Classifying Attack Traffic in IoT Environments via Few-Shot Learning

Giampaolo Bovenzi[a], Davide Di Monda[a,b], Antonio Montieri[a], Valerio Persico[a], Antonio Pescapé[a]

[a] *University of Napoli Federico II, Naples, Italy*
[b] *IMT School for Advanced Studies, Lucca, Italy*

Abstract

The Internet of Things (IoT) is a key enabler for critical systems, but IoT devices are increasingly targeted by cyberattacks due to their diffusion and hardware and software limitations. This calls for designing and evaluating new effective approaches for protecting IoT systems at the network level. While recent proposals based on machine- and deep-learning provide effective solutions to the problem of *attack-traffic classification*, their adoption is severely challenged by the amount of labeled traffic they require to train the classification models. In fact, this results in the need for collecting and labeling large amounts of malicious traffic, which may be hindered by the nature of the malware possibly generating little and hard-to-capture network activity. To tackle this challenge, we adopt *few-shot learning* approaches for attack-traffic classification, with the objective to improve detection performance for attack classes with few labeled samples. We leverage advanced deep-learning architectures to perform feature extraction and provide an extensive empirical study—using recent and publicly available datasets—comparing the performance of an ample variety of solutions based on different learning paradigms, and exploring a number of design choices in depth (impact of embedding function, number of classes of attacks, or number of attack samples). In comparison to non-few-shot baselines, we achieve a relative improvement in the F1-score ranging from 8% to 27%.

Keywords:
Attack-Traffic Classification, Internet of Things, Deep Learning, Few-Shot Learning, Network Security

1. Introduction

In recent years, the Internet of Things (IoT) is gaining momentum, enabling the connection and communication of a wide variety of devices and smart objects (e.g., cyber-physical systems or smart home devices), possibly leading to increased efficiency and cost savings in industrial sectors like manufacturing, transportation, healthcare, and smart cities. The number of IoT devices connected worldwide is estimated to reach 55.7B by 2025,[1] pushing for IoT security like never before. Indeed, the IoT environment is characterized by serious security concerns, as IoT devices are known to have low-cost production processes that increase the variety and the amount of related vulnerabilities [1]. Besides, severe cyberattacks are globally increased by 53% from 2018 to 2022—also driven by the Russo-Ukrainian (cyber-)war—with the *malware* category accounting for the 38% of the share.[2] Detailing, 1.5B attacks targeted IoT devices in the first half of 2021—doubling the number from the first half of 2020.[3] The *ENISA Threat Landscape* report[4] highlights that IoT devices are an essential initial attack surface for malicious actors. It further underscores this concern, revealing that *Mirai* botnets—notorious for hijacking IoT devices to launch large-scale attacks—were responsible for over 7M attacks in the early months of 2022 alone. Moreover, the report highlights a shift towards IoT devices for Distributed Denial-of-Service (DDoS) attacks due to the limited resources and weak security. To compound the problem, attackers increasingly leverage *zero-day* vulnerabilities to create new botnets and unleash even more sophisticated attacks[5]. As a consequence, the impact of the cost of cybercrime for businesses is estimated to reach $10.5T by 2025.[6]

Accordingly, monitoring the heterogeneous and highly dynamic traffic flowing across networks to understand its nature is a critical activity. Indeed, network-traffic classification (i.e. associating a traffic object to the application or the activity that generated it) underpins several use cases, such as billing, accounting, network provisioning, resource management, and also *network security* (e.g., detection and classification of anomalies, attacks, or malware).

Traffic classification approaches evolved over time according to the nature of networks and their traffic. Early solutions based on *port numbers* and *deep packet inspection* were severely challenged: the first by the usage of non-standard ports (not even

Email addresses: giampaolo.bovenzi@unina.it (Giampaolo Bovenzi), davide.dimonda@{unina.it, imtlucca.it} (Davide Di Monda), antonio.montieri@unina.it (Antonio Montieri), valerio.persico@unina.it (Valerio Persico), pescape@unina.it (Antonio Pescapé)

[1] https://bit.ly/idc-future-of-industry-ecosystems
[2] https://bit.ly/clusit-report-2022
[3] https://bit.ly/kaspersky-iot-attacks-doubling
[4] https://bit.ly/enisa-threat-landscape
[5] https://bit.ly/mirai-variant-iot-devices
[6] https://bit.ly/cybersecurity-ventures-cybercrime-cost-by-2025

present when IP datagrams do not encapsulate TCP/UDP segments, as may happen for attack traffic), the latter by traffic-encryption schemes, limiting the inspection to a reduced set of cleartext header fields, such as the Server Name Indication for TLS. Also, the rise of *Machine Learning* (ML) approaches was limited by the required domain-expert-driven feature-extraction process, leaving room for end-to-end learning capabilities (with automatic feature extraction from raw data) offered by *Deep Learning* (DL), at the cost of a large amount of labeled data needed to train supervised models.

Unfortunately, building large labeled network traffic datasets is a difficult and time-consuming task. It involves capturing network data, breaking it down into traffic objects, and adding labels. This often requires specialized setups [2] and raises concerns about user privacy and sensitive business information [3]. Additionally, because network traffic changes rapidly over time, this process must run continuously to maintain large, comprehensive, and up-to-date datasets.

These issues are exacerbated when dealing with *attack traffic*, where reproducing attack scenarios for attacks rarely observed (thus still having partially unknown dynamics, as in the case of sophisticated unknown/zero-day or morphed attacks), is even more problematic. Often, these situations require manual intervention and labeling by domain experts [4].

As a result, it is common that labeled data are scarce for some attack-traffic classes, an imbalanced situation that is well reflected by the dataset used in this work (i.e. `IoT-23` [4]), which encompasses attacks with less than 100 samples (cf. Sec. 4.1). This lack of sufficient data hinders the usage of ML and DL approaches that fall short with only limited knowledge available.

Few-Shot Learning (FSL) aims to tackle this challenge—i.e. learning to classify the traffic of attacks rarely observed without the need to collect a large amount of expensive labeled data—by leveraging non-few knowledge to build a model whose purpose is to generalize enough to new tasks with only few samples available. In other words, *FSL leverages an already trained DL model as a feature extractor and builds a classifier on top of it using few labeled samples for each new class*. The problem of learning with few samples has attracted the interest of research communities in a number of domains (e.g., computer vision), where FSL was initially investigated and explored in-depth. Accordingly, the taxonomy of the available solutions is particularly rich and consists of different strategies to capitalize on prior knowledge.

Given the promising results achieved in various domains, recent research endeavors have aimed to apply FSL techniques to network-traffic classification. In fact, most of the existing works leveraging FSL in this context are based on previously established computer vision methods, often with only minor modifications [5–9]. Notably, despite the extensive body of literature on FSL, the application of these techniques to network-traffic classification remains somewhat limited in its exploration of the full spectrum of FSL possibilities. Furthermore, the experimental assessments in these studies are often constrained, as they primarily compare proposed solutions against non-FSL counterparts [5, 6, 8, 10–14], thus emphasizing the general advantages of FSL without delving into the specific benefits offered by distinct FSL approaches. Accordingly, in this paper, we contribute as listed in the following.

- With the unprecedented heterogeneity and dynamicity of IoT networks that increase the variety and number of related vulnerabilities, we focus on the problem of classifying attack traffic in IoT environments. We consider the case of supervised learning with *limited supervised knowledge* for certain attacks, namely with only a few labeled samples available.
- We adopt advanced DL architectures to perform feature extraction (*embedding functions*) fed with network input proven informative for attack-traffic classification. Remarkably, such network input enables *early* attack-traffic classification, i.e. that provides verdicts based only on the initial part of the traffic objects, thus enhancing the responsiveness of the classifier.
- We leverage `IoT-23`, a public dataset containing the traffic of (rarely observed) attacks against IoT devices, whose adoption also fosters the reproducibility of our experimental analyses. Moreover, we consider three further datasets including both benign and malicious IoT traffic (i.e. `IoT-NID`, `Bot-IoT`, and `Edge-IIoTset`) to evaluate the generalization capability of considered approaches in other IoT environments.
- We provide extensive empirical results of FSL approaches for attack-traffic classification. We conduct an experimental assessment of their classification performance and compare them with non-few-shot baselines, obtaining improved performance figures.
- We evaluate a number of design choices with the aim of optimizing attack-traffic classification performance. We consider the impact of varying the *embedding function*, *number of ways* (i.e. number of classes considered), *number of shots* (i.e. number of samples available for each class), and *non-few prior knowledge*.

The rest of the paper is organized as follows. Section 2 provides the background on FSL and discusses the related work employing FSL for attack-traffic classification. Section 3 introduces the methodological framework for adapting and applying FSL to attack-traffic classification. Section 4 presents the experimental setup we consider, describing the dataset, the FSL setup, and the metrics we use for the evaluation. Section 5 describes the results of our experimental evaluation, with their main take-home messages. Finally, Section 6 concludes the paper, drawing the main remarks.

2. Few-Shot Learning Background and Related Work

This section offers a description and a taxonomy of the main concepts that will be discussed in the present paper (Sec. 2.1). It also summarizes the most related studies facing attack-traffic classification in a few-shot scenario (Sec. 2.2) and frames this work in relation to them (Sec. 2.3).

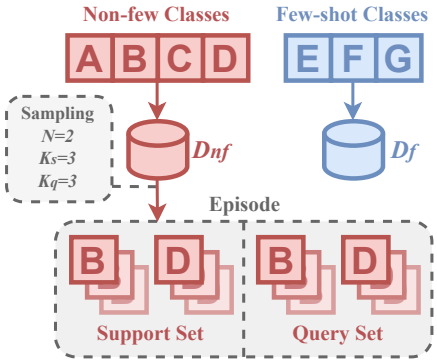


Figure 1: Example of creation of 2-way 3-shot episodes during meta-training. Each episode contains 12 samples: 6 for the support set and 6 for the query set.

2.1. Background on Few-Shot Learning

Few-Shot Learning (FSL) allows classification where only few (viz. very limited) labeled samples are available [15]. Expressly, the goal of FSL is to derive a model that is general enough to classify observations into classes characterized by the availability of (very) few samples, i.e. to fulfill a few-shot task, by capitalizing on the knowledge from related (non-few) tasks. This is different from traditional learning methods, which need a large number of samples for these classes to acquire knowledge during training and achieve satisfactory performance in the operational phase. Overall, FSL helps in dealing with the problem of strong label-distribution skewness in data (cf. Sec. 4.1) and can be extremely useful for several tasks related to traffic analysis, which suffer from this issue (e.g., encrypted and mobile-app traffic classification and malware identification). For instance, in the case of attack-traffic classification, FSL allows for the design of solutions capable of detecting zero-day attacks, namely a scenario where attack samples are very scarce and not observed during training. Indeed, while traditional ML and DL approaches fall short [16] and require retraining from scratch once enough data is available, FSL can learn from a limited number of labeled samples per class by extrapolating *prior knowledge* from already available and abundant data.

The way prior knowledge is utilized in FSL determines *three families of approaches* [17]: (i) *Algorithm-based approaches*: they employ prior knowledge to modify and optimize the strategy for searching new parameters; this serves as a valuable guideline tailored to the specific few-shot task at hand [18]. (ii) *Model-based approaches*: their focus is on jointly learning a set of related initial tasks; the goal is reducing the search space for optimal parameters by leveraging prior knowledge [19–21]. (iii) *Data-based approaches*: they exploit prior knowledge to augment the data associated with the few-shot tasks.

Moreover, FSL can be fulfilled by leveraging different *learning paradigms*. We provide the fundamentals of two popular FSL paradigms adopted herein: *meta learning* and *transfer learning*.

Meta-Learning Fundamentals. This paradigm enhances the performance on a novel task by distilling meta-knowledge obtained from various tasks through a meta-learner, whose goal is "learning to learn". In the context of FSL, meta-learning is used together with *episodic learning*, which involves organizing the training into a sequence of learning tasks named *episodes*. In each episode, a limited number of samples and classes is considered. This setup facilitates the model (*i*) in rapidly acquiring new knowledge by drawing insights from learning tasks faced and (*ii*) in effectively classifying unknown classes with only few samples.

Consequently, meta-learning approaches should preliminarily manage the creation of *N-way K-shot episodes*. More specifically, given a dataset $D = \{D_{nf}, D_f\}$, we assume that D_{nf} and D_f have a disjoint label space. D_{nf} includes the samples of "non-few" classes (i.e. the most populated ones). D_f encompasses the samples of "few-shot" classes (i.e. the least populated ones). The training phase and the operational phase exploit D_{nf} and D_f, respectively.[7] Going into detail, each episode is constructed by randomly sampling N classes (*N-way*) and then obtaining two partitions (non-overlapping): (*i*) the *support set* consisting of $N \times K_s$ samples (with K_s defining the *K-shot* setup) and (*ii*) the *query set* consisting of $N \times K_q$ samples. An example with $N = 2$ and $K = K_s = K_q = 3$ is reported in Fig. 1.

Starting from the definition of such episodes, (*i*) the model learns from a set of T classification tasks—i.e. *N-way K-shot tasks*—during *meta-training*: it uses the samples of the support set for training and measures the error on the query set; (*ii*) then, the model is operated on unknown few-shot classes to test its generalization ability: *meta-testing* emulates this operational phase by leveraging an akin episode-based procedure.

Transfer-Learning Fundamentals. Transfer learning seeks to learn generic features from a large dataset of non-few classes (T_{nf} task, also referred to as *pre-training*), and then specializes in a new task with only few labeled samples during the operational phase (T_f task, also referred to as *adaptation*). Consequently, in a similar way to the meta-learning, $D = \{D_{nf}, D_f\}$, with D_f encompassing the samples of the few-shot classes. Specifically, the T_{nf} task learns knowledge on the (pre-)training data extracted from D_{nf}. With an analogous learning procedure, the T_f task enriches (viz. *transfers*) the knowledge gained during T_{nf} with samples belonging to D_f (i.e. those from the few-shot classes not observed before).

