterns and robust representations from unlabeled data. The second phase introduces Few-shot learning with contrastive training. Few-shot learning enables the model to learn from a few labeled examples, thereby eliminating the dependency on a large amount of labeled data. Contrastive Training addresses the class imbalance issue by improving the discriminative power of the model. The third phase introduces a novel K-Nearest neighbor algorithm that sub-samples the majority class instances to further reduce imbalance and improve overall performance. Experimental results based on three publicly available benchmark datasets demonstrate the efficacy of FS3 in addressing the challenges posed by the limited availability of labeled data as well as class imbalance in datasets. Our proposed framework FS3, utilizing only 20% of labeled data, outperforms fully supervised state-of-the-art models by up to 42.39% and 43.95% with respect to the metrics precision and F_1 score, respectively.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**.

KEYWORDS

Intrusion detection, Few-Shot Learning, Internet of Things.

ACSAC '23, December 04–08, 2023, Austin, TX, USA

ACM ISBN 979-8-4007-0886-2/23/12.
https://doi.org/10.1145/3627106.3627193

ACM Reference Format:
Ayesha Siddiqua Dina, A.B. Siddique, and D. Manivannan. 2023. FS3: Few-Shot and Self-Supervised Framework for Efficient Intrusion Detection in Internet of Things Networks. In *Annual Computer Security Applications Conference (ACSAC '23), December 04–08, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3627106.3627193

1 INTRODUCTION

Internet of Things (IoT) has become an integral part of our lives, with billions of devices connected through wired and wireless networks. For example, the number of active IoT devices surpassed 10 billion in 2021. According to a report by Cisco [6], this number is projected to increase to over 500 billion by 2030. The proliferation of IoT devices has revolutionized various domains, ranging from healthcare to transportation, by enabling seamless connectivity and intelligent automation [11]. For instance, cell phones, thermostats, and doorbell cameras have already made a significant impact on various aspects of society, encompassing industry and everyday life. However, the rapid expansion of IoT networks has led to a significant increase in the attack surface for malicious actors, and hence these networks are increasingly becoming attractive targets for malicious actors. As a prime example, a series of distributed denial of service (DDoS) attacks took place in the United States in 2016, exploiting the vulnerabilities of IoT devices through the Mirai malware [7].

Intrusion detection systems (IDSes) are essential for protecting IoT networks from attacks and play a pivotal role in safeguarding IoT networks against unauthorized access, data breaches, and other security threats. Traditional IDSes often rely on signature-based approaches, which struggle to keep up with the evolving threat landscape. Furthermore, IoT devices commonly employ specialized protocols and display unique traffic patterns. These inherent characteristics capture the intricacies of IoT network traffic. As a result, the utilization of Machine Learning (ML) has become prevalent in securing IoT devices as well as optimizing various other aspects such as coordinating wireless devices for efficient spectrum usage [24]. ML models have demonstrated promising results in detecting anomalous patterns in network traffic and identifying potential intrusions [54]. However, the effectiveness of these models heavily relies on the availability of large amounts of labeled training data, which can be scarce to obtain in real-world IoT environments, especially for new attack vectors. Since annotating large datasets with accurate labels is a resource-intensive and often expensive task, especially in domains like cybersecurity, where expert knowledge is required. Additionally, datasets for IoT networks are often characterized by class imbalance, since the occurrence of intrusions targeting different device types and functionalities varies significantly. Specifically, within any given time window, we often

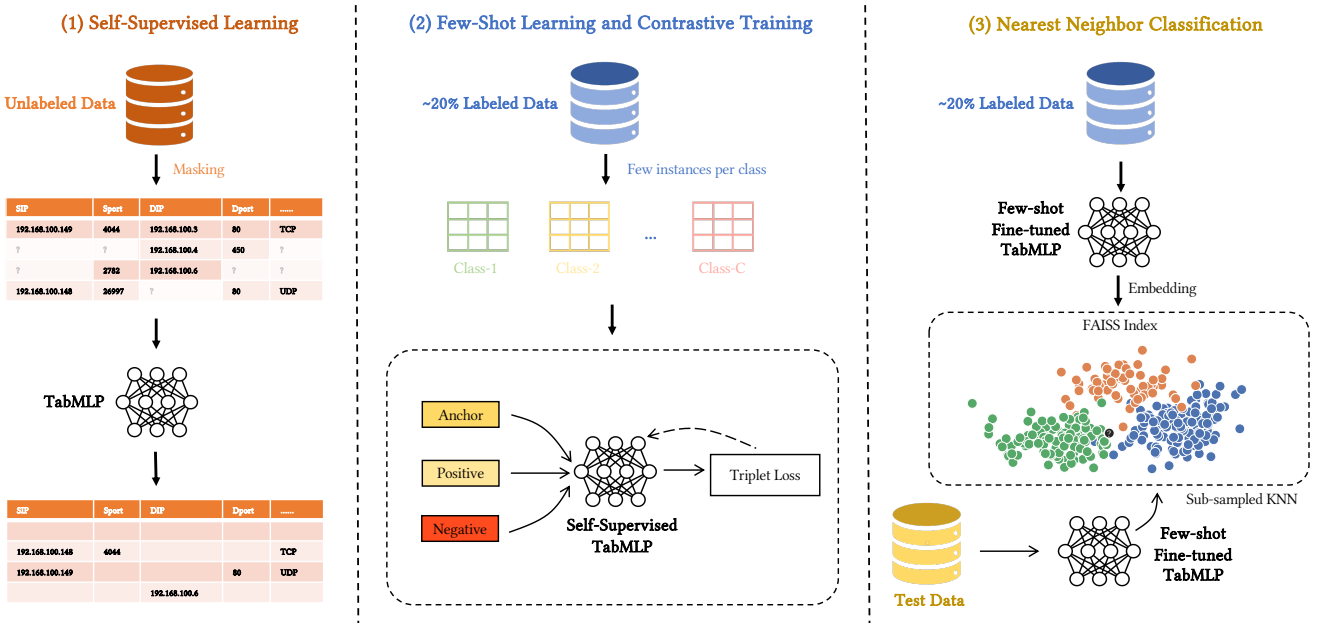


Figure 1: Overview of our proposed Few-Shot and Self-Supervised framework: FS3 that requires a small amount of labeled data.

observe two contrasting scenarios: either there is a significantly higher number of intrusion attempts, or there is a significantly smaller number of such events. This inherent variability in attack occurrences is a fundamental characteristic of IoT network traffic. Conversely, when there is a sudden surge in attacks (and the system was trained using balanced data), it can lead to poor detection systems. Our intent in focusing on class-imbalance datasets was precisely to reflect these real-world fluctuations in IoT intrusion detection. Notably, the ratio of the number of samples in the minority classes to that of the majority classes can vary from 1:100 to 1:1000, and beyond. Consequently, the minority classes representing these intrusions are often underrepresented in the training data, leading to biased model learning and suboptimal detection performance for these critical threats.

The issue of imbalanced datasets is often overlooked or addressed through oversampling and undersampling techniques, which can lead to overfitting or underfitting problems. Overfitting arises from the inclusion of exact replicas of original samples, while underfitting occurs due to inadequate data samples by undersampling. To avoid the overfitting problem that arises due to simple oversampling, adding synthetic data generated to minority classes has been proposed [12]. Furthermore, traditional loss functions, such as cross-entropy loss, do not properly attend to minority class instances during model training, since they perform averaged-gradient updates. To deal with this challenge, methods employing specialized loss function such as focal loss to allow dynamically scaled-gradient updates has been used [13]. This results in down-weighing of easy instances, thereby compelling the model to concentrate on difficult misclassified examples. Nonetheless, all of these approaches are fully supervised and require a huge amount of labeled training data. Class imbalance in datasets is a frequent issue in machine learning, arising when the distribution of samples in a dataset is skewed or biased. This can result in a model bias during training, ultimately affecting its performance adversely. Our objective is to reduce the influence of class imbalance problems in deep learning classifiers while also tackling the scarcity of large labeled training data in real-world scenarios.

We address these issues in this paper and propose FS3, a novel **F**ew-**S**hot and **S**elf-**S**upervised framework for intrusion detection in IoT networks. An overview of our proposed framework FS3 is presented in Figure 1. FS3 overcomes the limitations of state-of-the-art approaches by leveraging self-supervised learning (SSL), few-shot learning (FSL) with contrastive training, and a novel sub-sampled K-Nearest Neighbor (KNN) algorithm. In the first phase, FS3 employs SSL to extract latent patterns and robust representations from unlabeled data. By capitalizing on the inherent structure of the data, FS3 reduces the dependence on extensive labeling, alleviating the burden of manual annotation. Specifically, we leverage attentive interpretable tabular learning [3] and pre-train a tabular multilayer perceptron (TabMLP) [58] as the backbone encoder that learns robust embeddings of the categorical as well as continuous features using masking objective. The second phase introduces FSL with contrastive training. By learning from a small number of labeled examples (e.g., 5-10 instances per class), the model becomes adaptable to dynamic IoT environments, where acquiring extensive labeled data for all possible intrusion scenarios is impractical. Our proposed approach effectively uses IoT-specific features, making our method well-suited for the detection of intrusions that are characteristic of IoT environments. Specifically, we fine-tune the encoder contrastively using the triplet loss function [61] to enhance the model's discriminative power, particularly for the minority classes. Essentially, this loss function imposes a constraint on the model to learn feature representations of the input samples in a manner that promotes proximity among samples belonging to the same class within the feature space, while ensuring greater separation between samples belonging to different classes.

