mathematics

Article

Enhancing IoT Security: A Few-Shot Learning Approach for Intrusion Detection

Theyab Althiyabi *, Iftikhar Ahmad and Madini O. Alassafi

Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia; iakhan@kau.edu.sa (I.A.); malasafi@kau.edu.sa (M.O.A.)
* Correspondence: tmalthiyabi@stu.kau.edu.sa

Abstract: Recently, the number of Internet of Things (IoT)-connected devices has increased daily. Consequently, cybersecurity challenges have increased due to the natural diversity of the IoT, limited hardware resources, and limited security capabilities. Intrusion detection systems (IDSs) play a substantial role in securing IoT networks. Several researchers have focused on machine learning (ML) and deep learning (DL) to develop intrusion detection techniques. Although ML is good for classification, other methods perform better in feature transformation. However, at the level of accuracy, both learning techniques have their own certain compromises. Although IDSs based on ML and DL methods can achieve a high detection rate, the performance depends on the training dataset size. Incidentally, collecting a large amount of data is one of the main drawbacks that limits performance when training datasets are lacking, and such methods can fail to detect novel attacks. Few-shot learning (FSL) is an emerging approach that is employed in different domains because of its proven ability to learn from a few training samples. Although numerous studies have addressed the issues of IDSs and improved IDS performance, the literature on FSL-based IDSs is scarce. Therefore, an investigation is required to explore the performance of FSL in IoT IDSs. This work proposes an IoT intrusion detection model based on a convolutional neural network as a feature extractor and a prototypical network as an FSL classifier. The empirical results were analyzed and compared with those of recent intrusion detection approaches. The accuracy results reached 99.44%, which shows a promising direction for involving FSL in IoT IDSs.

Keywords: few-shot learning; intrusion detection system; cybersecurity; Internet of Things; prototypical networks

MSC: 68M25

check for updates

Citation: Althiyabi, T.; Ahmad, I.; Alassafi, M.O. Enhancing IoT Security: A Few-Shot Learning Approach for Intrusion Detection. *Mathematics* **2024**, *12*, 1055. https://doi.org/10.3390/math12071055

Academic Editors: Chinyang Henry Tseng, Woei-Jiunn Tsaur and Hsing-Chung Chen

Received: 19 February 2024
Revised: 15 March 2024
Accepted: 27 March 2024
Published: 31 March 2024

1. Introduction

The Internet of Things (IoT) is becoming increasingly important in various areas, such as smart cities, healthcare, manufacturing, smart government systems, and marketing. Connected IoT objects provide automation and aid in making fateful decisions. The rapid growth and heterogeneity of an increasing number of IoT devices have intensified cybersecurity challenges. The natural mobility of the IoT and the limited capability of computational resources are jointly regarded as security limitations that increase the severity of cybersecurity threats. The intrusion detection mechanism plays an essential role in securing IoT networks. Since the 1970s, traditional intrusion detection has been used mainly to detect attacks and intrusion in the computer field and to enhance security by monitoring networks [1]. In the past decade, network evolution has resulted in scaling and fasting of the network, which allows numerous applications to rely on the network and exchange critical data through the network. Traditional IDSs failed to fulfill the requirements of a large-scale advanced network in terms of detection accuracy. To overcome the capability issues in intrusion detection in traditional IDSs, researchers have focused on utilizing the advantages of machine learning (ML) and deep learning (DL) techniques, which have recently exploded in popularity. An ML-based intrusion detection system (IDS) is a simple system for classifying network traffic by using ML mainly to detect attack patterns or abnormal traffic [2]. Both ML and DL techniques have shown tremendous improvements in intrusion detection. However, even if ML is good for classification, other methods perform better in feature transformation. For instance, ML has limited tuning capabilities, whereas DL can handle large amounts of data and can be tuned in various ways. However, both the ML approach, which heavily relies on statistical characteristics, and the DL approach, powered by large high-quality datasets, encounter difficulties when attempting to learn from a small number of examples.

1.1. IDS Using ML and DL

The performance of ML and DL models is typically dependent on the size of the dataset; large datasets lead to good classification performance, whereas small datasets might lead to overfitting issues [3]. Network IDSs based on supervised DL techniques require a large amount of labeled data to be generalized successfully; however, gathering massive malicious data as samples to train DL classifiers is cost-prohibitive, and the field is continually developing, making these data useless or obsolete [4]. The process of data collection is difficult, challenging, and sometimes impossible. Security and privacy regulations are among the challenges faced in data collection, as is the lack of dataset availability attributable to rare occurrences, especially in IoT networks. Apart from the limitations and prohibitive dataset collection in DL and ML, the training technique is complex and consumes a large amount of computational resources, which constrains the IoT. Moreover, ML and DL IDS-based methods suffer from detection accuracy issues and cannot detect novel attacks in some cases, such as zero-day attacks [5,6]. Today's artificial intelligence (AI) applications (e.g., ML and DL) learn from large amounts of data, but the eventual objective of AI is to be as intelligent as humans. AI is inspired by the human brain; hence, machines should reduce the gap between AI and the human brain, avoid the complexity of the AI learning process, and easily learn new concepts by using only a few examples. Researchers have recently explored new AI techniques to fill this gap.

1.2. Few-Shot Learning

Few-shot learning (FSL) is the concept of accurately classifying objects by using a few samples in a training dataset. The first FSL model was proposed by Fink [7], who embedded samples in a kernel and successfully classified image objects to their proper class by using only one example instead of a large training dataset. FSL has since received much attention and is now a hot topic because of the scientific objective of AI in approaching human capabilities and the industry requirement for ensuring low learning costs. The following are some basic terminologies for FSL: N-way refers to the number of categories, while K-shot represents the number of samples in each category [8]. One-shot FSL simply means a one-sample-based configuration, while zero-shot FSL means the absence of samples, which means that the module learns from a distinctive defined feature. However, FSL is also the general term for all types of learning, known as low-shot learning. This method supposes the inclusion of one to three samples but is not limited to those numbers; in fact, it may contain fewer or more samples. The overall combination of N-way and K-shot is known as a support set, in which the Q-query is a test sample that is not embedded in the training set. An increase in the shot number (K-shot) improves the prediction process. Meanwhile, a low number of N-ways implies improved performance [9]. The embedding and use of FSL in IDS may aid in overcoming the issues of data collection, rapidly training the model by using only a few samples, and harnessing the ability to detect novel attacks. The advantages of few-shot learning classification include its strong adaptability, low overheads in terms of resources, and the ability to transfer across various scenarios easily [10]. However, existing FSL intrusion detection systems models are complex and may be inappropriate for low-power networks such as IoT, and their performance is still not satisfying. In IDSs based on Siamese networks, it is challenging to control randomness in pairwise selection and they perform poorly with imbalanced data. Several scholars' research has focused predominantly on studying FSL approaches in NIDS using common, well-known standard datasets. However, there is a discernible paucity of research specifically targeting Intrusion Detection Systems (IDSs) within the Internet of Things (IoT) domain. Even though the IoT network is distinguished by its extensive scale and diversity, it suffers from a diverse range of unpredictable attacks. Among the different types of FSL approaches, the most common are the Siamese network (i.e., suitable for one-shot models because it depends on two pairwise distance compressions) and the prototypical network [11]. This prototypical approach computes the distance between the query and support sets, in which the approaching component predicts the final result after comparing the query samples' distances to the shots of other support sets. This method differs from the Siamese network, which uses individual shots in a pairwise manner. The prototypical approach is suitable for multiclass classification; unlike the Siamese network, this approach is rarely studied in IoT IDSs. To the best of our knowledge, the dataset employed in our study has not previously been validated with few-shot learning (FSL) methodologies in IDSs. In this research, we propose a multiclass classification intrusion detection system (IDS) that leverages prototypical networks. The proposed model is specifically designed to investigate the performance of IDSs within Internet of Things (IoT) networks when only a few samples are available. Our model integrates few-shot learning (FSL) with a deep learning (DL) model, specifically a one-dimensional convolutional neural network (1D_CNN). This integration allows for efficient feature extraction and forms the backbone of our classification process.

The research is divided into six sections. The Section 1 presents the introduction. The Section 2 highlights most related literature reviews. The Section 3 illustrates the proposed method. The Section 4 shows the experimental implementation of the model and the results. The Section 5 discusses the results and highlights important findings. The Section 6 is the conclusion.

2. Literature Review

Intrusion detection is a rich area of study. Since the past decade, scientific researchers have focused on improving the performance of IDSs. Moreover, embedding various ML and DL algorithms in IDSs results in excellent performance and accuracy of the modeled algorithm compared with traditional algorithms. However, in the case of a shortage of anomalous samples, conventional ML cannot enhance the detection results, as ML and DL are highly dependent on a large amount of training data that enable the efficient training of models [12,13]. Classification is a difficult process. Several scholars have faced challenges in improving the accuracy of classification techniques in ML by using datasets of limited size because they have few features and because the classification model cannot generalize patterns shown in training data. Yang et al. [14] proposed a method for creating Gaussian distributions by using smoothness features. For outputs classified based on inputs, two inputs that are close to each other result in outputs that are also close to each other. However, the aforementioned methods suffer from issues of data replication and noise. Another approach involves two methods proposed by Andonie [15], who utilized fuzzy ARTMAP to train neural networks (NNs) on limited datasets. Nevertheless, the method is still limited and is unscalable.

Despite other attempts, FSL has achieved good classification results by using few samples. In contrast to typical ML approaches, FSL employs existing knowledge and experience to aid in the exploration of new tasks [16]. Intrusion detection approaches based on FSL can efficiently address the problem of a limited number of abnormal samples, reduce complexity, and enhance the detection rate and overall accuracy. Moreover, the application of FSL in intrusion detection has been reviewed in detail from different perspectives.

In terms of network intrusion detection, Chowdhury et al. [17] employed FSL by utilizing a CNN for feature extraction in attribute learning, and this approach was applied as an SVM input. An experiment was conducted using the KDD99 and NSL-KDD datasets, and experimental verification was performed. The findings suggest that the proposed model can effectively classify any type of attack into five primary categories with an accuracy level of 97.5%. The model can identify rare attack types by including only a few samples in learning and can handle the sample imbalance problem; the proposed model operates similarly to a hybrid CNN–SVM model. However, each model was trained separately. Hence, this kind of training is ineffective at learning the feature representation of sample data. In addition, scalability issues may occur when dealing with large amounts of data as a result of the complex structure caused by utilizing an SVM with a Gaussian kernel function as a classifier [18].

Similarly, YU and the Biennial Informatics Network (BIAN) [19] employed the advantages of deep neural networks (DNNs) and CNNs to model an IDS via FSL. The aim of the model was to overcome the lack of abnormal samples for effectively training advanced IDSs. Some attack categories involve an extremely limited number of samples, which hinders an IDS from detecting and referring to its proper class, especially for unbalanced datasets. The multistage model used only a few samples in the support and testing dataset and embedded DNN and CNN for use in feature extraction. The Euclidean distances between samples were also computed. With only 1% of the dataset used, the model achieved a high accuracy, which was 92.34% for the NSL–KDD dataset and 85.75% for the KDDTrainC + dataset. Moreover, compared with those of other approaches, the detection rates of rare analytes, such as U2R and R2L, were fivefold greater for a small sample size. Both the accuracy and false alarm rate were high, enhancing the overall IDS performance. In addition, FSL depended on a balanced dataset only.

Hindy et al. [20] presented a novel approach based on FSL (one shot) to enhance IDSs by using Siamese networks. The main challenge in current ML-based IDSs is that supervised ML largely depends on a large number of cyber-attack incidents to be learned as training requirements prior to achieving high accuracy detection. In addition, retraining is needed when a new class of cyber-attack cannot be observed (zero-day attack). The proposed approach utilized FSL to overcome the lack of cyber-attack incident numbers needed for training and used only a single incident for each class. Furthermore, Siamese networks use similarity pairs to differentiate classes. Similarity pair comparisons in Siamese networks replace distinctive features. In contrast to traditional ML approaches, a Siamese network needs to be directly connected to networks to learn similarities from previous classes, consequently allowing an adapted IDS to detect new attacks without the need for retraining. However, the model mentioned above did not support fewer than three classes in each dataset, and the model could not be trained effectively given the 50% similarity output. Furthermore, the technique of pair selection was performed randomly with the constraint that the number of similar and dissimilar pairs must be equal. Adding distinctive pairs or devolving a technique for pair combination might enhance the overall accuracy.

Furthermore, an FSL-based NIDS was designed by Xu et al. [21], who developed network intrusion detection based on meta-learning associated with FSL for traffic classification. This method was developed to overcome the limitations of conventional ML IDSs. However, numerous attack incidents, such as zero-day assaults, are needed for training; such incidents are rare in some cases and difficult to collect. On the basis of the meta-learning system, the advantages of DNNs were utilized to develop the approach known as FC-Net. FC-Net included two architectures: F-Net and C-Net. F-Net was used for feature extraction, while C-Net was used for feature compression in the network. This system could solve a binary classification problem by creating sufficient prior knowledge, which was achieved by generating large numbers of mapped pair features, comparing them, and calculating the proper classification of whether an attack was real. As the model was well trained on the original dataset, on the basis of sufficient prior knowledge, FSL could be applied using only a few samples and achieved good performance and high accuracy in detecting attacks. Furthermore, novel attacks could be detected. The model achieved 98.88% accuracy and 99.62% accuracy on different datasets. Although the model was scalable and could be applied using various algorithms, the approach ignored the worldwide spatial distance between classes, which is detrimental to the further development of recognition accuracy [22].

