Computer Modeling in Engineering & Sciences

Tech Science Press

https://doi.org/10.32604/cmes.2026.078467

ARTICLE

Systematic Evaluation of Few-Shot Learning for Unseen IoT Network Attack Detection

Liam Revell[1], **Hyunjae Kang**[1,*], **Jung Taek Seo**[2] **and Dan Dongseong Kim**[1]

[1]School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane, QLD, Australia
[2]Department of Smart Security, Gachon University, Seongnam-si, Gyeonggi-do, Republic of Korea

*Corresponding Author: Hyunjae Kang. Email: hyunjae.kang@uq.edu.au

Received: 31 December 2025; Accepted: 23 March 2026; Published: 27 April 2026

ABSTRACT: The rapid proliferation of Internet of Things (IoT) devices has increased the importance of network intrusion detection systems (NIDS) for protecting modern networks. However, many machine learning and deep learning based NIDS rely on large volumes of labeled attack data, which is often impractical to obtain for newly emerging or rare attacks. This paper presents a benchmark-style systematic evaluation of meta-learning-based Few-Shot Learning (FSL) classifiers for detecting previously unseen intrusions with limited labeled data. We investigate three representative FSL models, namely Prototypical Networks, Relation Networks, and MetaOptNet, and further examine two decision-level ensemble strategies based on majority voting and probability averaging. Experiments are conducted on the UQ-IoT-IDS-2021 dataset under four attack distributions and five K-shot settings ($K \in \{3, 5, 7, 10, 15\}$), with performance reported separately for seen and unseen attacks using class-balanced macro-averaged F1 score. The experimental results show that MetaOptNet consistently provides the strongest and most stable performance on unseen attacks. Furthermore, investigating decision-level ensembling reveals that probability averaging frequently matches or marginally outperforms the best base model by successfully mitigating weaker predictions, whereas majority voting demonstrates slight performance degradation due to strict consensus requirements. We also highlight prominent error modes, such as benign-attack ambiguity and cross-device attack correlations, alongside an analysis of model complexity and inference latency. These findings highlight the potential of few-shot learning for data-scarce intrusion detection and provide insights into model selection and architectural trade-offs under varying support set sizes and attack compositions.

KEYWORDS: Few-shot learning; IoT security; network intrusion detection; meta-learning

1 Introduction

The rapid proliferation of Internet of Things (IoT) networks has led to increasingly complex and heterogeneous network environments, ranging from smart home appliances to industrial control systems. While this connectivity enables advanced automation and monitoring, it also introduces a broad attack surface and new security vulnerabilities. Consequently, network intrusion detection systems (NIDS) play a critical role in protecting IoT networks by monitoring traffic and identifying malicious activities in real time.

Recent advances in machine learning (ML) and deep learning (DL) have significantly improved the detection accuracy of NIDS by enabling automated feature learning from network traffic data [1,2]. However, most ML/DL-based intrusion detection approaches rely on large volumes of labeled attack data to achieve robust performance [3]. In realistic IoT settings, such labeled data are often scarce, particularly for newly

emerging or rare attack types. Collecting and annotating network traffic requires extensive expert effort and raises privacy concerns [4,5], making it impractical to rapidly update models for novel attacks. As a result, conventional supervised learning approaches struggle to generalize to unseen intrusion scenarios.

Few-Shot Learning (FSL) has emerged as a promising paradigm for addressing data scarcity by enabling models to generalize to new classes using only a small number of labeled samples [6]. FSL models aim to recognize novel classes with limited supervision by leveraging prior knowledge learned across related tasks. Motivated by this capability, recent studies have begun exploring FSL-based intrusion detection, demonstrating encouraging results in detecting unseen attacks under constrained data settings [7]. Nonetheless, prior work remains limited in scope, often focusing on a single FSL architecture or evaluating performance on restricted experimental configurations.

In particular, while prior studies have explored FSL techniques for intrusion detection, the impact of varying attack distributions and support-set sizes across different FSL architectures remains insufficiently studied. Moreover, while ensemble learning has been widely known to improve robustness and generalization by combining multiple learners, its potential has not been thoroughly investigated in conjunction with FSL for intrusion detection. Given the heterogeneous nature of network attacks, combining diverse FSL decision mechanisms may offer improved resilience against unseen attack behaviors.

In this paper, we conduct a benchmark-style study using three FSL architectures: namely Prototypical Networks [8], Relation Networks [9], and MetaOptNet [10]. Our primary objective is to present a systematic and comprehensive evaluation of these representative FSL models for IoT network intrusion detection, rather than proposing a new model architecture. We also investigate whether simple decision-level ensemble strategies can enhance their performance, assessing both their individual and ensemble-based performance in our evaluation. Experiments are conducted on the UQ-IoT-IDS-2021 dataset [11] under four distinct attack distributions and multiple K-shot settings, enabling a detailed analysis of model robustness and generalization to unseen attacks. Our findings highlight both the potential and the limitations of applying few-shot learning to intrusion detection, offering practical benchmarking insights for deploying FSL-based NIDS in real-world IoT environments.

The main contributions of this paper are summarized as follows:

- **Systematic Benchmarking of FSL Paradigms:** We provide a comprehensive evaluation of three distinct few-shot learning paradigms—metric-based (Prototypical Networks), relation-based (Relation Networks), and optimization-based (MetaOptNet)—specifically for detecting previously unseen IoT network attacks.
- **Robustness Analysis across Realistic IoT Scenarios:** We evaluate model performance under four diverse attack distributions (rare class, stage-wise, new device, and scaled imbalance), revealing how attack composition and device heterogeneity impact generalization.
- **Evaluation of Decision-Level Ensembles:** We demonstrate that simple ensemble strategies, particularly probability averaging, can effectively mitigate the weaknesses of individual base models to improve detection stability in data-scarce settings.
- **Practical Deployment Insights:** Through detailed error and complexity analysis, we identify critical trade-offs between detection accuracy and inference latency, providing a roadmap for deploying FSL-based NIDS on resource-constrained IoT devices.

Acronyms used in this paper are listed in Table 1.

Table 1: List of acronyms used in this paper.

Acronym	Full Name
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
EnsAvg	Ensemble with Probability Averaging
EnsMaj	Ensemble with Majority Voting
FSL	Few-Shot Learning
IoT	Internet of Things
MCU	Microcontroller Unit
ML	Machine Learning
MONet	MetaOptNet
NIDS	Network Intrusion Detection System
ProtoNet	Prototypical Network
RelNet	Relation Network
SGD	Stochastic Gradient Descent
SNN	Siamese Neural Network
SVM	Support Vector Machine

2 Related Work

2.1 Intrusion Detection in IoT Environments

NIDS plays a critical role in securing IoT environments, which are characterized by heterogeneous devices, dynamic traffic patterns, and continuously evolving attack surfaces [12]. In such settings, intrusion detection models must often operate under limited labeled data and frequent distribution shifts, particularly when new or previously unseen attack types emerge [13]. These characteristics challenge conventional supervised learning approaches, which typically assume access to large, well-labeled, and stationary datasets.

A substantial body of prior work has applied ML and DL techniques to intrusion detection in IoT networks. Early learning-based approaches relied on classical classifiers and handcrafted features, while more recent studies have leveraged deep neural networks to automatically learn hierarchical traffic representations [1,2,14]. Despite their success under controlled settings, most existing intrusion detection systems are trained in fully supervised manners and require extensive labeled data, limiting their practicality in real-world IoT deployments [4,15].

Beyond the requirement for large datasets, IoT-edge environments frequently encounter severe class imbalance, as malicious traffic samples are often rare compared to the vast volume of benign traffic. Collecting representative on-field attack samples can be complex, time-consuming, or even infeasible in critical domains due to safety and privacy constraints. To address these hurdles, recent efforts have focused on deploying lightweight, similarity-based models on resource-constrained hardware. For instance, Fusco et al. [16] demonstrated that Siamese Neural Networks (SNNs) can maintain high detection rates even with limited and imbalanced data. Their approach highlights the feasibility of utilizing TinyML for on-board training and inference on Microcontroller Units (MCUs) like the ESP32, effectively mitigating the dual challenges of data scarcity and strict computational limits.

2.2 Few-Shot Learning and Meta-Learning for Intrusion Detection

To address data scarcity and poor generalization to unseen attacks, recent research has explored alternative learning paradigms that emphasize data efficiency and adaptability. Among these, FSL has gained increasing attention as a promising framework for scenarios in which only a small number of labeled samples are available for new attack classes [7]. Rather than training a single static classifier, FSL aims to learn transferable representations or decision mechanisms that enable rapid adaptation to new tasks with minimal supervision.

Few-shot learning is commonly realized through meta-learning, where models are trained over a distribution of tasks rather than individual data samples [6]. During training, models are exposed to episodic tasks that mimic the few-shot inference setting, each consisting of a small labeled support set and an unlabeled query set. This episodic formulation has been shown to be effective in low-data intrusion detection scenarios, and several studies have explored meta-learning-based frameworks to improve detection performance under limited supervision [3,17–20].

Depending on how adaptation is performed at inference, meta-learning-based FSL methods can be broadly distinguished by their underlying decision mechanisms. Metric-based approaches classify query samples by measuring distances in an embedding space, exemplified by Matching Networks [21] and Prototypical Networks [8]. Relation-based methods extend this idea by learning a parametric similarity function that explicitly models relationships between support and query instances, such as Relation Networks [9]. Optimization-based approaches, such as MAML [22] and MetaOptNet [10], formulate few-shot classification as a task-specific optimization problem by fitting a discriminative classifier on the support set during each episode.