Furthermore, FS3 introduces a novel sub-sampled KNN algorithm in the third phase. This algorithm selectively sub-samples instances from the majority class considering the distribution of the training data, reducing the class imbalance and further enhancing the performance of the model. By intelligently weighting the instances, FS3 achieves a more balanced representation of the classes, leading to improved intrusion detection capabilities across the entire spectrum of intrusion types. To optimize the process of similarity search and enhance the speed of inference, we employ Facebook AI Similarity Search (FAISS) [20] for storing the training samples (i.e., 20% of the labeled data). One of the primary motivations behind using only 20% of the training data is to reduce the labeling cost, which is a significant concern in many real-world scenarios. By effectively utilizing a smaller portion of the data, our approach addresses this practical constraint and can make machine-learning solutions more accessible to organizations with limited labeling resources. Specifically, using only 20% of the training data our approach tries to emulate where the machine learning practitioners have access to limited labeled training data. Similarly, another important consideration is the reduction in training time.

To evaluate the effectiveness of FS3, we conducted extensive experiments using three publicly available datasets from the IoT domain, namely, WUSTL-EHMS [17], WUSTL-IIoT [63], and BoT-IoT [23]. These datasets represent diverse IoT scenarios and encompass samples for wide range of attack classes. We compare the performance of FS3 with several state-of-the-art IDSes presented in the literature – CNN-BiLSTM [45], PB-DID [59], and DBN-IDS [5]. Furthermore, we trained a range of strong baseline models that employ traditional cross-entropy loss function, dice loss function, and random oversampling techniques. We evaluated both binary and multi-class classification intrusion detection tasks. The experimental results demonstrate the superior performance of FS3 with significant improvements over the state-of-the-art as well as baseline models in a wide range of experimental setups. Notably, our proposed framework FS3 outperforms fully supervised approaches by achieving up to 42.39% and 43.95% improvements with respect to precision and F_1 score, respectively, while utilizing only 20% of the labeled data. It is also important to emphasize that our fine-tuning phase (i.e., FSL) only used 5 and 10 labeled training examples per class. These results highlight the remarkable efficacy of FS3 in addressing the challenges posed by limited labeled data availability and data imbalance in datasets, enabling more robust and accurate intrusion detection in IoT networks. All the relevant code is available at: `github.com/MultifacetedIntrusionDetection/ID-FS3`.

Contributions of this paper can be summarized as follows:

- We propose a novel few-shot and self-supervised framework FS3 for intrusion detection in IoT networks – a highly critical yet underexplored area.
- FS3 effectively leverages unlabeled data and enhances the discriminative capacity of the model in scenarios wherein limited labeled data is available.
- We evaluate FS3 on three diverse publicly available datasets for both multi-class classification and binary classification tasks and show that it outperforms fully supervised state-of-the-art models with respect to precision and F_1 score by a large margin.

The remainder of the paper is organized as follows. In Section 2, we provide an overview of the learning paradigms employed in this paper. Section 3 introduces our proposed framework. The experimental setup is presented in Section 4 and the results of our performance evaluation are discussed in Section 5. Section 6 discusses the related work and Section 7 concludes the paper.

2 PRELIMINARIES

Our proposed framework FS3 employs a number of learning paradigms. In the following, we provide a brief overview of these paradigms.

2.1 Self-Supervised Learning

Self-supervised or unsupervised learning is a learning setting in machine learning that involves training models on unlabeled data without explicit guidance or supervision from human-labeled examples. In the context of deep learning, this learning approach focuses on discovering patterns, structures, and representations within the data without relying on predefined labels. Pretraining models in the fields of Natural Language Processing (NLP) and Computer Vision [19, 35, 36, 47] train deep learning models on large-scale datasets with the objective of learning general-purpose representations of the input data. These pre-trained models serve as a foundation for various downstream tasks and can significantly improve performance, especially when labeled data is limited. In this work, we use TabNet [3] to perform unsupervised learning using unlabeled IoT network traffic data. The model consists of an encoder-decoder structure, where the encoder learns to capture important features from the input data, and the decoder predicts the masked or target variable based on the learned features. TabNet leverages sparse instance-wise feature selection. The model employs a sequential multi-step architecture, where each step contributes to a portion of the decision-making process based on the selected features. It incorporates nonlinear processing of the selected features and the concept of ensembling through higher dimensions and multiple steps. These design choices contribute to TabNet's ability to effectively learn from unlabeled data, capture complex relationships, and improve overall performance in various tasks. In TabNet, the same D-dimensional features $f \in \mathbb{R}^{B \times D}$ are passed to each decision step, where B represents the batch size. This ensures that the model operates consistently on the input features across all decision steps within a given batch. An encoder is used to do the multi-step processing. TabNet incorporates a learnable mask, denoted as $M \in \mathbb{R}^{B \times D}$, to enable a soft selection of salient features in each step. By employing a sparse selection of the most important features, TabNet ensures that the learning capacity of each decision step is not wasted on irrelevant features. This approach enhances the model's parameter efficiency, allowing it to focus on the most relevant information and optimize its performance. Finally, A decoder is used to reconstruct the tabular features from the encoded representations produced by the encoder. The decoder component is responsible for transforming the encoded representations back into the original tabular feature space, enabling the reconstruction of the input data. We choose to utilize TabNet due to its self-supervised learning capabilities, which involve masking a portion of the elements in the dataset. This approach allows the model to gain an understanding of network traffic flow without

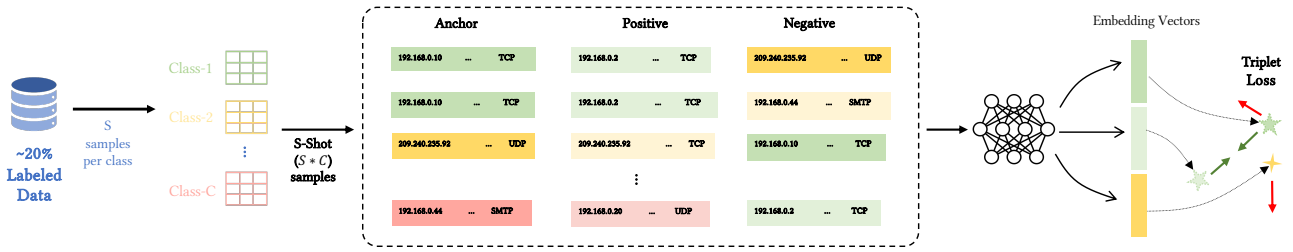


Figure 2: Overview of few-shot learning using contrastive training with triplet loss.

relying on explicit labels. In the second phase of our framework, the model is further trained contrastively, which enhances the discriminative power of the model. Finally, we introduce a sub-sampled KNN approach to make predictions and effectively detect intrusions in IoT networks. By leveraging these features of TabNet, we aim to improve the performance of our intrusion detection system.

2.2 Contrastive Training

Contrastive Training is a machine learning technique used for training models in a way that enhances their ability to discriminate between different instances or samples [53, 57]. It is commonly applied in tasks such as representation learning, where the objective is to learn a meaningful and compact representation of data. For example, the contrastive loss function is designed to minimize the distance between embeddings of positive pairs while maximizing the distance between embeddings of negative pairs. The objective is to update the weights of our embedding model (i.e., TabNet) in a way that satisfies this condition after each pass through the network. We use contrastive learning in such a way that it can tackle data imbalance effectively in the classifications task. In particular, enabling the model to focus on the distinctive features of minority class instances improves their representation learning process. In our work instead of using the traditional contrastive loss function, we use the triplet loss function.

2.3 FAISS and KNN

The FAISS library is designed to facilitate efficient similarity searches and clustering of dense vectors. It offers a wide range of comparison operations, including L2 distance, dot product, and cosine similarity. By incorporating indexing structures like hierarchical navigable small worlds (HNSW) [31] and navigating spreading-out graphs (NSG) [15], FAISS enables highly effective searching even in collections of billions of vectors. It is primarily implemented in C++ and relies on BLAS as its main dependency. Additionally, FAISS supports GPU acceleration for faster inference, allowing for both single and multi-GPU indexing using CUDA. Its Python interface ensures compatibility with all major deep learning frameworks. We utilize FAISS to index our training dataset (i.e., 20% of the total data) in the third phase of our proposed framework. Then, we employ our custom sub-sampled KNN classifier to make predictions, which are specifically tailored for our task and perform better than the classical KNN algorithm.