2.2. Related Work

The challenge of learning with only few samples available (thus minimizing the need for extensive data collection) has garnered significant attention from research communities across various domains. These include networking, where there is a growing interest in traffic analysis and attack-traffic classification. Indeed, recently FSL has gained considerable attention

[7] Commonly, the operational phase, dealing with samples belonging to few-shot classes not observed during the training, is also referred to as the testing phase since, during the latter, the trained model is "tested" in a few-shot scenario.

Table 1: Related work dealing with attack-traffic classification via Few-Shot Learning approaches. The papers are ordered by year. The last row summarizes the present work. Acronyms' meaning is reported at the bottom of the table.

Paper	Approach	EF	FB	Dataset	TO	Raw	Early	Input Data
Huang et al. [5], 2020	MN+gates	1D-CNN	○	NSL-KDD	B	○	○	Flow based-statistics
Zheng et al. [6], 2020	RN+DA	AE+2D-CNN	○	ISCX 2012 IDS, ISCX VPN-nonVPN	B	●	○	Raw flow sequence
Xu et al. [10], 2020	RN	DNN	○	ISCX 2012 FS, CIC-IDS 2017 FS	F	●	●	*M* packets per flow *N* bytes per packet
Zhou et al. [11], 2021	SN	SCNN	○	UNSW-NB15, *self-built*	B	○	○	Flow based-statistics
Feng et al. [12], 2021	MAML	AE+LSTM	○	CICAndMal 2017, CIC-IDS 2017	F	●	◐	Flow based-statistics Time-series features
Ouyang et al. [13], 2021	PN	2D-CNN	○	SCADA dataset	B	○	○	Flow based-statistics
Rong et al. [22], 2021	PN	2D-CNN	●	MCFP, USTC-TFC, CICInvesAndMal 2019, BEN-1, *self-built*	B	●	●	*N* bytes of a biflow
Wang et al. [7], 2021	RN+SN	SCN, 2D-CNN, ResNet18, VGG16	●	CIC-IDS 2017, UNSW-NB15	B	○	○	Flow based-statistics
Liang et al. [14], 2021	RN	LSTM+VBP	○	NSL-KDD, CIC-IDS 2017	B	○	○	Flow based-statistics
Guo et al. [23], 2022	MAML	2D-CNN	●	USTC-TFC2016	B	●	●	Raw flow sequence
Yang et al. [24], 2022	PN	AE+2D-CNN	○	CIC-IDS 2017	B	●	●	Flow based-statistics Raw flow sequence
Lu et al. [25], 2023	MAML	1D-CNN	●	*self-built* combining several public datasets	B	○	○	Flow based-statistics
Pawlicki et al. [26], 2023	SN	1D-CNN	●	CSE-CIC-IDS 2018	B	○	○	Flow based-statistics
This work	MN, PN, RN, MON, MAML, ANIL, FT, FZ	1D-CNN, 2D-CNN, 2D-CNN+LSTM, MIMETIC*	●	IoT-23, IoT-NID, Edge-IIoT, Bot-IoT	B	●	●	L4/L2 unbiased payload Per-packet header fields

Approach—Proposed FSL approach; **EF**—Embedding Function; **FB**—FSL Baseline; **TO**—Traffic Object: Biflow (B), Flow (F); **Raw**—Raw input data; **Early**—Early attack-traffic classification. **Acronyms**: AutoEncoder (AE), Almost No Inner Loop (ANIL), Convolutional Neural Network (CNN), Data Augmentation (DA), Deep Neural Network (DNN), Fine-Tuning (FT), Freezing (FZ), Long Short-Term Memory (LSTM), Model-Agnostic Meta-Learning (MAML), Matching Networks (MN), MetaOptNet (MON), Prototypical Networks (PN), Relation Network (RN), Siamese Network (SN), Siamese Capsule Network (SCN), Siamese Convolutional Neural Network (SCNN), Variational Bayesian Process (VBP).
"+" symbol indicates hybrid architectures; ● present, ◐ partial, ○ lacking.

as being nowadays an in-depth explored branch of scientific research, with its first applications in computer vision. Hence, most networking-related studies focusing on FSL employ solutions originally devised for computer vision, albeit with some minor adaptations being proposed.

Accordingly, in Tab. 1, we report and categorize the most relevant papers dealing with attack-traffic classification in a few-shot scenario. We underline the key aspects concerning the covered subject and any variations brought to the computer vision-tailored FSL approaches that served as inspiration. Interestingly, all the referenced works were published between 2020–2023, highlighting the recent interest of the networking community in FSL. Besides being the latest, we have selected these papers since they primarily deal with the task of classifying the traffic of *unseen attacks* with only few samples available. Indeed, in addition to the papers reported in Tab. 1, other studies [27–29] also leverage FSL methods for intrusion detection purposes; however, we do not consider them in our analysis since the proposed approaches are evaluated on a testing set that shares the same label space with the training set (i.e. seen attacks), namely, the paper cornerstone is simply mitigating the class imbalance in data.

The second column of Tab. 1 shows the specific **FSL approach** that each work utilizes for attack-traffic classification tasks. Firstly, we underline that the FSL approaches applied in the related studies fall into two families: *model-based* and *algorithm-based*. Additionally, the majority of them exploit the *meta-learning* paradigm for training the models. Moreover, we highlight that only five works [7, 22, 23, 25, 26] evaluate more than one FSL approach, as pointed out by the **FSL Baseline (FB)** column. This paramount aspect underlines the *lack of a comprehensive comparison* between the different FSL families and paradigms in related works. Hereinafter, we focus on *substantial novelties* that related studies bring to the computer vision techniques they utilize.

One of the first meta-learning algorithms tackling FSL is *Matching Networks*, firstly proposed in [19]. Matching Networks are based on an attention mechanism to extract useful prior knowledge and embed samples in a lower-dimensional space; then, they perform a generalized nearest-neighbor classification. Inspired by the latter, Huang et al. [5] add a gating technique to the base Matching Networks algorithm to deal with attack-traffic identification. Additional gates can be considered soft classifiers providing similarity scores to evaluate the importance of both known and unknown anomalies in classifying unlabeled samples. Matching Networks have also been used

in [22] as a baseline.

Prototypical Networks are the natural successors of Matching Networks and have been originally proposed and evaluated on computer vision data in [20]. Prototypical Networks classify new instances by computing their distances with class *prototypes* (i.e. a centroid representative of a certain class) and labeling them according to the closest prototype. In the networking field, Ouyang et al. [13] aim to recognize cyberattacks in network SCADA systems through Prototypical Networks. Similarly, Rong et al. [22] use the original Prototypical Networks to detect unseen malware; training is performed in an episodic fashion using traffic sessions converted to gray-scale images (i.e. traffic-to-image encoding). More recently, Yang et al. [24] propose an FSL framework still considering the traffic as gray-scale images; they apply Prototypical Networks on top of AutoEncoder-transformed features for malware traffic classification.

Relation Network is another meta-learning approach, commonly applied to address computer vision tasks, which has been initially described in [21]. Unlike Matching and Prototypical Networks, embedded training and test samples are concatenated and fed to a relation module consisting of a Convolutional Neural Network (CNN) that outputs a similarity score between 0 and 1. As shown in Tab. 1, Relation Network (and its variants) is the most common FSL approach applied for attack-traffic classification. Both Xu et al. [10] and Liang et al. [14] leverage the original Relation Network as-is for few-shot network intrusion detection. Differently, Zheng et al. [6] combine the Relation Network with a data-based method; especially, the authors augment the training set through the implementation of an "hallucinator", which creates new samples by adding noise to the data. Lastly, Wang et al. [7] also integrate a Siamese Network (as an embedding function) with the Relation Network to improve the detection of network attacks under imbalanced training data.

Model-Agnostic Meta-Learning (MAML) is a popular FSL approach [18] that continuously refines the model parameters via a meta-learning procedure enriched with a fine-tuning phase for rapid adaptation to new FSL tasks. Regarding our related literature, Feng et al. [12], Guo et al. [23], and Lu et al. [25] design an FSL model based on the original MAML to detect anomalous traffic with only few samples, while MAML is considered as a baseline in [22].

In addition to the specific FSL approach employed, the choice of the **embedding function** (i.e. the feature extractor) is highly important since the performance attained is closely related to an embedded space capable of emphasizing similarities/differences among samples in order to generalize. To this end, single-modal CNNs (both one- and two-dimensional) are the most popular choice; hybrid architectures (marked with + in the **EF** column) obtained via the composition of different layers are also used (e.g., AE+CNN [6, 24], AE+LSTM [12], and LSTM+VBP [14]). Notably, Zhou et al. [11] and Wang et al. [7] employ different embedding functions being variants of a Siamese Network (i.e. a CNN- and Capsule Neural Network-based architecture, respectively) to embed in tandem two samples and output comparable feature vectors. As for the FSL approaches, we underline that related literature *lacks an exhaustive benchmark of embedding functions*: only Wang et al. [7] evaluate multiple architectures.

Regarding the specific **datasets** leveraged, to the best of our knowledge, none makes use of the recent `IoT-23` (employed herein) to evaluate FSL for attack-traffic classification. On the other hand, CIC-IDS 2017 is the most used dataset [7, 10, 12, 14, 24]. Notably, four papers [5, 6, 10, 14] leverage ISCX 2012 or NSL-KDD, being outdated datasets collected one and more than two decades ago, respectively. Unfortunately, due to the rapid evolution of networks and related cyberattacks, these datasets hardly exhibit a current real-world traffic profile.

As for traffic segmentation, we notice that flows, and particularly bidirectional flows (*biflows*), are the most frequent options as **traffic objects**. Based on the latter, various **input data** are considered to feed FSL models. Firstly, in Tab. 1, we explicitly flag the works that effectively consider **raw input data** [6, 10, 12, 22–24]. Conversely, the remaining works counter-productively adopt ready-to-use features (commonly obtained from preprocessed datasets) such as flow-based statistics extracted from the sets of packet/payload lengths and inter-arrival times. Indeed, this shortcoming nullifies a key strength of DL architectures—widely employed in related works as embedding functions for feature extraction—namely the automatic extraction of knowledge from raw attack-traffic data. Finally, a notable advantage for network intrusion detection systems lies in intercepting attacks as they are still running. Hence, in Tab. 1, we also highlight the works attaining **early** attack-traffic classification [10, 22–24]. These works commonly capitalize on the first packets/bytes of each traffic object to promptly provide the classification verdicts, as opposed to the classification accomplished only once the malicious activity is completely over (i.e. "post-mortem" classification).

2.3. Positioning

Starting from the discussion of related work, we *position our work against the state-of-the-art*. The aspects detailed hereinafter are also summarized in the last row of Tab. 1.

- We consider a variety of FSL approaches based on the *meta-learning* paradigm, both employed in related studies (i.e. Matching Networks, Prototypical Networks, Relation Network, and MAML) and *novel to the attack-traffic classification field* (i.e. `MetaOptNet` [30] and `ANIL` [31]). In order to evaluate the effectiveness of other FSL paradigms for attack-traffic classification, we also enrich the roster of methods with approaches based on *transfer learning* (i.e. Fine-Tuning and Freezing). Furthermore, we take into account both *model-based* and *algorithm-based* families, thus *providing a comprehensive comparison of the effectiveness of various FSL families and paradigms* when dealing with attack-traffic classification tasks.
- We devise a *novel meta-training procedure* enabled for early-stopping—based on a meta-validation phase—to better select the best model and avoid overfitting, thus enhancing the approaches' ability to generalize.

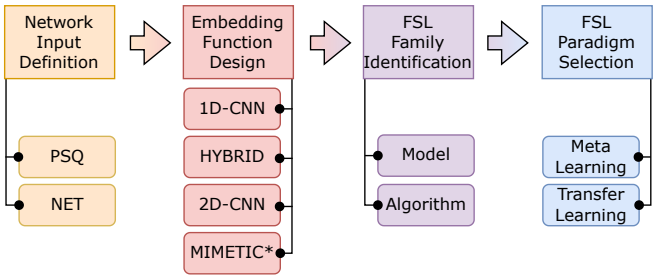


Figure 2: Methodological framework for few-shot IoT attack-traffic classification with considered options.

- We assess the performance of *different embedding functions* both single-modal and multimodal. Particularly, we have selected well-known and promising DL networks from the traffic classification domain.
- We leverage and *unbiased* and *effective* [16] input traffic data that enable fast (i.e. *"early"*) classification of attack traffic, ensuring that the performance is not inflated and the outcomes are not specific for just one traffic scenario, but rather generalizable to a variety of these.
- We validate the necessity of using (advanced) few-shot tailored solutions by comparing their classification performance with *non-few-shot baselines*.
- We employ four *publicly available and recent* (published between 2019 and 2022) *datasets*, namely `IoT-23`, `IoT-NID`, `Bot-IoT`, and `Edge-IIoTset`. They provide challenging IoT attack scenarios, encompassing both common and rarely observed attacks, for the proposed approaches.

