Received 3 August 2024; revised 18 September 2024; accepted 14 October 2024. Date of publication 16 October 2024; date of current version 30 October 2024.
Digital Object Identifier 10.1109/OJCOMS.2024.3481895

Few-Shot Class-Incremental Learning for Network Intrusion Detection Systems

DAVIDE DI MONDA[1,2], ANTONIO MONTIERI[1], VALERIO PERSICO[1], PASQUALE VORIA[1], MATTEO DE IESO[1], AND ANTONIO PESCAPÈ[1] (Senior Member, IEEE)

[1]Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione, University of Napoli Federico II, 80125 Naples, Italy
[2]IMT School for Advanced Studies, 55100 Lucca, Italy

CORRESPONDING AUTHOR: D. DI MONDA (e-mail: davide.dimonda@unina.it)

This work was supported in part by the Italian Research Program "PON *Ricerca e Innovazione* 2014–2020 (PON R&I) REACT-EU–Asse IV–Azione IV.4"; in part by the PNRR ICSC National Research Centre for High Performance Computing, Big Data and Quantum Computing through the NRRP MUR Program funded by the NextGenerationEU under Grant CN00000013; and in part by the MUR PRIN 2022 Program (D.D.104—02/02/2022) through the Project xInternet funded by the NextGenerationEU.

ABSTRACT In today's digital landscape, critical services are increasingly dependent on network connectivity, thus cybersecurity has become paramount. Indeed, the constant escalation of cyberattacks, including zero-day exploits, poses a significant threat. While Network Intrusion Detection Systems (NIDSs) leveraging machine-learning and deep-learning models have proven effective in recent studies, they encounter limitations such as the need for abundant samples of malicious traffic and full retraining upon encountering new attacks. These limitations hinder their adaptability in real-world scenarios. To address these challenges, we design a novel NIDS capable of promptly adapting to classify new attacks and provide timely predictions. Our proposal for attack-traffic classification adopts *Few-Shot Class-Incremental Learning* (FSCIL) and is based on the *Rethinking Few-Shot* (RFS) approach, which we experimentally prove to overcome other FSCIL state-of-the-art alternatives based on either *meta-learning* or *transfer learning*. We evaluate the proposed NIDS across a wide array of cyberattacks whose traffic is collected in recent publicly available datasets to demonstrate its robustness across diverse network-attack scenarios, including malicious activities in an Internet-of-Things context and cyberattacks targeting servers. We validate various design choices as well, involving the number of traffic samples per attack available, the impact of the features used to represent the traffic objects, and the time to deliver the classification verdict. Experimental results witness that our proposed NIDS effectively retains previously acquired knowledge (with over 94% F1-score) while adapting to new attacks with only few samples available (with over 98% F1-score). Thus, it outperforms non-FSCIL state of the art in terms of classification effectiveness and adaptation time. Moreover, our NIDS exhibits high performance even with traffic collected within short time frames, achieving 95% F1-score while reducing the time-to-insight. Finally, we identify possible limitations likely arising in specific application contexts and envision promising research avenues to mitigate them.

INDEX TERMS Attack-traffic classification, deep learning, few-shot class-incremental learning, network intrusion detection system, network security.

I. INTRODUCTION

In recent years, cybersecurity has rapidly emerged as a major concern for society at large, encompassing individuals, businesses, and governments on a global scale. As the reliance on technology grows, so does the potential for disruption and harm caused by cyber threats. The financial impact of these threats is staggeringly high, with public administration and healthcare among the most vulnerable sectors, potentially facing significant financial losses [1]. A recent survey revealed that 37% of companies that have experienced a cyberattack have lost more than \$100*k* [2]. Among the cyber threats, *Distributed Denial of Service (DDoS)* attacks have become particularly notorious. DDoS attacks involve overwhelming a machine or network resources with a high volume of network traffic, impairing its availability to the intended users. The number and variety of such attacks are growing rapidly. *Cloudflare* reports a 111% increase from 2021 to 2022 of DDoS attacks based on HTTP and 67% of Ransom DDoS [3]. An additional layer of complexity is added by the appearance of novel DDoS attacks capable of exploiting new vulnerabilities. This is exemplified by the record-breaking DDoS attack that leveraged the HTTP/2 Rapid Reset vulnerability [4].