3 FS3: INTRUSION DETECTION FRAMEWORK

3.1 Task Formulation

We represent the input data as a two-dimensional matrix $\mathcal{X} = (x_1, x_2, x_3, \cdots, x_N)$, where $x_i \in \mathbb{R}^D$ $(1 \leq i \leq N)$ is a vector with D dimensional feature space. Each x_i is associated with a label y_i $(1 \leq i \leq N)$, where $y_i \in \{1, \cdots, C\}$. Thus, we have an N-sample dataset with C distinct attack categories. The feature vector $x_i \in \mathbb{R}^D$ is mapped to a label $y_i \in \{1, \cdots, C\}$ using a learnable function $y = f(x)$. In the supervised learning setting, f is estimated using a labeled training dataset. Unlike traditional classification tasks with roughly equal number of instances in each class, our focus in this work is to overcome the challenge of class imbalance issues. The class imbalance phenomenon is prevalent in all IoT datasets. That is, a few classes dominate the dataset with a significant number of instances, while the remaining classes have only a few instances. Furthermore, acquiring labeled data for each attack type is laborious and expensive. To overcome the class imbalance in datasets and minimize the labeling effort, we reformulate the task and assume that we only have access to L labeled training examples, where $L \ll N$. In our experiments, we utilize 20% of the labeled data. Our objective is to train a model using a subset of the original dataset with L labeled samples. Now $x_i \in \mathbb{R}^D$ $(1 \leq i \leq L)$ associated with a label y_i $(1 \leq i \leq L)$. Our framework also utilizes a few-shot learning paradigm where we randomly sample S labeled examples from each class.

3.2 Phase 1: Self-Supervised Learning

We use TabNet [3] as the backbone model for self-supervised learning. Specifically, we used the masking objective to mask 20% of the features in the input data. The remaining features are embedded into a high-dimensional vector space and are used to reconstruct the masked features. Figure 1 (left part) illustrates the SSL process. We take the embeddings of the categorical features, along with the numerical features, as inputs. These inputs are then fed through a series of dense layers, which form the Multi-Layer Perceptron (MLP). Each dense layer consists of multiple neurons or units and applies a non-linear activation function to transform the input data. The output of each layer is passed as the input to the next layer until the final layer, which produces the desired output. During this phase, a pre-trained encoder that has been trained using an unlabeled dataset is produced. In our work, we incorporate two dense layers with a dropout rate of 0.1%. The two dense layers have 200 and 100 neurons, respectively. The encoder utilizes an embedding

dimension of 100 for two (WUSTL-IIoT and BoT-IoT) datasets, while the third (WUSTL-EHMS) dataset utilizes an embedding dimension of 70.

3.3 Phase 2: Few-Shot Learning and Contrastive Training

We leverage Few-shot learning (FSL) with contrastive training to further train the pre-trained model (from Phase 1) by utilizing only a few labeled samples per class. In our framework FS3, we randomly select S labeled samples (i.e., S-shot) for each of the C classes. To train the model, we employ contrastive training that focuses on learning discriminative features for each class. Specifically, we use triplet loss function [18, 41, 52] to train the encoder contrastively. Unlike the traditional contrastive loss function, the triplet loss function incorporates triplets within each training sample (see Figure 2). A triplet consists of an anchor data point, a positive data point (belonging to the same class as the anchor), and a negative data point (belonging to a different class than the anchor). By utilizing triplets, the objective is to minimize the distance between the anchor and the positive point while simultaneously maximizing the distance between the anchor and the negative point during each gradient update. We can define the triplet loss as:

$$\mathcal{L}_{tri} = \sum_{y_a=y_p\neq y_n} [D_{ap} - D_{an} + m] \quad (1)$$

where y_a is the label of anchor sample, y_p is the label of positive sample, and y_n is the label of negative sample. D_{ap} is the distance between the anchor and the positive samples in the embedding space and D_{an} is the distance between the anchor and the negative sample, and margin (m) is a hyperparameter that controls the minimum distance between the anchor-positive and anchor-negative pairs. In our experiments, we use the Euclidean distance to calculate the distance.

Figure 2 illustrates how training data is prepared for contrastive training of the model using FSL and provides an overview of the contrastive training using triplet loss. Specifically, the total number of training samples to train the model is $S * C$. S takes values between 5 and 10 in our experiments. We also use a miner function to identify hard pairs from the training samples [51]. We use Multi-Similarity Miner (MSM) [33], which selects both the hardest positive and hardest negative samples within each similarity margin for each anchor. The loss function is then computed based on these selected samples. This approach effectively addresses the data imbalance issue between the samples from majority and minority classes, improving the overall discriminative power of the model. We conduct the contrastive training procedure five times for both 5-Shot and 10-Shot scenarios. We report the average results of the five runs in our experiments to show the robustness of the method.

3.4 Phase 3: Nearest Neighbor Classification

Phase 3 of FS3 does not involve any training or finetuning. We employ the FAISS library to create an index for efficient and scalable retrieval of similar network traffic data. This indexed data is then utilized for making predictions. This phase only selects 20% of the training data, instead of the entire dataset. We feed the samples into the fine-tuned encoder (from Phase 2) to generate embedding vectors of training samples. To further reduce the class imbalance issue, we introduce a sub-sampled KNN algorithm. Specifically, we define the weight of each class as follows:

$$W_i = \max(1 - \sqrt{t/p_i}, a) \quad (2)$$

In Eq 2, W_i is the weight assigned to the ith class. t represents a hyperparameter that controls the sub-sampling process, p_i denotes the fraction of samples belonging to the ith class, and a is a constant that defines the minimum weight assigned to each class. In our experiments, the value for a is set to 0.1. Particularly, We use t to sub-sample only the majority classes and the weight for a class i will only be reduced if $p_i > t$. That is, minority classes are not affected by it, whereas the Eq 2 ensures that as the relative number of samples increases for a class, the corresponding probability of reducing its weight also increases. To illustrate, consider a dataset where t = 10-5, the total number of samples in the training data is 860011, and the total number of samples for the ith class type is 152 (minority class). In this case, we can calculate pi as 152/860011, resulting in Wi = 0.7621. If ith class type is majority class with 56379 samples, then calculated pi is 56379/860011 and resulting Wi = 0.9876. This is the way minority class type can get weight comparable to majority class to reduce the impact of imbalance in the dataset. Our proposed sub-sampled KNN algorithm provides a balanced weighting approach for each class, akin to Goldilocks, as opposed to classical KNN which is biased toward majority classes and the weighted-KNN variant by weighting it with the inverse of class size that gives equal weight to each class. For our experiments using the two datasets WUSTL-IIoT and BoT-IoT, we set the value of t to 10^{-5}, while in the other experiment using the dataset WUSTL-EHMS, we use $t = 10^{-4}$. To perform the final classification using sub-sampled KNN, we repeat the experiment five times for each shot, utilizing the five fine-tuned encoders from Phase 2. In all experiments, we use the same 20% labeled data.

4 EXPERIMENTAL SET UP

4.1 Evaluation Criteria

We utilized the following quantitative metrics to assess the performance of various ML classifiers: (i) Precision (**Pre**), (ii) Recall (**Rec**), and (iii) F_1-**Score** [22, 34]. In our experiments, we calculated the macro average for all three metrics, which is the recommended method for evaluating models on imbalanced datasets. **Pre** represents the algorithm's ability to predict different types of intrusions accurately. **Rec** indicates the proportion of actual intrusions correctly detected by the algorithm. The F_1-**Score** is the reciprocal of the arithmetic mean of **Pre** and **Rec**, representing the harmonic mean of the two metrics.

4.2 Datasets used in the Experiments

WUSTL-EHMS. The WUSTL-EHMS dataset [17] originates from a real-time Enhanced Healthcare Monitoring System (EHMS) that captures network flow metrics and patients' biometrics. It encompasses four components: medical sensors, gateways, networks, and control with visualization. Patient data is collected by sensors and transmitted through gateways, switches, and routers to the server. However, there is a risk of data interception before it reaches the server, particularly due to man-in-the-middle attacks involving

Table 1: Statistics of WUSTL-EHMS dataset.

Class	Train	(%)	Test	(%)
Normal	10275	87.47	2855	87.44
Attack	1472	12.53	410	12.56

Table 2: Statistics of WUSTL-IIoT dataset.

Class	Train	(%)	Test	(%)
Normal	797261	92.71	221462	92.71
DoS	56379	6.56	15661	6.56
Reconnaissance	5932	0.69	1648	0.69
Command Injection	185	0.02	52	0.02
Backdoor	152	0.02	43	0.02

spoofing and data injection. The dataset comprises 43 features, including 35 network flow features and 8 patients' biometric features. Samples are labeled as `Normal` or `Attack`, with attacker MAC addresses assigned a label of 1 and the rest labeled as 0 based on the Source MAC address feature. Similar to the WUSTL-IIoT dataset, the WUSTL-EHMS dataset lacks separate training and testing datasets. As a solution, the dataset is randomly split into training and testing datasets since it lacks a timestamp feature. Table 1 presents the distribution of the dataset after randomly splitting it into training and testing splits (Train and Test). As observed, a large portion of the samples belongs to the `Normal` class, indicating the prevalence of normal instances in the dataset.