3. Methodological Framework for IoT Attack-Traffic Classification via Few-Shot Learning

In this section, we introduce the methodological framework for the exploitation of FSL approaches for IoT attack-traffic classification. Specifically, when designing an FSL attack-traffic classifier, the following aspects must be covered: (*i*) the traffic object must be defined, together with the related pieces of information to be fed to the classification architecture as input (see Sec. 3.1); (*ii*) the process for extracting relevant features from the input data must be selected (see Sec. 3.2 that presents state-of-the-art embedding functions based on DL); (*iii*) the particular FSL approach must be chosen, requiring the selection of both the FSL family it belongs to and the paradigm to be leveraged (see Sec. 3.3 that deepens *model-based* and *algorithm-based* families, introducing the designed *meta-learning evaluation procedure*, and presents the specific *FSL approaches* employed in our work). Figure 2 summarizes these main aspects to be considered when exploiting FSL approaches for IoT attack-traffic classification. Note that these choices are partially intertwined, with the network input implying the feature-extraction process, and the FSL family possibly resorting to a specific FSL paradigm. Also, note that the downstream steps of the methodological framework—including the identification of the FSL family and the selection of the FSL paradigm—are more general and can be borrowed from domains beyond network traffic classification. On the other hand, the upstream steps—concerning the definition of the network input and the design of the embedding function—are domain-specific.

3.1. Traffic Object and Network Input

In our work, we leverage the *bidirectional flow* (*biflow*) as the traffic object. The choice of the traffic object defines how raw traffic is segmented into discrete traffic units, with biflow being the most common traffic object used when dealing with attack-traffic classification via DL (cf. Tab. 1). In detail, a biflow aggregates the network packets that share the same quintuple (i.e. source and destination IP addresses, source and destination ports, and transport-level protocol), considering the source and the destination as interchangeable, namely it models both the directions of communication.

Considering the biflow as the elementary sample of our classification task, the input data used to feed the FSL methods exploited herein are organized in two input sets: *packet-field sequences* (`PSQ`) and *network-packet bytes* (`NET`). We select such input data based on previous literature regarding both encrypted-traffic classification [32–34] and attack-traffic classification [10, 16, 22, 35, 36], and preliminary experimental evaluations conducted in both non-few and few-shot scenarios demonstrating their effectiveness. Both `NET` and `PSQ` are (min-max) normalized within the range $[0, 1]$.

`PSQ` **Input Type.** For each biflow, M informative unbiased fields are extracted from the first N_p packets of the sequence, resulting in an $N_p \times M$ matrix. In particular, the selected $M = 4$ fields are: (*i*) the number of bytes in the transport layer payload; (*ii*) the packet direction (-1 or 1); (*iii*) the TCP window size (0 for non-TCP packets); (*iv*) the time elapsed since the arrival of the previous packet (i.e. the inter-arrival time[8]).

`NET` **Input Type**. For each biflow, we extract the first N_b bytes (each arranged as an integer ranging from 0 to 255) of the network-layer header and payload data (i.e. starting from the IP header) of the sequence of packets in the biflow. Since the transport layer and network layer contain biased information (i.e. IP addresses, transport ports, and IP & transport checksums) [16, 32], we obfuscate (viz. we zero) the portion of bytes related to these fields to avoid inflating classification performance (more details on this procedure are given regarding the occlusion analysis performed in [16]).

We underline that we limit to the first N_p (resp. N_b) packets (resp. bytes) so that the `PSQ` (resp. `NET`) input type is suited for *early* attack-traffic classification [32], i.e. it can support traffic classification based on the observation of a limited portion of

[8] We underline that instead of considering the absolute time/timestamp of packets in each biflow, we take into account relative temporal information via the inter-arrival times. Indeed, the latter field carries unbiased information that does not depend on the specific collection environment/setup (e.g., an attack whose traffic has been collected only on certain days or periods) and does not risk wrongly inflating classification performance.

the traffic object to classify. This technique improves the responsiveness of the classifier, which is of particular practical interest in the case of IoT attack-traffic classification.

Moreover, it is worth mentioning that both N_p and N_b should be selected by striking a balance between classification performance and the efficiency of the traffic capture process. Opting for lower values of N_p or N_b reduces the computational load required for collecting network input data, but excessively low values can have a detrimental impact on classification results [16, 32].

3.2. Embedding Functions

FSL approaches extract relevant features via an *embedding function* by reducing the input data into a space of lower dimensions. Specifically, we exploit different state-of-the-art DL architectures that were proposed and used to perform (attack-)traffic classification [16, 32, 37]. A graphical representation of embedding functions considered herein is provided in Fig. C.16 of Appendix C.

1D-CNN. The first DL network is a single-modal 1D-CNN originally proposed in [34]. We leverage an architecture with the same layers and related hyperparameters as in [34]. Such a network consists of two 1D convolutional layers—each followed by a 1D max-pooling layer—and ends with two dense layers. 1D-CNN is fed with the `NET` input.

Hybrid. This DL network is a more complex single-modal Hybrid architecture, which combines 2D-CNN and Long Short-Term Memory (LSTM) layers. Such layers are concatenated by reshaping the output tensor of the 2D-CNN into a matrix that is provided as input to the LSTM. The specific architecture and hyperparameters are those proposed in [33]. According to the 2D nature of the convolutional layer, we feed Hybrid with the `PSQ` input.

2D-CNN. We also employ a version of Hybrid without the LSTM layer, namely a single-modal 2D-CNN also being proposed in [33]. We exploit 2D-CNN to evaluate the performance attained with an architecture simpler than Hybrid.

Mimetic*. The last leveraged architecture is a *multimodal* traffic classifier named Mimetic, originally proposed in [37]. Mimetic is made of two branches, with each branch corresponding to a different modality: one branch is fed with the `NET` input, the other with the `PSQ` one. The `NET` branch consists of two 1D convolutional layers—each followed by a 1D max-pooling layer—and a final dense layer. The `PSQ` branch encompasses a Bidirectional Gated Recurrent Unit and one dense layer. The features extracted by each branch are then concatenated and further elaborated via a shared dense layer. We underline that, differently from the original Mimetic classifier [37], herein we do not train Mimetic via a two-phase procedure but in a monolithic fashion, so as to assess a variant having a training procedure analogous to the other embedding functions. Hence, we refer to it as Mimetic*.

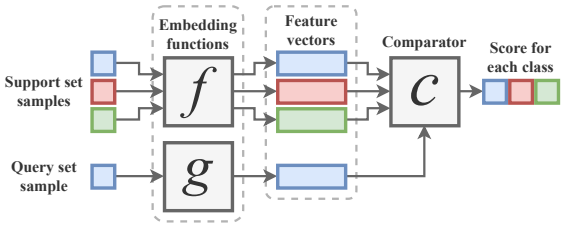


Figure 3: Model-based meta-learning general architecture. Both support and query samples are first compressed leveraging a dedicated embedding function (f and g, respectively). Then, a comparator component c classifies the embeddings of the query set against those of the support set.

3.3. Few-Shot Learning Approaches

This section introduces the main FSL approaches belonging to the two FSL families we deal with, namely model- and algorithm-based. We highlight that all the discussed approaches follow the meta-learning paradigm, except for transfer-learning ones. The section ends with the meta-learning procedure we have designed for attack-traffic classification.

Model-based Approaches. Model-based approaches have the objective of reducing the risk of overfitting by constraining the hypothesis space[9] to a smaller one exploiting the prior knowledge. Figure 3 depicts a general architecture for model-based meta-learning solutions. Such solutions apply *embedding functions* (i.e. f and g in Fig. 3) to the samples belonging to a d-dimensional space $\mathbb{R}^d$ to map them in an m-dimensional space $\mathbb{R}^m$, where $m < d$. This process ensures that similar samples are closer, whereas dissimilar samples become more easily distinguishable, thus effectively reducing the hypothesis space. The embedding functions are trained based on prior knowledge extracted from the meta-training tasks defined on D_{nf} (i.e. non-few classes).

Classification is performed via a *comparator* by measuring the similarity of $g(x_q)$ with $f(x_s)$, where $g(\cdot)$ and $f(\cdot)$ are the embedding functions applied to query and support instances, respectively, being $x_q, x_s \in D_*$. In the present work, we set $g(\cdot) = f(\cdot)$, and we use as embedding functions state-of-the-art DL networks widely employed and suited for attack-traffic classification (cf. Sec. 3.2).

The main difference among model-based approaches is the comparator (the c block in Fig. 3), namely the similarity measure/mechanism employed. In the present work, we use the following model-based approaches: (*i*) *Matching Networks* (`MatchingNet`) [19] are based on a one-nearest-neighbor classifier that leverages the Euclidean distance; (*ii*) *Prototypical Networks* (`ProtoNet`) [20] perform the classification by computing the Euclidean distance between the embedding of a sample and each per-class centroid—named *prototype*; (*iii*) *Rela-*

[9] Hypothesis space encompasses all potential solutions available for a given learning problem. Within this space, the learning algorithm seeks to discover the optimal model, essentially identifying the most suitable parameters/weights that effectively fit the data and exhibit strong generalization to new, unseen samples.

tion Network (`RelationNet`) [21] trains a *relation module*—based on a CNN—that computes a similarity score between the embeddings of query and support samples of every class; (*iv*) `MetaOptNet` [30] uses a *linear Support Vector Machine* trained on the samples of the support set, as a comparator, and classifies the samples of the query set by capitalizing on the resultant linear space separation.

Algorithm-based Approaches. The models of the algorithm-based family explore the hypothesis space to find the set of parameters that corresponds to the best hypothesis in this space. However, the scarce number of samples available for training in a few-shot scenario hinders the proper updating of the parameter set, leading to suboptimal performance. To address this issue, algorithm-based methods leverage prior knowledge to guide the search for the best parameter set.

In this work, we consider `MAML` [18] and `ANIL` [31], whose objective is to *refine meta-learned parameters* by learning an initial set of parameters through meta-learning and subsequently enhancing it via data from D_f (i.e. few-shot classes). They also use an embedding function as a feature extractor. `MAML` (*Model-Agnostic Meta-Learning*) iteratively adjusts the initial meta-learned parameter set on the basis of the performance achieved on episodic tasks (i.e. the so-called *inner-loop adaptation*). The aim is to find a set of parameters that is highly adaptable to different tasks. `ANIL` (*Almost No Inner Loop*) is a simpler variant of `MAML` that remains effective while reducing computational demands. `ANIL` avoids inner-loop updates for the embedding function during both meta-training and meta-testing and adopts these updates exclusively to the model head.[10]

Moreover, we utilize two *transfer-learning* approaches, outlined as follows. *Fine-Tuning* ($\texttt{TL}_{\texttt{FT}}$) entails two main steps: (*i*) it initially learns the model parameters by solving the task T_{nf} (i.e. training on D_{nf}) and then (*ii*) refines such parameters by learning the task T_f (i.e. leveraging only the samples in D_f). Despite this leads to significantly faster execution, $\texttt{TL}_{\texttt{FT}}$ is prone to the problem of *forgetting*, i.e. it may lose knowledge about old non-few classes. To deal with this issue, *Freezing* ($\texttt{TL}_{\texttt{FZ}}$) fixes the weights of the embedding function when a new task is introduced, allowing only the update of the weights associated with the model head (in contrast to $\texttt{TL}_{\texttt{FT}}$, where both the embedding function and the model head undergo updates). This results in preserving previously acquired knowledge while also adapting to few-shot classes if the model has proper generalization capability.

Meta-Learning Procedure. Algorithm 1 shows the meta-learning procedure we have defined for attack-traffic classification, which is followed by all model-based approaches along with `MAML` and `ANIL`. In addition to the common *meta-training* on D_{nf_1} (lines 1–6) and *meta-testing* on D_f (lines 14–17) described in Sec. 2.1, we perform a further *meta-validation* phase on D_{nf_2} (lines 7–13), which has a disjoint label space from the other two datasets, namely $D_{nf} = D_{nf_1} \cup D_{nf_2}$ | $D_{nf_1} \cap D_{nf_2} = \varnothing$. More precisely, meta-training is conducted for a given number of epochs in an episodic fashion, as described before. On the other hand, D_{nf_2} is utilized to implement an early-stopping procedure based on the achieved accuracy (lines 7–12) and to choose the model that obtains the highest performance after the completion of the training process (`restoreBestModel(valAccuracies)` in line 13 selects the best-performing model across all the epochs).[11] Finally, meta-testing—which emulates the operational phase—is performed on this best-performing model. It should be noted that during meta-testing the performance obtained by the chosen model is properly assessed using D_f.[12]

Algorithm 1 Few-Shot Meta Learning

Require: D_{nf_1}, D_{nf_2}, D_f, K, and N
1: **for** $i \leftarrow 1$ to *epochs* **do**
2: **for** $j \leftarrow 1$ to *episodes* **do** ▷ Meta-training
3: *support*, *query* ← getEpisode(D_{nf_1}, K, N)
4: *trainPreds* ← metaLearningAlgorithm(*support*, *query*)
5: *trainLoss*[j] ← computeLoss(*trainPreds*)
6: backpropagate(*trainLoss*)
7: **for** $j \leftarrow 1$ to *episodes* **do** ▷ Meta-validation
8: *support*, *query* ← getEpisode(D_{nf_2}, K, N)
9: *valPreds* ← metaLearningAlgorithm(*support*, *query*)
10: *valAccuracy*[j] ← computeAccuracy(*valPreds*)
11: *valAccuracies*[i] ← average(*valAccuracy*)
12: **if** earlyStopping(*valAccuracies*) **then** break
13: restoreBestModel(*valAccuracies*)
14: **for** $i \leftarrow 1$ to *episodes* **do** ▷ Meta-testing
15: *support*, *query* ← getEpisode(D_f, K, N)
16: *testPreds* ← metaLearningAlgorithm(*support*, *query*)
17: *testAccuracy*[i] ← computeAccuracy(*valPreds*)
return average(*testAccuracy*)

Finally, Fig. 4 depicts the overall workflow obtained for IoT attack-traffic classification via FSL, which results from the methodological and design aspects described in the present section.