Given the escalating threat landscape, there is an urgent need for effective countermeasures. *Network Intrusion Detection Systems (NIDSs)* aim to classify several types of network attack traffic and distinguish them from legitimate traffic, providing a line of defense against cyber threats. Traffic classification techniques have evolved significantly over the years. Initially reliant on *port-IP matching* and *deep packet inspection*—the former prone to evasion by sophisticated attacks and the latter challenged by encryption—NIDSs have seen the rise of techniques based on *Machine Learning (ML)* and *Deep Learning (DL)* [5]. While ML offers a powerful tool for traffic classification, it requires domain experts to craft effective features capable of modeling malicious traffic, which can be time-consuming and hardly automatable. On the other hand, end-to-end DL directly capitalizes on raw traffic data, hence eliminating the need for manual feature engineering. However, DL relies on a large amount of data to provide satisfactory results [6]. Collecting datasets for network traffic classification is a tough and expensive task. Indeed, the constant evolution of legitimate and malicious traffic patterns poses a significant challenge in maintaining accurate and up-to-date datasets. Moreover, gathering and labeling network traffic is resource-intensive, demanding both hardware infrastructure and specialized personnel.

Continuously and efficiently updating ML/DL-based NIDSs to incorporate the knowledge about new attacks collides with two kinds of practical constraints: *(i)* data on novel attacks are often limited, resulting in seldom having enough samples of such attacks to feed the training of the NIDS; *(ii)* the process for updating the NIDS cannot rely on costly retraining from scratch while also needing to retain the knowledge already incorporated into the classification model. To address these critical challenges, we propose a NIDS based on *Few-Shot Class-Incremental Learning* (`FSCIL`). `FSCIL` offers a solution by learning from a limited number of samples and incrementally updating the data-driven model at the core of the NIDS as new attacks emerge. More intuitively, `FSCIL` tries to resemble human-level general intelligence, empowering the NIDS to effectively generalize from limited labeled traffic samples (viz. few shots)—and thus excelling in detecting new, unseen attacks—without losing the NIDS' capability in dealing with known attacks (viz. class-incremental learning). In other words, `FSCIL` allows for rapid adaptation to new threats while preserving the knowledge of previous benign and malicious traffic patterns, thus avoiding the so-called *catastrophic forgetting* issue [7].

Based on such considerations, in this paper, we provide the contributions as outlined below.

- We design a *data-driven NIDS that can be swiftly updated to classify novel attacks not seen during the training phase* as soon as a limited set of samples for such attacks is available. With this goal in mind, *the NIDS is based on the `FSCIL` paradigm*. This choice allows for updating the core of the NIDS with minimal data and allows it to quickly incorporate new attack knowledge, reducing the time to make it capable of detecting new threats. More specifically, we validate the capability of the proposed NIDS to *adapt to various scenarios*—e.g., cyberattacks targeting servers and malicious activities in an Internet of Things (IoT) context—when facing *novel DDoS attacks not seen during the training phase and characterized by an extremely limited number of samples*. To this aim, we exploit two real attack-traffic datasets, namely `CSE-CIC-IDS2018` and `IoT-NID`.
- We adopt *different `FSCIL` approaches* based on either *meta-learning* or *transfer learning*, and we choose the best-performing *Rethinking Few-Shot (`RFS`)* as the central component of our NIDS. We also compare our proposal against *the non-`FSCIL` state of the art*, namely a NIDS trained *from scratch* each time a new class of attack becomes available, without constraints on limited data.
- We carefully investigate various design choices for our proposed NIDS, namely: (*i*) *the number of samples* needed to effectively classify new attacks while retaining acquired knowledge; (*ii*) the usage of *effective and robust features* used to represent the traffic objects (i.e., the input data) by identifying which features contribute the most to the NIDS' performance; (*iii*) the setting of a *temporal threshold* within which the classification verdict has to be provided to enable a *trade-off between the time-to-insight and classification effectiveness* in scenarios requiring rapid detection (e.g., when facing slow flooding attacks).

The paper is structured as follows. Section II provides background information and positions this study against related work. In Section III, we introduce the proposed NIDS based on `FSCIL`, along with the approaches used for attack-traffic classification. Section IV deepens the specific configurations employed, concerning datasets, `FSCIL` approaches, and other implementation details. The experimental results and corresponding take-home messages are presented in Section V. Finally, Section VI concludes the paper and outlines future directions. For the benefit of the reader, Table 1 summarizes the notations used throughout the manuscript.