Wang et al. [22] developed a novel approach for FSL based on the Siamese capsule network to enhance network intrusion detection with unsupervised training data. The lack of training data is one of the obvious challenges of advanced network IDSs, as it strongly affects the performance and detection accuracy. In addition, this approach has failed to detect unknown attacks. The proposed approach aimed to address the abovementioned issues. The training data consisted of three types: normal attack, sufficient attack, and scarce attack. In the training phase, K-means clustering was used to resample unsupervised row data from various known attack types and normal traffic to determine subtype numbers, which were subsequently used to construct FSL classes (C-way). The capsuleNet method was used to directly extract features. The resampled balance data extracted from the raw data and the data from scarce attacks were embedded as a training dataset to train the Siamese capsule network. The value K (number of samples in each class) was adaptive; this feature means that the number of samples in each class (C-way) varies. In an imbalanced dataset, FSL distinguishes between normal and abnormal sets based on learned distinctive features, whereas a balanced FSL training set represents a support set for testing to detect anomalous behavior. In the testing phase, a similarity measurement was used to classify test samples from the support set based on the most similar samples by using an extracted feature in the Siamese capsule network. Although the proposed approach achieved a high accuracy rate of 96.26%, the model was dependent on two parallel Siamese capsule network mechanisms, and the model is difficult to apply to actual IDSs. Furthermore, the approach requires considerable computing resources, and time information for the experiment and detection process was missing in the study.

Liang et al. [23] introduced a MicroserviceOriented intrusion detection model based on variational FSL to address distributed imbalanced data issues in the industrial IoT. Intrusion detection is a part of the microservice distributed industrial IoT environment. Traditional deep learning algorithms typically require a tremendous amount of training data to enhance detection performance in a large-scale interconnected IoT distributed environment. However, in such an IoT environment, the resources of each node are limited, and storage and computational constraints hinder the fulfillment of deep learning training data requirements. Furthermore, dealing with imbalanced datasets with low computing resources is another challenge for enhancing intrusion detection performance. Therefore, OICS-VFSL is proposed to overcome these gaps. The "optimized intra/interclass structure-based variational FSL" model incorporates FSL with variational feature representations. The model is simply divided into intraclasses and interclasses, the first for distance optimization by using variational feature representation and the other for feature concatenation. The detection rate of intelligent multiclasses by using imbalanced data exceeded 0.98 for attack data and 0.99 for normal data. However, additional experimental evaluation is needed because the results may change when the proposed approach is applied to a complex environment with a distributed IoT.

Similarly, Zhou et al. [24] presented FSL-SCNN, which is an intelligent image anomaly detection model that utilizes FSL based on a Siamese network and a CNN to cope with scarce imbalanced data in industrial CPSs. Several AI techniques may be used to enhance the detection of abnormal or attack signals generated from various surveillance and other sensor devices. However, in such a hybrid cyber-physical industrial environment, real-time detection may be challenging because of the large number of complicated structures and large amounts of data generated by various distributed devices. In an attempt to enhance the detection accuracy and mitigate overfitting issues when using only a few samples, the FSL-SCNN model was developed to calculate the distance based on the FSL input samples by using SCNN to determine the optimal feature representation and to effectively detect and classify cyber-physical attacks, including novel attacks utilized by the Siamese network. The overfitting problem was resolved by reducing the original features to a less dimensional representation. Although the model has high performance compared with other models, embedding large amounts of data and more complex scenarios in the experiment may change the results.

In another work, a novel network intrusion detection method integrating FSL was introduced by Iliyasu et al. [4] based on incorporating two learning techniques: a supervised autoencoder and discriminative representation learning. Current ML-based IDSs have a substantial rate of false alarms; as a result, this sector is a hotbed of research. In contrast, DL instruction detection approaches aid in mitigating the false alarm rate. However, large datasets are required for DL algorithms. Dataset collection is difficult, especially when the size of the dataset is large; moreover, in the field of cybersecurity, regular updates occur, and several new attacks emerge. Therefore, the proposed method aimed to overcome the limitations of DL and ML intrusion detection methods based on the use of an FSL technique, which can enhance IDS accuracy with only a few samples. The method includes two main phases: first, a supervised autoencoder is merged, and discriminative representation learning is used to develop a feature extraction model. In the second phase, a classifier is trained over a feature extractor; thus, the classifier model can generalize from only a few samples. Although the model reached a 99.8% detection rate by using only a few samples for certain types of attacks, it performed poorly for others. (Table 1) summarized literature review that utilized FSL in intrusion detection systems.

In an IoT MQTT network, Alaiz et al. [25] presented a novel intrusion detection method for detecting intrusion in an IoT environment that uses the MQTT protocol. The proposed model utilized XGBoost, LSTM, and GRU for a strong IDS to classify three classes of attacks. The model demonstrated remarkable validation performance on an MQTT dataset via multiclass classification, as 93.37% of the LSTM images were used, whereas 96.08% of the GRU images were used to indicate higher results. However, the model was tested on only three classes. On one dataset, additional classes and datasets may aid in better validation and lead to more accurate results.

Similarly, a deep learning IDS for IoT networks based on a DNN was proposed by Khan et al. [26], and the model utilizes an ANN on an IDS to distinguish between normal traffic and different types of attacks. The classification performance reached a high accuracy of 99% in binary classification and 98% in multiclass classification.

Prajisha et al. [27] presented an efficient IDS that combines the chaotic SALP swarm algorithm with LightGBM and achieved outstanding accuracy on a variety of datasets. The accuracy of ECSSA-LightGBM is 99.38% for MC-IoT; that of ECSSA-LightGBM is 98.91% for MQTT-IoT-IDS2020; and that of ECSSA-LightGBM is 98.35% for MQTTset. However, the proposed IDS is inapplicable for detecting CoAP attacks in the IoT network.

Hindy et al. [28] conducted several experiments to study the performance of an IDS based on ML on the MQTT-IOT-2020 dataset via six different ML techniques: logistic regression (LR), Gaussian naïve Bayes (NB), k-nearest neighbors (k-NN), support vector machines (SVMs), decision trees (DTs), and random forests (RFs). The results of the experimental study indicate significant accuracy. However, ML methods perform well only when the data are sufficient for training.

Zeghida et al. [29] developed an ensemble learning-based IDS that included bagging, boosting, and stacking to enhance the accuracy and strength of classifying attacks in the MQTT IoT network. The model has high accuracy. In terms of stacking, accuracy averages in the range of 95.43% to 95.38% are obtained for RF, bagging, AdaBoost, HistGradientboost, and the XGB classifier. Model tests that use one dataset and an additional dataset may provide accurate validation.

Similarly, Chesney et al. [30] presented an MQTT IoT- IDS based on both DL and ML techniques. The techniques for DL included random forest, logistic regression, decision tree, K-nearest neighbors, and support vector machine (SVM) methods; for DL techniques, a CNN was utilized. The experiments show good accuracy, with a binary average of 90%. However, the DL techniques used in this method failed to detect aggressive scan attacks, as the accuracy was 63%.

Mosaiyebzadeh et al. [31] proposed a model for NIDS DL based on IoT networks that combines three techniques: CNN, RNN, and LSTM. An experiment was conducted on the MQQ-IoT2020 dataset, and the average accuracy was 97%. However, given the nature of DL, the model requires a large dataset. The experiment should be extended to other IoT datasets for further verification.

Table 1. Summary of reviewed literature on IDSs based on FSL.

Reference	Dataset	Performance	Limitation
[17]	KDD99 NSL-KDD	Accuracy = 97.5%	The training is conducted in two totally separated processes which may affect the efficiency and the scalability.
[19]	NSL-KDD KDDTrainC + UNSW-NB15	Accuracy = 92.34% using less than 2% dataset	False alarm rate is high. FSL requires a balanced dataset.
[20]	CICIDS2017 KDD Cup'99 NSL-KDD	Accuracy = 84% for CICIDS2017 88% for KDD Cup'99 and 91% + for NSL-KDD	The model does not support less than three classes. Pair selection in SN requires enhanced technique as it is carried out randomly with equality constraints for both similarity and dissimilarity.
[21]	ISCX2012FS CICIDS2017FS (real traffic + source data)	Accuracy = 98.88%, 99.62%	The approach does not consider the worldwide spatial distance between classes, which is detrimental to the further development of recognition accuracy.
[22]	CICIDS-2017 UNSW_NB15	Accuracy = 95.25%, 96.26% 91.28%, 93.69%	Depends on two parallel Siamese capsule network mechanisms, making it challenging to apply to actual intrusion detection systems. Real-time detection is essential; detecting time info is missing in the paper. Requires huge computing resources.
[23]	NSL-KDD CIC-IDS2017	DR = 99%	Requires more experiment evaluation in complex IoT environments. Should have used a dataset designed especially for IoT.
[24]	UNSW-NB15	F1 = 93.06%	Applied only in a simple scenario. The model is required to be implemented in more complex scenarios for more evaluations and accuracy improvement.
[4]	CIC-IDS2017 NSL-KDD	Accuracy = 81%	Performance requires improvement. Dataset used in the comparison evaluation experiment is different than that used in the main model experiment.

Nonetheless, the model is quite complex and may consume large amounts of resources, which is a challenge in IoT networks.

3. Proposed Method

The proposed methodology consists of four phases: dataset selection, dataset preposessing, feature extraction, and classification. Figure 1. illustrates a broad overview of the methodology.

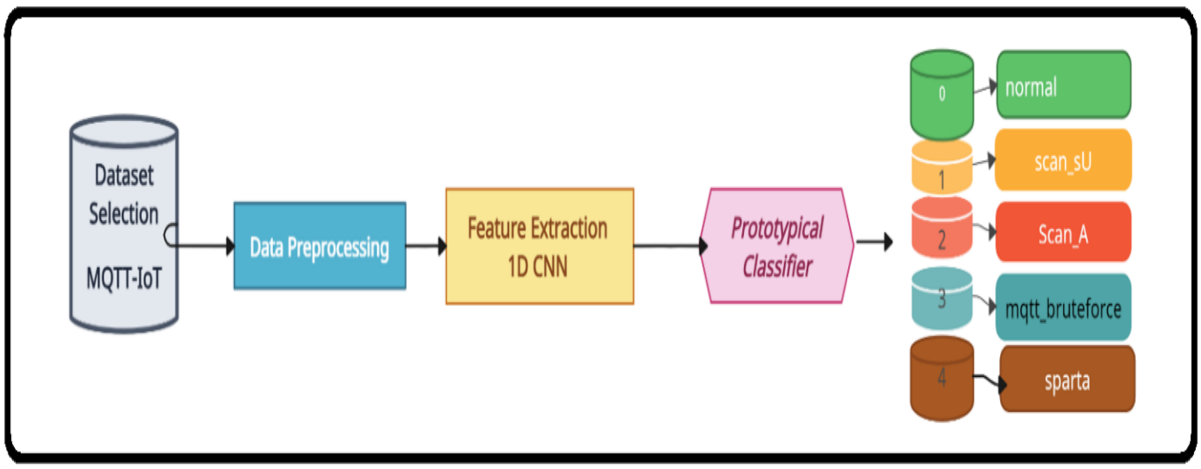

Figure 1. Proposed model 1D_CNN with prototypical networks.

3.1. Dataset Selection

Given that related work has focused mainly on IoT networks, we conducted comprehensive scholarly reviews of various types of IoT datasets. On the basis of these reviews, we selected recent, network-simulated, and various scenarios involving different types of attacks. The dataset should also be reliable and effective for IDS classification and feature extraction. Therefore, the experiments were conducted using the MQTT-IoT-IDS2020 dataset. Furthermore, we used a common IDS dataset to emphasize that our work can be evaluated with other works by using the FSL approach in IDS. To the best of our knowledge, the MQTT-IoT-IDS2020 dataset has not been used in FSL IDSs.

3.1.1. MQTT-IOT-IDS2020 Dataset

The MQTT-IoT-IDS2020 dataset is an IoT network dataset generated by Hendy et al. [32]. The dataset contributes to the IoT cybersecurity committee and is considered the first dataset simulated in the IoT MQTT network. The dataset is generated using various IoT sensor devices and aimed at recoding five different types of normal and attack behaviors. The dataset mainly consists of five classes: one normal sample record and four different types of attack. The first attack type is called Scan_A, which represents an aggressive scan attack. The second attack is recorded as Scan_sU, which refers to a UDP scan, and DDP stands for the user diagram protocol. The third one is Sparta, which is short for Sparta SSH brute force attack. The fourth record, MQTT_BF, represents MQTT brute force.

3.1.2. CIC_IDS2017 Dataset

The term deception in the "CICIDS2017" dataset typically consists of three parts: CIC, IDS, and 2017. CIC stands for the Canadian Institute for Cyber Security, while IDS refers to an IDS; the last number, 2017, is the date of release. The CICIDS2017 dataset [33] is one of the most common datasets used in IDSs. This scheme has been widely emphasized in several IDS studies. The dataset is rich in content because it contains various types of intrusion attacks. Among the 15 classes, 1 class is normal and the rest are different types of cyber intrusion attacks, such as botnet, SQL injection, and DDoS attacks.