4. Experimental Setup

In this section, we describe the setup utilized for the experimental campaigns conducted in this study. In detail, we first outline the dataset leveraged for the experimentation in Sec. 4.1. Then, in Sec. 4.2, we describe the FSL setup in terms of meta- and transfer-learning setups, and hyperparameters configuration. Finally, we discuss the metrics employed to evaluate the performance of considered models in Sec. 4.3.

4.1. Dataset Description

The `IoT-23` dataset [4] has been collected at the Stratosphere Laboratory of the Czech Technical University between 2018

[10] The term "model head" is commonly used to refer to a dense layer with softmax activation connected to the last layer of the embedding function.

[11] The validation accuracy is calculated for each episode, then the per-episode accuracy values are averaged to derive an average measure for each epoch. Finally, the model that exhibits the best performance based on this measure is selected.

[12] The performance could be evaluated also on D_{nf_1} and D_{nf_2}. However, the evaluation on D_f is the most interesting result, since the meta-testing is performed on previously unseen data, thus emulating an actual operational scenario of attack-traffic classification.

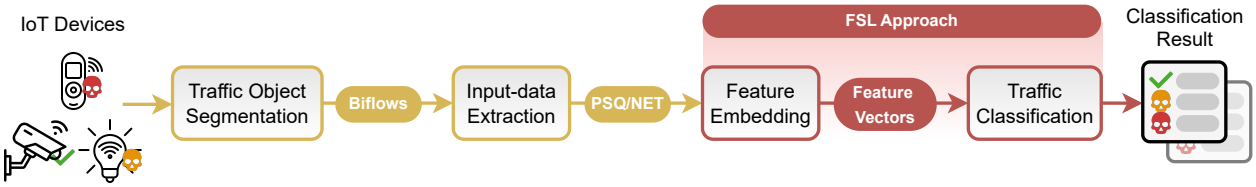


Figure 4: Workflow for IoT attack-traffic classification via Few-Shot Learning.

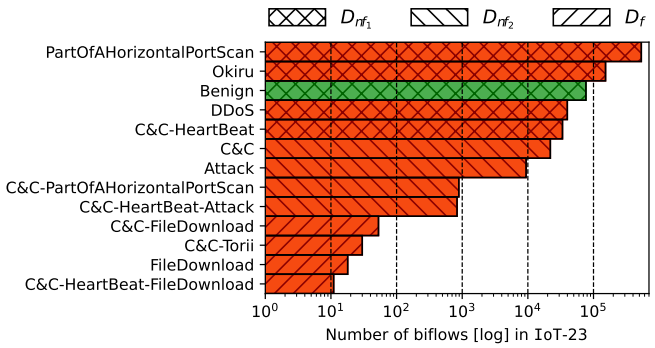


Figure 5: Number of per-class biflows (in log scale) in IoT-23 after preprocessing and dataset partitioning. For details on label meaning, please refer to the description in [4].

and 2019. It has been released as 23 PCAP traces along with ground-truth files obtained via Zeek.[13] A number of scientific and practical motivations led us to select it. First, as introduced in Sec. 1, IoT-23 dataset represents an actual attack-traffic classification scenario where the adoption of FSL is of practical interest, being characterized by scarcity of samples for training the classification models. Nevertheless, the dataset is rich enough (870.6*k* biflows and 13 classes after preprocessing) to constitute a representative evaluation test bench. Moreover, IoT-23 is a public and well-known dataset, which has been already used in a number of previous works [16, 38]. Thus, its adoption also favors the reproducibility of the experiments by the scientific community.

Going into detail, IoT-23 has been collected in a controlled IoT environment with an unrestrained network connection, with no defense solutions being enforced. A Raspberry Pi infected with specific malware is manipulated to produce malicious network traffic, while three actual IoT devices (i.e. a Somfy Smart Doorlock, a Philips HUE Smart LED Lamp, and an Amazon Echo Home) generate benign network traffic. The dataset is manually labeled at the biflow level, detailing the relationship between malicious flows and the corresponding malicious activities carried out. Conversely, traffic originating from non-malicious sources is simply labeled as "benign".

IoT-23 exhibits a severe class imbalance problem: the four most highly-populated classes present more than 15*M* biflows, the three least-populated ones have less than 10 biflows, whereas all the other classes present no more than 40*k* biflows. Hence, we randomly down-sampled (without replacement) the former majority classes to the 0.25% of the original dataset and removed the latter minority ones. After these pre-processing operations, the employed dataset encompasses 870.6*k* biflows distributed among 13 classes as depicted in Fig. 5: 12 attack-traffic classes (in red) and 1 benign class (in green). Overall, IoT-23 encompasses 99.85% TCP biflows, with the remaining UDP biflows mainly ($\approx$ 80%) belonging to the benign class.

It is worth noting that attack-traffic classes have *labels reporting one or more malicious activities*. As explained in [4], each malicious activity should be considered as a detail of the overall attack performed. For example, in *C&C-FileDownload*, "C&C" is the first activity conducted (i.e. Command & Control the infected victim), and "FileDownload" is what is done in that C&C. Table 2 describes the meaning of each malicious activity in IoT-23. We refer to the IoT-23 website [4] for further specific details.

The attacks in IoT-23 are also very diverse, covering all the classic phases of the life cycle of common botnets (e.g., Mirai, Reaper, BrickerBot, Okiru, and Hajime). In the following, we describe the relationship between each phase of a botnet life cycle and the malicious activities in IoT-23 (reported in parentheses):

i. *Infection* (Okiru and Torii), the devices get infected with malware, turning them into bots;

ii. *Propagation* (PartOfAHorizontalPortScan), the botnet spreads, infecting more devices;

iii. *Command and Control Setup* (C&C), the botmaster establishes a control infrastructure;

iv. *Exploitation* (DDoS, Attack, FileDownload), the botnet carries out attacks or malicious activities;

v. *Maintenance* (HeartBeat), the botnet is managed and updated by the botmaster.

It should be noted that our methodology is general and adapts to various types of IoT-network attacks, namely it is not necessarily bound to botnet traffic. In fact, while the IoT-23 dataset includes malicious activities associated with phases explicitly related to botnets' life cycle (e.g., *Infection*, *Command and Control Setup*, and *Maintenance*), it also covers more general phases like *Propagation* (viz. reconnaissance) and *Exploitation*. To further prove the generalizability of our methodology, we also consider a wider set of attack scenarios by selecting three additional datasets related to the IoT-security context, namely IoT-NID, Bot-IoT, and Edge-IIoTset (cf. Sec. 5.2).

[13] https://www.stratosphereips.org/datasets-iot23

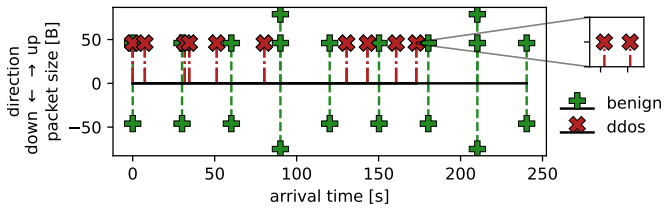


Figure 6: Example of benign and malicious (DDoS) biflows in terms of PSQ features (specifically, direction, packet size, and inter-arrival time). Packet direction is denoted via the sign associated with the packet size: positive (↑) for upstream and negative (↓) for downstream packets. The zoomed-in view (top right) reveals two closely-spaced, equally-sized consecutive upstream packets, which is characteristic of the DDoS instance. This pattern repeats for each DDoS marker in the figure.

Network Input Size. Regarding the network input data described in Sec. 3.1, we use $N_p = 20$ packets and $N_b = 576$ bytes for PSQ and NET, respectively. This choice is justified by the sensitivity analysis on the IoT-23 dataset we conducted in [16] with the same embedding functions considered herein (see Sec. 3.2). To further substantiate our choice, in Appendix A, we also report a dataset characterization showing the value distribution of PSQ and NET for the different classes of IoT-23.

In the case of IoT-23, we underline that the first 576 bytes of a biflow (i.e. the NET input) can be collected from the first few packets (at most 3 for 95% of the biflows). Therefore, the time required to collect the network input is dominated by the size of PSQ, i.e. 20 packets.

The network input collection for the biflows having 20 packets or more requires less than $\approx$ 16 seconds in 95% of the cases. Biflows shorter than 20 packets conclude within $\approx$ 1 second. Refining the data collection with a threshold set to 5 seconds to interrupt it in case of silence, the time required to collect network input data falls within 6–16 seconds for 95% of the biflows.

Figure 6 presents an example of benign and malicious (i.e. DDoS) biflows from the IoT-23 dataset, focusing on 3 representative fields from the PSQ input. From this example, a clear request/response periodic pattern emerges (green + markers in Fig. 6), characteristic of IoT device (benign) traffic. Conversely, the DDoS attack instance is characterized by pairs of equally sized consecutive packets sent in the upstream direction (red $\times$ markers in Fig. 6). Note that the figure is meant to report exemplificative patterns for the inputs, but it is not intended to capture all the intricacies of the data-driven model trained for feature embedding.

Dataset Partitioning. We split the classes of IoT-23 into three disjoint sets to be exploited for evaluating FSL attack-traffic classification approaches: C_{nf_1}, C_{nf_2}, and C_f corresponding to the classes considered to build D_{nf_1}, D_{nf_2}, and D_f, respectively. Figure 5 depicts the three sets of classes defined. More specifically, C_{nf_1} includes the 5 most populous classes of IoT-23; C_{nf_2} consists of 4 classes, the most populous besides those in C_{nf_1}; the last set C_f includes the 4 remaining less populated classes.

Table 2: Description of malicious activities in IoT-23.

Malicious Activity	Description
Attack	There is an attack from an infected device to another host (e.g., brute force, command injection).
C&C	The infected device has been connected to a *Command and Control* server.
DDoS	The infected device is the source of a *distributed denial of service* attack.
FileDownload	A file has been downloaded from the infected device.
HeartBeat	The packets sent on that connection are used to keep track of hosts infected by a C&C server.
Okiru	The connection has the characteristics of an *Okiru* botnet.
PartOfAHorizontal PortScan	The connections are used to do a *horizontal port scan* to obtain information for future attacks.
Torii	The connection has the characteristics of a *Torii* botnet.

To emulate an actual few-shot operational scenario, the classes in C_f have less than 100 samples, with the least populated class having just 11 samples.

4.2. Few-Shot Learning Setup

In the following, we describe the setup for both meta-learning and transfer-learning paradigms and the configuration of hyperparameters set for FSL approaches.

Meta-Learning Setup. To outline the meta-learning configuration, we provide details on the definition of *meta-learning episodes*, which involves the selection of the parameters N (*ways*) and K (*shots*) (see Sec. 2.1). During the meta-training, we set the values of N and K based on the specific objective of our analysis. Specifically, N is set to 4 (i.e. the number of classes in C_{nf_2} and C_f) in both meta-validation and meta-testing phases: we aim at solving a classification problem where the samples of 4 classes are available during the operational phase (i.e. at inference time). In order to mimic the latter phase, N is also set to 4 during meta-training, unless we evaluate the impact of performing meta-training with a varying number of classes. To select K, we are constrained by U_{min}, i.e. the minimum number of samples available in a class: $K_q + K_s \leqslant U_{min}$. Indeed, if we set $K_q = K_s = K$, then $K \leqslant \lfloor \frac{U_{min}}{2} \rfloor$. For IoT-23, $U_{min} = 11$ (and $K \leqslant 5$) for meta-testing, namely the number of samples in D_f of the least-populated *C&C-HeartBeat-FileDownload* class of C_f.

Transfer-Learning Setup. To implement the transfer-learning approaches, we use the *cross-evaluation procedure* as follows. We further split both D_{nf} and D_f into a *train set* and a *test set* that play roles analogous to the support set and the query set of meta-learning, respectively. In detail, we perform 10 different hold-outs as a cross-validation strategy (with 70%/30% train/test ratio). From each hold-out, we randomly generate 10

additional training runs, each one characterized by the selection of a uniquely drawn set of K samples per class (i.e. having the same cardinality as the support set used for meta-learning), for a total of 100 evaluation points.

Hyperparameters Configuration. We have tuned the hyperparameters characterizing the considered FSL approaches via a preliminary experimental campaign on D_{nf_2}. The optimal configuration obtained is characterized by 100 *training epochs* (each consisting of 120 *episodes* in the case of meta-learning) and utilizes the *Adam* optimizer with a learning rate of 10^{-4}. Moreover, we apply an *early-stopping* mechanism that monitors the validation accuracy to mitigate overfitting (with a patience of 20 epochs and a minimum delta of 0.01). Other approach-specific hyperparameters have been configured via similar preliminary experimentations. For completeness, we report them in the Appendix D, along with additional implementation details.

4.3. Performance Metrics

To assess the performance of FSL approaches, we employ the macro F1-score and the Davies-Bouldin index. For each metric, we show both its average value and the 95^{th} confidence interval (attained on D_f) over different runs, i.e. 120 and 100 for meta- and transfer-learning approaches, respectively.

The macro *F1-score* is harmonic mean of per-class precision ($prec_l$) and recall (rec_l), which are averaged over the L classes: $F1 = \frac{1}{L}\sum_{l=1}^{L}\frac{2*prec_l*rec_l}{prec_l+rec_l}$. We utilize the F1-score in FSL approaches based on both meta-learning (where episodes are inherently balanced) and transfer-learning, as well as for non-few-shot baselines (the latter two are subject to data imbalance). In fact, the F1-score proves to be more robust than the conventional accuracy when dealing with skewed datasets.