TABLE 1. List of the notations used in the manuscript.

Notation	Description
$\mathcal{C}_{old}$	Classes available during the initial training.
$\mathcal{C}_{new}$	Classes not seen during the initial training and characterized by few available samples.
$\mathcal{C}_{all}$	Overall set of classes (i.e. $\mathcal{C}_{all} = \mathcal{C}_{old} \cup \mathcal{C}_{new}$).
$\mathcal{D}_{old}$	Dataset containing the samples of $\mathcal{C}_{old}$.
$\mathcal{D}_{new}$	Dataset containing the samples of $\mathcal{C}_{new}$.
$\mathcal{D}_{all}$	Dataset containing the samples of both $\mathcal{C}_{old}$ and $\mathcal{C}_{new}$.
N	Number of classes sampled for each task (i.e. ways).
$K_s = K$	Number of samples per class (i.e. shots) for the support set.
K_q	Number of samples per class for the query set.
$\mathcal{T}_0$	Pre-training phase of transfer-learning approaches.
$\mathcal{T}_1$	Fine-tuning phase of transfer-learning approaches.
N_p	Number of packets in a biflow used as model input.
T_s	Temporal threshold limiting per-biflow packets considered.
θ	Embedding function.
ϕ	(Attack-)traffic classifier.
$\boldsymbol{x}$	Input of the embedding function.
y	Actual label of a biflow.
$\hat{\boldsymbol{y}}$	Probability vector over classes (i.e. soft values).
$\boldsymbol{v}$	Feature vector.
$\mathcal{L}$	Loss function.

II. BACKGROUND AND RELATED WORK

This section provides an overview of the key concepts of meta-learning and transfer learning (Section II-A). Then, it presents the most related state-of-the-art works dealing with *Few-Shot Learning* (FSL), *Class-Incremental Learning* (CIL), and the combination of both, i.e., FSCIL (Section II-B). Lastly, it outlines the novelty of the present manuscript by establishing its position within the current literature (Section II-C).

A. BACKGROUND

In this work, we leverage FSCIL approaches by starting from "traditional" FSL ones. The latter approaches exploit *non-few knowledge* of existing classes—where with *classes* we refer to different types of malicious traffic (i.e., different attacks) as well as to the benign traffic—to construct a model aimed at generalizing to new learning tasks with only few samples available. Specifically, by building on *meta-learning* and *transfer learning*, we incorporate the constraint of *incremental learning* from the CIL paradigm. This implies that the NIDS must retain the non-few knowledge previously acquired (i.e., to avoid catastrophic forgetting) while also learning to identify new attack classes for which only limited data are collected.

Meta-Learning Fundamentals. *Meta-learning* aims to build models with high generalization ability realizing a *learning-to-learn* paradigm. Specifically, the training process is conducted across a wide range of learning tasks to achieve optimal performance in a broad spectrum of scenarios, including potentially unseen tasks. In other words, the goal is to distill meta-knowledge that can enhance performance on new tasks, even in the presence of few samples for these latter. Indeed, meta-learning is particularly used in the context of FSL through the application of *episodic-learning*. Such a technique involves organizing the training process to mimic the operational conditions of FSL through a large number of classification tasks (aka episodes) with limited available data. Since the goal is to build models that can effectively generalize on novel tasks, the learning phases (i.e., training, validation, and testing) are typically conducted on data with distinct label spaces. Consequently, the initial step involves splitting the dataset so that each partition encompasses different sets of classes. Each partition is then utilized for a specific phase: *meta-training*, *meta-validation*, and *meta-testing*.

Like standard training, meta-training is performed in multiple epochs. During an epoch, a series of independent episodes/tasks are drawn from the partition designated for meta-training. Similar procedures are followed for meta-validation and meta-testing, where a fixed number of episodes/tasks are sampled. The final score used to evaluate the meta-learning model's performance is the average of the per-episode scores.