3.2. Dataset Preprocessing

Preprocessing is a crucial step before training. Data preprocessing steps generally include filtering, cleaning, normalization, and removing unnecessary values in the datasets. Accuracy and classification performance mainly depend on the dataset's reliability. A dataset that contains noise or inaccurate data may cause misclassification and affect the prediction efficiency [34].

A representation of the correlation matrix of numeric features in the MQTT-IoT2020 dataset is illustrated in Figure 2, and Table 2 shows the total number of samples in the dataset. In the cleaning process, all nonvalues were removed; three columns were discarded: namely, the SADDR and DADDRS columns. Regarding properly handled categorical information, a prototype can be encoded with one hot encoding, the port number for (sport and dport) resulting in high cardinality; thus, one hot encoding would be extremely sparse. Instead, we treated them numerically, which is not a straightforward progress because it contains hexadecimal values. We also converted the Boolean data into integer values to normalize the categorical data immediately after encoding. Furthermore, any class can be included because each class has sufficient samples for FSL. Furthermore, in the CICIDS2017 dataset, we additionally excluded three classes to reduce the number of classes and avoid overnormalizing these classes. The total number of CICIDS2017 records in the dataset is 2,803,743, with 79 feature dimensions. In addition, we removed the unnecessary columns such as: "Flow Packets/s", and "Flow Bytes/s".

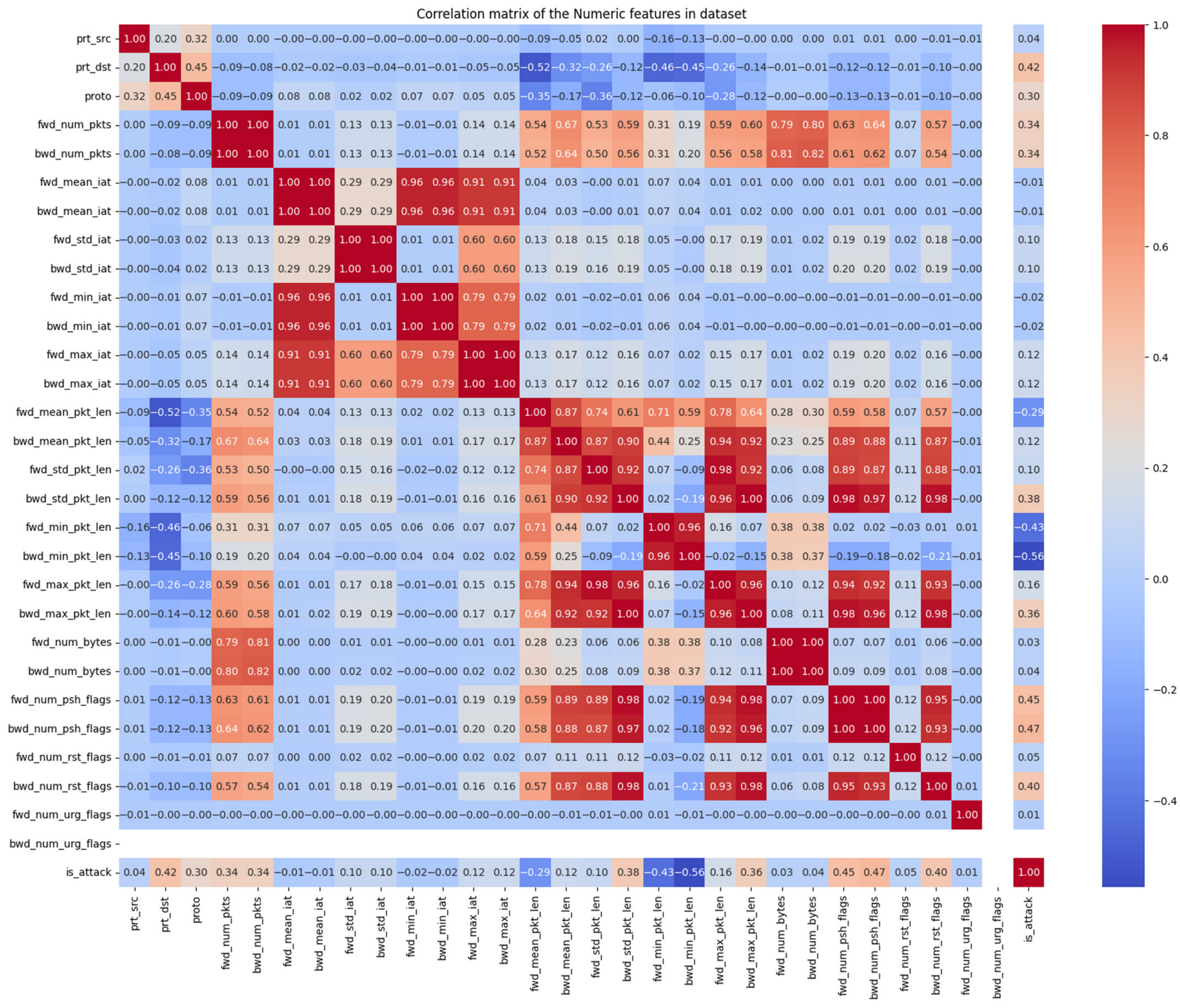

Figure 2. Correlation matrix of the numeric feature in MQTT-IoT2020 dataset.

Table 2. Total number of samples in the dataset.

Sample Type	Number of Samples
normal	188,378
scan_sU	22,434
scan_A	19,907
mqtt_bruteforce	14,544
sparta	14,116

3.3. Proposed FSL Model Architecture

The proposed model consists of two main steps. The first is a 1D CNN with five layers initially utilized for feature extraction. The second is where the output data are transferred to an FSL classifier; in our case, we applied a prototypical network technique for multiclass classification.

3.3.1. Feature Extraction (1D CNN)

Recently, numerous scholars have developed IDSs based on the CNN technique. Various methods have been studied using CNNs, such as 1D CNNs, two-dimensional CNNs, and three-dimensional CNNs. The 1D FSL (1D CNN) is a deep learning model that can analyze historical and sequential data and network traffic flows; this model has demonstrated outstanding performance in IDSs and has been extensively studied in various IDS studies. On the basis of previous reviews by scholars, we believe that 1D CNN is suitable for feature extraction, especially for complex IoT networks, for the following reasons:

- The ability to extract vital exceptional patterns from raw data and excellent performance in predicting anomalies in packets in the traffic flow.
- The complexity and dimensionality of the input data are minimized, which can contribute to improving the efficiency and accuracy of an IDS.
- Overcoming the two major challenges in an IDS, which are noise and an imbalanced dataset [35].
- Adapting to various kinds of network attacks, such as DoS, DDoS, and PortScan attacks, by employing a variety of structures and parameters [36].

For feature extraction, a preprocessed dataset was passed to one dimension of the 5-layer convolutional neural network model. The dataset was divided into 60% training set and 40% temporary set: X_train, X_temp, y_train, and y_temp. In addition, the temporary set was split equally into validation and test sets: (X_val, X_test, y_val, y_test). Furthermore, the training data were balanced using RandomOverSampler. Balanced data allow the next step to be taken to take advantage of preparing the weighted training, test, and validation datasets. Therefore, we created DataLoader without the sampler (as the data are already balanced). The training rate was set to (0.01), and to avoid overfitting issues, we set the regularization parameter to (0.0001) and the batch size to 128.

The five layers of a 1D CNN are utilized to extract complex features from the MQTT-IoT2020 dataset in IoT networks. Figure 3 shows additional details about each layer. The input data are expected to have a dimension of input_dim. The 1D CNN is reshaped to have a single channel (1) because it is a 1D convolution neural network, Conv1d. The conv1d is applied with 64 filters (out_channels), a kernel size of 3, and a padding of 1. This step is followed by batch normalization and ReLU activation. This process is repeated with increasing numbers of filters (64 > 128 > 256 > 256 > 128), batch normalization, and rectified linear unit (ReLU) activation after each convolution. Convolutional layers: the convolutional layers are designed to capture hierarchical features from the input data. Fully connected (linear) layers: after the convolutional layers, the data are flattened (torch.flattening) and passed through a fully connected layer with input_dim * 128 input features (128 features from each of the 128 filters in the last convolutional layer). Output layer: the output layer consists of a linear layer with num_classes output units for classification. The sample class shape is torch. Size (25, 28) torch, Size (25) torch, Size (50, 28) torch, and Size (50). The 1D five-layer CNN is the backbone of the prototypical network classifier and acts as an input.

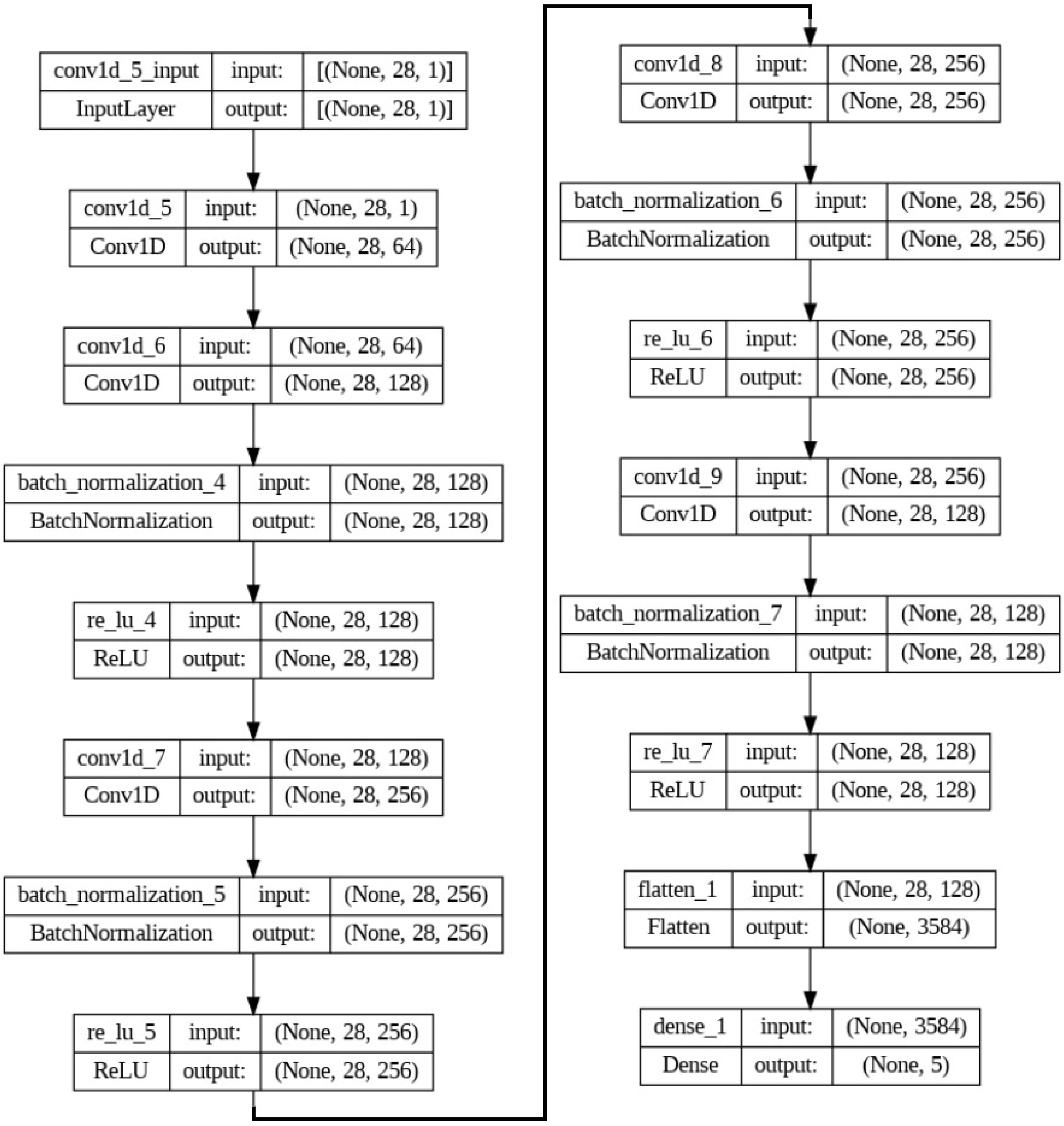
conv1d_5_input | input: | [(None, 28, 1)]
InputLayer | output: | [(None, 28, 1)]
conv1d_5 | input: | (None, 28, 1)
Conv1D | output: | (None, 28, 64)
conv1d_6 | input: | (None, 28, 64)
Conv1D | output: | (None, 28, 128)
batch_normalization_4 | input: | (None, 28, 128)
BatchNormalization | output: | (None, 28, 128)
re_lu_4 | input: | (None, 28, 128)
ReLU | output: | (None, 28, 128)
conv1d_7 | input: | (None, 28, 128)
Conv1D | output: | (None, 28, 256)
batch_normalization_5 | input: | (None, 28, 256)
BatchNormalization | output: | (None, 28, 256)
re_lu_5 | input: | (None, 28, 256)
ReLU | output: | (None, 28, 256)
conv1d_8 | input: | (None, 28, 256)
Conv1D | output: | (None, 28, 256)
batch_normalization_6 | input: | (None, 28, 256)
BatchNormalization | output: | (None, 28, 256)
re_lu_6 | input: | (None, 28, 256)
ReLU | output: | (None, 28, 256)
conv1d_9 | input: | (None, 28, 256)
Conv1D | output: | (None, 28, 128)
batch_normalization_7 | input: | (None, 28, 128)
BatchNormalization | output: | (None, 28, 128)
re_lu_7 | input: | (None, 28, 128)
ReLU | output: | (None, 28, 128)
flatten_1 | input: | (None, 28, 128)
Flatten | output: | (None, 3584)
dense_1 | input: | (None, 3584)
Dense | output: | (None, 5)

Figure 3. The 1D CNN (five layers).