The *Davies-Bouldin index* (*DB index*, in brief) is used to measure the embedding function effectiveness since it quantifies how similar an object is to its own cluster compared to the other ones. DB index ranges from 0 (best) to $+\infty$ (worst). This aspect holds great importance, as most meta-models operate akin to a nearest-neighbor classifier within the embedded space. Hence, a key point of good performance is to define a clearly circumscribed space (viz. cluster) for each class. Formally, the DB index is defined as the average of per-cluster indexes $(s_l + s_k)/d_{lk}$ over all the clusters (viz. classes): $DB = \frac{1}{L}\sum_{l=1}^{L}\max_{k\neq l}\frac{s_l+s_k}{d_{lk}}$, where s_l is the average distance between each point of cluster l and the centroid of that cluster, d_{lk} is the distance between the cluster centroids l and k, and L is the number of clusters.

5. Experimental Evaluation

In Sec. 5.1, we perform a comprehensive comparison of embedding functions aiming to select the best feature extractor. In Sec. 5.2, we analyze the applicability of the FSL approaches on different datasets, with the goal to validate their generalizability. Then, in Sec. 5.3, we conduct multiple sensitivity analyses to assess the impact of various parameters (e.g., the number of ways and shots) on performance. To validate the effectiveness of FSL approaches, in Sec. 5.4, we show the performance of non-few-shot solutions. Finally, in Sec. 5.5, we evaluate the impact of using a larger training dataset.

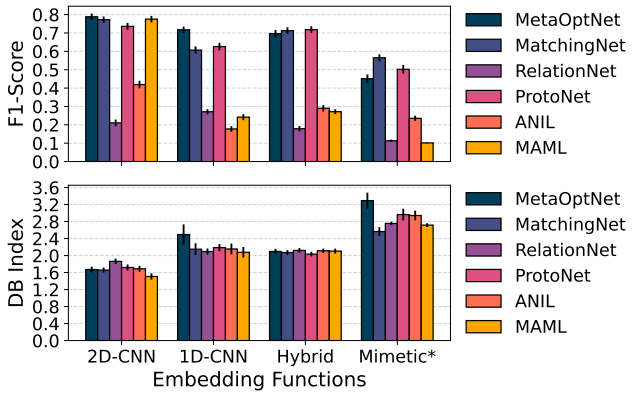


Figure 7: Performance according to F1-score (top) and DB index (bottom) when using different embedding functions.

5.1. Embedding Function Selection

This analysis assesses the performance of FSL approaches with varying embedding functions. We employ the DL architectures described in Sec. 3.2: 2D-CNN, HYBRID, 1D-CNN, and MIMETIC*.[14] We set the number of ways (N) and shots ($K = K_q = K_s$) to 4 and 5, respectively (i.e. 4-way 5-shot).

Figure 7 (top) illustrates the F1-score achieved on D_f by the six FSL approaches with different embedding functions. We can notice that FSL approaches (mostly) show the same relative performance when varying the embedding function: `MatchingNet`, `MetaOptNet`, and `ProtoNet` are the best performing, whereas `RelationNet` and `ANIL` are the worst ones, regardless of the embedding function. `MetaOptNet` reaches the overall best F1-score (up to 0.80) using the 2D-CNN. Focusing on the specific embedding functions, those fed with `PSQ` input data (i.e. 2D-CNN and HYBRID) exhibit better performance. Interestingly, single-modal embedding functions result in a better F1-score w.r.t. MIMETIC* for all FSL approaches but `ANIL`. This outcome highlights the need to design optimized multi-modal DL solutions (e.g., in terms of architecture and learning procedure) to deal with attack-traffic classification in an FSL scenario.

The outcomes that can be drawn when looking at the DB index (Fig. 7, bottom) are in line with those provided by the F1-score, with 2D-CNN significantly outperforming other embedding functions. However, in this case, `MAML` shows slightly better results than other FSL approaches, although it never reaches a DB index lower than 1.5. This denotes that the common training procedure minimizing categorical cross-entropy loss is struggling in obtaining a generalizable and separable embedding space. This call for a more informative loss function with

[14] When employing the 1D-CNN, we opt for *First-Order `MAML`* to reduce the computational load. This variant, although simpler, achieves nearly comparable performance to `MAML` by exclusively utilizing first-order gradients during parameter optimization [18].

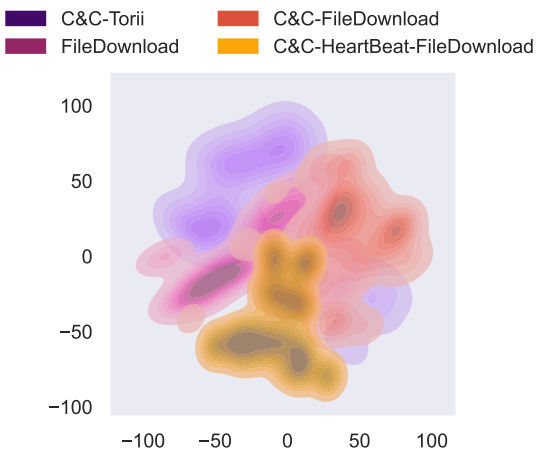


Figure 8: Kernel Density Estimate (KDE) plot obtained projecting the feature vectors produced by `MetaOptNet` with 2D-CNN as embedding function via t-SNE.

the objective of improving the separability of classes in the latent space (e.g., by including an optimization term based on the DB index).

Also, we estimate the complexity of the embedding functions via the number of Trainable Parameters (TP) of each DL architecture. The best-performing 2D-CNN (145k TP) shows significantly lower (viz. better) computational complexity w.r.t. MIMETIC* (1.2M TP) and 1D-CNN (4.0M TP) and similar to HYBRID (119k TP). Therefore, in the following analyses, we leverage the 2D-CNN as embedding function.

To provide an example of the effectiveness of 2D-CNN, Fig. 8 depicts the projection in a two-dimensional space of the embedded biflows obtained using the best-performing approach, i.e. `MetaOptNet`. This projection is achieved using *t-Distributed Stochastic Neighbor Embedding (t-SNE)*, which aims to preserve the local structure of the embedded space. From this figure, it is evident that sufficiently separable clusters can be identified. In fact, the cluster that tends to overlap more with others in certain regions is the *C&C-FileDownload*, highlighting a similarity with the other classes in C_f.

5.2. Generalization Performance

In this section, we delve into the generalization capability of the chosen FSL approaches considering a wider set of attack scenarios. To this aim, we have selected three additional datasets related to the IoT-security context: `IoT-NID` [39], `Bot-IoT` [40], and `Edge-IIoTset` [41].[15] To confirm the broad applicability of the considered FSL approaches for classifying attack traffic in IoT environments, we exploit the best embedding function based on the previous analysis (i.e. 2D-CNN) and evaluate its performance on these three datasets. Note that although such datasets all contain attack traffic involving IoT devices, their nature and composition are different. Thus, the resulting classification problems may be characterized by a different complexity.

[15] Further details regarding these datasets are reported in Appendix B.

Dataset Partitioning. As for `IoT-23`, we have applied two criteria to define the partitioning of the dataset: (*i*) C_f includes all underrepresented classes (i.e. the minority ones in their respective datasets); (*ii*) D_{nf_1} and D_{nf_2} have a number of classes equal to C_f, at least. This choice influences the number of ways, since N is set equal to C_f. On the other hand, $K(= K_s)$ is kept fixed for all the datasets to keep the analysis consistent in terms of shots. Below, we deepen the partitioning and the configuration of the episodes for each dataset:

- `IoT-NID` consists of 10 highly imbalanced classes. The 3 least frequent classes (i.e. *OS Detection*, *Man-in-the-Middle (MITM) ARP Spoofing*, and *Mirai Host Bruteforce*) have less than 500 biflows each, while the most frequent class (i.e. *Benign*) has over 80k biflows. These minority classes are included in C_f. Consequently, for meta-training, meta-validation, and meta-testing, we define 3-way 5-shot tasks ($K = K_s = 5$, $K_q = 50$).[16]
- `Bot-IoT` gathers traffic from 10 different attacks along with benign traffic, for a total of 11 classes. As for `IoT-NID`, C_f includes the 3 minority classes (i.e. *DDoS HTTP*, *Keylogging*, and *Data Exfiltration*) having under 2k samples, against a majority class (i.e. *OS Fingerprint*) with more than 50k samples. The episode setup is the same as `IoT-NID`.
- Unlike the other datasets, `Edge-IIoTset` contains a larger number of 24 classes, enabling an evaluation scenario that considers a larger number of ways. Therefore, we choose the classes with less than 2k samples for C_f (i.e. *Water Level*, *Vulnerability Scanner*, *XSS*, *Mobdus*, *Backdoor*, *Ransomware*, *OS Fingerprint*, and *MITM*). On the other hand, the majority class (i.e. *Sound Sensor*) has $\approx$ 95k samples. In this case, we set episodes with $N = 8$, $K_s = 5$, and $K_q = 50$.

Generalization Results. Overall, the experimental results suggest that FSL has a wide range of applications in the domain of attack-traffic classification. With the exception of `Edge-IIoTset`—which results in a unique setup and a harder classification problem when applying the aforementioned criteria—the observed effectiveness of FSL methods is comparable (`IoT-NID`) or better (`Bot-IoT`) when compared to that achieved on `IoT-23`. Table 3 presents the F1-score and the DB index for the 6 meta-learning approaches on the different datasets considered. Starting with `IoT-NID`, we observe F1-score and DB index values relatively similar to those obtained on `IoT-23`, except for `RelationNet` and `MatchingNet`, which tend to improve (+46% F1-score) and deteriorate (−14% F1-score), respectively. The performance is good both in terms of F1-score and DB index when the approaches are tested on `Bot-IoT`. Particularly, the best approach, `MetaOptNet`, achieves an F1-score of 83%. On the other hand, classification results on `Edge-IIoTset` are worse, likely due to a more complex classification

[16] Unlike `IoT-23`, the minority class has enough biflows to consider a larger value for K_q.

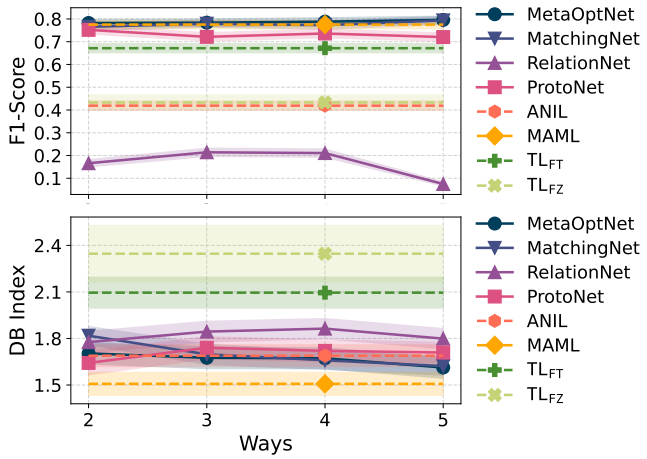


Figure 9: Sensitivity analysis to train ways (N) in terms of F1-score (top) and DB index (bottom).

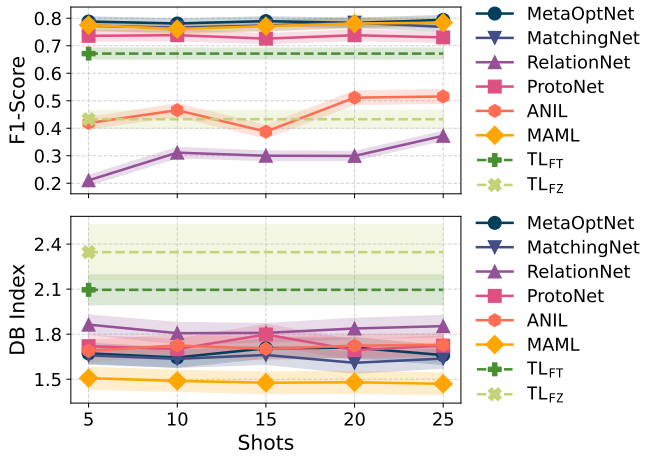


Figure 10: Sensitivity analysis to train shots (K) in terms of F1-score (top) and DB index (bottom).

task given the higher number of ways ($N = 8$). This highlights the need for a larger number of shots, as results show that having 5 samples for 8 classes is insufficient. Notably, some approaches (i.e. `MetaOptNet`, `MatchingNet`, and `ProtoNet`) report slight variability across different datasets. Conversely, the remaining (i.e. `RelationNet`, `ANIL`, and `MAML`) exhibit greater sensitivity to different data, thus suggesting that an additional hyperparameter tuning phase for the specific scenario would be beneficial, as also stated in [42].

5.3. Sensitivity Analyses

This section presents sensitivity analyses to the number of ways (N) and shots (K) and concludes by relating the embedding function effectiveness with classification performance.

Sensitivity to the Number of Ways (N). This analysis aims to investigate the impact of the number of meta-training ways N looking at the trend of the F1-score and DB index. More specifically, N ranges from 2 to 5 for meta-training tasks, while it is set to 4 for meta-testing and meta-validation ones. The number of shots remains constant at $K = 5$. Likewise, in the case of $\texttt{TL}_{\texttt{FT}}$ and $\texttt{TL}_{\texttt{FZ}}$, we present the outcomes obtained on C_f classes employing $K = 5$ samples for each class. Conversely, `MAML` and `ANIL` require the same value of N for meta-training, meta-validation, and meta-testing, precluding this specific analysis. However, to provide an additional benchmark, we show the performance of `MAML` and `ANIL` with 4-way 5-shot tasks for each meta-learning phase. Figure 9 shows an almost constant trend with only minor oscillations for both metrics when changing N. `MetaOptNet` reaches the highest F1-score of ≈ 0.80 and DB index of ≈ 1.61 with 5 ways, although the scores of `ProtoNet` and `MatchingNet` are very close. Conversely, `RelationNet` is the worst-performing approach. Regarding transfer-learning-based approaches, F1-score and (mostly) DB index show poor performance for both $\texttt{TL}_{\texttt{FT}}$ (< 0.70 F1-score and 2.09 DB index) and $\texttt{TL}_{\texttt{FZ}}$ (< 0.50 F1-score and 2.35 DB index).