More formally, the task sampling procedure is determined by the triplet of values $\langle N, K_s, K_q \rangle$, where: *(i)* N represents the number of classes to be selected for each task; *(ii)* K_s denotes the number of samples per class that are allocated for training the model: the resulting *support set* has $N \times K_s$ samples in total; *(iii)* K_q stands for the number of samples per class that are designated for evaluating the model error rate: the resulting *query set* has $N \times K_q$ samples in total. The outcome of this procedure is referred to as an *N-way K-shot task*, where $K = K_s$. We refer to our previous work [8] for a more comprehensive overview (including the pseudocode of the learning procedure) on meta-learning.

Transfer-Learning Fundamentals. The objective of *transfer learning* is to capitalize on the general knowledge acquired from an extensive dataset in the *pre-training* phase ($\mathcal{T}_0$). Specifically, the model optimized in $\mathcal{T}_0$—aka the *base model*—is then specialized for a new target classification task, during the so-called *fine-tuning* phase ($\mathcal{T}_1$). As in the meta-learning paradigm, the label space is not shared across the two phases (i.e., $\mathcal{T}_0$ and $\mathcal{T}_1$ have a disjoint label space) when dealing with FSL. Conversely, in the case of FSCIL, the label space can partially overlap.

B. RELATED WORK

The literature on automated intrusion detection techniques is flourishing, with a strong focus on designing DL-based NIDS for *attack-traffic classification*. These works primarily utilize two approaches: anomaly detection and misuse detection. The former models the anomalies as deviations from the profile of benign traffic and thus exploits semi-supervised [22] or unsupervised [23], [24] approaches to detect malicious (viz. anomalous) traffic samples. In contrast, misuse detection directly identifies known attack patterns by training on both benign and malicious samples; it thus can identify specific threats via single-modal [25] or multimodal [26] supervised approaches.

TABLE 2. Related studies focusing on attack-traffic classification using few-shot learning, class-incremental learning, and the combination of both. Papers are listed chronologically by publication year. The final row provides an overview of the present work. The meaning of acronyms is provided at the bottom of the table.

Paper	FSL	CIL	Approach	Dataset	Raw	Early	Input Data
Constantinides et al. [9], 2019	○	●	SVM+SOINN	NSL-KDD	○	○	Flow based-statistics
Huang et al. [10], 2020	●	○	MN+Gates	NSL-KDD	○	○	Flow based-statistics
Ouyang et al. [11], 2021	●	○	PN	SCADA dataset	○	○	Flow based-statistics
Rong et al. [12], 2021	●	○	PN	MCFP, USTC-TFC, CICInvesAndMal2019, BEN-1, self-built benign	●	●	N bytes of a biflow
Wang et al. [13], 2021	●	●	MAML+MM	NSL-KDD	○	○	Flow based-statistics
Liang et al. [14], 2021	●	○	RN	NSL-KDD, CIC-IDS2017	○	○	Flow based-statistics
Data and Aritsugi [15], 2021	○	●	T-DFNN	CIC-IDS2017	○	○	Flow based-statistics
Data and Aritsugi [16], 2022	○	●	AB-HT	CIC-IDS2017	○	○	Flow based-statistics
Lu et al. [17], 2023	●	○	MAML	FSIDS-IOT combining: NSL-KDD, UNSW-NB15, CIC-IDS2017, CSE-CIC-IDS2018, CIC-DDoS2019	○	○	Flow based-statistics
Song et al. [18], 2023	○	●	RNN-MG	CIC-IDS2017, ISCXVPN2016	●	●	Per-packet header fields
Pawlicki et al. [19], 2023	●	○	SN	CSE-CIC-IDS2018	○	○	Flow based-statistics
Cerasuolo et al. [20], 2023	○	●	FT w/ & w/o Mem, LwF, iCaRL+, BiC	IoT-23	●	●	Per-packet header fields
Di Monda et al. [21], 2023	●	○	MN, PN, MON, FT, FZ, BL++, RFS, NM	IoT-23	●	●	Per-packet header fields
Bovenzi et al. [8], 2024	●	○	MN, PN, RN, MON, ANIL, MAML, FT, FZ	IoT-23, IoT-NID, Edge-IIoT, Bot-IoT	●	●	L4/L2 unbiased payload Per-packet header fields
This work	●	●	MN, PN, MON, FT, FZ, RFS	CSE-CIC-IDS2018, IoT-NID	●	●	Per-packet header fields