WUSTL-IIoT. The WUSTL-IIoT dataset [63] focuses on industrial Internet of Things (IIoT) data, generated using a supervisory control and data acquisition architecture in an IIoT testbed [62]. The testbed is designed to closely mimic real-world industrial systems and enable realistic cyber-attacks. The dataset comprises approximately 2.7*GB* of data collected over a period of around 53 hours. In accordance with the authors' recommendations, certain features are excluded from the dataset, namely 'StartTime', 'LastTime', 'SrcAddr', 'DstAddr', 'sIpId', and 'dIpId', as they uniquely identify attacks and could potentially bias the model. The dataset underwent preprocessing and cleansing, involving the removal of rows with missing or corrupted values and extreme outliers. The resulting dataset contains 39 attributes and covers four distinct attack types: Denial of Service (`DoS`), `Command Injection`, `Reconnaissance`, and `Backdoor`. It is important to note that the WUSTL-IIoT dataset does not have separate training and testing datasets. To overcome this, the dataset is sorted based on the timestamp of the data items. The first 80% of the samples are allocated for training, while the remaining samples were used for testing. Table 2 provides an overview of the distribution of data pertaining to different attack types in the training and testing splits. From the table, it can be observed that the `Normal` class comprises approximately 93% of the dataset. In contrast, the dataset contains a much smaller percentage of samples related to `DoS`, `Reconnaissance`, `Command Injection`, and `Backdoor` attacks, accounting for approximately 7%, 7%, 0.02%, and 0.01% respectively.

Bot-IoT. The Bot-IoT dataset [23], generated using simulated botnet scenarios, is structured into directories based on attack types,

Table 3: Statistics of Bot-IoT dataset.

Class	Train	(%)	Test	(%)
DDoS	1233052	52.52	385309	52.51
DoS	1056118	44.98	330112	44.99
Reconnaissance	58335	2.48	18163	2.48
Normal	296	0.01	107	0.015
Theft	52	0.002	14	0.002

with packet capture files collected. The data generation process involved the use of simulated IoT sensors. Statistical analysis techniques, including Correlation Coefficient and Joint Entropy, were applied to analyze the dataset features. To enhance prediction model performance, additional features were extracted and incorporated into the dataset [42]. It is recommended to label these extracted features, such as attack flow, categories, and sub-categories, to improve the model's effectiveness. The Bot-IoT dataset encompasses four sub-components within the IoT testbed: simulation, networking platform, feature extraction, and forensics analytics. Alongside `Normal` traffic data, the dataset includes instances of various attack types, including `DoS`, Distributed Denial of Service (`DDoS`), `Reconnaissance`, and `Theft`. In total, the dataset comprises 15 features. The Bot-IoT dataset comprises both a training dataset and a testing dataset. The distribution of data items across different classes in the training and testing datasets of Bot-IoT is illustrated in Table 3. Notably, the `Normal` class has 296 training samples (0.01% of the total), and the `Theft` class has 52 training samples (0.002% of the total). On the other hand, the training dataset has a higher percentage of `DDoS` and `DoS` samples, accounting for 53% and 45% of the dataset, respectively.

4.3 Competing Methods

4.3.1 State-of-the-art Models. **CNN-BiLSTM.** The CNN-BiLSTM architecture proposed by Sinha et al. [45] consists of multiple layers including a 1D-CNN layer, batch normalization, and Bi-LSTM layers. The 1D-CNN layer in CNN-BiLSTM utilizes the ReLU activation function and employs a maximum pool size of five. Batch normalization is applied to expedite the training process. Bi-LSTM layers are incorporated throughout the model in a progressive manner, doubling the kernel size at each iteration. Specifically, the first Bi-LSTM layer comprises 64 units, followed by a second layer with 128 units, and a final Bi-LSTM layer with 128 units. The last layer, which is a dense layer, is fully connected and employs the softmax activation function. To evaluate the performance of CNN-BiLSTM, we utilized the open-source code provided at the CNN-BiLSTM repository [8].

PB-DID. To leverage the benefits of shared features between datasets, we employ PB-DID [59] by utilizing an auxiliary dataset that shares similarities with the main dataset. Our approach consist of two experiments. In the first experiment, we train PB-DID on the Bot-IoT and WUSTL-IIoT datasets, which have four common features. We merge the samples of the Normal, DoS, and Reconnaissance class types from the WUSTL-IIoT dataset into the Bot-IoT dataset. The merged dataset is used for training the model, while the original Bot-IoT testing set is used for evaluation. In the second experiment,

we train PB-DID on the WUSTL-IIoT and WUSTL-EHMS datasets, which share ten common features. Similar to the first experiment, we merged the attack class samples from both datasets and the normal class samples from both datasets. Performance evaluation of PB-DID in this experiment is conducted using the provided open-source code [9].

DBN-IDS. We also compare our work with recently published work DBN-IDS [5]. In our work, we utilize the proposed architecture of the DBN model for all three datasets. The model consists of five RBMs stacked with (49, 128), (128, 256), (256, 128), (128, 128), and (128, 64) visible/hidden nodes per RBM, respectively. The output from the last RBM is connected to a fully connected layer with 5 nodes and 2 for both multi-class and binary classification using the Softmax function. To address the data imbalance issue, we employ a combination of SMOTE and undersampling techniques. For the performance evaluation of BBN-IDS in this experiment, we utilize the provided open-source code [10].

CTGANSamp: models were trained on the training datasets balanced using synthetic samples. To address data imbalance in the datasets, we also utilize synthetic instances generated using CTGAN [12, 55]. CTGAN employs a GAN-based model to generate synthetic tabular data based on the original tabular data. For the Bot-IoT training dataset, we add synthetic data using CTGAN, resulting in a total of 69215 samples for the Normal attack type and 71238 samples for the Theft attack type. However, we choose not to introduce additional synthetic samples for the DDoS, DoS, and Reconnaissance attack classes due to their already sufficiently large sample sizes. In the WUSTL-IIoT dataset, synthetic samples are added to the Reconnaissance, Command Injection, and Backdoor attack classes, resulting in 54447, 70867, and 60231 training samples, respectively. As Normal and DoS already have a sufficient number of samples, therefore no new samples are added to them. In the case of the WUSTL-EHMS training dataset, synthetic samples were only added to the attack class, increasing the total number of samples in that class to 9472. For our experiments, we trained the models on these balanced datasets using the cross-entropy loss function. We refer to these models as FNN-CTGANSamp and CNN-CTGANSamp.

Focal: models were trained using focal loss function. To evaluate our method, we utilize the focal loss function [13, 26], originally designed for object detection tasks. The focal loss is especially beneficial when there is a substantial class imbalance between foreground and background classes during training. In our implementation, we adopt the focal loss as a specialized loss function. The focal loss has two hyperparameters: γ and α. During training on the Bot-IoT dataset, we set α and γ as 2 and 1 for FNN, and 5 and 5 for CNN, respectively. When training on the WUSTL-IIoT dataset, we set γ and α as 2.5 and 0.15 for FNN, and 2.0 and 0.3 for CNN, respectively. Finally, for training on the WUSTL-EHMS dataset, we set γ and α as 2 and 2 for FNN, and 2 and 0.2 for CNN, respectively. In our experiments, we denote the models trained using the focal loss function as FNN-Focal and CNN-Focal, representing the FNN and CNN architectures, respectively.

4.3.2 Baseline Models. **ORG: models were trained using original datasets (i.e., without balancing the dataset).** For training the classifiers, we utilized the original training samples from the datasets and employed a traditional loss function, specifically the cross-entropy loss function.

RND: models were trained on the datasets, balanced using random oversampling. To address the class imbalance in the training datasets, we employed random oversampling. This technique involves duplicating samples from the minority classes randomly, thereby balancing the dataset. During the oversampling process, a representative sample from each subject was selected independently to maintain the integrity of the population [43]. After applying random oversampling to balance the training datasets, the number of training samples for each class type in Bot-IoT, WUSTL-IIoT, and WUSTL-EHMS became approximately 1233052, 797261, and 10275, respectively. In our experiments, we denote these models as FNN-RND and CNN-RND, respectively. Both models were trained using the cross-entropy loss function.

Dice: models were trained using dice loss function. To tackle data imbalance in image segmentation tasks, the widely used dice loss [44, 48] is employed. This loss is based on the dice coefficient, which measures the similarity between predicted and ground truth segmentation masks. The dice coefficient is computed by dividing the sum of the ground truth and predicted values by twice the intersection of the ground truth and predicted values. This coefficient serves as the foundation for calculating the dice loss. In our study, we applied the dice loss function to all three original training datasets for training deep learning (DL) models in both multi-class and binary-class classification tasks. The dice loss is used as one of the methods for performance comparison, with a smoothing parameter of 1×10^{-7} incorporated in the calculation. In our experiments, we refer to the models trained using the dice loss function as FNN-Dice and CNN-Dice, corresponding to the FNN and CNN architectures, respectively.

While all the baseline and state-of-the-art models are trained in a fully supervised manner, our proposed FS3 *takes a different approach by utilizing only* 20% *of the labeled data for the final classification task. This reduction in labeled data usage distinguishes our method from the others and highlights its potential for achieving comparable or even superior performance with a significantly smaller labeled dataset.*

5 RESULTS OF PERFORMANCE EVALUATION

5.1 Quantitative Analysis

Table 4 provides a comparison of the performance of all competing methods, including our proposed FS3, with respect to precision, recall, and F_1 score across all three datasets. For FS3, we present two types of few-shot learning: 5-Shot and 10-Shot. We conduct the Phase 2 experiment five times by randomly selecting 5 or 10 samples from the training sets. In Phase 3, we use the same 20% of labeled data to perform sub-sampled KNN classification five times and we report the average of the results obtained from these five iterations.