3.3.2. Classifier Model (Prototypical Networks)

In this research, we use a multiclassification method. In the classification process for our model, we utilized the prototypical network approach, where the Euclidean distance plays an essential role in computing the distance metrics among all support sets and queries. The prototype Euclidean distance produces optimal cluster representations given the support set labels. Figures 4 and 5 briefly show that the classification process uses a prototypical network. All the output weights extracted from the 1D_CNN feature extraction passed the classifier, which counted all feature vectors and computed the ratio at the end. The FSL model is implemented in different scenarios based on the number of samples (k) per class (N). The first structure of FSL using a prototypical network is the N-way K-shot, where N = 5 refers to the number of classes, and K = 5, meaning each class in the N-way consists of 5 samples, thus forming a 5-way 5-shot scenario. The second scenario is a 5-way 10-shot setting, where the number of classes (way) remains the same, and the number of samples in each class is K = 10. The query set, on the other hand, is configured as a 10-query set for evaluating the model. The 5-way classes represented in the support set are the classes of sample types: normal, scan_sU, scan_A, mqtt_bruteforce, and spart. However, the query set represents the evaluation samples that should be correctly classified to the corresponding class based on the nearest calculated distance to the support set prototype.

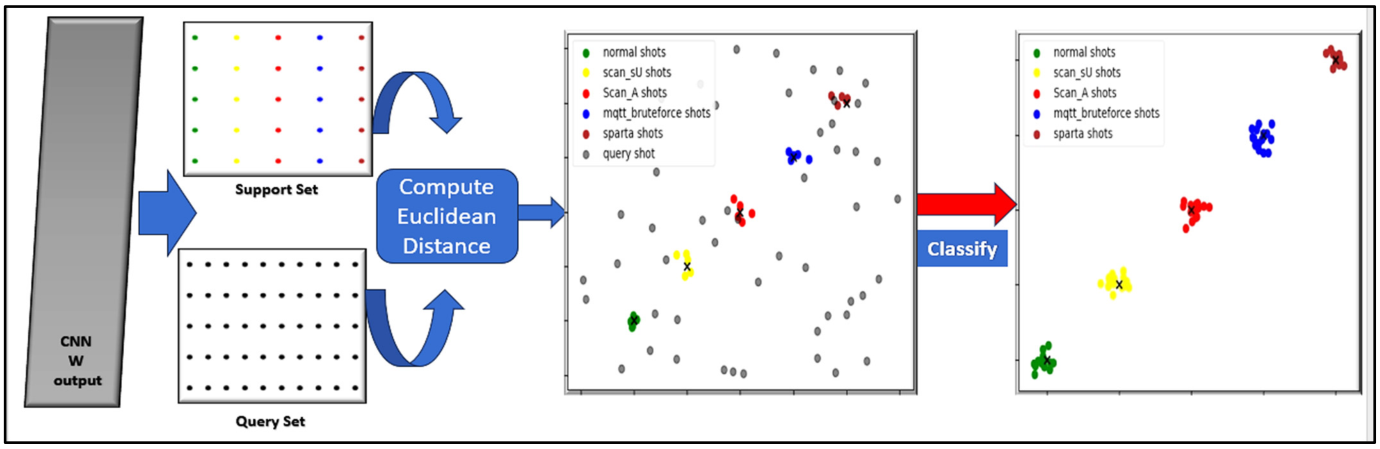

Figure 4. Process of the proposed model (1D_CNN) with prototypical networks.

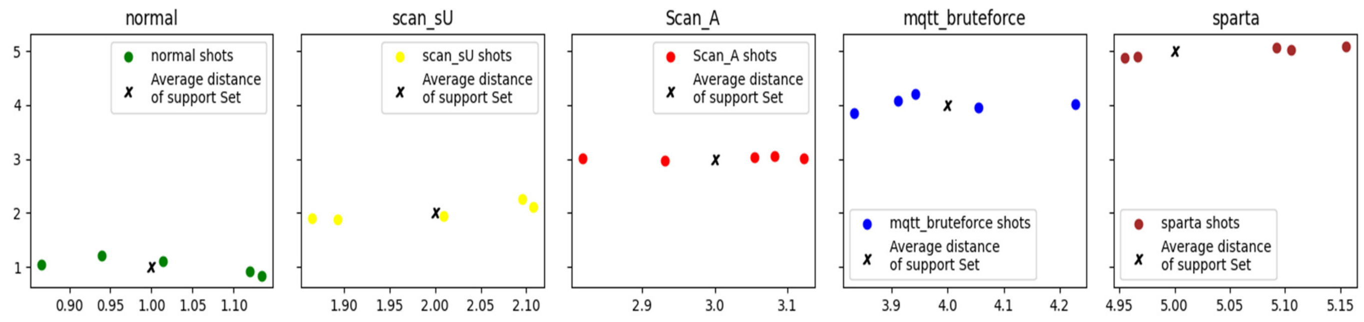

Figure 5. Correlation and distance averages for prototypical and support set shots.

The Euclidean distance computes the distance of each sample in the query set and matches it to the nearest distance of the corresponding class. Algorithm 1 provides a high-level illustration of the process of training loss computation for prototypical networks for each episode as follows:

1. Centroid Calculation:

In centroid calculation, Equation (1) is used to compute the centroid by averaging the feature embeddings $f(x_s)$ of support samples $(x_s,\ y_s)$ belonging to class k. The term "Euclidean distance" is implied in the calculation of distances.

$$c_k = \frac{1}{\left|D_{\{S_k\}}\right|} \sum\nolimits_{(x_s,\ y_s) \in\ D_{\{S_k\}}} f(x_s) \tag{1}$$

c_k: The centroid or prototype of class k. This point represents the "average" of all samples in class k.
$D_S k$: The set of support samples for class k. These samples are used to calculate the centroid.
$|D_S k|$: The number of support samples for class k.
x_s: A support sample.
$f\ (x_s)$: The feature embedding of support sample, x_s, represented as a vector in the feature space.
$\sum_{(x_s,\ y_s)\in\ D_{\{S_k\}}} f(x_s)$: The sum of feature embeddings of all support samples in class k.
$\frac{1}{\left|D_{\{S_k\}}\right|}$: Used to calculate the average of feature embeddings.

2. Loss L_1 Calculation:

This computes the loss L_1 for a query sample x_q belonging to class k with respect to its centroid c_k. The squared Euclidean distance between the feature embedding $f(x_q)$ of the query sample and the centroids of all classes is used in the calculation. It involves the SoftMax operation and negative logarithm to obtain the cross-entropy loss; the equation is as follows.

$$L_1 = -\log\left(\frac{\exp\left(-\parallel f\left(x_q\right) - c_k \parallel^2 /2\sigma^2\right)}{\sum_{k'} exp\left(-\parallel f(x\_q) - c\_(k')\ \parallel^2 /2\sigma^2\right)}\right) \tag{2}$$

L_1: The loss for query sample x_q belonging to class k.
$f\ (x_q)$: The feature embedding of query sample x_q.
c_k: The centroid or prototype of class k.
$\parallel f\left(x_q\right) - c_k \parallel^2$: The squared Euclidean distance between the feature embedding of the query sample and the centroid of class k.
$\exp\left(-\parallel f\left(x_q\right) - c_k \parallel^2 /2\sigma^2\right)$: The exponential of the negative squared distance, scaled by $2\sigma^2$. Measures similarity between the query sample and class $k's$ centroid.
$\sum_{k'} exp\left(-\parallel f(x\_q) - c\_(k^{'})\ \parallel^2 /2\sigma^2\right)$: Sum of similarities between the query sample and centroids of all classes.
$\frac{\exp\left(-\|f\left(x_q\right)-c_k\|^2/2\sigma^2\right)}{\sum_{k'} exp(-\|f(x\_q)-c\_(k')\ \|^2/2\sigma^2)}$: SoftMax of negative squared distances, yielding a probability distribution over classes.

3. Intra-class Loss $L_{\{\text{in}\}}$ Calculation:

$$L_{\text{in}} = -\log\left(\frac{\exp\left(-\parallel f\left(x_q\right) - c_k \parallel^2 /2\sigma^2\right)}{\sum_{k'} \exp\left(-\parallel f\left(x_q\right) - c_{k'} \parallel^2 /2\sigma^2\right)}\right) \tag{3}$$

This computes the loss L_{in} for a query sample x_q within the same class. The loss is calculated similarly to L_1, but with a slight difference in the calculation of distances. It also involves comparing the Euclidean distances between the feature embedding $f(x_q)$ of the query sample and the centroids of all classes.

4. Out-of-distribution Loss L_{out} Calculation:

$$L_{\text{out}} = -\log\frac{1}{|\ D_{\text{OOD}}\ |}\sum_{(x_s,y_s)\in D_{\text{OOD}}}\frac{\exp\left(-\parallel f\left(x_q\right) - c_{k'} \parallel^2\right)}{\sum_{k''} \exp\left(-\parallel f\left(x_q\right) - c_{k''} \parallel^2\right)} \tag{4}$$

This equation computes the loss L_{out} for out-of-distribution samples. It is based on the Euclidean distance between the feature embeddings of the query sample and the centroids, similar to L_1 and L_{in}. However, it involves summing over all out-of-distribution samples and the centroids, where D_{OOD} is the set of out-of-distribution samples, $F(x_Q)$ is the feature embedding of a query sample $x_{q'}$, and $c_{k'}$ is the centroid for class k'.

Algorithm 1: Training Episode Loss Computation for Prototypical Networks

```
Require: Meta-training set : $D_train = (x_i, y_i)_{i=1}^{N_tain}$, where $y_i \in \{j\}_{j=1}^{M_train}$, and Mtrain $> 2$.
Require: Nway Number of classes in the meta-training set.
Require: Nshot Number of query samples per class.
Require: Nquery Number of classes in the meta-training set.
Ensure: The loss L for a randomly generated training episode.
1.  $V_{ID}$ ←RandomSample jj = 1Mtrain,K ▷ Randomly select K classes for the
episode.
2.  for $k$ in {1, 2, . . ., K} do
3.  Dk ← RandomSample (DVk,Nshot + Nquery) ▷ Sample support and
query samples.
4.  DSk,DQk←SplitDk, Nshot ▷ Split samples into support and query
sets.
5.  Calculate ck by using Equation (1)
6.  End for
7.  VOOD ← RandomSample jj = 1Mtrain VID,1 ▷ Select a class not in VID for
out-of-distribution samples.
8.  DOOD ← RandomSample DVOOD, NOOD. ▷ Sample out-of-distribution samples.
9.  L ← 0
10. for $k$ in {1, 2, . . ., K} do
11. for (xq, yq) in DQk do
12. Calculate L1 and Lin by using Equations (2) and (3);
13. $L \leftarrow L + \frac{1}{K \cdot N_{query}} \cdot (L_1 + L_{in})$
14. end for
15. end for
16. for (xq, yq) in DOOD do
17. Calculate Lout by using Equation (4) on DOOD;
18. $L \leftarrow L + \frac{1}{N_{OOD}} \cdot L_{out}$
19. end for
```

The model learns to classify query samples based on their similarity to the corresponding centroids of each class, which are computed from the support samples. The loss functions encourage the model to correctly classify both in-distribution and out-of-distribution samples. The model is trained to minimize these loss functions. The input of the model contains the original feature extracted from 1D_CNN and sets a prototype per class. Sequentially, the forward method defines how the prediction is performed after removing the fully connected layer and computes the main vectors and feature distance for each query shot and the prototype for all shots in each class to make it ready for prediction. Once the forward method is successfully applied, score calculations are performed later to classify the query shot as belonging to the correlated classes, and the query is classified as the class with the highest score, which is the lowest computed distance. In terms of the obtained optimal accurate classification, the training process is trained and evaluated for several tasks in 20 epochs. The model parameters are adjusted to backpropagate the gradients, and the loss is computed for each iteration. Additionally, the algorithm calculates and outputs the epochs and calculates the average loss. In the tuning process, we monitored the best result from the epochs; we defined this as the best validation accuracy to reach the optimal value fund during the iteration.

4. Experiment and Results

In this research, we initially conducted the experiment by using an HP notebook with an Intel Gen8 CPU and 12 GB of RAM. Eventually, we moved the experiment to Google Colab for mobility. The language used was Python 3.10, and the main libraries used were as follows: torch, numpy, pandas, matplotlib, matplotlib, seaborn, sklearn, and easyfsl.