FSL—Few-Shot Learning; **CIL**—Class-Incremental Learning; **Approach**—Proposed FSL, CIL or FSCIL approach; **Raw**—Raw input data; **Early**—Early attack-traffic classification.
Acronyms: Support Vector Machine (SVM), Self-Organizing and Incremental Neural Network (SOINN) Hoeffding Tree-based AdaBoost (AB-HT), Model Merging (MM), Tree Deep Feedforward Neural Networks (T-DFNN), Recurrent Neural Network based on Model Growth (RNN-MG) Siamese Network (SN), Matching Networks (MN), Prototypical Networks (PN), Relation Network (RN), MetaOptNet (MON), Model-Agnostic Meta-Learning (MAML), Almost No Inner Loop (ANIL), Fine-Tuning (FT), Freezing (FZ), Baseline++ (BL++), Rethinking Few-Shot (RFS), Negative Margin (NM), Memory (Mem), Learning Without Forgetting (LwF), Incremental Classifier and Representation Learning (iCaRL+), Bias Correction (BiC).
"+" symbol indicates hybrid architectures; ● present, ○ lacking.

Recent years have witnessed a noticeable surge in the research interest in cybersecurity, as highlighted by a +200% rise in the use of relevant keywords related to this domain [27], along with a trend towards advanced solutions for addressing the challenges inherent to DL (and ML)-based attack-traffic classification. Within this domain, specific issues have gained substantial attention, influencing the direction of research endeavors:

i) *New attack classes handling:* this issue entails developing approaches capable of classifying threats previously unseen during the training phase of ML/DL methods with the fewest possible samples, aiming to promptly enforce countermeasures.

ii) *Class imbalance management:* intrusion detection datasets are often characterized by a skewed class distribution, since certain attack types generate a high number of samples (e.g., flooding, probing), while others yield fewer instances (e.g., data theft, *command-and-control* connections).

iii) *Knowledge incorporation for continuous learning:* existing approaches need to constantly update their knowledge of both emerging attacks and normal traffic patterns; this process must integrate new information without erasing previously acquired knowledge, all while being time-efficient to avoid retraining from scratch with old and new data.

Addressing the first and second challenges often involves the application of FSL techniques, while the third challenge finds solutions in CIL. Currently, to the best of our knowledge, the combination of these two paradigms (i.e., FSCIL) emerges as relatively under-explored in the attack-traffic classification literature despite its natural fit.

Table 2 collects and categorizes the most relevant papers that leverage FSL, CIL, and FSCIL for attack-traffic classification. The majority of these studies were published between 2020 and 2023 (with one exception [9] published in 2019), demonstrating a growing interest in these topics.

The second and third columns of Table 2 indicate whether the works apply **FSL and/or CIL**. Specifically, eight works [8], [10], [11], [12], [14], [17], [19], [21] employ FSL approaches, five [9], [15], [16], [18], [20] apply CIL, and only one [13] proposes a solution combining both (viz. FSCIL). This underscores the *lack of analyses regarding methodologies based on* FSCIL *for attack-traffic classification* in related literature, and thus the absence of an effective solution that can simultaneously tackle the issues mentioned at the beginning of this section.