While episodic meta-learning represents a dominant paradigm, some researchers advocate for more computationally efficient strategies, such as fine-tuning and knowledge distillation, particularly within Few-Shot Class-Incremental Learning (FSCIL) frameworks. For instance, Cerasuolo et al. [23] analyzed fine-tuning-based incremental approaches, demonstrating that mechanisms like Bias Correction (BiC) can effectively integrate knowledge of 0-day attacks while achieving training time savings of up to 80% compared to traditional learning-from-scratch. Similarly, Du et al. [24] introduced a method (BFS-NID) that utilizes self-supervised Vision Transformers and branch fusion strategies, enabling NIDS to learn emerging attack classes from a few samples without forgetting previously seen threats. These developments highlight a shift toward lighter, incremental update mechanisms that reduce the computational burden of few-shot adaptation in dynamic IoT environments.

Comparative analyses of these approaches have been extensively conducted in domains such as computer vision, providing insights into their relative strengths under standardized benchmarks [6,25]. However, analogous comparative investigations in the context of intrusion detection remain limited. In particular, the impact of domain-specific characteristics, such as high intra-class variability, overlapping class distributions, and distribution shifts, has not been systematically examined across different meta-learning paradigms.

2.3 Ensemble Approaches for Intrusion Detection

Ensemble learning has been widely adopted in intrusion detection to improve accuracy, robustness, and generalization by combining multiple classifiers with complementary strengths [15,26]. In IoT intrusion detection, Alhowaide et al. [27] demonstrated that majority voting over selected base learners consistently outperforms individual classifiers across multiple datasets. Similarly, Cao et al. [28] showed that a stacking-based ensemble combining Naive Bayes and LightGBM significantly improves detection accuracy compared to single-model baselines.

While these studies confirm the effectiveness of ensemble strategies under fully supervised settings, their application to few-shot intrusion detection remains limited.

2.4 Positioning of this Work

Motivated by the above observations, this work serves as a benchmark-style study focusing on three representative meta-learning models, namely Prototypical Networks, Relation Networks, and MetaOptNet, as analytical tools to study few-shot intrusion detection. Unlike recent studies that primarily focus on proposing specialized architectures for TinyML deployment or developing session-based incremental learning modules to prevent forgetting, our primary objective is to present a systematic and comprehensive evaluation of existing paradigms within a unified IoT intrusion detection framework.

Rather than proposing new meta-learning architectures, we intentionally restrict adaptations to embedding-level adjustments, preserving the core learning mechanisms of each model. This allows us to isolate and compare the effects of metric-based, relation-based, and optimization-based decision strategies under varying attack distributions—including rare class, stage-wise, and cross-device scenarios—which have not been thoroughly examined in previous model-centric literature. Through this unified analysis, we seek to provide clearer comparative insights of existing few-shot learning paradigms for realistic IoT intrusion detection environments.

The key technical differences and contributions of our study are summarized in Table 2. While recent works focus on proposing specific similarity-based architectures or incremental update modules to handle limited data, our work provides a systematic evaluation that identifies which fundamental decision logic (metric, relation, or optimization) is most effective when provided with extremely small support sets of unseen attacks.

Table 2: Summary and comparison of related works in IoT intrusion detection.

Category	Key Refs.	Primary Focus	IoT Benefit	Data Scarcity Handling
Supervised	[1,14]	Hierarchical feature learning via DNNs/CNNs.	High accuracy in stationary environments.	**Low:** Requires large labeled datasets.
Ensemble	[27,28]	Combining heterogeneous learners for robustness.	Improved detection stability.	**Low:** Typically used in full-data settings.
TinyML	[16]	Few-shot similarity learning using SNN for class imbalance.	On-board training on resouce-constrained MCUs.	**High:** Resilient to extreme data scarcity (0.07% training set).
Incremental FSL	[23,24]	Fine-tuning and distillation for 0-day adaptation.	Rapid adaptation with reduced training time.	**High:** Learns new classes from 5–200 shots.
FSL & Ensemble	**This Study**	**Systematic benchmark of FSL paradigms.**	**Insights into robust strategies for IoT NIDS.**	**High:** Learns new classes from 3–15 shots.

3 Proposed Methodology

This section describes the proposed meta-learning framework for few-shot intrusion detection, as illustrated in Fig. 1.

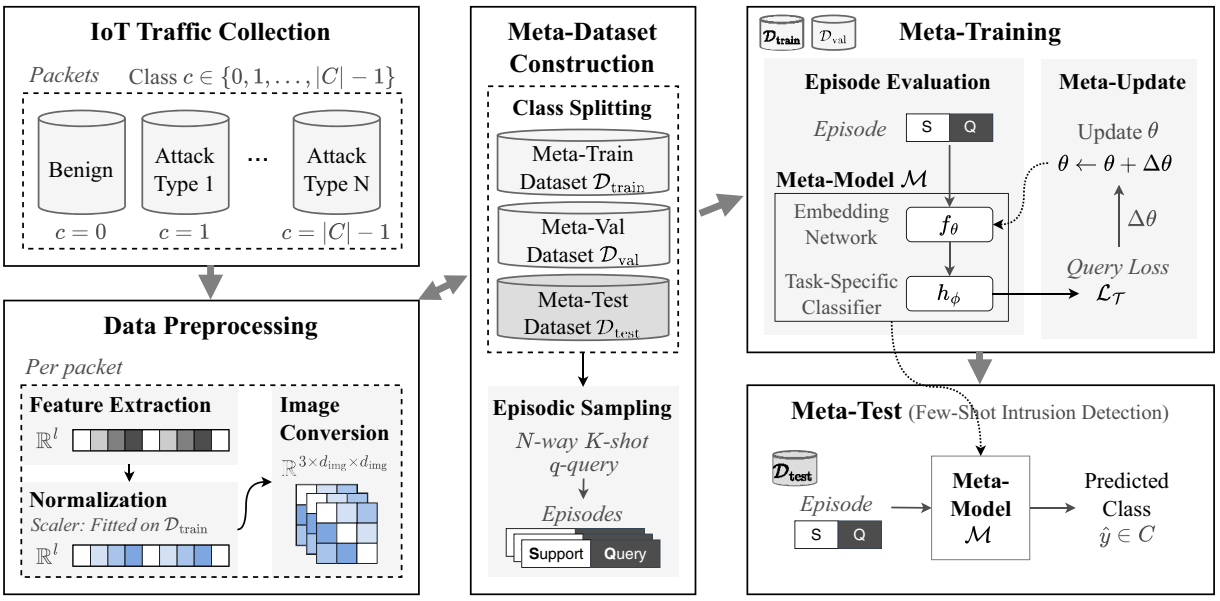

Figure 1: Overall meta-learning framework for few-shot intrusion detection. Meta-train, meta-validation, and meta-test datasets are constructed via class-level splitting. Episodic meta-training is performed using support and query samples, and the trained meta-model is evaluated on unseen attack types.

3.1 Problem Definition and Notations

We consider intrusion detection in IoT networks as a few-shot multi-class classification problem under the meta-learning paradigm. Given network traffic samples collected from IoT environments, the objective is to correctly identify benign traffic and various attack types, including those that are *unseen* during training, using only a *limited number of labeled examples*.

The main notations used in this paper are summarized in Table 3.

Table 3: Summary of notations.

Notation	Description
C	Set of traffic classes (benign and attack types)
$c \in C$	Class label
$\mathcal{D}_{\text{train}}$	Meta-train dataset
$\mathcal{D}_{\text{val}}$	Meta-validation dataset
$\mathcal{D}_{\text{test}}$	Meta-test dataset
$\mathcal{T}$	An episode (task) in meta-learning
S	Support set of an episode
Q	Query set of an episode
N	Number of classes per episode (N-way)
K	Number of support samples per class (K-shot)
q	Number of query samples per class (q-query)

(Continued)

Table 3 (continued)

Notation	Description
x	Input IoT traffic sample
y	Ground-truth class label of a sample
$\hat{y}$	Final predicted class label
l	Dimension of the temporal-statistical feature vector
d_{img}	Spatial resolution of the converted RGB image
$\mathcal{L}_{\mathcal{T}}$	Query loss for episode $\mathcal{T}$
$\mathcal{M}^{(m)}$	m-th meta-model in the ensemble
$\mathbf{p}^{(m)}$	Class probability vector output by the m-th meta-model
$f_\theta(\cdot)$	Embedding network with parameters θ
$h_\phi(\cdot)$	Task-specific classifier
$\mathcal{F}(\cdot)$	Ensemble fusion function

3.2 Data Preprocessing

To evaluate FSL methods for IoT intrusion detection, raw network traffic samples are preprocessed to preserve both packet-level attributes and their temporal context. In network intrusion detection, the interpretation of an individual packet is highly dependent on surrounding traffic patterns, as a packet that appears benign in isolation may contribute to malicious behavior when observed within a sequence of similar packets. To capture such contextual information, temporal-statistical feature extraction is employed.

Specifically, we adopt the Kitsune feature extractor [29], which processes raw network packets and applies damped incremental statistics to derive implicit contextual features from the traffic stream. This approach enables the extraction of temporal-statistical representations that encode both the characteristics of the current packet and the recent behavioral context of the network traffic. As a result, each packet is represented as a feature vector of length $l = 100$.

The extracted features exhibit varying scales, which can negatively impact the training of few-shot learning models. Therefore, a Min-Max normalization is applied to rescale all features to the range $[0, 1]$. To maintain the statistical independence of each split and eliminate potential leakage from future data, normalization parameters are determined based on the meta-training set and then applied to the meta-validation and meta-test sets.

FSL architectures, such as the Prototypical Networks and MetaOptNet evaluated in this study, typically utilize CNNs as embedding backbones designed for image-based domains. However, the temporal-statistical features extracted from network traffic are one-dimensional, posing a structural mismatch with standard 2D CNN backbones. To address this without introducing unintended performance artifacts or errors from extensive architectural redesigns, we transform the 1D feature vectors into 2D image-like tensors. This approach preserves each model's original detection logic and enables an "as-is" comparative evaluation of their decision mechanisms within a unified framework. The feature-to-image conversion is performed on a per-sample basis, ensuring that the transformation of a specific sample does not rely on statistics from other samples.