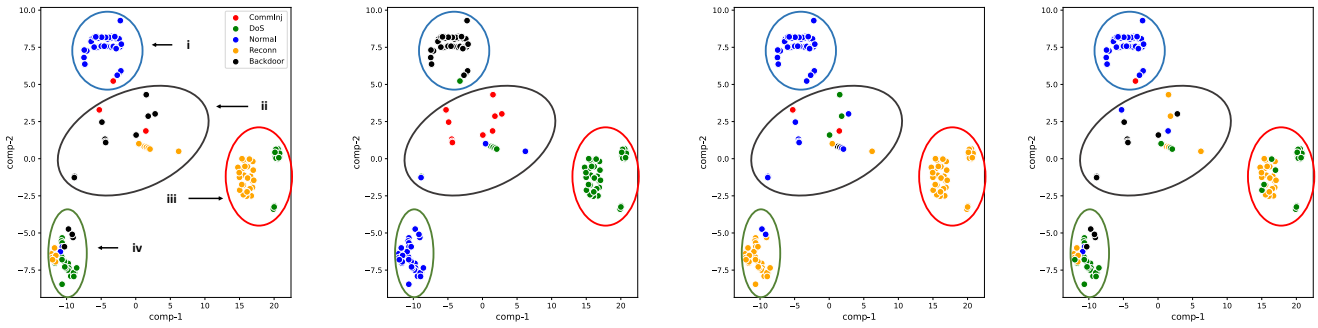
WUSTL-EHMS. FS3 shows significant improvement in performance over all the baseline and state-of-the-art models on the WUSTL-EHMS dataset with respect to all metrics. Specifically, in the AVG 5-Shot configuration, we observe substantial increases of 4% in precision, 33% in the recall, and 22% in F_1 score compared to

Table 4: Performance comparison of all methods on WUSTL-EHMS, WUSTL-IIoT, and BoT-IoT datasets.

DL Models	Classifier's	WuSTl-EHMS			WuStl-IIoT			BoT-IoT		
	Name	Pre	Rec	F_1	Pre	Rec	F_1	Pre	Rec	F_1
State-of-the-art Models	CNN-BiLSTM [45]	0.9010	0.7305	0.7851	0.7222	0.4349	0.5086	0.2477	0.2563	0.0778
	PB-DID [59]	0.4372	0.4998	0.4664	0.2105	0.1110	0.0214	0.1717	0.2037	0.1448
	DBN-IDS [5]	0.7362	0.7105	0.7222	0.4631	0.6339	0.3851	0.1185	0.5582	0.1652
	FNN-CTGANSamp [12]	0.9294	0.7364	0.7962	0.6628	0.3437	0.4050	0.4991	**0.8652**	0.5540
	CNN-CTGANSamp [12]	0.9107	0.7360	0.7921	0.7025	0.5122	0.5533	0.4298	0.7988	0.4536
	FNN-Focal [13]	0.9524	0.7369	0.8011	0.3854	0.293	0.3194	0.5559	0.6380	0.5784
	CNN-Focal [13]	0.9423	0.7338	0.7963	0.8198	0.6617	0.6974	0.6165	0.6325	0.5853
Baseline Models	FNN-ORG	0.9382	0.7359	0.7975	0.5254	0.4151	0.4578	0.5073	0.6345	0.5436
	FNN-RND	0.9339	0.7367	0.7974	0.5834	0.7630	0.5850	0.4990	0.4990	0.5275
	FNN-Dice	0.9336	0.5000	0.4665	0.1854	0.2000	0.1924	0.0900	0.2000	0.1241
	CNN-ORG	0.9284	0.7327	0.7927	0.7486	0.6142	0.6558	0.4434	0.5347	0.4211
	CNN-RND	0.9272	0.7362	0.7956	0.5894	**0.7720**	0.5942	0.5349	0.7843	0.5680
	CNN-Dice	0.0628	0.5000	0.1116	0.1854	0.2000	0.1924	0.0900	0.2000	0.1241
FS3 **(This work)**	AVG 5-Shot	**0.9794**	0.9812	0.9801	**0.8897**	0.6804	0.7017	0.6198	0.6030	0.5960
	AVG 10-Shot	0.9698	**0.9941**	**0.9809**	0.7847	0.7050	**0.7144**	**0.6314**	0.6297	**0.6046**

FNN-ORG. AVG 10-Shot exhibits significant superiority over CNN-ORG with respect to precision, recall, and F_1 score. Furthermore, when compared to specialized loss functions such as Dice and Focal loss, both AVG 5-Shot and AVG 10-Shot outperform with respect to all metrics. For instance, AVG 10-Shot achieves improvements of 1% in precision, 33% in the recall, and 22% in F_1 score compared to FNN-Focal. Similarly, AVG 5-Shot shows improvements of 3% in precision, 33% in the recall, and 23% in F_1 score when compared to CNN-Focal. Our FS3, under both AVG 5-Shot and AVG 10-Shot, outperforms all state-of-the-art models with respect to all metrics. When compared to CNN-BiLSTM, AVG 5-Shot exhibits a remarkable improvement of 8% in precision, 36% in the recall, and 24% in F_1 score. Furthermore, our FS3 consistently outperforms PB-DID (i.e., F_1 score of 0.4664), DBN-IDS (i.e., F_1 score of 0.7222), and DBN-IDS (i.e., F_1 score of 0.7222) by a substantial margin. FS3 leverages self-supervised learning in Phase 1, allowing the encoder to gain a deep understanding of the network traffic data without relying on class labels. In Phase 2, the encoder is trained contrastively using a small number of instances, enabling it to learn discriminative representations for the samples. Ultimately, FS3 achieves a better balance between precision and recall using the hyperparameter t, resulting in an improved overall performance of the model, leading to a notable improvement over other competing models.

WUSTL-IIoT. Our proposed approach exhibits substantial improvements over all the baseline and state-of-the-art models on the WUSTL-IIoT dataset, particularly with respect to precision and F_1 score. Specifically, in the AVG 5-Shot configuration, we observe a significant increase of 69% in precision, 63% in the recall, and 53% in the F_1 score compared to FNN-ORG. We observe a similar trend with both FNN-CTGANSamp and CNN-CTGANSamp. Similarly, AVG 10-Shot demonstrates significant superiority over CNN-ORG in terms of precision, recall, and F_1 score. Additionally, when compared to specialized loss functions such as Dice and Focal loss, both AVG 5-Shot and AVG 10-Shot outperform with respect to all metrics. For instance, AVG 10-Shot has improved 93% in precision, 140% in the recall, and 123% in F_1 score, compared to FNN-Focal. Similarly, AVG 5-Shot has improved 8% in precision, 2% in the recall, and 0.61% in F_1 score, compared to CNN-Focal. Both AVG 5-Shot and AVG 10-Shot outperform all state-of-the-art models with respect to all metrics. When compared to CNN-BiLSTM, AVG 5-Shot exhibits a notable improvement of 23% in precision, 56% in the recall, and 37% in F_1 score. Moreover, our proposed FS3 outperforms both PB-DID (F_1 of 0.0214) and DBN-IDS (F_1 of 0.3851) by a significant margin. Although our proposed approach may not achieve the same level of recall as CNN-RND and FNN-RND (i.e., 0.7720 and 0.7630, respectively), it outperforms them with respect to precision and F_1 score. The reason for the improved precision lies in Phase 2 of our approach, where we train the encoder contrastively using a small number of instances. Furthermore, by adjusting the hyperparameter t, we can achieve an optimal trade-off between precision and recall, leading to an overall enhancement in the model's performance. In contrast, random oversampling or undersampling approaches such as CNN-RND, FNN-RND, and DBN-IDS involve duplicating or removing existing samples, which can result in overfitting or underfitting of the model and ultimately lead to a lower F_1 score. Specialized loss functions like Dice and focal loss can have a potential drawback where they tend to prioritize the minority class, which may result in decreased performance on the majority class. The reason for the poor performance of other models could be their architecture is not robust and use traditional methods to overcome the data imbalance issue. Based on the performance using the WUSTL-IIOT dataset, we conclude that FS3 consistently



(a) Ground Truth **(b) CNN-RND** **(c) CNN-Focal** **(d) FS3**

Figure 3: Qualitative analysis of CNN-RND, CNN-Focal, and FS3 (5-shot) on WUSTL-IIoT dataset.

outperforms all competing models with respect to robust metrics such as precision and F_1 score. Additionally, it successfully finds a better balance between precision and recall.