Sensitivity to the Number of Shots (K). In this analysis, we simultaneously vary the number of shots of query and support sets ($K_q = K_s = K$) used during meta-training from 5 to 25 with a step of 5, while for meta-validation and meta-testing $K = 5$. The number of ways is $N = 4$. Again, for $\texttt{TL}_{\texttt{FT}}$ and $\texttt{TL}_{\texttt{FZ}}$, we report the results on C_f using $K = 5$ samples per class (i.e. equal to K used in the operational phase of meta-learning approaches). The F1-score in Fig. 10 (top) reports that the best-performing approaches (`MetaOptNet`, `MatchingNet`, `ProtoNet`, and `MAML`) do not show a significant increment for larger values of K. On the other hand, the approaches with lower performance (`ANIL` and `RelationNet`) improve in a more significant way with K, still not providing performance comparable to the others ($\approx$ 0.50 F1-score at most). `MetaOptNet` achieves the best F1-score (0.80) when $K = 25$, although the improvement is limited ($\approx +0.01$ with respect to $K = 5$), especially considering that the computational effort when $K = 25$ is much higher. DB index is depicted in Fig. 10 (bottom) and highlights patterns similar to the F1-score. In this case, `MAML` has the best results with a DB index equal to 1.47 when $K = 25$, whereas `RelationNet` is the worst approach (1.86 when $K = 5$).

In addition, we wonder if using a higher number of support samples in the operational phase (i.e. meta-testing) may result in performance improvement. Therefore, hereinafter we consider the best-performing approach, namely `MetaOptNet`, and jointly vary $K = K_s = K_q$ in $\{5, 10, 15, 20, 25\}$ in all the meta-learning phases. Because the maximum number of samples for classes in C_f may be lower than the required K, we set the number of support samples to the highest possible for each class in the meta-testing phase, namely we use $K = \min\{K, U_l - 1\}$, where U_l is the total number of samples of the class l. Results are shown in Fig. 11 as confusion matrices, one for each tested value of K. From the F1-score values on top of each matrix, a clear trend emerges: the performance degrades for higher K, i.e. F1-score passes from 0.79 with $K = 5$ to 0.56 with $K = 25$. Nevertheless, confusion matrices exhibit diverse fine-grained behaviors. The main finding is that the more populous classes obtain a performance boost when passing from $K = 5$

Table 3: F1-score and DB index achieved by the FSL approaches on `IoT-NID`, `Bot-IoT`, `Edge-IIoTset`, and `IoT-23`.

Dataset	Metric	FSL Approaches					
		`MetaOptNet`	`MatchingNet`	`RelationNet`	`ProtoNet`	`ANIL`	`MAML`
`IoT-NID`	F1-Score [%]	72.89	63.53	67.50	74.44	44.83	77.23
	DB Index	1.54	1.39	1.25	1.88	1.61	1.37
`Bot-IoT`	F1-Score [%]	83.80	80.11	65.43	81.27	75.50	83.02
	DB Index	1.34	1.43	1.27	1.58	1.43	1.22
`Edge-IIoTset`	F1-Score [%]	64.57	67.86	31.03	66.70	19.88	19.21
	DB Index	2.97	1.79	2.15	2.11	2.06	1.99
`IoT-23`	F1-Score [%]	78.80	77.27	21.06	73.60	41.90	77.51
	DB Index	1.67	1.65	1.86	1.72	1.68	1.50

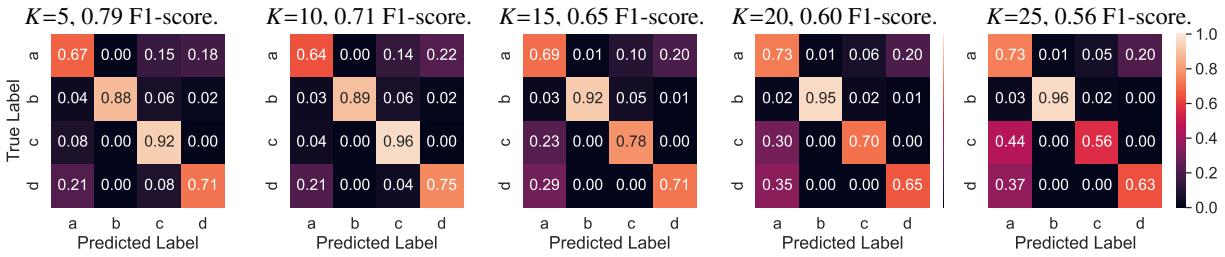


Figure 11: Confusion matrices for best performing FSL approach `MetaOptNet` when varying the number of both support-set and query-set sizes $K \in \{5, 10, 15, 20, 25\}$ in all meta-learning phases.
Label encoding: a = C&C-FileDownload, b = C&C-Torii, c = FileDownload, and d = C&C-HeartBeat-FileDownload.

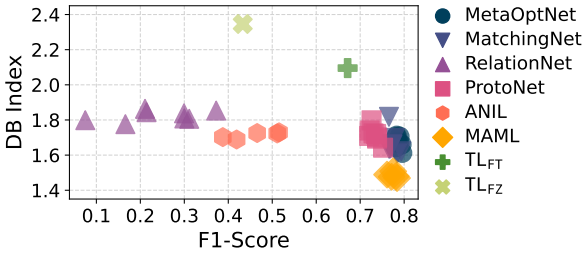


Figure 12: F1-score against DB index of FSL approaches. Results for all the N-way K-shot considered are reported.

to $K = 25$: the recall score passes from 0.67 (0.88) to 0.73 (0.96) for the *C&C-FileDownload* (*C&C-Torii*). On the other hand, less populated classes experience performance degradation, likely due to the higher imbalance of the support-set size result of the higher number of shots.

Classification Performance vs. Embedding Function Effectiveness. Figure 12 further deepens the relation between the F1-score and DB index. In detail, a correlating trend can be identified—particularly shown by `MetaOptNet`, `MatchingNet`, `MAML`, `ProtoNet`, TL_{FT}, and TL_{FZ}—where the higher (resp. the lower) the classification accuracy the better (resp. the worse) the separation effectiveness of the embedding function. The remaining approaches show peculiar behaviors: `RelationNet` results in a slightly worse embed space for higher F1-scores, thus proving its poor adaptability to attack-traffic classification; differently, `ANIL` obtains a good embedding separation, but it suffers in terms of F1-scores, highlighting the chance to improve its performance, e.g., by devising a loss function explicitly optimizing the DB index.

5.4. *Performance of Non-Few-Shot Approaches*

This section aims to validate the necessity of applying few-shot tailored solutions for performing attack-traffic classification. With this goal in mind, we leverage the Random Forest (RF), the eXtreme Gradient Boosting (XGB), and the best-performing 2D-CNN (hereinafter, referred to as `Scratch`) as non-few-shot approaches, being these models successfully used in the traffic-classification domain [3, 32]. As input data, we selected the `PSQ` input type defined in Sec. 3.1 properly flattened to feed RF and XGB. To perform this analysis we employ (*i*) a non-few-shot-based masking procedure and (*ii*) traditional class-imbalance techniques (namely, *random oversampling* and *SMOTE*), both detailed in Appendix E. We evaluate the classification performance of such solutions via the same cross-validation procedure used for transfer-learning.

Regarding the *non-few-shot-based masking procedure*, RF, XGB, and `Scratch` obtain an average F1-score of 0.74, 0.71, and 0.69, respectively (see the central dashed-delimited block in Fig. 13). Accordingly, all non-few-shot approaches featuring the masking inference procedure present poorer classification performance than the best FSL ones (i.e. `MetaOptNet`, `MatchingNet`, and `MAML`), which show a relative improvement of 8%–16% F1-score.

Moreover, regarding the application of *traditional class-imbalance techniques*, namely random oversampling and

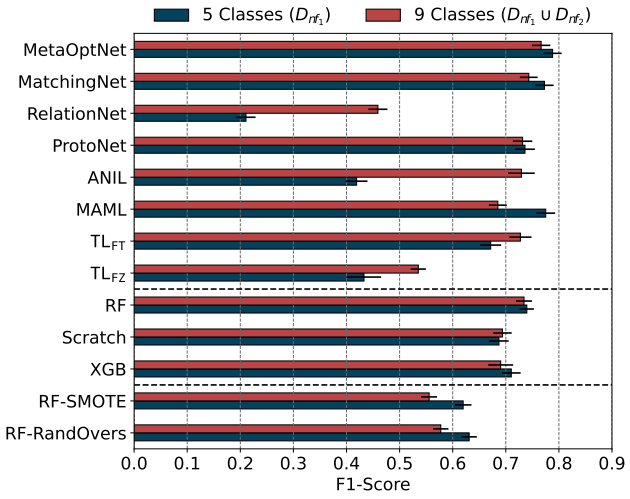


Figure 13: Comparison of F1-score attained by both few-shot and non-few-shot approaches when using training datasets with a different number of classes.

SMOTE, we incorporate them into the training of RF, which is the best non-few-shot model compared to XGB and `Scratch` (see Fig. 13). Specifically, `RF-RandOvers` uses random oversampling, while `RF-SMOTE` leverages synthetic instances generated by SMOTE. Both are used to augment the $K = 5$ samples of few-shot classes, resulting in 5000 training samples for each class in C_f (cf. Appendix E).

Looking at the bottom dashed-delimited block in Fig. 13, `RF-RandOvers` and `RF-SMOTE` yield lower F1-scores (0.63 and 0.62, respectively) than those of RF trained with the former masking technique for overfitting mitigation (0.74 F1-score). This highlights the limitations of these methods for extremely imbalanced scenarios. In fact, despite one can argue that 5000 augmented samples might be not enough because the 2-order-of-magnitude gap with the majority class ($\approx 10^5$ samples, as depicted in Fig. 5), creating a higher number of samples from a limited starting pool (i.e. $K = 5$ samples) may result in very low variability for minority classes, inevitably leading to overfitting in real-world scenarios. Therefore, this investigation is impractical.

In the next section, we further deepen the comparison between FSL approaches and non-few-shot baselines when varying the training requirements.

5.5. Performance when Augmenting Prior Knowledge

This experimental campaign is aimed at evaluating the performance obtained when taking advantage of augmented knowledge during the initial training phase, that is using a larger set of non-few classes. Formally, we employ the training dataset $D_{nf_1} \cup D_{nf_2}$ counting 9 classes ($C_{nf_1} \cup C_{nf_2}$) and compare the corresponding performance with that obtained with D_{nf_1} counting 5 classes (C_{nf_1}). Notably, for meta-learning approaches, this analysis is a proxy to evaluate the effectiveness of the proposed meta-learning procedure (see Sec. 3.3) because merging D_{nf_1} and D_{nf_2} prevents the enforcement of early stopping: the meta-training phase is carried out for all the 100 epochs. For both meta-training and meta-testing, we set $N = 4$ and $K = 5$ (i.e. 4-ways 5-shots tasks), an analogous configuration is adopted for transfer-learning-based and non-few-shot approaches.

Figure 13 reports the classification performance in terms of F1-score when using a different number of classes during meta-training. Results show a performance boost (up to $+0.30$ F1-score) when leveraging 9 training classes mainly for those algorithms underperforming in previous analyses (i.e. `RelationNet`, `ANIL`, TL_{FZ}, and TL_{FT}). On the other hand, typically well-performing methods (i.e. `MetaOptNet`, `MatchingNet`, `ProtoNet`, and `MAML`) show a slight performance degradation, i.e. up to -0.09 F1-score by `MAML`.

However, because increasing the number of training classes is expected to lead to performance improvement, the worsening experienced by such approaches is likely related to overfitting, due to the absence of the early-stopping mechanism. Interestingly, non-few-shot approaches (i.e. RF, XGB, and `Scratch`) obtain similar performance regardless of the number of training classes. Differently, `RF-SMOTE` and `RF-RandOvers` exhibit lower F1-score values when trained with the wider class pool. This is likely attributable to overfitting phenomena. Noteworthy, these models struggle to obtain performance figures comparable to the three best FSL approaches, namely `MAML`, `MatchingNet`, and `MetaOptNet`, underlying the necessity of few-shot-tailored attack-traffic classification solutions.

6. Conclusions and Future Perspectives

In this work, we adopted FSL approaches for IoT attack-traffic classification. We used FSL as a means to cope with the problem of scarcity of training data for certain classes when dealing with the classification of traffic generated by cyberattacks. Accordingly, the evaluation relied on a recent and publicly available attack-traffic dataset collected in an IoT environment (`IoT-23`).

The analyses assessed (*i*) the impact of using different embedding functions for extracting features in the latent space, (*ii*) the generalization ability of FSL approaches to wider IoT attack scenarios, (*iii*) the sensitivity to the number of classes and the number of samples for each class, and (*iv*) the performance attained when augmenting the prior knowledge acquired from related non-few tasks. Both single-modal and multimodal state-of-the-art embedding functions based on DL have been considered, and results show that the 2D-CNN fed with packet-sequence input data performs the best in learning new tasks, on average. The experimentation covered the main families of approaches and paradigms to tackle the FSL problem, including meta-learning FSL approaches (`ProtoNet`, `MatchingNet`, `RelationNet`, `MetaOptNet`, `MAML`, and `ANIL`) and transfer-learning ones (Fine-Tuning and Freezing). In addition, various non-few-shot baselines are also evaluated.