Concretely, each 1D feature vector of length $l = 100$ is reshaped into a 2D matrix of size $d_{\text{img}} \times d_{\text{img}}$, where ${d_{\text{img}}}^2 = l$. Thus, $d_{\text{img}} = 10$. For few-shot classifiers that require three-channel image inputs, the resulting matrix is replicated across three channels to form an RGB image tensor of shape $3 \times d_{\text{img}} \times d_{\text{img}}$.

These image representations are then used as inputs to the shared embedding network in the proposed meta-learning framework.

3.3 Meta-Dataset Construction

To support few-shot learning under the meta-learning paradigm, the preprocessed IoT traffic data are organized into meta-datasets using a class-aware splitting strategy. This strategy is specifically chosen to prevent label-related leakage; the dataset was partitioned at the class level prior to data preprocessing. This ensures that attack types designated as "unseen" in the meta-test set ($\mathcal{D}_{\text{test}}$) are strictly excluded from the meta-training ($\mathcal{D}_{\text{train}}$) and meta-validation ($\mathcal{D}_{\text{val}}$) phases. The meta-test dataset is constructed to include both classes observed during meta-training and previously unseen attack types, while ensuring that these samples are not used in the meta-training and meta-validation phases. This design reflects realistic intrusion detection scenarios, in which known traffic patterns and novel attacks may coexist at deployment time.

Following class-level partitioning, episodic sampling is applied to construct few-shot tasks from each meta-dataset. Each episode consists of a support set and a query set sampled according to an N-way K-shot protocol. Following the standard meta-learning protocol, each task maintains a rigorous separation between the support set S and the query set Q. An overview of the meta-dataset construction and episodic sampling process is illustrated in Fig. 2.

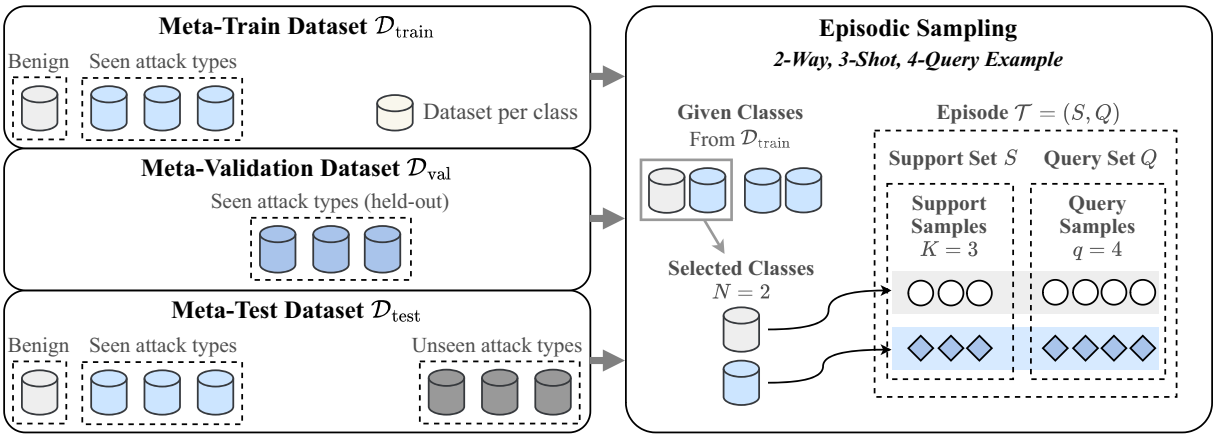

Figure 2: Meta-dataset construction and episodic sampling process. The dataset is split into meta-train, meta-validation, and meta-test sets, followed by episode construction with support and query samples. An example of 2-way 3-shot 4-query episodic sampling from the meta-train dataset is shown.

3.4 Meta-Training

Meta-training aims to learn a transferable embedding function that enables rapid adaptation to new intrusion detection tasks with limited labeled data. Given an episode, the shared embedding network $f_\theta(\cdot)$ maps each input sample to a fixed-dimensional latent representation.

Using the support set embeddings, a task-specific classifier h_ϕ is instantiated for the episode. The classifier parameters ϕ are derived solely from the support set and are specific to the current task. For each query sample x, the meta-model produces a class probability vector $\mathbf{p} = h_\phi\big(f_\theta(x)\big)$.

The meta-learning objective is computed based on the prediction loss over query samples. This loss is used to update only the shared embedding parameters θ via gradient-based optimization. The task-specific parameters ϕ are discarded after each episode, ensuring that knowledge is accumulated exclusively in the shared embedding network.

3.5 Meta-Test

Meta-testing evaluates the generalization capability of the trained meta-model under few-shot intrusion detection scenarios. The learned embedding network f_θ is fixed during this stage, and no parameter updates are performed.

For each evaluation task, a task-specific classifier h_ϕ is instantiated from the available support samples using the same mechanism as in meta-training. Query samples are then mapped to the embedding space and classified by the task-specific classifier to produce class probabilities.

3.6 Adaptation of Meta-Models for Intrusion Detection

FSL models are primarily developed for computer vision tasks, where inputs are natural images with well-defined spatial structures and low intra-class variability. In contrast, network intrusion detection relies on network traffic data that differ substantially in structure and semantics and often exhibit high variability even within the same attack class. To assess the applicability of FSL under these conditions, we consider multiple task-specific meta-learning models during meta-training. Specifically, Prototypical Networks [8], Relation Networks [9], and MetaOptNet [10] are adopted as base meta-models, representing metric-based, relation-based, and optimization-based paradigms for few-shot classification.

Prototypical Networks perform classification by measuring distances to class prototypes under the assumption that samples from the same class form compact clusters in the embedding space. Since intrusion detection data often violate this assumption due to protocol diversity and contextual dependence, the prototype-based formulation is retained while the convolutional embedding architecture is adapted to the reduced input resolution of network traffic features. This isolates the effectiveness of distance-based representations in this domain. Fig. 3 illustrates the adapted embedding function architecture of the Prototypical Network.

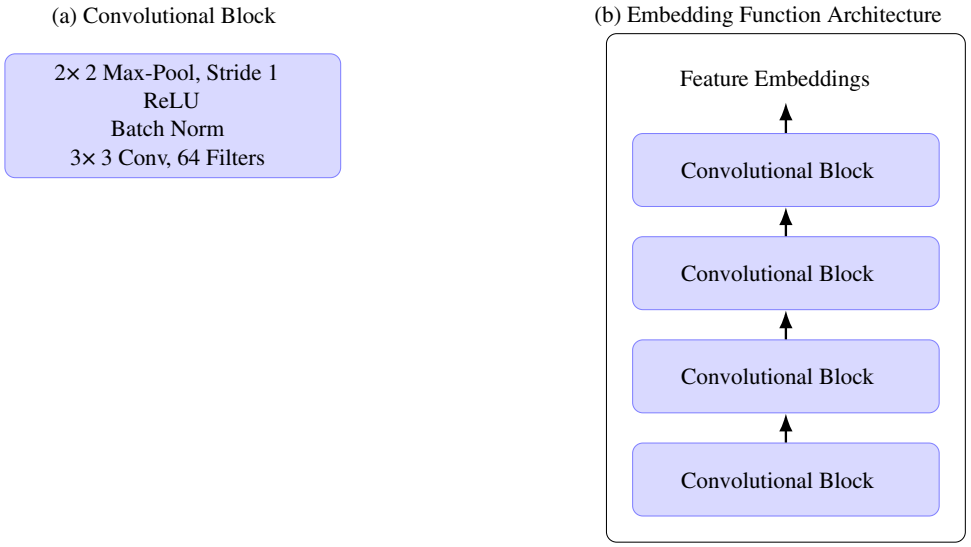

Figure 3: Prototypical network embedding function architecture.

Relation Networks classify samples using a learned similarity function between support and query instances, which can be sensitive to noise and support set composition in heterogeneous traffic data. To reduce information loss, the embedding architecture is modified through adjusted padding and pooling strategies, while the relation module and its regression-based loss are preserved. This enables an evaluation

of relational learning robustness for intrusion detection. Fig. 4 illustrates the adapted embedding function architecture of the Relation Network.

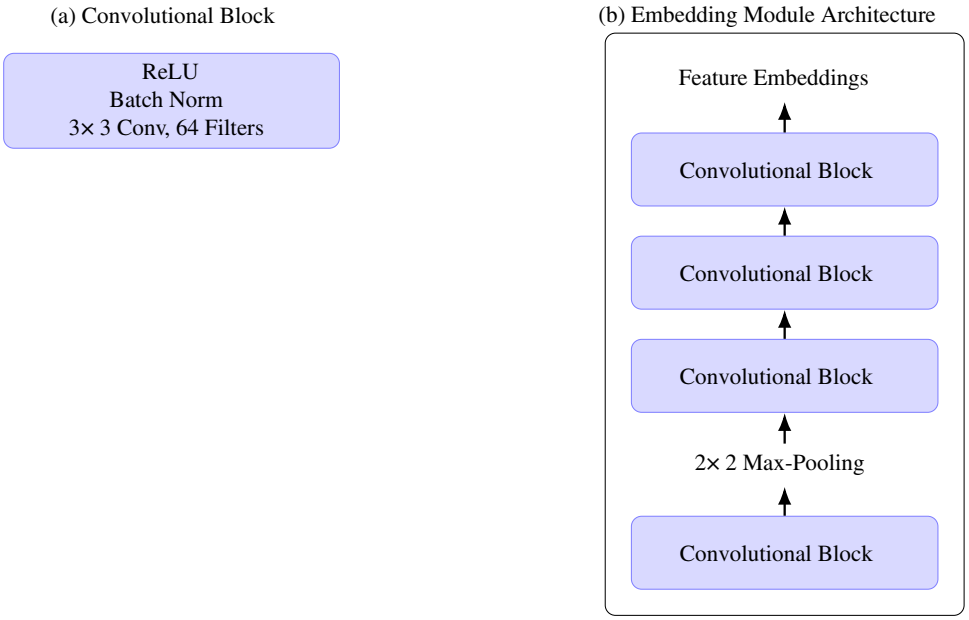

Figure 4: Relation network embedding module architecture.