BoT-IoT. In terms of precision and F_1 score, FS3 demonstrates significant improvement over all baseline and state-of-the-art models on the BoT-IoT dataset. Specifically, in the AVG 10-Shot setting, we observe a precision improvement of 24% and an F_1 score improvement of 11% compared to FNN-ORG. Similarly, FS3 AVG 10-Shot achieves a precision, recall, and F_1 score that are 42%, 17%, and 43% higher, respectively than CNN-ORG. Notably, the AVG 10-Shot and AVG 5-Shot, consistently outperform the FNN-Dice and CNN-Dice with respect to all metrics. In a similar fashion, our proposed approach FS3 AVG 10-Shot performs better than FNN-Focal with respect to precision and F_1 score by 13% and 4%, respectively. The improvement in precision can be attributed to the contrastive training of the encoder in FS3, which utilizes a small number of instances. This training approach allows the model to focus on extracting relevant and discriminative features. Additionally, by tuning the hyperparameter t, we can find an optimal balance between precision and recall, resulting in an overall enhancement in the model's performance. The same trend can be observed for CNN-Focal. Furthermore, our approach consistently outperforms the state-of-the-art models (CNN-BiLSTM, PB-DID, DBN-IDS) across all metrics. Although our proposed approach may not attain the same level of recall as FNN-CTGANSamp (i.e., 0.8652), it is worth noting that FNN-CTGANSamp exhibits poor precision and F_1 score. The reason for this discrepancy could be CTGANSamp, which adds a significant number of synthetic samples to the original training set, resulting in an increased overhead for the model and potential overfitting issues.

5.2 Qualitative Analysis

We draw 200 samples from the WUSTL-IIoT testing set and perform t-SNE projection to analyze how different methods performed on this sample. For qualitative analysis, we specifically chose CNN-RND, CNN-Focal, and FS3 (5-Shot), which achieved F_1 scores of 59.42%, 69.74%, and 74.45%, respectively, on the WUSTL-IIoT dataset. In Figure 3(a), the ground truth is visualized, where each data sample in the five attack classes is correctly marked using the colors red, green, blue, yellow, and black. To aid visual interpretation, we grouped the selected samples into four distinct groups, regardless of their attack class types. These groups are represented by encircled regions in four different colors: (i) blue, (ii) black, (iii) red, and (iv) green. Within the blue circle of the ground truth, the majority of samples belong to the DoS attack type, while a few of them are of the Command Injection type. When applying CNN-RND, all the samples within the blue circle are incorrectly classified as Backdoor. However, both CNN-Focal and FS3 correctly classify the DoS, and CNN-Focal missed a command injection. The black circle in the ground truth contains a large number of samples belonging to the Backdoor attack type, along with some samples of the Command Injection and Reconnaissance types. When using CNN-RND, all the Backdoor samples within the black circle are misclassified as Command Injection or DoS. Similarly, CNN-Focal classifies all of them as either normal or DoS. On the other hand, FS3 predicts the samples similar to the ground truth, correctly identifying most of them as Backdoor. However, it does misclassify some samples as Normal and DoS instead of Command Injection and Reconnaissance, respectively. The majority of samples within the red circle in the ground truth (Figure 3(a)) correspond to the Reconnaissance attack type, with some samples being DoS. However, in Figure 3(b) and (c), CNN-RND and CNN-Focal fail to correctly classify the samples within the green circle. On the other hand, FS3 demonstrates more accurate classification for most of these samples, except for a few cases where Reconnaissance attacks are misclassified as DoS attacks, as depicted in Figure 3(d). The samples enclosed within the green circle in the ground truth (Figure 3(a)) consist of attack types such as DoS, Reconnaissance, Normal, and Backdoor. In Figure 3(b), CNN-RND misclassifies all the Reconnaissance samples as Normal. Similarly, in Figure 3(c), CNN-Focal also misclassifies these samples as Reconnaissance. In contrast, FS3 (Figure 3(d)) appears to correctly classify all the samples within the green circle.

5.3 Ablation Study

In our implementation, we incorporate various components within our model, each of which plays a distinct role in determining the overall performance. Consequently, it becomes crucial to establish methods for quantifying the individual contributions of these

Table 5: Ablation study of proposed framework: FS3**.**

Components Name	Phase of FS3	Different Strategy of KNN	WUSTL-EHMS			WUSTL-IIoT			BoT-IoT		
			Pre	Rec	F_1	Pre	Rec	F_1	Pre	Rec	F_1
Self-Supervised Encoder	Phase 1	Classical	0.8434	0.7703	0.8007	0.7249	0.6488	0.6769	0.5248	0.5388	0.5043
		Inverse of Class Size	0.8567	0.8179	0.8357	0.6460	0.6829	0.6569	0.4790	0.6381	0.4891
Fine-tuned Encoder	Phase 2	Classical	0.9294	**0.9708**	0.9488	0.7095	0.7023	0.6925	0.5581	0.7952	0.5645
		Inverse of class Size	0.7965	0.9477	0.8458	0.6276	**0.7457**	0.6713	0.5518	**0.7995**	0.5531
	Phase 3	Sub-Sampled KNN	**0.9571**	0.9533	**0.9552**	**0.9774**	0.6734	**0.7154**	**0.5904**	0.7176	**0.5871**

components toward the overall model performance. This allows us to gain a deeper understanding of how each part influences the model's effectiveness and aids in the evaluation and optimization of our approach. Table 5 presents the ablation study of our proposed approach FS3 . We employ different strategies for applying the KNN algorithm after each phase of our proposed method. We conduct a comparison between our proposed sub-sampled KNN with classical KNN (without considering weights), and the Inverse of class size approach (weighting class types based on their inverse frequency). The results obtained in Phase 2, where the encoder is trained contrastively with either a 5-shot or 10-shot approach, show a significant improvement over the results obtained in Phase 1 across all metrics for all the KNN strategies employed in the final classification. This improvement highlights the effectiveness of training the encoder using a few instances with triplet loss function and the impact it has on the overall performance of the model. For instance, in Phase 3, utilizing our Sub-Sampled KNN on WuStl-EHMS, we observe significant improvements in terms of precision, recall, and F_1 score compared to classical KNN in Phase 1. Specifically, our proposed approach achieves approximately 13% higher precision, 23% higher recall, and 19% higher F_1 score on the WuStl-EHMS dataset. In Phase 3, when utilizing our proposed sub-sampled KNN, we observe significant improvements over classical KNN in terms of precision and F_1 score. Specifically, our approach achieves approximately 38% improvement in precision and 3% improvement in F_1 score compared to classical KNN. Although the recall score in Phase 3 using our approach is not as high as that of classical KNN or the inverse of class size strategy in Phase 2, it is still comparable to them. The notable advantage of our approach is the improved F_1 score achieved using the proposed Sub-Sampled KNN. These results highlight the effectiveness of our method in enhancing the overall performance of intrusion detection.

6 RELATED WORKS

6.1 Machine Learning Frameworks

Several ML frameworks have been proposed by researchers to address security issues in IoT [2, 4, 27, 30, 56]. Yang *et al.*[56] developed an intelligent IoT network ML framework using Software Defined Network (SDN) and Network Function Virtualization. Bagaa *et al.*[4] developed an ML framework using SDN and Network Function Virtualization to handle various IoT threats. Arachchige *et al.*[2] proposed PriModChain, a framework for ensuring the privacy of Industrial Internet of Things (IIoT) data. Liu *et al.*[27] presented a malicious node detection framework for handling a specific type of insider attack in IoT, called a conditional packet manipulation attack. Makkar *et al.*[30] proposed an ML framework for detecting spam in IoT networks. In addition, Dina *et al.*[12] utilized a DL model to balance data by incorporating synthetic data.

6.2 Using ML for Solving Miscellaneous Problems in IoT

Roy *et al.* [38] propose a two-layer hierarchical intrusion detection mechanism for IoT networks that uses machine learning. This model can effectively detect intrusions while satisfying the resource constraints of the IoT. By deploying multi-layered feedforward neural networks in the fog-cloud infrastructure, this model can utilize the resources in the fog layer to detect network attacks. Liang *et al.* [25] discuss the vulnerabilities of ML algorithms in detecting intrusions and how these algorithms can be used in launching cyber-attacks. Sun *et al.* [49] model botnet attacks using ML. Amouri *et al.* [1] present an intrusion detection system for mobile IoT, while Sivananthan *et al.* [46] combine SDN and ML techniques to manage IoT devices. Finally, Zheng *et al.* [60] discuss the challenges in applying privacy-preserving ML methods developed for cloud computing systems in the context of IoT.

Jha *et al.* [39] propose a technique for detecting unknown system vulnerabilities in IoT. Guerra *et al.* [16] observe that network traffic becomes obsolete over time, as attackers change their types and behavior. Wahab *et al.* [50] employ the Principle Component Analysis (PCA) method to study the change in the variance of the features across the intrusion detection data streams in IoT and present an online deep neural network that dynamically adjusts the sizes of the hidden layers to cope with these changes. Khan *et al.* [21] point out that the existing ML models used in cyber-security follow the black-box model and propose a method to solve this problem.