4.1. Implementation

In our task, we implemented our model by using various scenarios to test and evaluate the module. Initially, we started by setting up the FSL architecture. First, FSL is set as 5-way, 5-shot, and 10-query sets. The second step increases the number of samples per class to 10; the details are shown in Table 3. For evaluation and verification, by using two FSL architectures (i.e., 5-way 5-shot architecture and 5-way 10-shot architecture), we implemented the original feature space, which refers to the features that are not extracted from 1D_CNN in the dataset. Furthermore, we tested this approach by using random initialization, which means that the 1D_CNN model is not trained but rather given random weights; then, on those random weights, we extracted features.

Table 3. Total number of samples for each class.

Classes	Shots Per Class	Query Set	Classes Attack Description
normal	5 shots and 10 shots	10	Being normal
scan_sU	5 shots and 10 shots	10	The attack aims to compromise port services using UDP
scan_A	5 shots and 10 shots	10	Attack the explorer port for malicious purposes
mqtt_bruteforce	5 shots and 10 shots	10	Attack target systems in the MQTT network to perform brute force
sparta	5 shots and 10 shots	10	The attack technique aims to analyze space and associated cyber threats.

The main purpose of our research is to propose an IDS based on the FSL model by using an IoT dataset; based on our knowledge, no previous work has used MQTT-IOT-IDS2020 in FSL. Therefore, we could not compare our FSL results with those of similar studies under the same FSL architectures and experimental circumstances. However, we conducted a comprehensive comparison with several scholars who used deep learning and machine learning-based intrusion detection systems on the MQTT-IOT-IDS2020 dataset. Furthermore, we extended the experiment by using the CICIDS2017 dataset with a 5-shot scenario, testing it with both original feature spaces and randomly initializing it for evaluation. In addition, the 5-shot results are compared with another work used by the same CICIDS2017 dataset in an IDS model based on FSL by using a 5-shot in a support set.

4.2. Results

In this section, we will illustrate all the generated results. The implementation was conducted on two different datasets with three experiments per dataset. We start with the main MQTT-IoT-IDS2020 dataset. The model is a multiclass classification model. Hence, the result provided is an average measurement and is not converted to a binary classification.

4.2.1. MQTT-IOT-IDS2020 Dataset Results

The results for the MQTT-IoT-IDS2020 dataset are shown in Figure 6 and Table 4. The proposed 1D_CNN + prototypical network model achieved very good results—99.28% accuracy—and 99.28% for Presidion. The recall was 99.26%, while the F1-score was 99.27%. Only 5-shot was used for the 5-way 5-shot and 5-way 10-shot FSL query sets. In addition, when five shot in the support set were evaluated with a random model weight, the accuracy was 85.93%, the precision was 86.62%, the recall was 85.93%, and the F1-score was 86.00%. The third set of experimental results obtained using the original feature with 5 shots yielded 86.09% accuracy, 87.84% precision, and 86.09% recall; finally, the F1-score was 86.18%.

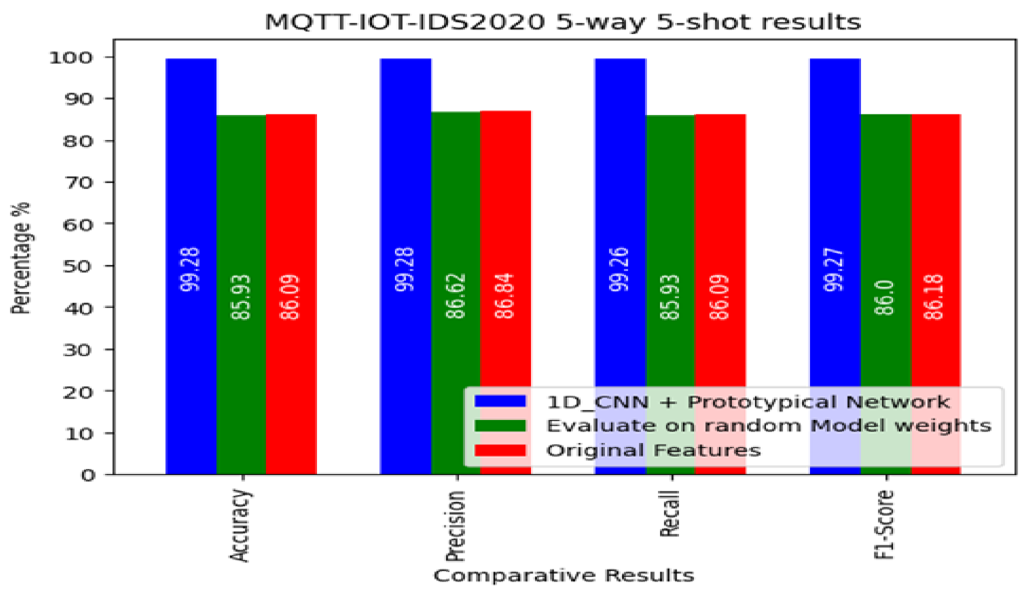

Figure 6. The 5-shot result for the MQTT-IoT2020 dataset.

Table 4. The 5-shot results for MQTT-IoT-IDS2020 dataset.

Method	Accuracy	Precision	Recall	F1-Score
Original features	86.09	86.84	86.09	86.18
Random model weights	85.93	86.62	85.93	86
1D_CNN + prototypical	99.28	99.28	99.26	99.27

The other experiment was conducted after increasing the number of shots per class from 5 to 10, as shown in Figure 7 and Table 5. The experimental results generated from our proposed model are as follows: the accuracy increased to 99.44%, the precision increased to 99.44%, the recall increased to 99.40%, and the F1-score increased to 99.42%. The results of the experiment that implemented the random model weight yielded 86.38% accuracy and 87.09% precision, while the recall was 86.37% and the F1-score was 86.46%.

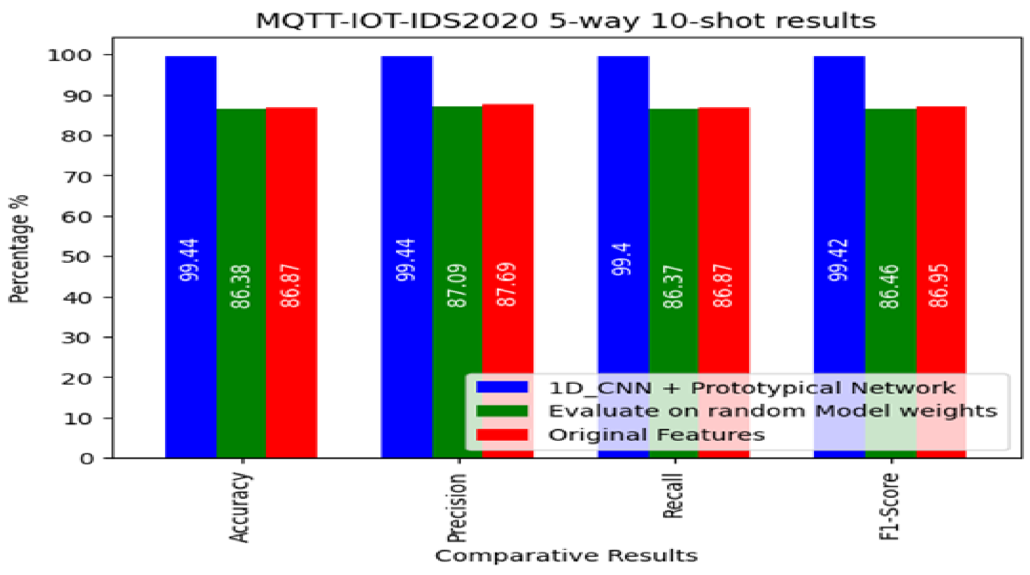

Figure 7. The 10-shot result for the MQTT-IoT2020 dataset.

Table 5. The 10-shot results for MQTT-IoT-IDS2020 dataset.

Method	Accuracy	Precision	Recall	F1-Score
Original features	86.87	87.69	86.87	86.95
Random model weights	86.38	87.09	86.37	86.46
1D_CNN + prototypical	99.44	99.44	99.4	99.42

The MQTT-IoT2020 5-way 5-shot and 10-shot comparative analysis results are illustrated in Figure 8 and Table 6. The comparative performance results for the MQTT-IoT2020 dataset using 5-shot and 10-shot are illustrated in Figures 9–12.

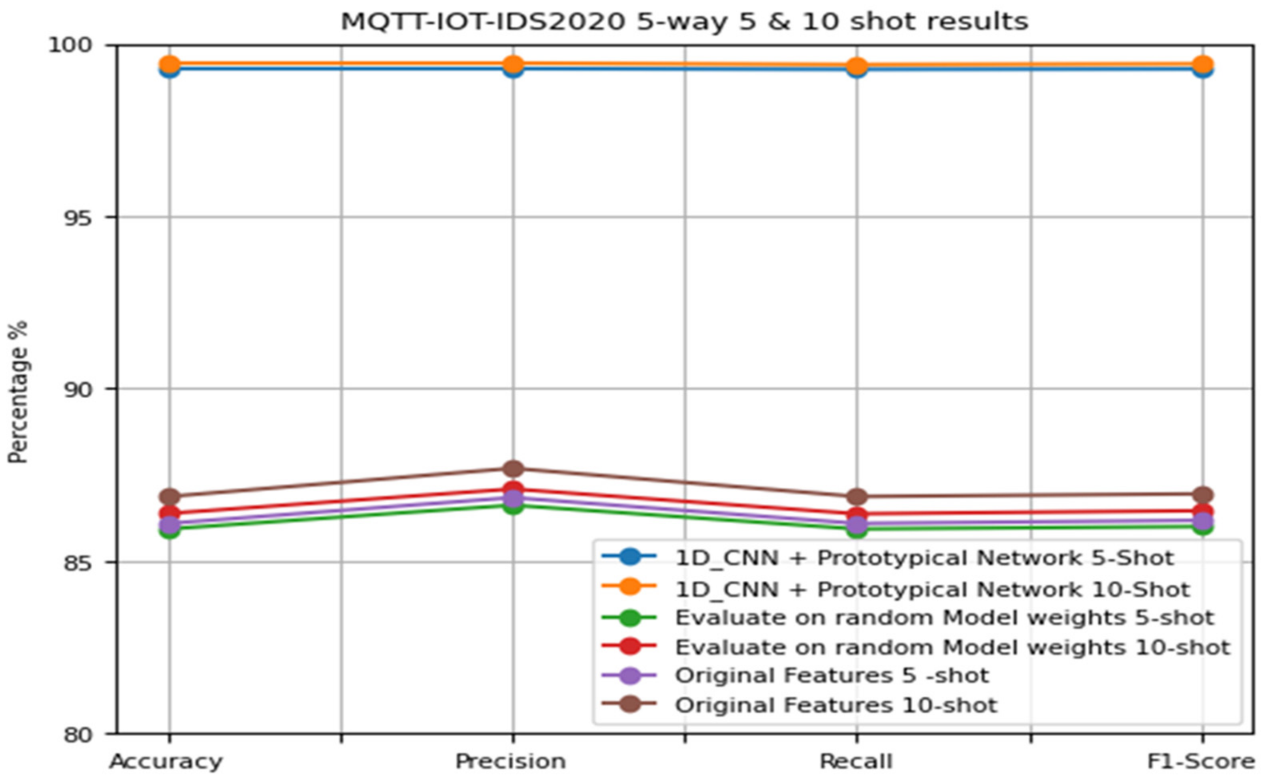

Figure 8. The 5-way 5-shot and 5-way 10-shot comparative analytic results for the MQTT-IoT2020 dataset.

Table 6. The 10-shot and 5-shot results for the MQTT-IoT-IDS2020 dataset.

Method	Accuracy	Precision	Recall	F1-Score
Original features 5-shot	86.09	86.84	86.09	86.18
Original features 10-shot	86.87	87.69	86.87	86.95
Random model weights 5-shot	85.93	86.62	85.93	86
Random model weights 10-shot	86.38	87.09	86.37	86.46
1D_CNN + prototypical 5-shot	99.28	99.28	99.26	99.27
1D_CNN + prototypical 10-shot	99.44	99.44	99.4	99.42

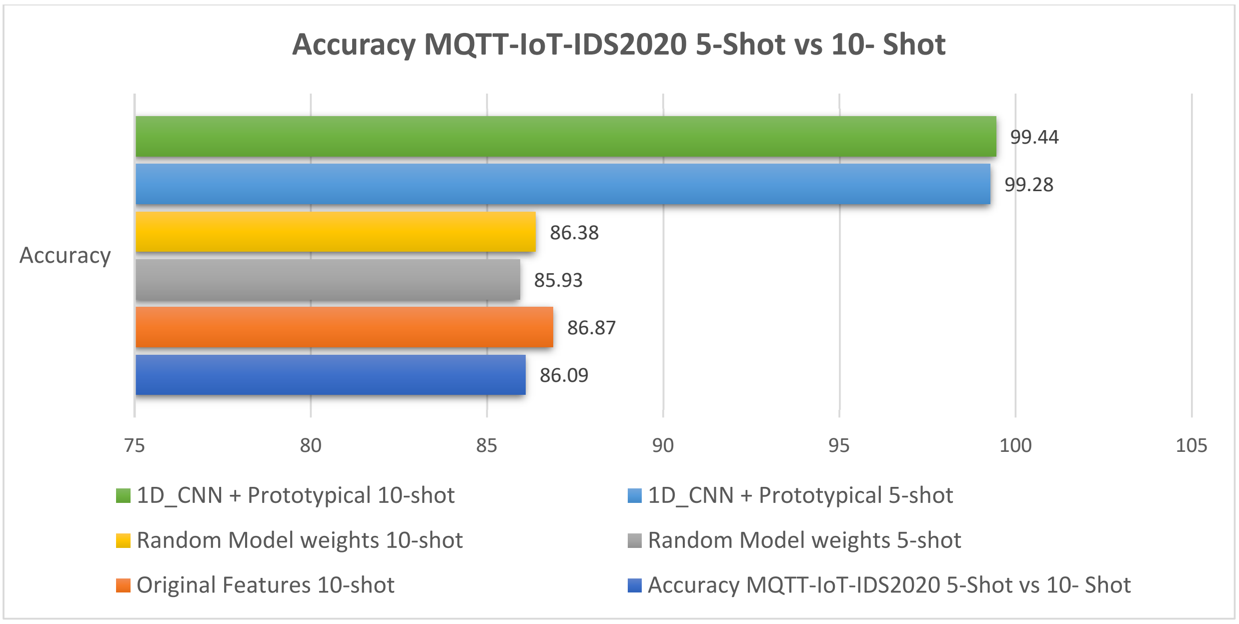

Figure 9. The 5-shot and 10-shot accuracy results for the MQTT-IoT-IDS2020 dataset.