MetaOptNet frames the few-shot classification as a discriminative learning problem by fitting an SVM to the support set for each episode. This formulation is well suited for intrusion detection, where decision boundaries between benign and malicious traffic are often more informative than compact class representations. Accordingly, the ResNet-based embedding is adapted to smaller input dimensions, while the SVM-based classification head remains unchanged to explicitly learn support-constrained decision boundaries. We omit an architecture figure for MetaOptNet, as its embedding follows a standard backbone.

Overall, all adaptations are intentionally limited to embedding-level architectural adjustments, preserving the fundamental learning mechanisms of each FSL model. This ensures that performance comparisons reflect intrinsic differences between prototype-based, relation-based, and boundary-based learning approaches, rather than artifacts arising from domain incompatibility.

3.7 Ensemble of Meta-Models

In addition to evaluating individual meta-learning models, we investigate the potential of ensemble-based decision fusion in the context of few-shot intrusion detection. Due to differences in inductive bias and task-specific decision mechanisms, distinct meta-models may produce complementary or conflicting predictions when operating under limited labeled data.

Fig. 5 illustrates the ensemble process. Given a query sample, each meta-model independently produces a class probability vector based on its episode-specific classifier. In this work, we apply and compare two decision-level ensemble fusion strategies: (1) probability averaging, which aggregates class probability vectors across meta-models and predicts the class with the highest averaged probability, and (2) majority voting, which selects the class predicted by the majority of meta-models [30,31]. We intentionally scope our ensemble analysis to simple decision-level fusion strategies to assess the intrinsic complementarity of

different FSL paradigms without introducing additional trainable parameters or modifying the underlying meta-learning models.

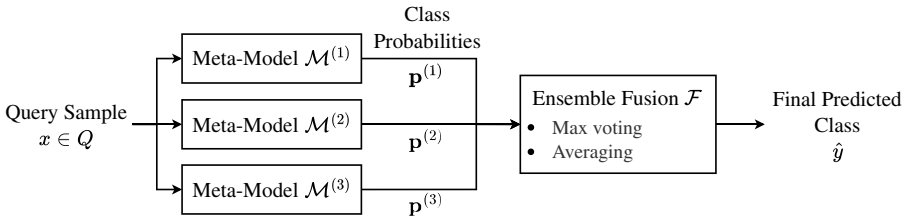

Figure 5: Ensemble of multiple meta-models for few-shot intrusion detection. Each meta-model outputs class probabilities for a query sample, which are aggregated by an ensemble fusion strategy to produce the final predicted traffic class.

4 Experimental Setup

4.1 Dataset and Preprocessing

We evaluate the proposed few-shot intrusion detection framework on the UQ-IoT-IDS-2021 dataset [11], which comprises benign traffic and a diverse set of real-world IoT attack scenarios collected from an emulated smart-home environment. The dataset is designed to reflect realistic IoT network behavior by incorporating heterogeneous devices (e.g., smartphones, a smart TV, an IP camera, a smart speaker, and a Raspberry Pi), multiple attack categories, and naturally imbalanced traffic distributions. These characteristics make the dataset a suitable benchmark for assessing generalization to both seen and unseen attacks under limited labeled data.

As described in Section 3.2, raw PCAP traffic is transformed into image-like tensors of size $[3, d_{\text{img}}, d_{\text{img}}]$ to match the standard input format of CNN-based meta-learning backbones.

4.2 Attack Distributions and Data Splits

To mitigate computational overhead while preserving a realistic and controlled evaluation environment, we utilized a representative subset of devices and attack types from the original dataset. Specifically, we selected network traffic from an IP camera (Cam 1), a Raspberry Pi, and a smart bulb (Bulb). Due to the substantial class imbalance inherent in the dataset, large-scale classes were down-sampled without replacement to mitigate skewness and ensure comparability across classes. Addressing this imbalance is critical, as it is recognized to interact negatively with few-shot learning paradigms, potentially disproportionately impairing generalization performance for minority classes [32].

To assess the robustness of the few-shot learners in various scenarios, we consider four attack distributions:

- **Distribution A (Rare class scenario):** Motivated by [20], this scenario assigns larger attack classes to $\mathcal{D}_{\text{train}}$ and smaller classes to $\mathcal{D}_{\text{test}}$. This setting reflects a practical environment where frequently observed attacks provide sufficient labeled data, whereas rare attacks remain data-scarce.
- **Distribution B (Stage-wise scenario):** This scenario evaluates the model's capacity to generalize from foundational or representative attacks to more complex variants within the same functional stage of the cyber kill chain. Attacks are categorized into three stages: reconnaissance (Host Discovery, Port Scanning, Service Detection), exploitation (ARP Spoofing, Telnet Brute Force), and denial of service (UDP, SYN, ACK, and HTTP Flooding). Classes within each stage are distributed across $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$,

and $\mathcal{D}_{\text{test}}$ to test whether the FSL model can identify unseen, advanced variants of similar malicious behaviors during inference.

- **Distribution C (New device scenario):** Specifically designed to assess cross-device robustness, this distribution partitions attack types based on their source IoT device. It assigns traffic from the IP camera to $\mathcal{D}_{\text{train}}$, traffic from the Raspberry Pi to $\mathcal{D}_{\text{val}}$, and traffic from the smart bulb to $\mathcal{D}_{\text{test}}$ to ensure the evaluation of device-independent feature learning.
- **Distribution D (Scaled imbalance scenario):** Based on the class composition of Distribution A, this scenario utilizes significantly less down-sampling for $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$ to preserve the natural, extreme class imbalance typical of real-world network traffic. The number of samples in $\mathcal{D}_{\text{test}}$ is kept constant to ensure a fair performance comparison with the other distributions.

For attacks selected for meta-training, we apply a 70:30 split so that the same attack types appear in both $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ (*seen* during meta-test), while additional attack types are held out and introduced only during testing (*unseen*). Tables 4–7 summarize the resulting splits and the number of samples used in the experiments.

Table 4: Distribution A: rare class scenario.

Set	Type	Traffic Class	# Samples
$\mathcal{D}_{\text{train}}$	Seen	Benign	70,000
		Telnet Brute Force (Raspberry Pi)	70,000
		ACK Flooding (Cam 1)	70,000
		UDP Flooding (Cam 1)	70,000
$\mathcal{D}_{\text{val}}$	Seen (held-out)	Host Discovery	22,775
		SYN Flooding (Cam 1)	100,000
		HTTP Flooding (Raspberry Pi)	100,000
$\mathcal{D}_{\text{test}}$	Seen	Benign	30,000
		Telnet Brute Force (Raspberry Pi)	30,000
		ACK Flooding (Cam 1)	30,000
		UDP Flooding (Cam 1)	30,000
	Unseen	Port Scanning (Cam 1)	4213
		Service Detection (Cam 1)	7533
		ARP Spoofing (Cam 1)	621

Table 5: Distribution B: stage-wise scenario.

Set	Type	Traffic Class	# Samples
$\mathcal{D}_{\text{train}}$	Seen	Benign	70,000
		Host Discovery	15,943
		ARP Spoofing (Cam 1)	435
		UDP Flooding (Cam 1)	70,000

(Continued)

Table 5 (continued)

Set	Type	Traffic Class	# Samples
$\mathcal{D}_{\text{val}}$	Seen (held-out)	Port Scanning (Cam 1)	4213
		SYN Flooding (Cam 1)	100,000
		ACK Flooding (Cam 1)	100,000
$\mathcal{D}_{\text{test}}$	Seen	Benign	30,000
		Host Discovery	6832
		ARP Spoofing (Cam 1)	186
		UDP Flooding (Cam 1)	30,000
	Unseen	Service Detection (Cam 1)	7533
		Telnet Brute Force (Raspberry Pi)	100,000
		HTTP Flooding (Raspberry Pi)	100,000

Table 6: Distribution C: new device scenario.

Set	Type	Traffic Class	# Samples
$\mathcal{D}_{\text{train}}$	Seen	Benign	70,000
		Service Detection (Cam 1)	5273
		ARP Spoofing (Cam 1)	435
		UDP Flooding (Cam 1)	70,000
$\mathcal{D}_{\text{val}}$	Seen (held-out)	Service Detection (Raspberry Pi)	4949
		ARP Spoofing (Raspberry Pi)	220
		UDP Flooding (Raspberry Pi)	100,000
$\mathcal{D}_{\text{test}}$	Seen	Benign	30,000
		Service Detection (Cam 1)	2260
		ARP Spoofing (Cam 1)	186
		UDP Flooding (Cam 1)	30,000
	Unseen	Service Detection (Bulb)	5002
		ARP Spoofing (Bulb)	759
		UDP Flooding (Bulb)	100,000

Table 7: Distribution D: scaled imbalance scenario based on distribution A.