In Table 2, we also report the **specific FSL, CIL, or FSCIL approaches** employed in each work. The investigation of the papers dealing with FSL reveals that most of them resort to "traditional" FSL meta-learning approaches originally introduced in the field of computer vision in 2016-2018, such as *Matching Networks* [28], *Prototypical Networks* [29], *Relation Network* [30], and *Model-Agnostic Meta-Learning (MAML)* [31]. These methods are often employed with only minor modifications from the original versions. Notably, our previous works [8], [21] stand out as the only ones that compare meta-learning approaches with those based on transfer learning. Differently from the rest of the state of the art, Pawlicki et al. [19] solely leverage a *Siamese neural network* as a few-shot learner for network intrusion detection. The Siamese neural network embeds traffic samples by taking pairs of samples from all the classes to generate feature vectors that can be easily compared; namely, it is capable of discerning between two classes even when a limited number of samples from a novel class are available. *Matching Networks* are employed in [8], [21] along with other FSL approaches for IoT-attack-traffic classification, and in [12] as a baseline. Huang et al. [10] enhance the original Matching Networks by incorporating a gating technique. These additional gates act as soft classifiers that detect whether a test sample belongs to a known or unknown attack type via a sigmoid function. Ouyang et al. [11] employ *Prototypical Networks* within the context of cyberattacks against SCADA networks by orchestrating principal component analysis for reducing feature dimension, one-hot encoding, and feature embedding via a 2D-convolutional neural network. Prototypical Networks are also utilized as the reference FSL approach in [12]—fed with traffic data converted into grayscale images—and in [8], [21] as aforementioned. Liang et al. [14] propose an adaptation of the *Relation Network* for attack-traffic classification in distributed IoT systems; specifically, reconstructed feature embeddings are added to the original Relation Network aiming at intra-class and inter-class distance optimization when dealing with an imbalanced (viz. few-shot) attack dataset. Relation Network is among the FSL approaches considered in [8] in an IoT-attack scenario. In the network security context, *MAML* (or its updated version *ANIL* [32]) has been used in [8], [12], [17] for FSL and in [13] for FSCIL. Particularly, the latter work enables MAML for FSCIL-based attack-traffic classification via a model expansion technique that combines the logits of old-class and new-class classifiers. *MetaOptNet* [33] is a more recent promising approach derived from computer vision that has been explored within the context of IoT-attack detection via FSL only in our previous researches [8], [21], demonstrating good performance compared to the other meta-learning approaches. In [21], we also adopt various FSL approaches based on the *transfer-learning* paradigm, mainly differing on the operations performed during the fine-tuning phase. These include (*i*) *Fine-Tuning*, (*ii*) *Freezing* (both also used in [8]), and three recent approaches adapted from the computer vision domain: (*iii*) *Baseline++* [34], (*iv*) *Rethinking Few-Shot* [35], and (*v*) *Negative Margin* [36].

Concerning the works that face attack-traffic classification using incremental learning, we can notice a less tight connection with the approaches originally proposed in computer vision. Indeed, such works commonly propose CIL approaches purposely devised and tailored for the considered task. Cerasuolo et al. [20] offer a comparison of different CIL techniques for IoT-attack-traffic classification: (*i*) *Learning Without Forgetting* [37] and *BiC* [38] borrowed from computer vision, (*ii*) *iCaRL+* [39] expressly proposed for encrypted traffic classification, and (*iii*) two versions of *Fine-Tuning* (i.e., with and without memory storing the samples of "old" classes). Two approaches based on ML have been proposed in [9], [16]. Constantinides et al. [9] describe a method based on a Support Vector Machine (SVM) equipped with multiple pairs of self-organizing and incremental neural networks, which use an incremental clustering method that can handle supervised data. Similarly, Data and Aritsugi [16] propose an ensemble incremental learning algorithm, where the Hoeffding Trees are exploited as the stumps of the incremental AdaBoost model. The same authors leverage also a CIL approach based on DL that expands the model into a tree-structured architecture allowing it to effectively integrate new attack-traffic knowledge [15]. DL is also adopted in the traffic classification solutions described in [18], whose authors utilize a recurrent neural network that expands its architecture when new classes are available (viz. *model growth*).

Investigating the attack-traffic **datasets** employed in the state of the art and comparing them with those used in the present work, Pawlicki et al. [19] and Bovenzi et al. [8] employ CSE-CIC-IDS2018 and IoT-NID, respectively; Lu et al. [17] only preprocess CSE-CIC-IDS2018 to construct their FSIDS-IOT. Conversely, the most common datasets in the related literature are the older CIC-IDS2017 [14], [15], [16], [17], [18] and NSL-KDD [9], [10], [13], [14], [17]. The latter particularly, despite being considered a general benchmark for attack-traffic classification, is over two decades old and no longer represents the current profiles of both benign and attack traffic. The choice of other datasets depends heavily on the use case considered in the related work. For instance, intrusions in SCADA networks [11], Android malware (e.g., CICInvesAndMal2019) [12], and attacks against IoT devices (e.g., IoT-23, Bot-IoT) [8], [20], [21].

Another essential aspect to take into account is the effective utilization of **raw input data** enabling a paramount advantage of DL classifiers, that is the automatic extraction of knowledge (in the form of highly expressive features) from attack-traffic data. In Table 2, we explicitly flag the studies that effectively exploit such **input data** in the form of per-network-flow bytes [8], [12] or per-packet informative fields (e.g., payload length, time-to-live, and TCP flags) [8], [18], [20], [21]. Conversely, the rest of the works adopt pre-processed features, such as flow-based statistics derived from the whole set of packet/payload lengths or inter-arrival times, which are typically obtained from pre-processed datasets. When combined with DL classifiers, the latter choice undermines their aforementioned peculiarity, resulting in suboptimal solutions.