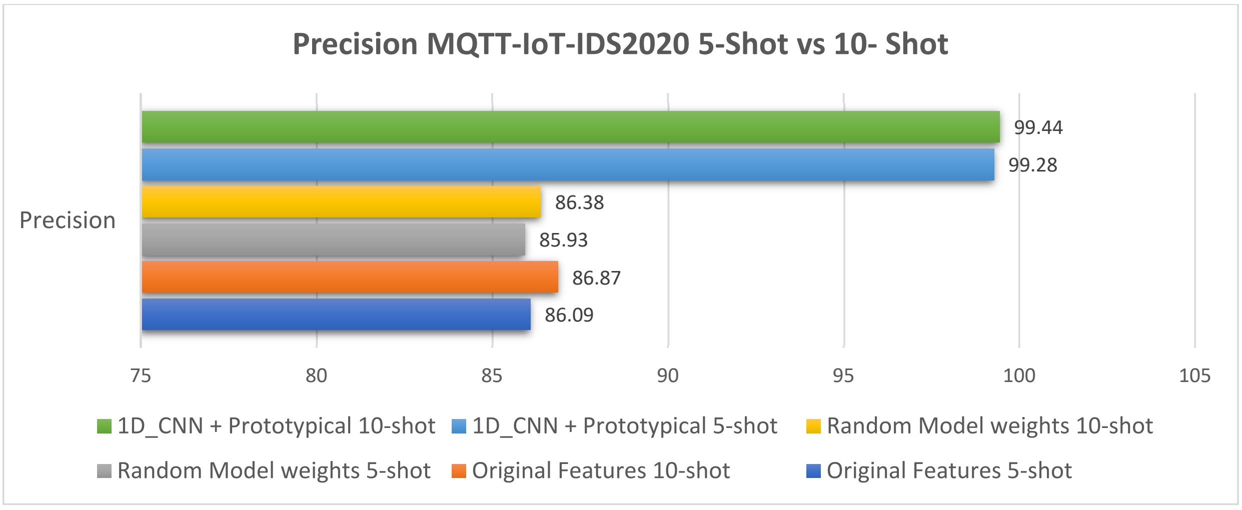

Figure 10. The 5-shot and 10-shot precision results for the MQTT-IoT-IDS2020 dataset.

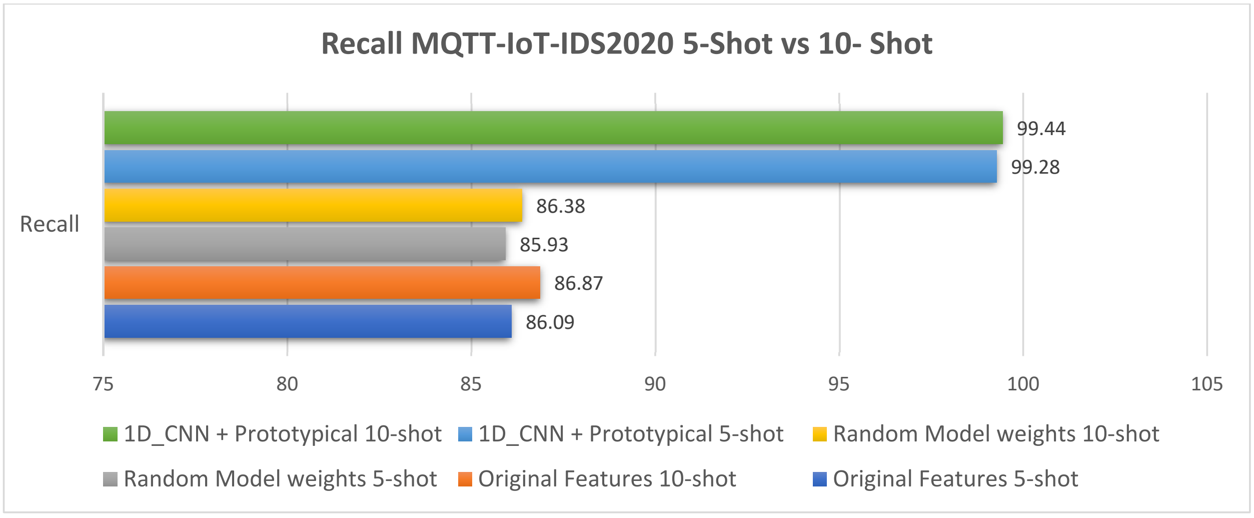

Figure 11. The 5-shot and 10-shot recall results for the MQTT-IoT-IDS2020 dataset.

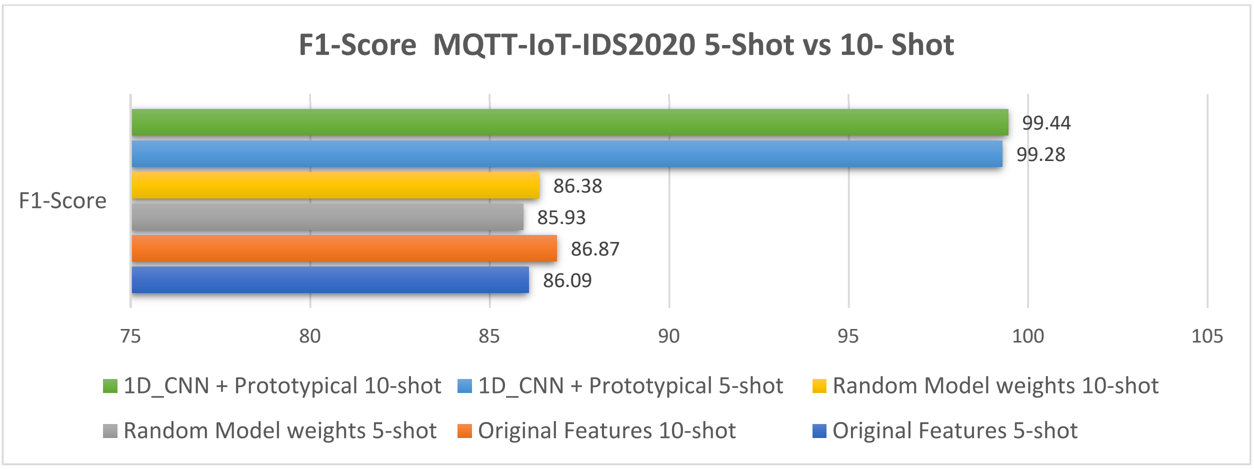

Figure 12. The 5-shot and 10-shot F1-scores for the MQTT-IoT-IDS2020 dataset.

4.2.2. CICIDS2017 Dataset Results

In this experiment, we used one FSL architecture that consisted of 12-way, 5-shot, and 10 query sets.

The results are illustrated in Figure 13 and Table 7. The first result for 5 shots is the proposed 1D_CNN + prototypical network method, which is 93.13% accurate, 93.46% precise, has 93.13% recall, and has an F1-score of 92.40%. The second result is for the evaluation of random model weights, for which the accuracy is 44.27%, the precision is 45.40%, the recall is 44.27%, and the F1-score is 40.20%. The third results for the conducted experiment by using the original features are as follows: 42.94% accuracy, 43.68% precision, 42.94% recall, and 38.67% F1-score.

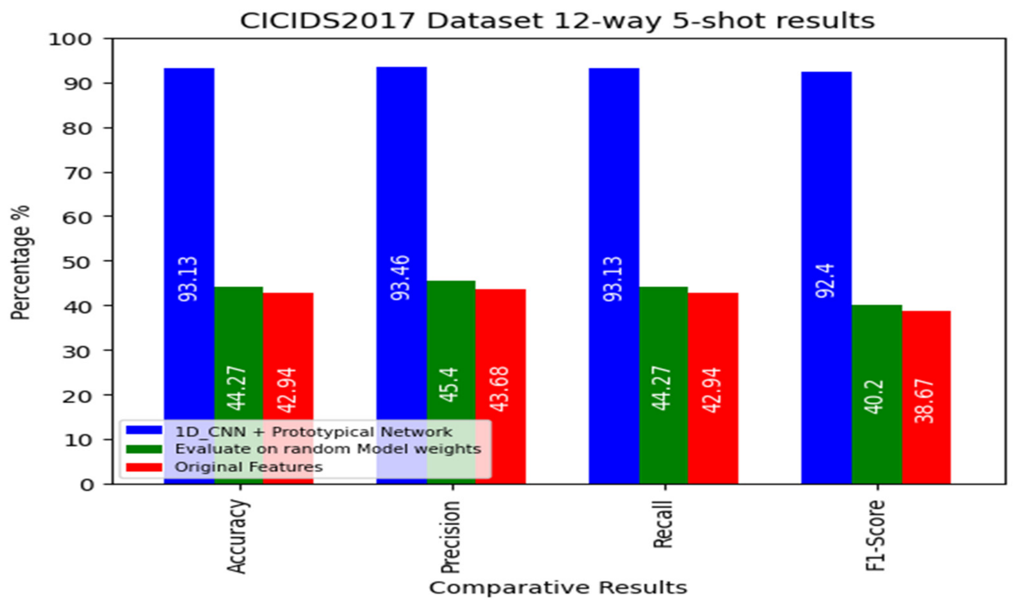

Figure 13. The 5-shot results for the CICIDS-IDS2017 dataset.

Table 7. The 12-way 5-shot results for the CICIDS-IDS2017 dataset.

Method	Accuracy	Precision	Recall	F1-Score
IDS17—original features 5-shot	42.94	43.68	42.94	38.67
IDS17—random model weights 5-shot	44.27	45.4	44.27	40.2
IDS17—1D_CNN + prototypical 5-shot	93.13	93.46	93.13	92.4

5. Discussion

The results for the MQTT-IoT-IDS2020 dataset revealed small gaps in accuracy among the three experiments conducted: namely, the 1D_CNN + prototypical network, which were evaluated on random model weights; moreover, the original features are shown in Figure 8. The accuracy of the 1D_CNN + prototypical network is high at 99.44%, while the average accuracy of the random model weights and original features is 86%. The gap is approximately 13%, which is considered a small gap. This scheme refers to the natural status of the dataset itself. The MQTT-IoT-IDS2020 dataset was tested with various IDSs based on ML and DL models, and a notably high performance was reached. However, even though reaching high performance with only a few samples in training is challenging, we compared our FSL model with modules that use large datasets in the training and testing stages. Meanwhile, the difference between our proposed model and both evaluated scenarios with (random weights and original features) is large in the CICIDS2017 dataset experiments (approximately 50%), as illustrated in Figure 14. The proposed model accuracy is 93.13%, while the other experiment has an average accuracy of 40%. This finding emphasizes that the model functions well and that the accuracy varies from one dataset to another. The comparative results between the two datasets were added to Table 8, as was the CICIDS2017 12-way 5-shot comparative analysis result shown in Figure 14. The accuracy comparison, recall, precision, and F1-score are illustrated in Figure 15, Figure 16, Figure 17, and Figure 18, respectively.

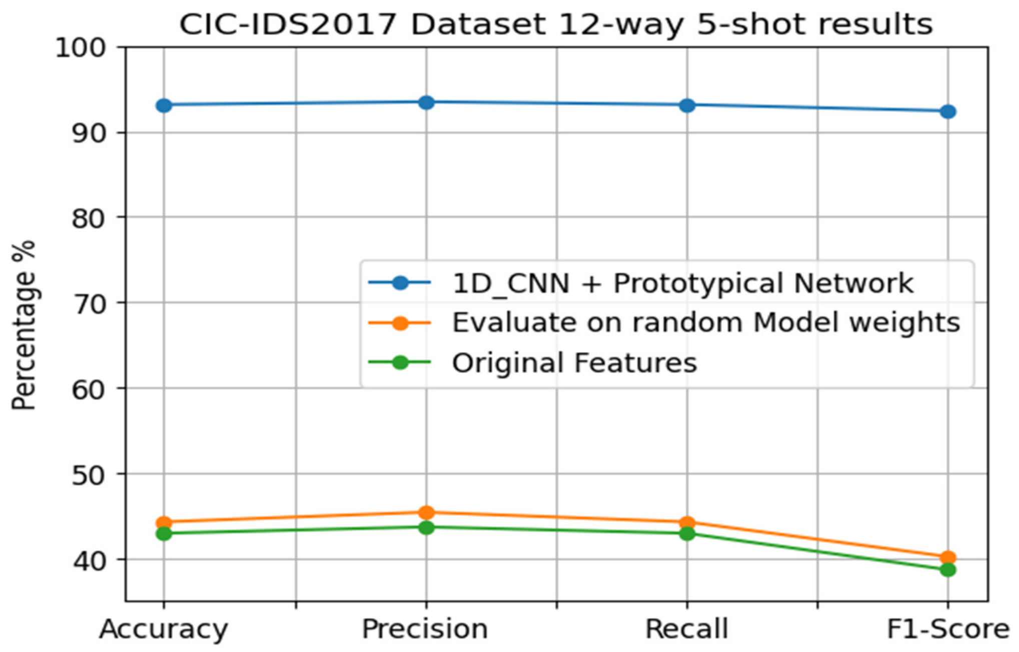

Figure 14. The 12-way 5-shot comparative analysis results for the CICIDS2017 dataset.