Set	Type	Traffic Class	# Samples
$\mathcal{D}_{\text{train}}$	Seen	Benign	1,000,000
		Telnet Brute Force (Raspberry Pi)	98,653
		ACK Flooding (Cam 1)	1,000,000
		UDP Flooding (Cam 1)	1,000,000

(Continued)

Table 7 (continued)

Set	Type	Traffic Class	# Samples
$\mathcal{D}_{\text{val}}$	Seen (held-out)	Host Discovery	22,775
		SYN Flooding (Cam 1)	797,217
		HTTP Flooding (Raspberry Pi)	683,408
$\mathcal{D}_{\text{test}}$	Seen	Benign	30,000
		Telnet Brute Force (Raspberry Pi)	30,000
		ACK Flooding (Cam 1)	30,000
		UDP Flooding (Cam 1)	30,000
	Unseen	Port Scanning (Cam 1)	4213
		Service Detection (Cam 1)	7533
		ARP Spoofing (Cam 1)	621

4.3 Few-Shot Task Configuration

We follow the episodic meta-learning protocol described in Section 3 and apply an identical task configuration across all evaluated meta-learning models. During meta-training and meta-validation, we use 3-way ($N = 3$) classification episodes to learn transferable representations and task-level decision rules from seen attack classes. For the K-shot setting, we evaluate $K \in \{3, 5, 7, 10, 15\}$ to analyze the impact of support-set size on generalization. In training and validation episodes, the query set contains $q = 15$ samples per class.

During meta-testing, we evaluate models on a more challenging 7-way ($N = 7$) classification setting that includes both seen and unseen attack classes. Each test episode contains $N \times K$ support samples and $N \times q$ query samples, where $K \in \{3, 5, 7, 10, 15\}$ and $q = 15$. All results are averaged over 1000 test episodes. This asymmetric task configuration is adopted to reflect a realistic deployment scenario in which a model is trained on a limited number of well-characterized attack types, but is required to discriminate among a larger and more diverse set of attacks at test time.

To ensure fair comparison, all meta-models and ensemble variants are evaluated using the same episodic splits, with shared support and query sets constructed via a bootstrapped data loader.

4.4 Compared Methods and Implementation Details

We evaluate three representative meta-learning approaches: Prototypical Networks, Relation Networks, and MetaOptNet. For Prototypical Networks and Relation Networks, we use CNN-based embedding functions adapted to the reduced input size, consistent with the architectures illustrated in Figs. 3 and 4. For MetaOptNet, we use an SVM classification head with a ResNet-12 embedding function, with minor architectural adjustments to accommodate the smaller image input.

Model-specific hyperparameters follow the original papers and the thesis implementation. Prototypical Networks and Relation Networks are trained with Adam [33] (learning rate 10^{-3}). MetaOptNet is trained with SGD (initial learning rate 0.1, Nesterov momentum 0.9, weight decay 5×10^{-4}).

We follow the original training protocols of each few-shot learner, as their implementations adopt different notions of an "epoch" and update schedules. For Prototypical Networks, we train for 40 epochs, where each epoch consists of 200 training episodes (i.e., 8000 episodes in total); we also utilize a StepLR scheduler with a step size of 10 and a decay factor (γ) of 0.5. For Relation Networks, we train for 20,000 episodes using a single iterative loop and evaluate on the validation set every 500 episodes. For MetaOptNet,

we train for 40 epochs, where each epoch consists of 500 training episodes (i.e., 20,000 episodes in total). For all models, we select the checkpoint that achieves the highest validation accuracy, where each validation phase is evaluated over 500 episodes.

4.5 Ensemble Evaluation

We evaluate two decision-level ensemble fusion strategies: (i) majority voting and (ii) probability averaging (Section 3.7). Both fusion methods combine the predictions of independently trained meta-models without introducing additional trainable parameters.

5 Evaluation

For brevity, figures use the following abbreviations for the evaluated models: ProtoNet (Prototypical Networks), RelNet (Relation Networks), MONet (MetaOptNet), EnsAvg (ensemble with probability averaging), and EnsMaj (ensemble with majority voting).

5.1 Metrics

We evaluate few-shot intrusion detection performance separately on *seen* and *unseen* classes. For seen classes, we report accuracy and macro-averaged F1 score to reflect both overall correctness and class-balanced performance. For unseen classes, which are the primary focus of this study, we report accuracy as well as macro-averaged precision, recall, and F1 score.

Let C be the set of evaluated classes (either seen or unseen), and let TP_c, FP_c, and FN_c denote the true positives, false positives, and false negatives for class $c \in C$. Per-class precision, recall, and F1-score are defined as:

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c}, \quad \text{Recall}_c = \frac{TP_c}{TP_c + FN_c}, \quad \text{F1}_c = \frac{2\,\text{Precision}_c\,\text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}.$$

Macro-averaged metrics are computed by equally weighting classes:

$$\text{Macro-Precision} = \frac{1}{|C|}\sum_{c\in C}\text{Precision}_c,\ \text{Macro-Recall} = \frac{1}{|C|}\sum_{c\in C}\text{Recall}_c,\ \text{Macro-F1} = \frac{1}{|C|}\sum_{c\in C}\text{F1}_c.$$

5.2 Comparison across Attack Distributions

As detailed in Section 4.2, we evaluated the detection performance of the Few-Shot Learning (FSL) models across four distinct scenarios: Distribution A (Rare Class), B (Stage-wise), C (New Device), and D (Scaled Imbalance). Fig. 6 presents the macro-averaged F1 scores for all evaluated models across 3, 5, 7, 10, and 15-shot settings. We utilize the macro-averaged F1 score as our primary metric to provide a balanced assessment of detection performance across all classes, ensuring that the results are not disproportionately skewed by the inherent class imbalance of the dataset.

Comparison of FSL Models:

Across all evaluated distributions, MONet and EnsAvg consistently achieved the highest F1 scores, reaching peak performance between 0.71 and 0.80 depending on the specific distribution (indicated by the red dashed lines in Fig. 6). ProtoNet generally exhibited slightly lower performance compared to the leading models, with F1 scores trailing by up to 0.08. In contrast, RelNet demonstrated the poorest performance among all candidates, with scores largely concentrated in the 0.5–0.6 range. This indicates that RelNet struggled to effectively classify traffic within the test sets, failing to learn the underlying patterns of the vehicular network traffic in these few-shot scenarios.

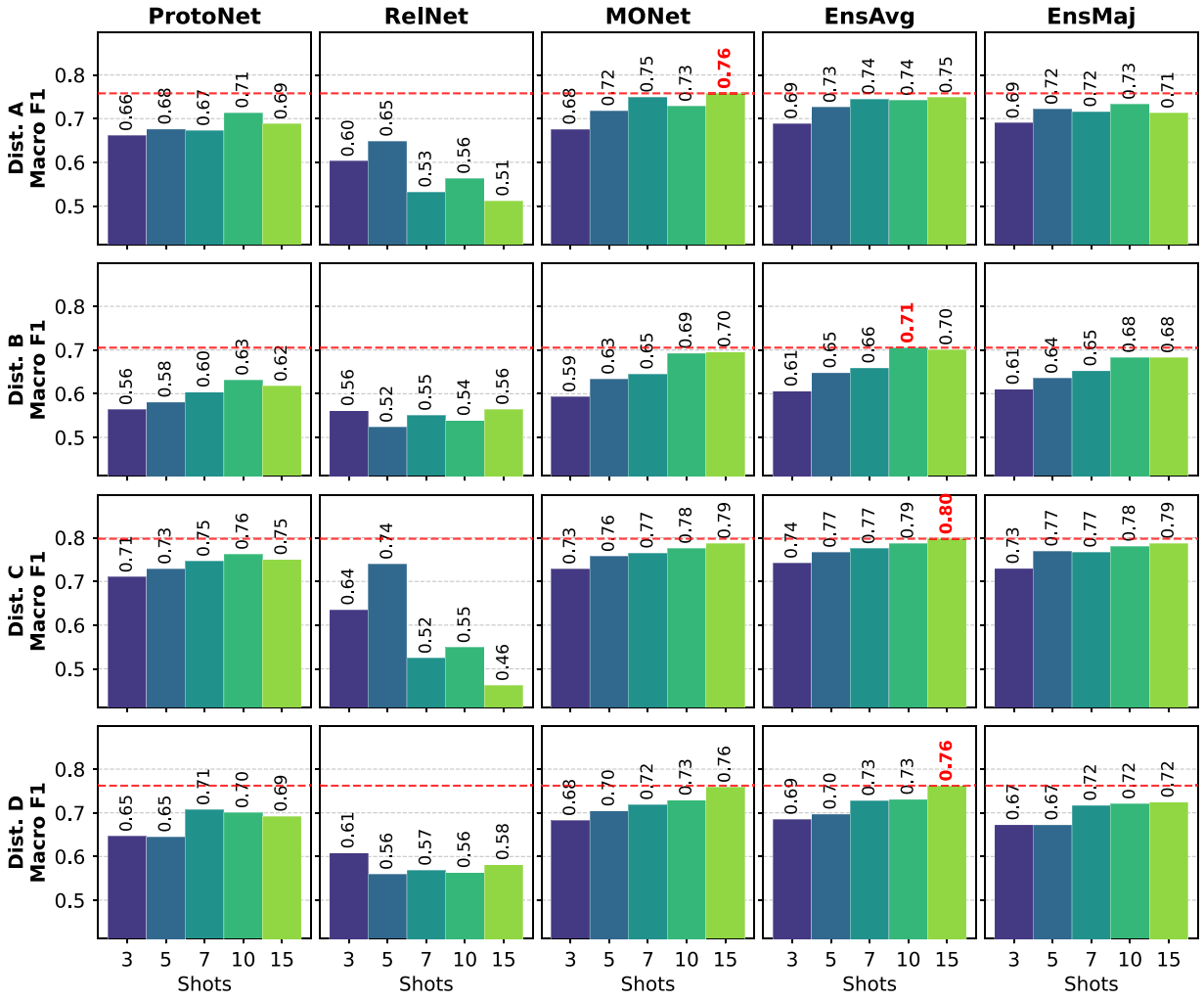

Figure 6: Overall F1 scores under Distribution A–D across models and K-shot.