Table 2 points out that the works exploiting raw input data can also deal with **early** attack-traffic classification [8], [12], [18], [20], [21]. Indeed, such studies typically focus on the initial bytes/packets of a given traffic object, being able to promptly detect an attack (ideally when it is still running). On the other hand, the other works conduct a "post-mortem" attack-traffic classification, starting only after the malicious activity has ended.

C. POSITIONING

The last row of Table 2 summarizes the main aspects of our study. This helps us to position it against the related work, underlining its novelty.

Among the surveyed literature, only Wang et al. [13] propose an FSCIL approach for attack-traffic classification. However, it suffers from a number of issues that hinder its reproducibility—which is why we do not consider this proposal as a baseline in our experimentation. Specifically, (*i*) the embedding function is not clearly described and thus it is not reproducible; (*ii*) the proposed approach is not naturally compatible with our input designed for early traffic classification as it is intended to work with flow-based features in a tabular format (thus suited only for "post-mortem" intrusion detection); and (*iii*) it employs the model growth paradigm, which is not scalable as the number of classes (i.e., new attacks to be identified) increases.

Compared to our most related previous works [8], [21], in this study, we introduce stricter real-world constraints that led to the need to adopt incremental learning besides considering a few-shot scenario. This entails the design of a NIDS that effectively leverages *FSCIL approaches*. Also, differently from related literature commonly focusing only on a single paradigm, we evaluate a *wide range of approaches for FSCIL attack-traffic classification based on meta-learning and transfer-learning paradigms*, as the core for designing our adaptive NIDS exploiting RFS.

As opposed to related literature focusing on *post-mortem* (viz. offline) attack-traffic classification [9], [10], [11], [13], [14], [15], [16], [17], [19] and better-targeted than the works performing *early* attack-traffic classification [8], [12], [18], [20], [21], we further stress the latter capability by explicitly taking into account the time required to collect packets before delivering the classification verdict, and its trade-off with the efficacy.

Finally, we assess the *generalization capability* and *broad applicability* of the designed NIDS in different network contexts. To this end, we show the performance attained on two *recent* and *publicly-available* datasets, namely CSE-CIC-IDS2018 and IoT-NID, unlike studies using outdated traffic data [9], [10], [13], [14], [15], [16], [17], [18]. Also, to the best of our knowledge, we are the first to jointly exploit such datasets to *cross-validate our NIDS* by operating it on the attack traffic belonging to the dataset not used for training.

III. DESIGNING CLASS-INCREMENTAL NIDS BASED ON FSCIL

This section first formalizes the problem statement and outlines the challenges addressed in the present work (Section III-A). Then, we introduce the foundational blocks of the comprehensive workflow for the proposed class-incremental NIDS that uses only few samples of new attack classes (Section III-B). Finally, we describe the FSCIL approaches exploited for the NIDS design and how we have adopted them starting from standard FSL methods (Section III-C).

A. PROBLEM STATEMENT

The state of the art highlights the effectiveness of DL-based NIDS for attack-traffic classification [26], [40], [41], especially when large amounts of data are available for model training. However, attacks rapidly evolve in a real-world cybersecurity scenario, often leading to the emergence of attacks different from those available during the training of the NIDS. This constant evolution of the threat landscape requires the update of the NIDS to deal with previously unknown attacks.

Unfortunately, training a traffic classifier using classic DL approaches every time new attack class samples become available is a time-consuming process. Until this training is complete, the NIDS cannot mitigate such unseen attacks. This process is hindered by a number of interrelated practical issues: *(i)* training a classification model *from scratch* every time the traffic of a previously unseen attack becomes available is a resource-demanding task in terms of both time and computing power; *(ii)* retraining the model with the full previous knowledge along with a limited number of samples of new attacks generates significant *class imbalance*, thereby limiting the NIDS ability to detect these new attacks; *(iii)* capturing sufficient samples of new attack traffic takes time, further extending the period during which the NIDS is *ineffective*. To overcome such issues, in this study, *we design a NIDS based on the FSCIL paradigm*. We evaluate its effectiveness in the mentioned scenario, i.e., when the classification model has to adapt to face the emergence of new attacks for which *only few samples are available*.