Overall, the analysis of sensitivity against K (viz. the number of shots/instances per class) and N (viz. the number of sampled ways/classes) witnesses that no major impact on performance is observed. `MatchingNet`, `MetaOptNet`, and `MAML`

attain the best performance in terms of F1-score, which settles to 0.75–0.8. On the other hand, `RelationNet` provides the worst performance (always lower than 0.4 F1-score, at best). These performance figures generally improve when using a larger set of non-few classes, which positively impacts the worst-performing approaches (`RelationNet` and `ANIL`). Transfer-learning approaches show F1-scores lower than 0.7 at best, failing to achieve performance figures as good as the meta-learning ones. We also proved the effectiveness of few-shot-tailored attack-traffic classification solutions by showing that the best FSL approach (i.e. `MetaOptNet`) obtains a relative gain of F1-score in the range 8%–27% when compared with non-few-shot baselines.

Based on the outcomes of the experimental evaluation conducted herein, in future work, we plan to explore: (*i*) the optimization of the embedding function, e.g., by considering more advanced architectures; (*ii*) the improvement of the learning objective, e.g., by employing more advanced loss functions to enhance the quality of embeddings; (*iii*) the expansion of the current methodology to address multi-stage attacks that involve multiple phases for completion, e.g., by identifying and correlating botnet activities in specific stages; (*iv*) the comparison of algorithm- and model-based FSL approaches with data-based ones exploiting generative models, e.g., generative adversarial networks and variational autoencoders. Finally, (*v*) we plan to evaluate the generalizability of the proposed framework in application domains beyond IoT attack-traffic classification.

Acknowledgments

This work is partially supported by the Italian Research Program "PON *Ricerca e Innovazione* 2014-2020 (PON R&I) REACT-EU – Asse IV – Azione IV.4", and by the "PNRR ICSC National Research Centre for High Performance Computing, Big Data and Quantum Computing (CN00000013)", under the NRRP MUR program funded by the NextGenerationEU. Also, this work is partially carried out within the "xInternet" Project supported by the MUR PRIN 2022 program (D.D.104—02/02/2022) funded by the NextGenerationEU. This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.

References

[1] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani. Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations. *IEEE Commun. Surveys Tuts.*, 21(3), 2019.

[2] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé. MIRAGE: Mobile-app Traffic Capture and Ground-truth Creation. In *IEEE ICCCS*, 2019.

[3] C. Wang, A. Finamore, L. Yang, K. Fauvel, and D. Rossi. AppClassNet: A commercial-grade dataset for application identification research. *ACM SIGCOMM CRR*, 52(3), 2022.

[4] S. Garcia, A. Parmisano, and M. J. Erquiaga. IoT-23: A labeled dataset with malicious and benign IoT network traffic, 2020.

[5] S. Huang, Y. Liu, C. Fung, W. An, R. He, Y. Zhao, H. Yang, and Z. Luan. A Gated Few-shot Learning Model For Anomaly Detection. In *IEEE ICOIN*, 2020.

[6] W. Zheng, C. Gou, L. Yan, and S. Mo. Learning to Classify: A Flow-Based Relation Network for Encrypted Traffic Classification. In *ACM Web Conference*, 2020.

[7] Z.-M. Wang, J.-Y. Tian, J. Qin, H. Fang, and L.-M. Chen. A Few-Shot Learning-Based Siamese Capsule Network for Intrusion Detection with Imbalanced Training Data. *Hindawi Computat. Intell. Neurosci.*, 2021, Sep 2021.

[8] Z. Zhao, Y. Lai, Y. Wang, W. Jia, and H. He. A Few-Shot Learning Based Approach to IoT Traffic Classification. *IEEE Commun. Lett.*, 26(3), 2022.

[9] G. Bovenzi, D. Di Monda, A. Montieri, V. Persico, and A. Pescapé. Few shot learning approaches for classifying rare mobile-app encrypted traffic samples. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2023.

[10] C. Xu, J. Shen, and X. Du. A Method of Few-Shot Network Intrusion Detection Based on Meta-Learning Framework. *IEEE Trans. Inf. Forensics Security*, 15, 2020.

[11] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin. Siamese Neural Network Based Few-Shot Learning for Anomaly Detection in Industrial Cyber-Physical Systems. *IEEE Trans. Ind. Informat.*, 17(8), 2021.

[12] T. Feng, Q. Qi, J. Wang, and J. Liao. Few-Shot Class-Adaptive Anomaly Detection with Model-Agnostic Meta-Learning. In *IFIP Networking*, 2021.

[13] Y. Ouyang, B. Li, Q. Kong, H. Song, and T. Li. FS-IDS: A Novel Few-Shot Learning Based Intrusion Detection System for SCADA Networks. In *IEEE ICC*, 2021.

[14] W. Liang, Y. Hu, X. Zhou, Y. Pan, and K. I-Kai Wang. Variational Few-Shot Learning for Microservice-Oriented Intrusion Detection in Distributed Industrial IoT. *IEEE Trans. Ind. Informat.*, 2021.

[15] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. In *ICLR*, 2019.

[16] A. Nascita, F. Cerasuolo, D. Di Monda, J. Garcia, A. Montieri, and A. Pescapè. Machine and Deep Learning Approaches for IoT Attack Classification. In *IEEE INFOCOM*, 05 2022.

[17] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3), 2020.

[18] C. Finn, P. Abbeel, and S. Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *PMLR ICML*, volume 70, 2017.

[19] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. *NIPS*, 29, 2016.

[20] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. *NIPS*, 30, 2017.

[21] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE CVPR*, 2018.

[22] C. Rong, G. Gou, C. Hou, Z. Li, G. Xiong, and L. Guo. UMVD-FSL: Unseen Malware Variants Detection Using Few-Shot Learning. In *IEEE IJCNN*, 2021.

[23] H. Guo, X. Zhang, Y. Wang, B. Adebisi, H. Gacanin, and G. Gui. Few-shot malware traffic classification method using network traffic and meta transfer learning. In *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*. IEEE, 2022.

[24] J. Yang, H. Li, S. Shao, F. Zou, and Y. Wu. Fs-ids: A framework for intrusion detection based on few-shot learning. *Computers & Security*, 122, 2022.

[25] C. Lu, X. Wang, A. Yang, Y. Liu, and Z. Dong. A few-shot based model-agnostic meta-learning for intrusion detection in security of internet of things. *IEEE Internet of Things Journal*, 2023.

[26] M. Pawlicki, R. Kozik, and M. Choraś. Improving siamese neural networks with border extraction sampling for the use in real-time network intrusion detection. In *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023.

[27] M. M. U. Chowdhury, F. Hammond, G. Konowicz, C. Xin, H. Wu, and J. Li. A few-shot deep learning approach for improved intrusion detection. In *IEEE UEMCON*, 2017.

[28] Y. Yu and N. Bian. An Intrusion Detection Method Using Few-Shot Learning. *IEEE Access*, 8, 2020.

[29] L. Yu, J. Dong, L. Chen, M. Li, B. Xu, Z. Li, L. Qiao, L. Liu, B. Zhao, and C. Zhang. PBCNN: Packet Bytes-based Convolutional Neural Network for Network Intrusion Detection. *Elsevier Comput. Netw.*, 194, 2021.

[30] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *IEEE/CVF CVPR*, 2019.

[31] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. In *ICLR*, 2019.

[32] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Trans. Netw. Service Manag.*, 16(2), 2019.

[33] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE Access*, 5, 2017.

[34] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang. End-to-end encrypted Traffic Classification with one-dimensional convolution neural networks. In *IEEE ISI*, 2017.

[35] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu. Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE access*, 6, 2017.

[36] Z. Song, Z. Zhao, F. Zhang, G. Xiong, G. Cheng, X. Zhao, S. Guo, and B. Chen. I^2 rnn: An incremental and interpretable recurrent neural network for encrypted traffic classification. *IEEE Transactions on Dependable and Secure Computing*, 2023.

[37] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè. MIMETIC: Mobile encrypted traffic classification using multimodal deep learning. *Elsevier Comput. Netw.*, 165, 2019.

[38] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. den Hartog. ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets. *IEEE Internet Things J.*, 9(1), 2021.

[39] K. Hyunjae, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. Kim. Iot network intrusion dataset. *IEEE Dataport*, 2019.

[40] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100, 2019.

[41] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke. Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning. *IEEE Access*, 10, 2022.

[42] A. Antoniou, H. Edwards, and A. Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

Appendix A. IoT-23 Dataset Characterization

In this section, we delve into the choice of the PSQ and NET inputs (cf. Sec. 3.1) through a characterization of IoT-23 in the light of these two types of input. Specifically, this analysis witnesses how the two inputs represent a good choice for characterizing the peculiarity of the (IoT attack-traffic) classes composing the datasets. For a given class, we report the average value on all biflows for each packet/byte index (i.e. N_p and N_b, respectively) in Fig. A.14.

Interestingly, the packet length (PL) (see Fig. A.14(a)) proves to be particularly useful in distinguishing minority classes (i.e. *C&C-FileDownload*, *C&C-Torii*, *FileDownload*, and *C&C-HeartBeat-FileDownload*). The differences between "high frequency" attacks (e.g., *DDoS* and *PartOfAHorizontalPortScan*) are less relevant, as the PL remains at similar and low values.

The packet direction (DIR) exhibits interesting patterns for several classes (see Fig. A.14(b)). For example, *DDoS* is mainly composed of outgoing packets. Also, we note that this class encompasses various types of DDoS attacks from UDP-based to TCP Null attacks (i.e. having packets with all 6 TCP flags unset, as can be seen from $N_p = 11$ in Fig. A.14(e)). Other attacks (e.g., *C&C-HeartBeat-Attack* and *C&C-HeartBeat-FileDownload*) present several incoming packets from the C&C server (e.g., to keep track of the infected device).

The inter-arrival time (IAT) also presents diverse values for different classes (see Fig. A.14(c)). For instance, it tends to be higher for *Benign* and attacks like *C&C-HeartBeat* and *DDoS*, while it shows lower values for attacks with fewer samples.

The TCP window size (WIN) provides helpful information, especially in the first packets, up to $N_p = 5$ (see Fig. A.14(d)). The pattern exhibited by certain *DDoS* and *PartOfAHorizontalPortScan* biflows is interesting, with an average value of ≈ 2000 Bytes.

In addition to the 4 previously introduced fields, we include 2 more fields for this characterization introduced in [1], namely, the base-10 representation of the flag field in the TCP header (FLG) and the Time-to-Live (TTL).

FLG is particularly relevant for all those attacks exploiting TCP header malformations. In fact, as already mentioned, TCP Null Attack DDoS sets all flags to 0, whereas other DDoS attacks set various flags to 1 (see Fig. A.14(e), from $N_p = 3$ to $N_p = 6$).

Finally, there is a variance in TTL values (see Fig. A.14(f)) between *DDoS*, *PartOfAHorizontalPortScan*, and minority classes. However, compared to the other 4 features, FLG and TTL contribute less significantly. Hence, we have chosen to utilize PL, DIR, IAT, and WIN.

Regarding the NET input (see Fig. A.14(g) and Fig. A.14(i)), there is a clear difference between "high frequency" and "low frequency" attacks, where the latter, characterized by file download phases, have on average a longer payload in terms of bytes. Additionally, the effect of obfuscation conducted on biased fields [1] (e.g., IP addresses and L4 ports) can also be observed in the figure.

From Figs. A.14(h) and A.14(i) it also emerges that, in the case of IoT-23, there are no "late-starting" attacks, namely attacks that start after N_p packets or N_b bytes. Nonetheless, we remark that in the event of needing to address these attacks, it is sufficient to appropriately set N_p and N_b.

Appendix B. Details of IoT-NID, Bot-IoT, and Edge-IIoTset Datasets

Herein, we provide more details regarding the three additional datasets introduced in Sec. 5.2. Figure B.15 depicts the class population in terms of biflows for IoT-NID, Bot-IoT, and Edge-IIoTset. Additionally, we illustrate the partitioning of classes for D_{nf_1}, D_{nf_2}, and D_f (according to the criteria detailed in Sec. 5.2) using distinct hatch patterns. Normal and malicious classes are differentiated by coloring, the former in green and the latter in red. Notably, these datasets encompass both TCP and UDP traffic (e.g., *DDoS UDP* and *Mirai UDP* are attacks generating exclusively UDP traffic).

Appendix C. Embedding Functions Details

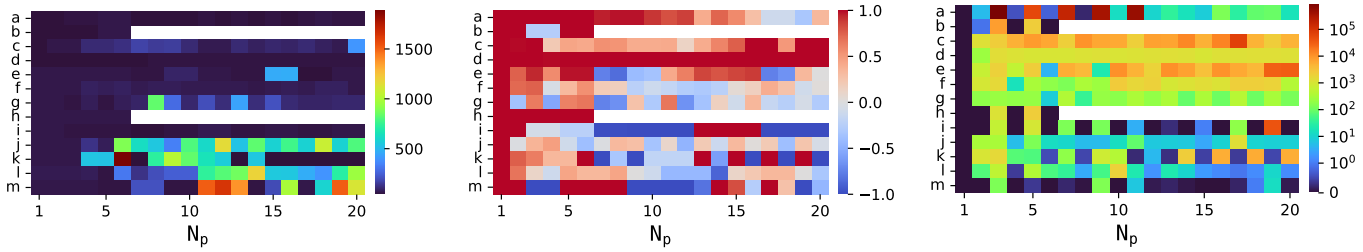
We provide additional details on Deep Learning architectures used as embedding functions described in Sec. 3 of the main manuscript. Figure C.16 depicts the connections between the layers of considered architectures, reporting also the references of their original proposals in the field of (attack-)traffic classification. We recall that 1D-CNN, HYBRID, and 2D-CNN are single-modal, whereas MIMETIC* is multimodal.