Performance of Ensemble Models:

The results for the ensemble models, EnsAvg and EnsMaj, closely mirror the performance of MONet, suggesting that the ensembles are heavily influenced by the most robust base model. EnsAvg frequently matched or marginally outperformed MONet, demonstrating that averaging class probabilities can effectively mitigate the impact of lower-performing models within the ensemble, leading to more accurate global decisions. Conversely, EnsMaj showed a slight performance degradation of 0.02 to 0.04 compared to the top-tier scores. This suggests that the "hard voting" mechanism is more sensitive to individual model misclassifications, as the relatively strict consensus requirement allows incorrect predictions from weaker models to more easily adversely affect the final output.

Comparison of Scenarios:

Among the four scenarios, Distribution B (Stage-wise Scenario) proved to be the most challenging for all FSL models. Models trained on foundational or basic attack stages (e.g., Host Discovery, ARP Spoofing) struggled to generalize to more advanced or sophisticated variants (e.g., Service Detection, Telnet Brute Force) within the same functional category.

Conversely, the highest performance was observed in Distribution C (New Device Scenario). While this scenario introduces a previously unseen device (the smart bulb), the attack types themselves were consistent with those encountered during the training phase (IP camera and Raspberry Pi). This discrepancy indicates that the consistency of attack patterns—rather than the device-specific characteristics—is the primary driver of intrusion detection performance in vehicular and IoT environments.

Finally, we compared Distribution A (balanced via down-sampling) and Distribution D (natural extreme imbalance). Interestingly, the performance gap between these two scenarios was negligible, typically ranging between 0.01 and 0.03 for the same models. For instance, both MONet and EnsAvg reached an identical peak F1 score of 0.76 in their respective best settings. We attribute this minimal difference to two factors: first, the episodic sampling mechanism in FSL ensures that the model is exposed to a consistent number of samples per task regardless of total class size; second, the 70,000 samples per class provided in Distribution A were likely sufficient to statistically represent the core feature space, making further increases in data volume less impactful.

Impact of Support Size:

Taken together, the results show that increasing K does not universally yield monotonic improvements across methods. While MONet generally benefits from larger support sets (with consistent gains in macro-F1 under most distributions), the effects are weaker or inconsistent for ProtoNet and RelNet.

5.3 Comparison across Traffic Types

The performance of the models varies significantly depending on the traffic class, as shown in Tables 8–11. Volumetric attacks, such as UDP Flooding and ACK Flooding, consistently achieve high detection rates (F1 scores > 0.95) across all models and distributions. This high performance can be attributed to their distinct traffic profiles, which are characterized by high packet transmission rates and repetitive packet structures that are easily captured by the temporal-statistical features. Conversely, probing and scanning attacks, such as Service Detection and Port Scanning, present a greater challenge. These attacks often exhibit traffic patterns that closely resemble benign background traffic or other reconnaissance activities, leading to lower and more variable F1 scores.

Table 8: Performance comparison on Distribution A (15-shot). The rows for the unseen classes are in gray. The best results across the models are in **bold**.

Class	ProtoNet			RelNet			MONet			EnsAvg			EnsMaj		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Benign	0.695	0.543	0.610	0.335	0.218	0.264	0.778	0.594	0.674	**0.779**	**0.614**	**0.687**	0.773	0.581	0.663
Telnet Brute Force	0.822	0.861	0.841	0.788	0.903	0.842	0.838	0.884	0.860	**0.839**	0.902	**0.869**	0.824	**0.907**	0.863
ACK Flooding (Cam 1)	0.817	0.966	0.885	0.724	**0.978**	0.832	**0.993**	0.959	**0.976**	0.904	0.965	0.934	0.817	0.970	0.887
UDP Flooding (Cam 1)	1.000	0.989	**0.994**	**1.000**	0.989	0.994	0.995	**0.990**	0.993	0.999	0.990	0.994	0.999	0.989	0.994
Port Scanning (Cam 1)	0.613	0.434	0.508	0.423	0.209	0.280	0.642	**0.829**	**0.723**	**0.651**	0.769	0.705	0.633	0.465	0.536
Service Detection (Cam 1)	0.358	0.326	0.341	0.211	**0.478**	0.293	**0.575**	0.292	**0.387**	0.538	0.270	0.359	0.392	0.340	0.364
ARP Spoofing (Cam 1)	0.547	0.755	0.634	0.434	0.044	0.080	0.590	**0.839**	**0.693**	**0.590**	0.824	0.688	0.590	0.807	0.681

Table 9: Performance comparison on Distribution B (15-shot). The rows for the unseen classes are in gray. The best results across the models are in **bold**.

Class	ProtoNet			RelNet			MONet			EnsAvg			EnsMaj		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Benign	0.370	0.153	0.216	0.331	0.218	0.263	0.428	**0.407**	**0.417**	**0.433**	0.323	0.370	0.429	0.265	0.328
Host Discovery	0.373	0.391	0.382	0.465	0.401	0.430	0.504	0.510	0.507	**0.520**	**0.540**	**0.530**	0.515	0.485	0.500
ARP Spoofing (Cam 1)	0.563	0.807	0.663	**0.666**	**0.839**	**0.743**	0.595	0.766	0.670	0.600	0.803	0.686	0.620	0.825	0.708
UDP Flooding (Cam 1)	0.982	0.983	0.983	**1.000**	**0.986**	**0.993**	0.989	0.982	0.985	0.993	0.984	0.989	0.994	0.985	0.990
Service Detection (Cam 1)	0.488	0.592	0.535	0.301	0.334	0.316	**0.735**	0.579	0.648	0.729	0.606	**0.662**	0.583	**0.622**	0.602
Telnet Brute Force	0.622	0.655	0.638	0.512	0.397	0.447	**0.673**	0.661	0.667	0.660	**0.712**	**0.685**	0.646	0.692	0.668
HTTP Flooding	0.988	0.834	0.904	0.659	0.888	0.756	0.983	0.968	0.975	**0.995**	0.970	**0.983**	0.992	**0.972**	0.982

Table 10: Performance comparison on Distribution C (15-shot). The rows for the unseen classes are in gray. The best results across the models are in **bold**.

Class	ProtoNet			RelNet			MONet			EnsAvg			EnsMaj		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Benign	0.885	0.812	0.847	**0.930**	0.848	**0.887**	0.865	0.868	0.867	0.872	0.869	0.870	0.875	**0.871**	0.873
Service Detection (Cam 1)	0.935	0.655	0.770	0.728	0.009	0.018	0.907	0.669	0.770	0.961	**0.680**	**0.796**	**0.969**	0.625	0.760
ARP Spoofing (Cam 1)	0.582	0.379	0.459	**0.638**	0.021	0.040	0.518	**0.688**	**0.591**	0.561	0.494	0.525	0.550	0.482	0.514
UDP Flooding (Cam 1)	0.996	0.995	0.996	**1.000**	0.982	0.991	0.998	0.995	0.996	0.998	**0.996**	**0.997**	0.999	0.995	0.997
Service Detection (Bulb)	0.703	0.980	0.819	0.545	0.918	0.684	0.848	0.946	0.894	**0.860**	0.966	**0.910**	0.819	**0.981**	0.893
ARP Spoofing (Bulb)	0.388	**0.594**	0.469	0.438	0.083	0.140	**0.457**	0.369	0.408	0.440	0.572	**0.497**	0.437	0.560	0.491
UDP Flooding (Bulb)	**0.999**	0.803	0.890	0.311	**0.990**	0.473	0.979	0.987	0.983	0.991	0.986	**0.988**	0.982	0.988	0.985

Table 11: Performance comparison on Distribution D (15-shot). The rows for the unseen classes are in gray. The best results across the models are in **bold**.

Class	ProtoNet			RelNet			MONet			EnsAvg			EnsMaj		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Benign	0.690	0.526	0.597	0.314	0.334	0.324	0.716	0.600	0.653	**0.749**	**0.636**	**0.688**	0.748	0.601	0.666
Telnet Brute Force	0.787	0.864	0.824	0.736	0.873	0.799	0.824	0.806	0.815	**0.840**	0.862	**0.851**	0.801	**0.891**	0.844
ACK Flooding (Cam 1)	0.919	0.963	0.940	0.849	**0.977**	0.909	**0.994**	0.959	**0.976**	0.963	0.964	0.964	0.926	0.968	0.946
UDP Flooding (Cam 1)	**1.000**	**0.989**	**0.994**	**1.000**	0.984	0.992	0.993	**0.989**	0.991	0.998	**0.989**	**0.994**	0.999	**0.989**	**0.994**
Port Scanning (Cam 1)	0.600	0.454	0.517	0.595	0.349	0.440	**0.667**	**0.800**	**0.727**	**0.667**	0.775	0.717	0.634	0.502	0.560
Service Detection (Cam 1)	0.359	0.323	0.340	0.224	0.321	0.264	**0.569**	**0.375**	**0.452**	0.564	0.339	0.424	0.399	0.353	0.374
ARP Spoofing (Cam 1)	0.536	0.774	0.633	0.457	0.275	0.343	**0.603**	0.823	**0.696**	**0.603**	**0.825**	**0.696**	0.593	0.813	0.685

Interestingly, the evaluation of cross-device generalizability in Distribution C (Table 10) reveals that models can successfully detect unseen variants of known attacks originating from novel devices. For instance, the ensemble models achieved an F1 score of 0.910 for the unseen Service Detection (Bulb) class, which is comparable to or even better than the performance on the seen Service Detection (Cam 1) class (F1 score of 0.796). This suggests that the few-shot learners are capable of capturing device-agnostic, attack-specific features rather than over-fitting to device-specific network signatures.