Ferrag *et al.* [14] conducted a study to compare the performance of centralized and federated deep learning with three popular deep learning approaches, using three different datasets. Zolanvari *et al.* [62, 63] recognized the significance of machine learning (ML) and big data analytics in securing both IoT and IIoT. They built a real-world testbed to conduct cyber-attacks and develop an IDS that uses ML algorithms to detect backdoors, command injection, and SQL injection attacks. Moustafa [32] proposed a testbed architecture that allows the creation of dynamic testbed networks for IoT, enabling the interaction of edge, fog, and cloud tiers. They tested the architecture by executing real-world scenarios, including

normal and attack situations, and collected a labeled dataset named ToN_IoT. Li *et al.* [24] present an ML framework for automated decision-making in spectrum sharing for regulatory spectrum management. Rehm *et al.* [37] develop a clinical decision support system for managing patients in intensive care units, whereas Meidan *et al.* [40] present an ML-based method for detecting unauthorized Internet of Things (IoT) device types and Salman *et al.* [40] propose an ML framework for identifying device type and malicious traffic in IoT. Liu *et al.* [28] conduct a survey of ML methods for identifying and detecting compromised IoT devices, and Ma *et al.* [29] propose an ML-based method for evaluating the trustworthiness of IoT devices.

In a separate study, Zeeshan *et al.* [59] propose a Protocol-Based Deep Intrusion Detection (PB-DID) architecture that compares features from UNSW-NB15 and Bot-IoT datasets using flow and Transmission Control Protocol (TCP). To address the imbalance issue in the dataset, the authors combine both datasets by considering features that fall into both flow and TCP categories. They classify the combined dataset into three categories: normal, DoS attacks, and DDoS attacks, using a deep neural network. Sinha *et al.* [45] propose a hierarchical model that combines 1D-CNN and Bi-LSTM layers. The LSTM is used to identify long-term temporal patterns in a dataset, while CNN is used to identify spatial features. The authors perform binary and multi-class classification on two state-of-the-art datasets, NSL-KDD and UNSW-NB15. To balance the two datasets, they use random oversampling. Belarbi *et al.* [5]propose a Deep Belief Networks (DBNs) based multi-class classification Network Intrusion Detection System (NIDS). DBN, a generative graphical model composed of stacked Restricted Boltzmann Machines (RBMs), serves as the foundation for their approach. To tackle the data imbalance problem, they employ both SMOTE and undersampling techniques. In our experiments, we compare the performance of FS3 with PB-DID, CNN-BiLSTM, and DBN-IDS.

Many of the works mentioned above ignore the issue of data imbalance, while others address it by adding synthetic data or using random oversampling or random undersampling to balance the datasets. In this paper, we use three unbalanced datasets from three different IoT domains and propose a novel framework to detect intrusions in IoT network traffic data. Experimental evaluation shows that our approach performs better than state-of-the-art approaches on all three datasets.

7 CONCLUSIONS

In the last decade, researchers have extensively utilized Machine Learning (ML) techniques to develop and deploy intrusion detection systems for computer networks. However, the focus on intrusion detection for IoT has been comparatively limited. We propose FS3, a novel framework for intrusion detection in IoT networks. FS3 comprises three phases: (i) self-supervised learning, which utilizes SSL to extract latent patterns and robust representations from unlabeled data, (ii) few-shot learning (FSL) and contrastive training, which enables the model to learn from a small number of labeled examples (i.e., 5-10 instances per class), and (iii) sub-sampled KNN-based classification, which selectively sub-samples instances from the majority class based on the distribution of the training data. Our proposed framework FS3 leverages only 20% of the labeled training samples for making predictions, reducing the reliance on a large amount of labeled data as well as minimizing the startling effect of extreme class imbalance. Through extensive experimental analysis on three diverse IoT datasets, our framework consistently outperforms fully supervised baseline approaches and state-of-the-art models in terms of precision and F_1 score. The proposed framework FS3, while utilizing only 20% of labeled data, achieves notable performance improvements of up to 42.39% and 43.95%, with respect to precision and F_1 score respectively, compared to fully supervised state-of-the-art models. These results highlight the effectiveness and efficiency of FS3 in detecting intrusions in IoT networks.

REFERENCES

[1] Amar Amouri, Vishwa T. Alaparthy, and Salvatore D. Morgera. 2020. A Machine Learning Based Intrusion Detection System for Mobile Internet of Things. *Sensors* 20 (2020), 461. Issue 2. https://doi.org/10.3390/s20020461

[2] Pathum Chamikara Mahawaga Arachchige, Peter Bertok, Ibrahim Khalil, Dongxi Liu, Seyit Camtepe, and Mohammed Atiquzzaman. 2020. A Trustworthy Privacy Preserving Framework for Machine Learning in Industrial IoT Systems. *IEEE Transactions on Industrial Informatics* 16, 9 (2020), 6092–6102. https://doi.org/10.1109/TII.2020.2974555

[3] Sercan Ö Arik and Tomas Pfister. 2021. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. Archive, 6679–6687.

[4] Miloud Bagaa, Tarik Taleb, Jorge Bernal Bernabe, and Antonio Skarmeta. 2020. A Machine Learning Security Framework for IoT Systems. *IEEE Access* 8 (2020), 114066–114077. https://doi.org/10.1109/ACCESS.2020.2996214

[5] Othmane Belarbi, Aftab Khan, Pietro Carnelli, and Theodoros Spyridopoulos. 2022. An intrusion detection system based on deep belief networks. In *Science of Cyber Security: 4th International Conference, SciSec 2022, Matsue, Japan, August 10–12, 2022, Revised Selected Papers*. Springer, 377–392.

[6] Emilie Bout, Valeria Loscri, and Antoine Gallais. 2022. How Machine Learning Changes the Nature of Cyberattacks on IoT Networks: A Survey. *IEEE Communications Surveys and Tutorials* 24, 1 (2022), 248–279. https://doi.org/10.1109/COMST.2021.3127267

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2016. BeyoundTrust. https://www.beyondtrust.com/blog/entry/iot-bots-cause-october-21st-2016-massive-internet-outage/.

[8] code link. 2020. CNN-BiLSTM Repo. https://github.com/razor08/Efficient-CNN-BiLSTM-for-Network-IDS.

[9] code link. 2021. PB-DID Repo. https://github.com/nuttysunday/Protocol-Based-Deep-Intrusion-Detection-for-DoS-Normal-and-DDoS-Attacks.

[10] code link. 2022. DBN based IDS Repo. https://github.com/othmbela/dbn-based-nids/tree/master.

[11] Ayesha S. Dina and D. Manivannan. 2021. Intrusion detection based on Machine Learning techniques in computer networks. *Internet of Things* 16 (2021), 100462. https://doi.org/10.1016/j.iot.2021.100462

[12] Ayesha Siddiqua Dina, AB Siddique, and D Manivannan. 2022. Effect of balancing data using synthetic data on the performance of machine learning classifiers for intrusion detection in computer networks. *IEEE Access* 10 (2022), 96731–96747.

[13] Ayesha S Dina, AB Siddique, and D Manivannan. 2023. A deep learning approach for intrusion detection in Internet of Things using focal loss function. *Internet of Things* (2023), 100699.

[14] Mohamed Amine Ferrag, Othmane Friha, Leandros Maglaras, Helge Janicke, and Lei Shu. 2021. Federated Deep Learning for Cyber Security in the Internet of Things: Concepts, Applications, and Experimental Analysis. *IEEE Access* 9 (2021), 138509–138542. https://doi.org/10.1109/ACCESS.2021.3118642

[15] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2017. Fast approximate nearest neighbor search with the navigating spreading-out graph. *arXiv preprint arXiv:1707.00143* (2017).

[16] Jorge Luis Guerra, Carlos Catania, and Eduardo Veas. 2022. Datasets are not enough: Challenges in labeling network traffic. *Computers & Security* 120 (2022), 102810. https://doi.org/10.1016/j.cose.2022.102810

[17] A. A. Hady, A. Ghubaish, T. Salman, D. Unal, and R. Jain. 2020. Intrusion Detection System for Healthcare Systems Using Medical and Network Data: A Comparison Study. *IEEE Access* (2020), 106576 – 106584. https://www.cse.wustl.edu/~jain/ehms/index.html

[18] Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737* (2017).

[19] Elizabeth A Holm, Ryan Cohn, Nan Gao, Andrew R Kitahara, Thomas P Matson, Bo Lei, and Srujana Rao Yarasi. 2020. Overview: Computer vision and machine learning for microstructural characterization and analysis. *Metallurgical and Materials Transactions A* 51 (2020), 5985–5999.

[20] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[21] Izhar Ahmed Khan, Nour Moustafa, Imran Razzak, M. Tanveer, Dechang Pi, Yue Pan, and Bakht Sher Ali. 2022. XSRU-IoMT: Explainable simple recurrent units for threat detection in Internet of Medical Things networks. *Future Generation Computer Systems* 127 (2022), 181–193. https://doi.org/10.1016/j.future.2021.09.010

[22] Youness Khourdifi and Mohamed Bahaj. 2019. Heart disease prediction and classification using machine learning algorithms optimized by particle swarm optimization and ant colony optimization. *International Journal of Intelligent Engineering and Systems* 12, 1 (2019), 242–252.

[23] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. 2019. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. https://research.unsw.edu.au/projects/bot-iot-dataset. *Future Generation Computer Systems* 100 (2019), 779–796.