Table 8. The 5-shot results for the MQTT-IoT-IDS2020 and CIC-IDS2017 datasets.

Method	Accuracy	Precision	Recall	F1-Score
MQTT—original features 5-shot	86.09	86.84	86.09	86.18
IDS17—original features 5-shot	42.94	43.68	42.94	38.67
MQTT—random model weights 5-shot	85.93	86.62	85.93	86
IDS17—random model weights 5-shot	44.27	45.4	44.27	40.2
MQTT_1D_CNN + prototypical 5-shot	99.28	99.28	99.26	99.27
IDS17-1D_CNN + prototypical 5-shot	93.13	93.46	93.13	92.4

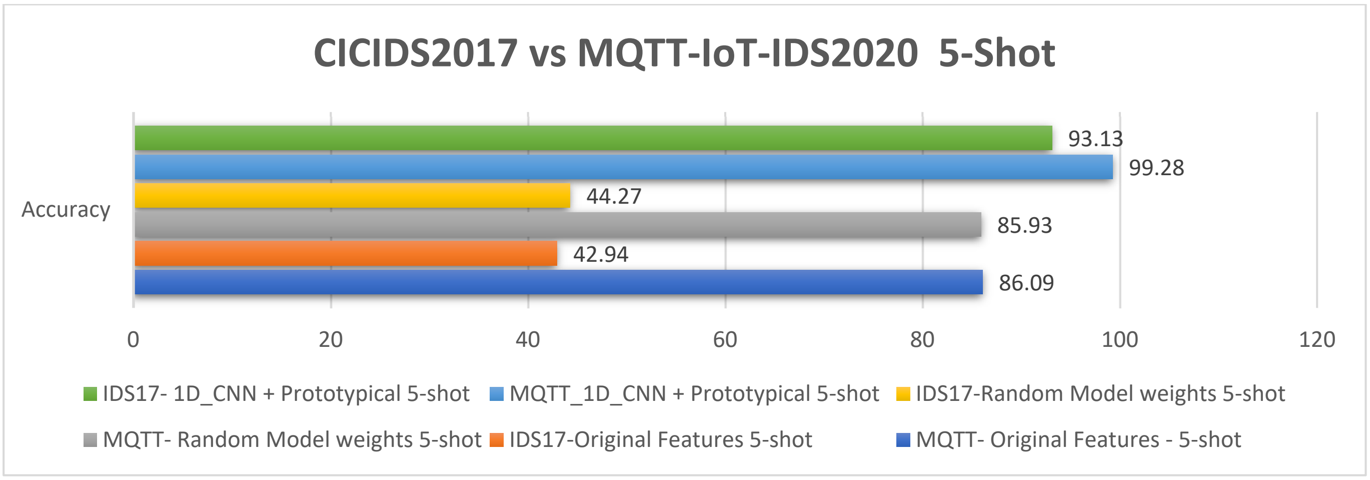

Figure 15. The 5-shot accuracy analysis results for the MQTT-IoT-IDS2020 and CICIDS2017 datasets.

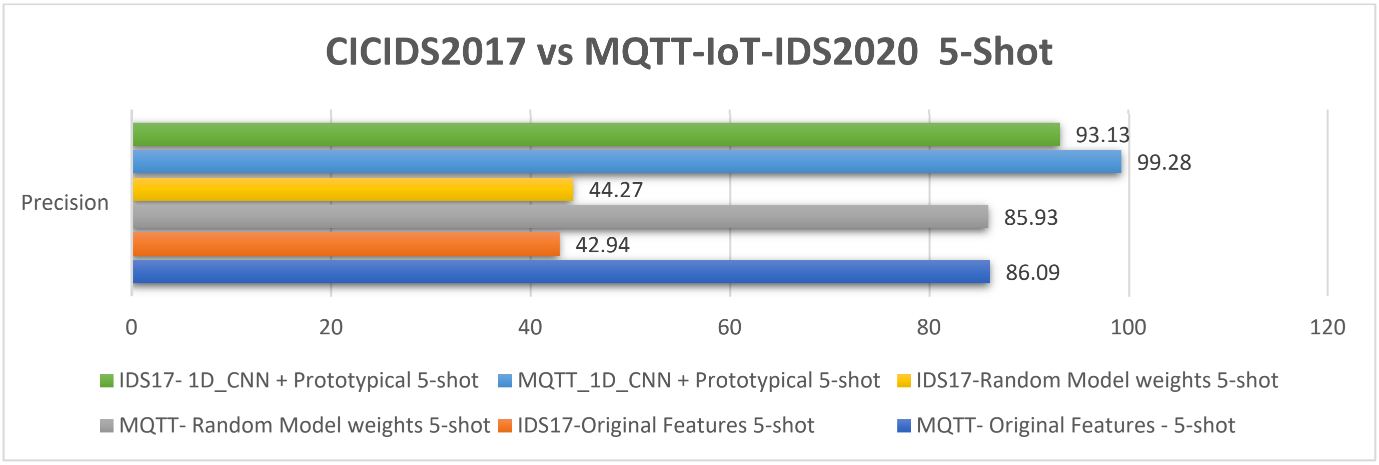

Figure 16. The 5-shot precision analysis results for the MQTT-IoT-IDS2020 and CICIDS2017 datasets.

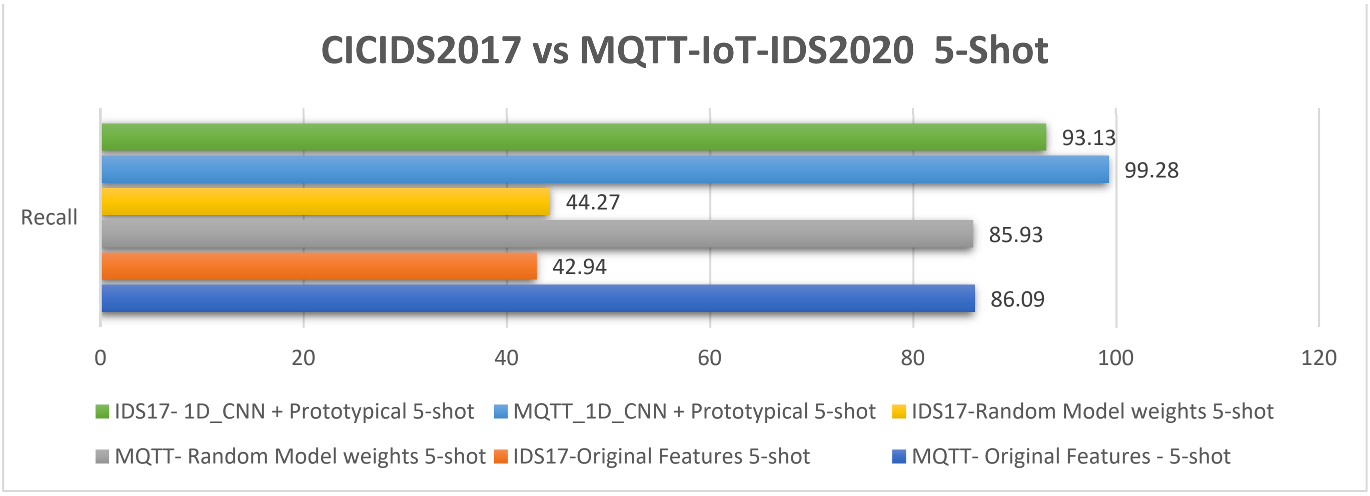

Figure 17. The 5-shot recall analysis results for the MQTT-IoT-IDS2020 and CICIDS2017 datasets.

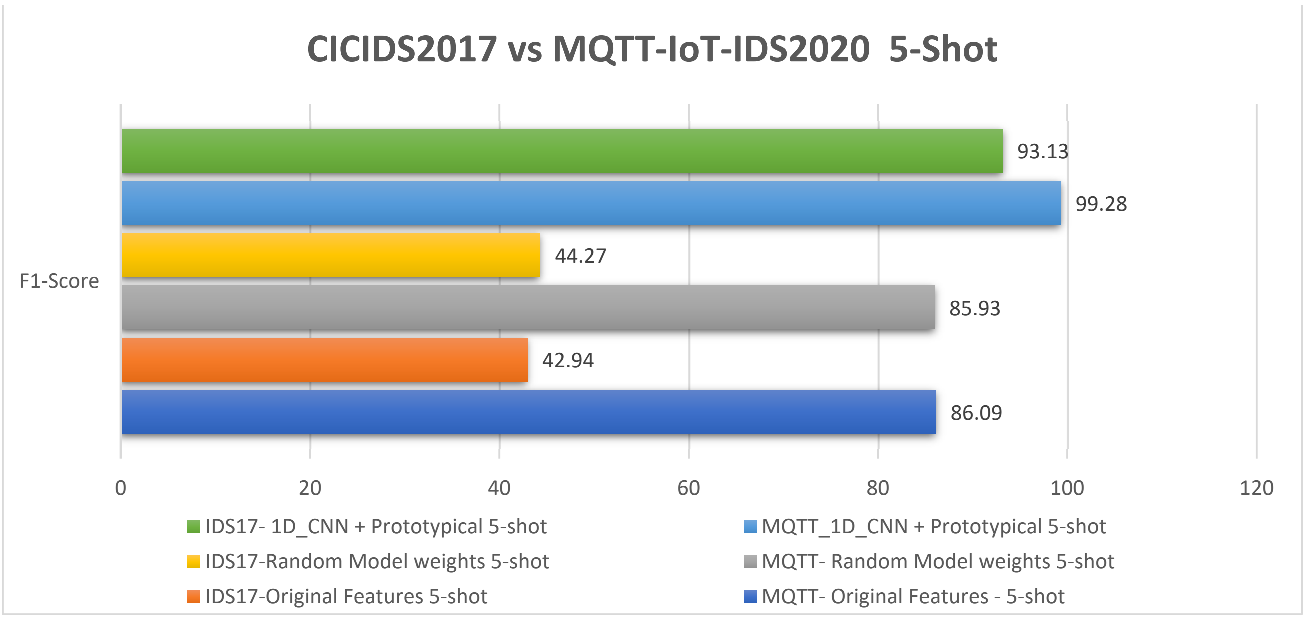

Figure 18. The 5-shot F1-score analysis results for the MQTT-IoT-IDS2020 and CICIDS2017 datasets.

In addition, we conducted a comparison for verification between our proposed model and other IDS models that use the MQTT-IoT-IDS2020 dataset. We compared the experimental results with those of other IDS models that use ML- and DL-based IDSs, to the best of our knowledge, no existing studies have employed the MQTT-IoT-IDS2020 dataset within the context of Few-Shot Learning (FSL) models. A comparison summary has been added to Table 9.

Table 9. Comparative analysis of IDS performance for the MQTT-IOT dataset.

Reference	Year	Dataset	Method	Result
[25]	2019	Their own gendered dataset using MQTT protocol	IDS based on XGBoost, RNN, LSTM, and GRU	LSTM 93.37% GRUs 96.08%
[26]	2021	MQTT-IoT-IDS2020	DNN (IDS based on ANN)	97.13%
[27]	2022	MC-IoT dataset, MQTT-IoT-IDS2020 dataset, MQTTset dataset	Chaotic salp swarm optimization algorithm (ECSSA) and LightGBM classifier	98.91% accuracy for MQTT-IoT-IDS2020
[28]	2020	MQTT-IoT-IDS2020	6 different ML techniques	Accuracy avg—+90%
[29]	2023	MQTT dataset	Ensemble learning (EL) including bagging, boosting, and stacking	Accuracy 95.38%
[30]	2022	MQTT-IoT-IDS2020 dataset	Various ML and DL, RF, LR, DT, K-N, SVM, DNN, CNN	Binary avg —+90%
[31]	2021	MQTT-IoT-IDS2020 dataset	(DNN), (LSTM), and mix of (CNN-RNN-LSTM)	Accuracy = 97.09%
[PROPOSED MODEL]	2024	MQTT-IoT-IDS2020 dataset	CNN (1D_5CNN) + FSL (prototypical network)	Accuracy 10-shot = 99.44% 5-shot = 99.28%

Another comparison was conducted on the CICIDS2017 dataset with other IDS models; the comparison was conducted on an IDS based on FSL, which used only a few samples for classification, and the comparison has been added to Table 10. For further clarification of the results, even though other models may reach higher accuracy, we set the proposed model as a multiclass classification, and this type of average classification's accuracy is usually lower than that of binary classification. Additionally, in the CICIDS2017 dataset, in our experiment, 12 classes were used, and other models may use fewer classes, which may increase the accuracy.