5.4 Error Analysis

To better understand the remaining misclassification patterns, we analyzed the 15-shot confusion matrices for the evaluated models. Several consistent error modes emerged across the distributions.

First, a prominent source of error is the *Benign-Attack Ambiguity*. Across all methods, benign traffic was frequently misclassified as attack traffic. For example, in Distribution A under MONet, approximately 25% of the benign query samples were incorrectly classified as ARP Spoofing. This ambiguity is primarily due to the diverse and unconstrained nature of benign IoT traffic, which can occasionally mimic the temporal patterns of certain attacks, making it difficult to construct a tight bounding representation for the benign class from a limited number of support samples.

Second, the models exhibited notable *Inter-Class Confusion*, particularly between highly correlated probing attacks. In Distributions A and D, a significant portion of the unseen *Service Detection* samples were misclassified as *Port Scanning* (e.g., up to 37% under MetaOptNet). Because both attacks involve systematic probing of target ports and services, their temporal-statistical footprints are highly similar, causing the algorithms to conflate the two categories when explicit decision boundaries are difficult to infer from few examples.

Finally, an analysis of the new device scenario (Distribution C) highlighted a compelling *Cross-Device Correlation*. When the models were tested on the unseen *ARP Spoofing (Bulb)* class, nearly 47% of the samples were misclassified as the seen *ARP Spoofing (Cam 1)* class instead of being categorized as a different attack or benign traffic. Rather than representing a failure of detection, this pattern indicates that the models successfully abstracted the core structural characteristics of the ARP Spoofing attack. The misclassification occurred primarily at the device label level rather than the attack type level, further demonstrating the models' capacity to learn generalizable attack representations across heterogeneous IoT devices.

5.5 Model Complexity Analysis

This section evaluates the model size and inference latency of the studied architectures. All measurements were conducted on a workstation featuring an Intel Core i9-14900K CPU and an NVIDIA GeForce RTX 4080 Super GPU. The models were implemented using the PyTorch library in a Python environment.

Table 12 shows the architectural differences between the models. ProtoNet and RelNet employ a shallow CNN structure with four convolutional blocks, maintaining a minimal memory footprint of less than 1 MB. Conversely, MONet and its ensemble variants (EnsAvg and EnsMaj) demonstrate significantly higher complexity, exceeding 12 million parameters and requiring approximately 48 MB of storage. This increase is attributed to the ResNet-based backbone, which provides the capacity to capture high-dimensional patterns in vehicular network traffic, albeit at the cost of higher resource consumption.

Table 12: Comparison of model parameters and size (independent of N-way K-shot settings and distributions).

Model	Parameters	Size (MB)
ProtoNet	113,088	0.43
RelNet	224,593	0.86
MONet	12,424,321	47.40
EnsAvg	12,762,002	48.69
EnsMaj	12,762,002	48.69

Consistent with the model size, the inference speeds of MONet and the ensemble variants are markedly slower than those of ProtoNet and RelNet (Table 13). Although MONet and EnsAvg offer enhanced detection accuracy—improving F1 scores by 0.02 to 0.08—they are over 90 times slower than ProtoNet. This indicates a clear trade-off: while MONet is superior for high-accuracy requirements, ProtoNet represents a more feasible solution for IoT devices with limited computational overhead.

Table 13: Comparison of inference speed (ms per episode) across 7-way K-shot 15-query settings (consistent across Distributions with negligible variation).

Model	3-shot	5-shot	7-shot	10-shot	15-shot
ProtoNet	0.51	0.55	0.54	0.54	0.54
RelNet	0.60	0.67	0.65	0.65	0.65
MONet	23.29	29.48	33.19	40.80	52.83
EnsAvg	24.40	30.70	34.38	41.99	54.02
EnsMaj	24.40	30.70	34.38	41.99	54.02

Regarding real-time applicability, the UQ-IoT-IDS dataset features benign traffic at 28 packets/s and peak attack traffic (UDP Flooding) at 8110 packets/s. Given that MONet and the ensemble models currently process fewer than 288 packets/s, they are expected to be capable of monitoring benign traffic and low-intensity attacks. However, further optimization or the adoption of hardware acceleration may be necessary to maintain real-time performance during high-volume flooding scenarios.

6 Discussion

6.1 Comparing the Performance of Few-Shot Learning Approaches

Across both attack distributions and all K-shot settings, MetaOptNet consistently achieved the strongest overall performance, including the highest macro-averaged F1 score on unseen attacks. This observation suggests that optimization-based few-shot learning may be particularly well suited to packet-level intrusion detection, where class boundaries are often overlapping and intra-class variability is high. By fitting a discriminative classifier on the support set in each episode, MetaOptNet can explicitly adapt decision boundaries to the available examples, rather than relying solely on compact class representations or pairwise similarity estimates. This flexibility may enable more stable generalization when limited and heterogeneous support samples are provided, as is common in few-shot intrusion detection scenarios. A key strength of this boundary-based approach appears to be its superior generalization on unseen attacks, as evidenced by MetaOptNet's peak F1-scores. However, a significant practical weakness lies in the trade-off between this

accuracy and computational demand; our analysis indicates that the ResNet-based backbone is markedly slower than simpler metric-based models, which may limit its immediate utility on low-resource IoT devices.

6.2 Challenges of Few-Shot Learning for Packet-Level IDS

Although the evaluated models demonstrate encouraging generalization capabilities, packet-level few-shot IDS inherently poses several structural challenges. Although the Kitsune feature representation captures multi-scale temporal statistics, these features may not always be sufficient to fully separate benign and malicious behavior without richer contextual information, such as session-based behavior or payload-level characteristics. Moreover, certain attack classes (e.g., Service Detection) exhibit substantial intra-class variability, where packets associated with a single attack scenario include multiple protocols and message types. Under a few-shot regime, the limited number of support samples may therefore be insufficient to represent the full attack scenario, occasionally leading to less stable inference outcomes.

Another commonly observed pattern is the misclassification of benign traffic as attack traffic. In realistic IoT environments, benign packets are inherently diverse, reflecting a wide range of protocols, payload sizes, and communication behaviors. Whereas traditional supervised IDS models benefit from abundant benign data to capture this diversity, few-shot learners must approximate benign behavior from a small number of support examples. Consequently, benign patterns that deviate from the limited support set may be more likely to be flagged as attacks, highlighting an important area for further refinement in packet-level few-shot IDS.

The benign-attack ambiguity identified in Error Analysis represents a structural weakness of applying FSL to packet-level detection, as a small support set may not sufficiently capture the inherent diversity of normal IoT traffic.

6.3 Sensitivity to Attack Composition and Class Imbalance

Performance varied substantially between Distributions A–D, supporting the view that few-shot generalization is highly sensitive to which attack types are included during meta-training and wihch attack types are newly introduced during meta-testing. Although class imbalance may partially contribute to the observed variance, it does not fully explain the magnitude of the performance gap, indicating that other factors, such as the intrinsic separability and variability of the selected attack types, also play an important role. These observations highlight the importance to evaluate few-shot IDS under multiple, carefully designed training/testing compositions rather than relying on a single split.

A specific limitation observed in our study is the difficulty of "stage-wise" generalization; models appear to struggle when task-shifting from basic reconnaissance to more advanced functional variants within the same attack category. On the other hand, a notable strength identified in our cross-device analysis (Distribution C) suggests that these models can successfully abstract attack-specific features that remain robust across different type of devices.

6.4 Effectiveness of Decision-Level Ensembling in Few-Shot IDS

Both ensemble strategies (probability averaging and majority voting) generally improved over weaker base learners, but did not consistently outperform the strongest single model (i.e., MetaOptNet). This outcome suggests that, in this setting, the base learners share common weaknesses on specific attack types, limiting the potential benefits of decision-level fusion. Especially, for majority voting, improvement requires that a majority of base learners perform adequately on each class; when multiple learners fail on the same class, voting can propagate the shared error. Nevertheless, the effectiveness of probability averaging a strategic

strength of ensembling: it can mitigate the impact of individual base-model weaknesses by successfully diluting uncalibrated or noisy predictions.

6.5 Future Directions

The findings of this study point to several promising directions for improving few-shot intrusion detection, particularly in addressing the evaluation trade-offs and error modes identified.

Lightweight Optimization-Based Models. While MetaOptNet achieved the highest overall detection performance, its ResNet-based backbone and SVM classification head result in substantial model complexity and inference latency. Future work should explore lightweight optimization-based meta-learners or knowledge distillation techniques (e.g., from MetaOptNet to Prototypical Networks) to enable real-time, accurate inference on resource-constrained IoT devices.

Richer Context for Subtle Attacks. The error analysis revealed significant inter-class confusion among probing and scanning attacks (e.g., Service Detection and Port Scanning), which often mimic benign communication patterns. Incorporating richer contextual information, such as flow-level aggregations or payload characteristics, could help disentangle the temporal-statistical similarities between these attacks and background traffic.

Asymmetric Support Allocation. To mitigate the prominent Benign-Attack Ambiguity observed across the distributions, future episodic setups could adopt an asymmetric sampling strategy. By allocating a substantially larger support set to the benign class, models could establish a more comprehensive definition of normal behavior, thereby reducing the rate of false positives caused by the inherent diversity of benign IoT traffic.

Device-Agnostic Feature Optimization. The cross-device correlation identified in Distribution C demonstrates the potential for learning generalized, structure-based attack patterns across heterogeneous hardware. Future research should explicitly focus on cross-device testing and emphasize domain-generalization techniques to further isolate device-agnostic representations from device-specific networking artifacts.