Appendix D. Configuration of FSL Approaches

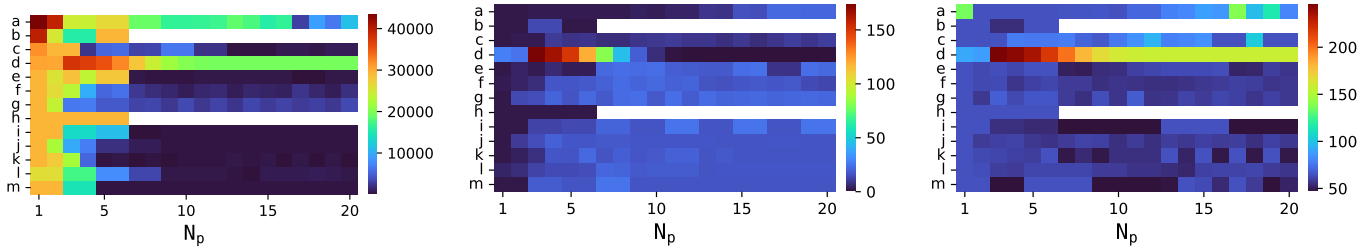
Hereinafter, the particular configuration of each FSL approach is provided in terms of approach-specific hyperparameters. Furthermore, for the sake of completeness and reproducibility, we give their implementation details.

Approach-Specific Hyperparameters. Different hyperparameters can be configured for the FSL approaches being investigated. We recall that the choices made herein are based on both state-of-the-art results and preliminary analyses performed. Both ProtoNet and MatchingNet employ the Euclidean distance, MatchingNet uses the *fully conditional embeddings with one layer* for the bidirectional Long Short-Term Memory and 2 *unrolling steps* for the attention Long Short-Term Memory. RelationNet uses the default relation module proposed in [5]. MetaOptNet is set with 15 maximum Support Vector Machine iterations and a *regularization parameter* c equal to 0.1. MAML and ANIL are calibrated with 8 *inner loop iterations* and an *inner/adaptation learning rate* equal to 0.01. The *Cross Entropy Loss* measures the error in all the models, except for MatchingNet and RelationNet, which utilize the *Negative Log Likelihood Loss* and the *Mean Squared Error*, respectively.

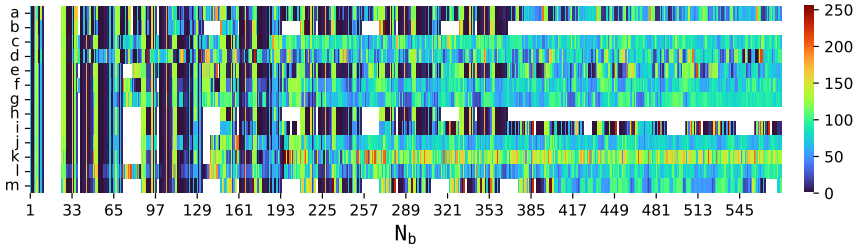
Implementation Details. We have adapted the implementation of FSL approaches initially designed to work with image



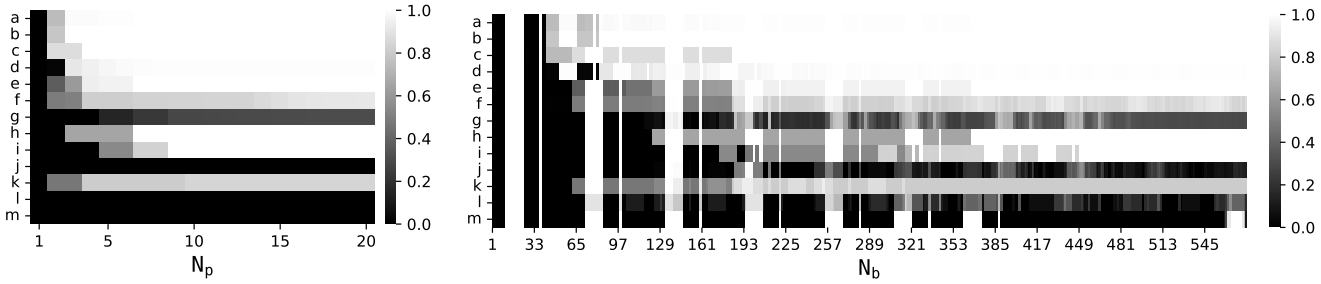
(a) Packet Length (PL) [B] (b) Packet Direction (DIR) (c) Inter-arrival Time (IAT) [ms]



(d) TCP Receive Window (WIN) [B] (e) TCP Flags (FLG) [B] (f) Time To Live (TTL) [# of hops]



(g) NET [B]



(h) Padding ratio for PSQ input (i) Padding/obfuscation ratio for NET input

Figure A.14: Characterization of the biflows in IoT-23 according to the 4 considered informative features of PSQ with the addition of FLG and TTL, and to the bytes in NET (from (a) to (g)). Values are averaged ignoring padding values and, in the case of IAT, transformed with the natural logarithm ($log(1 + x)$). Figures (h) and (i) provide information regarding the average amount of padding (and obfuscation for NET) for each class (in this case, "0" indicates no padding, while "1" indicates its presence).

Label encoding:

a = PartOfAHorizontalPortScan	b = Okiru	c = Benign	d = DDoS	e = C&C-HeartBeat
f = C&C	g = Attack	h = C&C-PartOfAHorizontalPortScan	i = C&C-HeartBeat-Attack	j = C&C-FileDownload
k = C&C-Torii	l = FileDownload	m = C&C-HeartBeat-FileDownload		

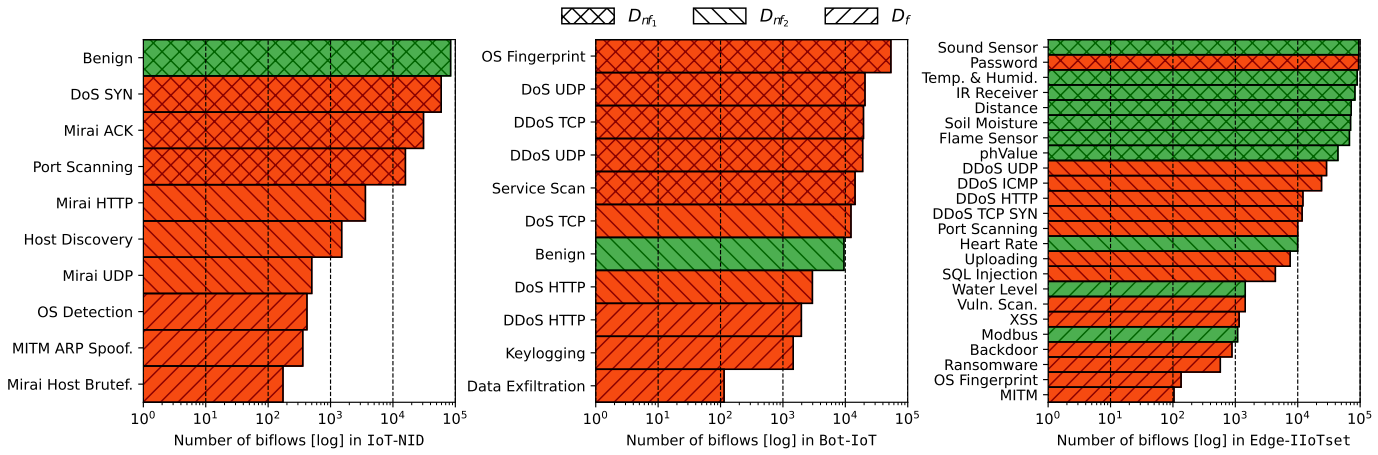


Figure B.15: Number of per-class biflows (in log scale) in `IoT-NID` (left), `Bot-IoT` (middle), and `Edge-IIoTset` (right). Bars are colored in green for benign/legitimate traffic, whereas bars in red for malicious traffic.

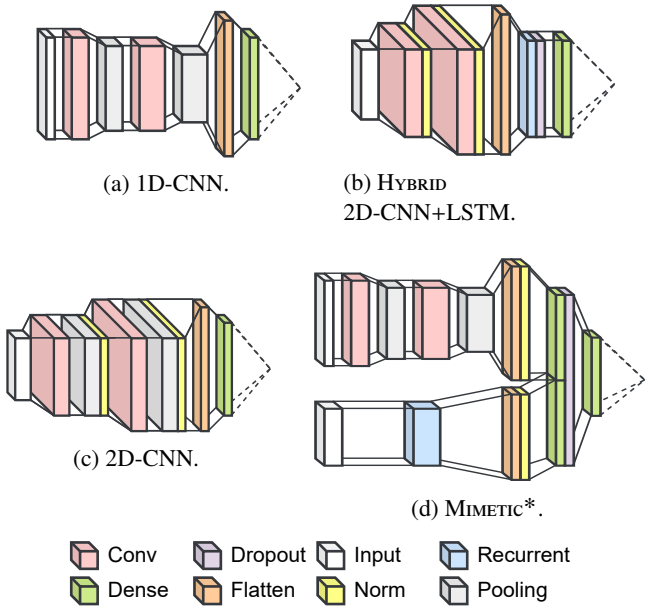


Figure C.16: Embedding functions used to extract features by reducing input traffic data into lower-dimensional space: (a) 1D-CNN [2], (b) HYBRID [3], (c) 2D-CNN [3], and (d) MIMETIC* [4]. Each embedding function uses a specific group of input data according to their internal structure: (a) `NET`, (b and c) `PSQ`, and (d) `NET`+ `PSQ`.

data, whose code is freely available. Most of the approaches are included in the *learn2learn* framework [6] while missing ones (i.e. `MatchingNet` and `RelationNet`) are derived from established GitHub repositories [5, 7]. `Scratch`, Fine-Tuning, and Freezing are self-built. All the implementations are based on *PyTorch/PyTorch Lightning* and have been significantly adapted and harmonized to work together within a common umbrella framework for attack-traffic classification.

Appendix E. Non-Few-Shot Training Procedures

With the goal of defining non-few-shot baselines to validate the necessity for FSL solutions, this appendix aims at deepening how we adopt non-few-shot classifiers when having only few samples available. Firstly, we recall the (preliminary experimentally-validated) assumption that Machine Learning and Deep Learning models built by exploiting few-available data for training are very prone to introduce *overfitting*. To mitigate this issue, FSL approaches propose to augment the available knowledge by exploiting non-few classes, thus obtaining a model that has a better generalization capability.

Following this direction, a naïve way to reduce the overfitting is adopting a **training procedure** consisting of feeding Machine/Deep Learning models by *mixing samples of non-few and few-shot classes*. More specifically, we leverage all the samples of non-few classes and K samples of each few-shot class. This procedure allows us to train a model capable of classifying both non-few and few-shot classes; however, it clearly suffers from a *huge class-imbalance issue*.

Because we do not aim at classifying non-few classes, a mitigation strategy to reduce the induced class imbalance is leveraging an **inference procedure** based on *masking the scores associated with non-few classes* and taking only the scores for samples of few-shot ones. In this way, the magnitude imbalance in weights that exists between non-few and few-shot classes is removed. The adoption of these procedures results in a model that (*i*) is capable of classifying few-shot classes, (*ii*) is less prone to overfitting, and (*iii*) significantly reduces the interference of non-few classes exploited during training.

A second approach to tackle the class imbalance problem involves applying **oversampling** (via re-sampling or synthetic instance generation techniques) to the dataset [8, 9]. In addition to the masking technique mentioned earlier, we train a Machine Learning baseline with a training set encompassing: (*i*) the samples of the *non-few classes*, (*ii*) the K samples of the few classes augmented up to obtain $K \times G$ (synthetic) samples—i.e. $N \times K \times G$, with $G = 1000$. The model trained in this way

is then evaluated solely on C_f. We chose two well-known techniques to perform augmentation: *Random Oversampling* and *Synthetic Minority Over-sampling Technique (SMOTE)* [10].

We underline that classifiers trained on synthetic data may face challenges when applied in real-world contexts. This is exacerbated with severely limited data available (e.g., $K = 5$), where samples generated from such a small set may not effectively represent the diversity of a real-world population. On the other hand, FSL approaches do not rely solely on those few samples, but aim to exploit prior knowledge to build sufficiently general knowledge to classify new classes with few samples.

Appendix References

[1] A. Nascita, F. Cerasuolo, D. Di Monda, J. Garcia, A. Montieri, and A. Pescapè. Machine and Deep Learning Approaches for IoT Attack Classification. In *IEEE INFOCOM*, 05 2022.

[2] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang. End-to-end encrypted Traffic Classification with one-dimensional convolution neural networks. In *IEEE ISI'17*, 2017.

[3] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE Access*, 5, 2017.

[4] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè. MIMETIC: Mobile encrypted traffic classification using multimodal deep learning. *Elsevier Computer Networks*, 165, 2019.

[5] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[6] S. M. Arnold, P. Mahajan, D. Datta, I. Bunner, and K. S. Zarkias. learn2learn: A library for meta-learning research. *arXiv preprint arXiv:2008.12284*, 2020.

[7] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[8] A. Tesfahun and D. L. Bhaskari. Intrusion detection using random forests classifier with smote and feature reduction. In *2013 International conference on cloud & ubiquitous computing & emerging technologies*. IEEE, 2013.

[9] T. Al-Shehari and R. A. Alsowail. Random resampling algorithms for addressing the imbalanced dataset classes in insider threat detection. *International Journal of Information Security*, 22(3), 2023.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 2002.