Table 10. Comparative analysis of FSL IDS performance for the CICIDS2017 dataset.

Reference	Year	Number of Shots	Method	Result
[37]	2023	1-shot	Siamese network	Overall accuracy = 80–85%
[21]	2022	5-shot	FC-NET (DL + Siamese N)	Accuracy = 89.09%
[38]	2022	10-shot	FS-IDS flow data encoding + feature fusion mechanism	Accuracy: 93.60
[22]	2021	5-shot	Siamese capsule network	Accuracy = 93.87
[39]	2022	fewer than 20	Global-aware prototypical network (GP-Net)	Accuracy: 94.58
[40]	2022	10-shot	FSL—parallelized triplet network	Recall: 94.57
[41]	2023	5-shot	CharNet (neuro-immune + character embedding)	Accuracy: 95.94
[PROPOSED MODEL]	2024	5-shot	1D CNN + prototypical network	Accuracy = 93.13%

6. Conclusions

This study presented an IDS based on an FSL 1D_CNN prototypical network. The model utilized a 1-dimensional 5-layer CNN to extract vital features that act as the backbone of the FSL classifier, which is a prototypical network. The model mainly focused on leveraging IDSs in IoT networks and overcoming the limitation of acquiring enormous datasets for DL training, which is scarce in some cases, especially for IoT networks, and mitigating the need for labeling enormous amounts of data or oversampling and undersampling approaches, which can cause some issues in performance. Two standard datasets are used in the model: mainly the IoT dataset (MQTT-IoT-IDS2020) and extended to the CICIDS2017 dataset for more validation. The proposed model achieved an accuracy of 99.44% for the MQTT-IoT-IDS2020 dataset and 93.13% for the CICIDS2017 dataset by using only 10 shots (samples). Applying the few-shot learning technique to the IDS could enhance the detection performance in IoT IDSs and overcome the challenging issues in IoT networks of collecting and labeling huge amounts of data. However, the proposed model was applied to only one IoT dataset as well as lacking the ability to detect novel attacks; our future work will involve assessing its performance across diverse IoT datasets. Additionally, we plan to extend the model by incorporating a new class specifically designed to detect unknown attacks, thereby enhancing its ability to classify novel attacks as intrusions.

Author Contributions: Conceptualization, T.A. and I.A.; methodology, T.A.; software, T.A.; validation, I.A. and M.O.A.; formal analysis, T.A.; investigation, T.A.; resources, I.A.; data curation, M.O.A.; writing—original draft preparation, T.A.; writing—review and editing, I.A.; visualization, T.A. and I.A.; supervision, I.A.; project administration, I.A.; funding acquisition, I.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research work was funded by Institutional Fund Projects under grant no. (IFPIP-1280-611-1443) from the Ministry of Education and King Abdulaziz University, DSR, Jeddah, Saudi Arabia.

Data Availability Statement: Data are contained within the article.

Acknowledgments: This research work was funded by Institutional Fund Projects under grant no. (IFPIP-1280-611-1443). Therefore, authors gratefully acknowledge technical and financial support from the Ministry of Education and King Abdulaziz University, DSR, Jeddah, Saudi Arabia.

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

References

1. Al-Hadhrami, Y.; Hussain, F.K. Real time dataset generation framework for intrusion detection systems in IoT. *Futur. Gener. Comput. Syst.* **2020**, *108*, 414–423. [CrossRef]
2. Min, E.; Long, J.; Liu, Q.; Cui, J.; Cai, Z.; Ma, J. SU-IDS: A semi-supervised and unsupervised framework for network intrusion detection. In *Cloud Computing and Security*; Sun, X., Pan, Z., Bertino, E., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 322–334.
3. Althnian, A.; AlSaeed, D.; Al-Baity, H.; Samha, A.; Dris, A.B.; Alzakari, N.; Abou Elwafa, A.; Kurdi, H. Impact of dataset size on classification performance: An empirical evaluation in the medical domain. *Appl. Sci.* **2021**, *11*, 796. [CrossRef]
4. Iliyasu, A.S.; Abdurrahman, U.A.; Zheng, L. Few-shot network intrusion detection using discriminative representation learning with supervised autoencoder. *Appl. Sci.* **2022**, *12*, 2351. [CrossRef]
5. Chawla, S. *Deep Learning-Based Intrusion Detection System for Internet of Things*; University of Washington: Seattle, WA, USA, 2017; p. 72.
6. Samaila, M.G.; Neto, M.; Fernandes, D.A.B.; Freire, M.M.; Inácio, P.R.M. Security challenges of the Internet of things. In *Beyond the Internet of Things: Everything Interconnected*; Batalla, J.M., Mastorakis, G., Mavromoustakis, C.X., Pallis, E., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 53–82.
7. Fink, M. Object classification from a single example utilizing class relevance metrics. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2004; Volume 17.
8. Wang, Y.; Yao, Q.; Kwok, J.; Ni, L.M. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.* **2020**, *53*, 1–34. [CrossRef]
9. Bontonou, M.; Béthune, L.; Gripon, V. Predicting the accuracy of a few-shot classifier. *arXiv* **2020**, arXiv:2007.04238.
10. Miao, G.; Wu, G.; Zhang, Z.; Tong, Y.; Lu, B. SPN: A Method of Few-Shot Traffic Classification with Out-of-Distribution Detection Based on Siamese Prototypical Network. *IEEE Access* **2023**, *11*, 114403–114414. [CrossRef]
11. Snell, J.; Swersky, K.; Zemel, R.S. Prototypical Networks for Few-shot Learning. *arXiv* **2017**, arXiv:1703.05175.
12. Wu, K.; Chen, Z.; Li, W. A novel intrusion detection model for a massive network using convolutional neural networks. *IEEE Access* **2018**, *6*, 50850–50859. [CrossRef]
13. Yin, C.; Zhu, Y.; Fei, J.; He, X. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
14. Yang, J.; Yu, X.; Xie, Z.-Q.; Zhang, J.-P. A novel virtual sample generation method based on gaussian distribution. *Knowl. Based Syst.* **2011**, *24*, 740–748. [CrossRef]
15. Andonie, R. Extreme data mining: Inference from small datasets. *Int. J. Comput. Commun. Control* **2010**, *5*, 280. [CrossRef]
16. Fei-Fei, L.; Fergus, R.; Perona, P. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 594–611. [CrossRef] [PubMed]
17. Chowdhury, M.M.U.; Hammond, F.; Konowicz, G.; Xin, C.; Wu, H.; Li, J. A few-shot deep learning approach for improved intrusion detection. In Proceedings of the 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), New York, NY, USA, 19–21 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 456–462.
18. Wu, Y.; Lee, W.W.; Gong, X.; Wang, H. A hybrid intrusion detection model combining sae with kernel approximation in internet of things. *Sensors* **2020**, *20*, 5710. [CrossRef] [PubMed]
19. Yu, Y.; Bian, N. An intrusion detection method using few-shot learning. *IEEE Access* **2020**, *8*, 49730–49740. [CrossRef]
20. Hindy, H.; Tachtatzis, C.; Atkinson, R.; Bayne, E.; Bellekens, X. Developing a siamese network for intrusion detection systems. In Proceedings of the 1st Workshop on Machine Learning and Systems, ACM, New York, NY, USA, 26 April 2021; pp. 120–126.
21. Xu, C.; Shen, J.; Du, X. A method of few-shot network intrusion detection based on metalearning framework. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3540–3552. [CrossRef]
22. Wang, Z.-M.; Tian, J.-Y.; Qin, J.; Fang, H.; Chen, L.-M. A few-shot learning-based siamese capsule network for intrusion detection with imbalanced training data. *Comput. Intell. Neurosci.* **2021**, *2021*, 7126913. [CrossRef]
23. Liang, W.; Hu, Y.; Zhou, X.; Pan, Y.; Wang, K.I.-K. Variational few-shot learning for microservice-oriented intrusion detection in distributed industrial IoT. *IEEE Trans. Ind. Inform.* **2021**, *18*, 5087–5095. [CrossRef]
24. Zhou, X.; Liang, W.; Shimizu, S.; Ma, J.; Jin, Q. Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems. *IEEE Trans. Ind. Inform.* **2021**, *17*, 5790–5798. [CrossRef]
25. Alaiz-Moreton, H.; Aveleira-Mata, J.; Ondicol-Garcia, J.; Muñoz-Castañeda, A.L.; García, I.; Benavides, C. Multiclass Classification Procedure for Detecting Attacks on MQTT-IoT Protocol. *Complexity* **2019**, *2019*, 6516253. [CrossRef]
26. Khan, M.A.; Khan, M.A.; Jan, S.U.; Ahmad, J.; Jamal, S.S.; Shah, A.A.; Pitropakis, N.; Buchanan, W.J. A Deep Learning-Based Intrusion Detection System for MQTT Enabled IoT. *Sensors* **2021**, *21*, 7016. [CrossRef]
27. Prajisha, C.; Vasudevan, A.R. An efficient intrusion detection system for MQTT-IoT using enhanced chaotic salp swarm algorithm and LightGBM. *Int. J. Inf. Secur.* **2022**, *21*, 1263–1282. [CrossRef]
28. Hindy, H.; Bayne, E.; Bures, M.; Atkinson, R.; Tachtatzis, C.; Bellekens, X. Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset). *arXiv* **2020**, arXiv:2006.15340.
29. Zeghida, H.; Boulaiche, M.; Chikh, R. Securing MQTT protocol for IoT environment using IDS based on ensemble learning. *Int. J. Inf. Secur.* **2023**, *22*, 1075–1086. [CrossRef]

30. Chesney, S.; Roy, K. AI Empowered Intrusion Detection for MQTT Networks. In Proceedings of the 2022 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), Durban, South Africa, 4–5 August 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6. [CrossRef]
31. Mosaiyebzadeh, F.; Araujo Rodriguez, L.G.; Macedo Batista, D.; Hirata, R. A Network Intrusion Detection System using Deep Learning against MQTT Attacks in IoT. In Proceedings of the 2021 IEEE Latin-American Conference on Communications (LATINCOM), Santo Domingo, Dominican Republic, 17–19 November 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6. [CrossRef]
32. Hindy, H.; Tachtatzis, C.; Atkinson, R.; Bayne, E.; Bellekens, X. Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset). In *Selected Papers from the 12th International Networking Conference, INC 2020*; Ghita, B., Shiaeles, S., Eds.; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2021; Volume 180. [CrossRef]
33. Sharafaldin, I.; Habibi Lashkari, A.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the International Conference on Information Systems Security and Privacy, Funchal, Portugal, 22–24 January 2018.
34. Shitharth, S.; Kshirsagar, P.R.; Balachandran, P.K.; Alyoubi, K.H.; Khadidos, A.O. An Innovative Perceptual Pigeon Galvanized Optimization (PPGO) Based Likelihood Naïve Bayes (LNB) Classification Approach for Network Intrusion Detection System. *IEEE Access* **2022**, *10*, 46424–46441. [CrossRef]
35. Liu, G.; Zhang, J. CNID: Research of Network Intrusion Detection Based on Convolutional Neural Network. *Discret. Dyn. Nat. Soc.* **2020**, *2020*, 4705982. [CrossRef]
36. Qazi, E.U.H.; Almorjan, A.; Zia, T. A One-Dimensional Convolutional Neural Network (1D-CNN) Based Deep Learning System for Network Intrusion Detection. *Appl. Sci.* **2022**, *12*, 7986. [CrossRef]
37. Hindy, H.; Tachtatzis, C.; Atkinson, R.; Brosset, D.; Bures, M.; Andonovic, I.; Michie, C.; Bellekens, X. Leveraging Siamese networks for one-shot intrusion detection model. *J. Intell. Inf. Syst.* **2023**, *60*, 407–436. [CrossRef]
38. Yang, J.; Li, H.; Shao, S.; Zou, F.; Wu, Y. FS-IDS: A framework for intrusion detection based on few-shot learning. *Comput. Secur.* **2022**, *122*, 102899. [CrossRef]
39. Guo, J.; Cui, M.; Hou, C.; Gou, G.; Li, Z.; Xiong, G.; Liu, C. Global-Aware Prototypical Network for Few-Shot Encrypted Traffic Classification. In Proceedings of the 2022 IFIP Networking Conference (IFIP Networking), Catania, Italy, 13–16 June 2022; pp. 1–9. [CrossRef]
40. Tian, J.-Y.; Wang, Z.-M.; Fang, H.; Chen, L.-M.; Qin, J.; Chen, J.; Wang, Z.-H. Few-Shot Learning-Based Network Intrusion Detection through an Enhanced Parallelized Triplet Network. *Secur. Commun. Netw.* **2022**, *2022*, 3317048. [CrossRef]
41. Ma, Z.; Chen, Z.; Zheng, X.; Wang, T.; You, Y.; Zou, S.; Wang, Y. A Biological Immunity Based Neuro Prototype for Few-Shot Anomaly Detection with Character Embedding. *Cyborg Bionic Syst.* **2023**, *5*, 0086. [CrossRef]