[24] Li Li and Amir Ghasemi. 2019. IoT-Enabled Machine Learning for an Algorithmic Spectrum Decision Process. *IEEE Internet of Things Journal* 6, 2 (2019), 1911–1919. https://doi.org/10.1109/JIOT.2018.2883490

[25] Fan Liang, William Grant Hatcher, Weixian Liao, Weichao Gao, and Wei Yu. 2019. Machine Learning for Security and the Internet of Things: The Good, the Bad, and the Ugly. *IEEE Access* 7 (2019), 158126–158147. https://doi.org/10.1109/ACCESS.2019.2948912

[26] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[27] Liang Liu, Xiangyu Xu, Yulei Liu, Zuchao Ma, and Jianfei Peng. 2021. A Detection Framework Against CPMA Attack Based on Trust Evaluation and Machine Learning in IoT Network. *IEEE Internet of Things Journal* 8, 20 (2021), 15249–15258. https://doi.org/10.1109/JIOT.2020.3047642

[28] Yongxin Liu, Jian Wang, Jianqiang Li, Shuteng Niu, and Houbing Song. 2022. Machine Learning for the Detection and Identification of Internet of Things Devices: A Survey. *IEEE Internet of Things Journal* 9, 1 (2022), 298–320. https://doi.org/10.1109/JIOT.2021.3099028

[29] W. Ma, X. Wang, M. Hu, and Q. Zhou. 2021. Machine Learning Empowered Trust Evaluation Method for IoT Devices. *IEEE Access* 9 (2021), 65066–65077. https://doi.org/10.1109/ACCESS.2021.3076118

[30] Aaisha Makkar, Sahil Garg, Neeraj Kumar, M. Shamim Hossain, Ahmed Ghoneim, and Mubarak Alrashoud. 2021. An Efficient Spam Detection Technique for IoT Devices Using Machine Learning. *IEEE Transactions on Industrial Informatics* 17, 2 (2021), 903–912. https://doi.org/10.1109/TII.2020.2968927

[31] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[32] Nour Moustafa. 2021. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society* 72 (2021), 102994. https://doi.org/10.1016/j.scs.2021.102994

[33] Kevin Musgrave. 2023. PyTorch Metric Learning. https://kevinmusgrave.github.io/pytorch-metric-learning/.

[34] Thuy TT Nguyen and Grenville Armitage. 2008. A survey of techniques for internet traffic classification using machine learning. *IEEE communications surveys & tutorials* 10, 4 (2008), 56–76.

[35] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507* (2017).

[36] Alan Ramponi and Barbara Plank. 2020. Neural unsupervised domain adaptation in NLP—a survey. *arXiv preprint arXiv:2006.00632* (2020).

[37] Gregory B. Rehm, Sang Hoon Woo, Xin Luigi Chen, Brooks T. Kuhn, Irene Cortes-Puch, Nicholas R. Anderson, Jason Y. Adams, and Chen-Nee Chuah. 2020. Leveraging IoTs and Machine Learning for Patient Diagnosis and Ventilation Management in the Intensive Care Unit. *IEEE Pervasive Computing* 19, 3 (2020), 68–78. https://doi.org/10.1109/MPRV.2020.2986767

[38] Souradip Roy, Juan Li, and Yan Bai. 2022. A Two-layer Fog-Cloud Intrusion Detection Model for IoT Networks. *Internet of Things* 19 (2022), 100557. https://doi.org/

[39] Tanujay Saha, Najwa Aaraj, Neel Ajjarapu, and Niraj K. Jha. 2022. SHARKS: Smart Hacking Approaches for RisK Scanning in Internet-of-Things and Cyber-Physical Systems Based on Machine Learning. *IEEE Transactions on Emerging Topics in Computing* 10, 2 (2022), 870–885. https://doi.org/10.1109/TETC.2021.3050733

[40] Ola Salman, Imad H. Elhajj, Ali Chehab, and Ayman Kayssi. 2019. A machine learning based framework for IoT device identification and abnormal traffic detection. *Transactions on Emerging Telecommunications Technologies* (September 2019). https://doi.org/10.1002/ett.3743

[41] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[42] Muhammad Shafiq, Zhihong Tian, Ali Kashif Bashir, Xiaojiang Du, and Mohsen Guizani. 2021. CorrAUC: A Malicious Bot-IoT Traffic Detection Method in IoT Network Using Machine-Learning Techniques. *IEEE Internet of Things Journal* 8, 5 (2021), 3242–3254. https://doi.org/10.1109/JIOT.2020.3002255

[43] Gaganpreet Sharma. 2017. Pros and cons of different sampling techniques. *International Journal of Applied Research* 3, 7 (2017), 749–752.

[44] Chen Shen, Holger R Roth, Hirohisa Oda, Masahiro Oda, Yuichiro Hayashi, Kazunari Misawa, and Kensaku Mori. 2018. On the influence of Dice loss function in multi-class organ segmentation of abdominal CT using 3D fully convolutional networks. *arXiv preprint arXiv:1801.05912* (2018).

[45] Jay Sinha and M Manollas. 2020. Efficient deep CNN-BiLSTM model for network intrusion detection. In *Proceedings of the 3rd International Conference on Artificial Intelligence and Pattern Recognition*. 223–231.

[46] Arunan Sivanathan, Hassan Habibi Gharakheili, and Vijay Sivaraman. 2020. Managing IoT Cyber-Security Using Programmable Telemetry and Machine Learning. *IEEE Transactions on Network and Service Management* 17, 1 (2020), 60–74. https://doi.org/10.1109/TNSM.2020.2971213

[47] Ryan Steed and Aylin Caliskan. 2021. Image representations learned with unsupervised pre-training contain human-like biases. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. 701–713.

[48] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. 2017. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 240–248.

[49] Peiyuan Sun, Jianxin Li, Md Zakirul Alam Bhuiyan, Lihong Wang, and Bo Li. 2019. Modeling and clustering attacker activities in IoT through machine learning techniques. *Information Sciences* 479 (2019), 456–471. https://doi.org/10.1016/j.ins.2018.04.065

[50] Omar Abdel Wahab. 2022. Intrusion Detection in the IoT under Data and Concept Drifts: Online Deep Learning Approach. *IEEE Internet of Things Journal* (2022), 1–1. https://doi.org/10.1109/JIOT.2022.3167005

[51] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. 2019. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5022–5030.

[52] Zhenyao Wang and Tonghai Liu. 2022. Two-stage method based on triplet margin loss for pig face recognition. *Computers and Electronics in Agriculture* 194 (2022), 106737.

[53] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. 2020. What should not be contrastive in contrastive learning. *arXiv preprint arXiv:2008.05659* (2020).

[54] Yang Xin, Lingshuang Kong, Zhi Liu, Yuling Chen, Yanmiao Li, Hongliang Zhu, Mingcheng Gao, Haixia Hou, and Chunhua Wang. 2018. Machine learning and deep learning methods for cybersecurity. *IEEE Access* 6 (2018), 35365–35381.

[55] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional GAN. *arXiv preprint arXiv:1907.00503* (2019).

[56] Kai Yang, Yuanming Shi, Yong Zhou, Zhanpeng Yang, Liqun Fu, and Wei Chen. 2020. Federated Machine Learning for Intelligent IoT via Reconfigurable Intelligent Surface. *IEEE Network* 34, 5 (2020), 16–22. https://doi.org/10.1109/MNET.011.2000045

[57] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* 33 (2020), 5812–5823.

[58] Javier Zaurin and Pavol Mulinka. 2023. Pytorch WideDeep. https://pytorch-widedeep.readthedocs.io/en/latest/index.html.

[59] Muhammad Zeeshan, Qaiser Riaz, Muhammad Ahmad Bilal, Muhammad K Shahzad, Hajira Jabeen, Syed Ali Haider, and Azizur Rahim. 2021. Protocol-Based Deep Intrusion Detection for DoS and DDoS Attacks Using UNSW-NB15 and Bot-IoT Data-Sets. *IEEE Access* 10 (2021), 2269–2283.

[60] Mengyao Zheng, Dixing Xu, Linshan Jiang, Chaojie Gu, Rui Tan, and Peng Cheng. 2019. Challenges of Privacy-Preserving Machine Learning in IoT. In *Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*. ACM, 1–7. https://doi.org/10.1145/3363347.3363357

[61] Yaoyao Zhong and Weihong Deng. 2019. Adversarial learning with margin-based triplet embedding regularization. In *Proceedings of the IEEE/CVF international conference on computer vision*. 6549–6558.

[62] Maede Zolanvari, Marcio A. Teixeira, Lav Gupta, Khaled M. Khan, and Raj Jain. 2019. Machine Learning-Based Network Vulnerability Analysis of Industrial Internet of Things. *IEEE Internet of Things Journal* 6, 4 (2019), 6822–6834. https://doi.org/10.1109/JIOT.2019.2912022

[63] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain. 2021. WUSTL-IIOT-2021 Dataset for IIoT Cybersecurity Research, Washington University in St. Louis, USA, October 2021. https://www.cse.wustl.edu/ jain/iiot2/index.html.