7 Conclusion

We have presented a comprehensive evaluation of few-shot learning for IoT intrusion detection by converting temporal-statistical traffic representations into image-like inputs and applying meta-learning under a unified episodic protocol. We have evaluated three representative few-shot learners (Prototypical Networks, Relation Networks, and MetaOptNet) and two decision-level ensemble strategies (majority voting and probability averaging) under four attack distributions and five K-shot settings. Across the evaluated settings, MetaOptNet achieved the strongest and most consistent macro-averaged F1 on unseen attacks. Our evaluation of ensemble strategies demonstrated that probability averaging effectively mitigates the impact of lower-performing models, frequently matching or marginally outperforming MetaOptNet. Conversely, majority voting slightly degraded overall performance due to its sensitivity to individual model misclassifications. Our extended error and complexity analysis also revealed that while MetaOptNet excels in accuracy, it entails substantial inference latency compared to distance-based counterparts, and that all models grapple with benign-attack ambiguity and cross-device correlations. Overall, these results demonstrate the practical viability of packet-level few-shot learning for IoT intrusion detection and underscore the critical trade-offs between detection efficacy, model complexity, and architectural vulnerability in few-shot IDS settings.

Acknowledgement: This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government (MOTIE) (A Study on Development of Cyber-Physical Attack

Response System and Security Management System for Maximizing Availability of Real-Time Distributed Resources, RS-2023-00303559).

Funding Statement: This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government (MOTIE) (A Study on Development of Cyber-Physical Attack Response System and Security Management System for Maximizing Availability of Real-Time Distributed Resources, RS-2023-00303559).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Liam Revell, Jung Taek Seo and Dan Dongseong Kim; methodology, Liam Revell; software, Liam Revell; validation, Liam Revell and Hyunjae Kang; formal analysis, Liam Revell; investigation, Liam Revell and Hyunjae Kang; data curation, Liam Revell; writing—original draft preparation, Liam Revell; writing—review and editing, Hyunjae Kang and Dan Dongseong Kim; visualization, Hyunjae Kang; supervision, Jung Taek Seo and Dan Dongseong Kim; project administration, Dan Dongseong Kim. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The source code for this study is available at https://github.com/trifle19/FSL-IoT-NIDS. All experiments were conducted using the publicly available UQ-IoT-IDS-2021 dataset at https://doi.org/10.48610/17b44bb.

Ethics Approval: Not applicable.

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

References

1. Shone N, Ngoc TN, Phai VD, Shi Q. A deep learning approach to network intrusion detection. IEEE Trans Emerg Topics Compu Intell. 2018;2(1):41–50. doi:10.1109/TETCI.2017.2772792.
2. Diro AA, Chilamkurti N. Distributed attack detection scheme using deep learning approach for Internet of Things. Future Generat Comput Syst. 2018;82(6):761–8. doi:10.1016/j.future.2017.08.043.
3. Jia W, Wang Y, Lai Y, He H, Yin R. FITIC: a few-shot learning based IoT traffic classification method. In: 2022 International Conference on Computer Communications and Networks (ICCCN). Piscataway, NJ, USA: IEEE; 2022. p. 1–10. doi:10.1109/ICCCN54977.2022.9868887.
4. Khraisat A, Alazab A. A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. Cybersecurity. 2021;4(1):18. doi:10.1186/s42400-021-00077-7.
5. Torres JLG, Catania CA, Veas E. Active learning approach to label network traffic datasets. J Inf Secur Appl. 2019;49(11):102388. doi:10.1016/j.jisa.2019.102388.
6. Wang Y, Yao Q, Kwok JT, Ni LM. Generalizing from a few examples: a survey on few-shot learning. ACM Comput Surv. 2020;53(3):63. doi:10.1145/3386252.
7. Duan R, Li D, Tong Q, Yang T, Liu X, Liu X. A survey of few-shot learning: an effective method for intrusion detection. Secur Commun Netw. 2021;2021(1):4259629. doi:10.1155/2021/4259629.
8. Snell J, Swersky K, Zemel R. Prototypical networks for few-shot learning. In: Advances in neural information processing systems, Vol. 30, Red Hook, NY, USA: Curran Associates, Inc.; 2017 [cited 2026 Mar 20]. Available from: https://proceedings.neurips.cc/paper_files/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html.
9. Sung F, Yang Y, Zhang L, Xiang T, Torr PHS, Hospedales TM. Learning to compare: relation network for few-shot learning. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ, USA: IEEE; 2018. p. 1199–208. doi:10.1109/CVPR.2018.00131.
10. Lee K, Maji S, Ravichandran A, Soatto S. Meta-learning with differentiable convex optimization. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ, USA: IEEE; 2019. p. 10649–57. doi:10.1109/CVPR.2019.01091.
11. He K, Kim D, Zhang Z, Ge M, Lam U, Yu J. UQ IoT IDS dataset 2021; 2022. doi:10.48610/17b44bb.

12. Chaabouni N, Mosbah M, Zemmari A, Sauvignac C, Faruki P. Network intrusion detection for IoT security based on learning techniques. IEEE Commun Surv Tutor. 2019;21(3):2671–701. doi:10.1109/COMST.2019.2896380.
13. Arthur MP, Ramachandran G, Sood K, Kaarthik P, Sridhar S, Chowdhury M. Empirical study of hierarchical intrusion detection systems for unknown attacks. IEEE Trans Netw Serv Manag. 2025;22(6):5564–81. doi:10.1109/TNSM.2025.3600378.
14. Roy S, Li J, Choi BJ, Bai Y. A lightweight supervised intrusion detection mechanism for IoT networks. Future Generat Comput Syst. 2022;127:276–85. doi:10.1016/j.future.2021.09.027.
15. Saied M, Guirguis S, Madbouly M. Review of artificial intelligence for enhancing intrusion detection in the internet of things. Eng Appl Artif Intell. 2024;127(3):107231. doi:10.1016/j.engappai.2023.107231.
16. Fusco P, Montefusco A, Rimoli GP, Palmieri F, Ficco M. TinyML-based intrusion detection system for handling class imbalance in IoT-edge domain using Siamese neural network on MCU. In: Barolli L, editor. Advanced information networking and applications. Cham, Switzerland: Springer Nature; 2025. p. 389–402. doi: 10.1007/978-3-031-87769-8_34.
17. Huang S, Liu Y, Fung C, An W, He R, Zhao Y, et al. A gated few-shot learning model for anomaly detection. In: 2020 International Conference on Information Networking (ICOIN). Piscataway, NJ, USA: IEEE; 2020. p. 505–9. doi:10.1109/ICOIN48656.2020.9016599.
18. Xu C, Shen J, Du X. A method of few-shot network intrusion detection based on meta-learning framework. IEEE Trans Inform Forens Secur. 2020;15:3540–52. doi:10.1109/TIFS.2020.2991876.
19. 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. doi:10.1016/j.cose.2022.102899.
20. Bovenzi G, Di Monda D, Montieri A, Persico V, Pescapè A. Classifying attack traffic in IoT environments via few-shot learning. J Inf Secur Appl. 2024;83(3):103762. doi:10.1016/j.jisa.2024.103762.
21. Vinyals O, Blundell C, Lillicrap T, Kavukcuoglu K, Wierstra D. Matching networks for one shot learning. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16. Red Hook, NY, USA: Curran Associates Inc.; 2016. p. 3637–45. doi:10.5555/3157382.3157504.
22. Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning, ICML'17. London, UK: PMLR; 2017. p. 1126–35.
23. Cerasuolo F, Bovenzi G, Ciuonzo D, Pescapè A. Attack-adaptive network intrusion detection systems for IoT networks through class incremental learning. Comput Netw. 2025;263(7):111228. doi:10.1016/j.comnet.2025.111228.
24. Du L, Gu Z, Wang Y, Wang L, Jia Y. A few-shot class-incremental learning method for network intrusion detection. IEEE Trans Netw Serv Manag. 2024;21(2):2389–401. doi:10.1109/tnsm.2023.3332284.
25. Chen WY, Liu YC, Kira Z, Wang YCF, Huang JB. A closer look at few-shot classification. arXiv:1904.04232. 2019.
26. Polikar R. Ensemble learning. In: Zhang C, Ma Y, editors. Ensemble machine learning. New York, NY, USA: Springer; 2012. p. 1–34. doi:10.1007/978-1-4419-9326-7_1.
27. Alhowaide A, Alsmadi I, Tang J. Ensemble detection model for IoT IDS. Internet Things. 2021;16(1):100435. doi:10.1016/j.iot.2021.100435.
28. Cao Y, Wang Z, Ding H, Zhang J, Li B. An intrusion detection system based on stacked ensemble learning for IoT network. Comput Electr Eng. 2023;110(5):108836. doi:10.1016/j.compeleceng.2023.108836.
29. Mirsky Y, Doitshman T, Elovici Y, Shabtai A. Kitsune: an ensemble of autoencoders for online network intrusion detection. In: Network and Distributed Systems Security (NDSS) Symposium 2018; 2018 Feb 18–21; San Diego, CA, USA. p. 1–15. doi:10.14722/ndss.2018.23204.
30. Mohammed A, Kora R. A comprehensive review on ensemble deep learning: opportunities and challenges. J King Saud Univ-Comput Inform Sci. 2023;35(2):757–74. doi:10.1016/j.jksuci.2023.01.014.
31. Brown G. Ensemble learning. In: Sammut C, Webb GI, editors. Encyclopedia of machine learning and data mining. Boston, MA, USA: Springer; 2017. p. 393–402. doi:10.1007/978-1-4899-7687-1_252.
32. Ochal M, Patacchiola M, Vazquez J, Storkey A, Wang S. Few-shot learning with class imbalance. IEEE Trans Artif Intell. 2023;4(5):1348–58. doi:10.1109/TAI.2023.3298303.
33. Kingma DP, Ba JL. Adam: a method for stochastic optimization. arXiv:1412.6980. 2015.