Formally, we refer to the traffic classes available during the initial training as $\mathcal{C}_{old}$ (including both malicious and benign traffic), and to those not seen during this training as $\mathcal{C}_{new}$. We assume that large amounts of samples belonging to $\mathcal{C}_{old}$ are initially available. On the other hand, later in the process only a limited number of samples (e.g., $K \leq$

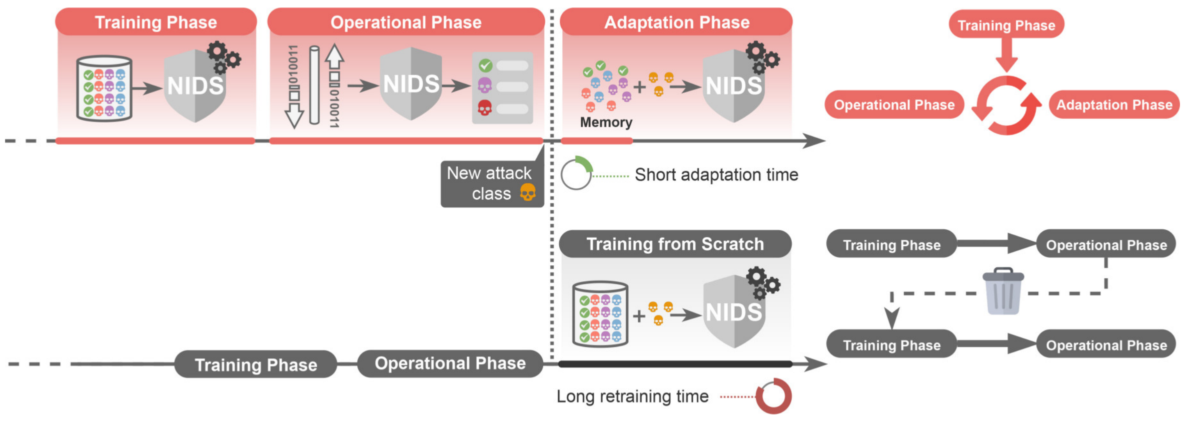


FIGURE 1. Comparison between a NIDS based on FSCIL (top, in red) and non-FSCIL alternative (bottom, in gray). The first two phases—i.e., (initial) *training* and *operational*—are common in both cases. The NIDS is first trained to classify known attacks and benign traffic (i.e., $\mathcal{C}_{old}$) using a large dataset and then it becomes operational. During the operational phase, the availability of attack traffic of a new type (i.e., $\mathcal{C}_{new}$) requires the NIDS to be retrained to detect it. The FSCIL NIDS needs a short *adaptation phase* using few samples of $\mathcal{C}_{new}$ and a small memory of $\mathcal{C}_{old}$. Differently, the non-FSCIL alternative must be *(re)trained from scratch*—hence the old model is discarded—using (sufficient) samples of $\mathcal{C}_{new}$ plus the entire original training dataset (i.e., all samples of $\mathcal{C}_{old}$), resulting in a much longer process.

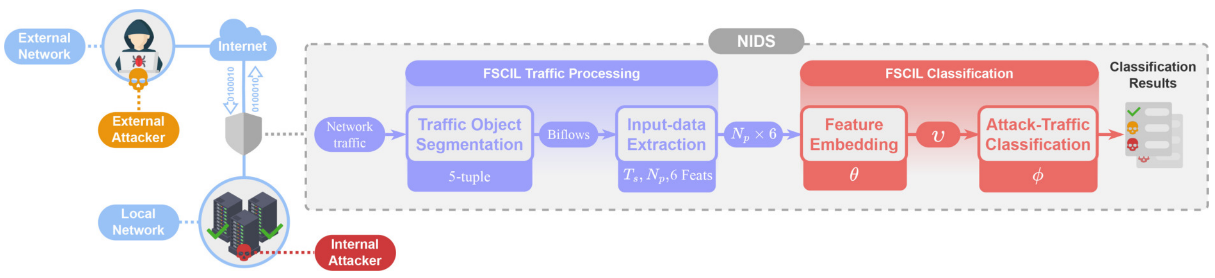


FIGURE 2. Workflow of the FSCIL NIDS. The methodological blocks regarding attack-traffic processing are colored in blue. The methodological blocks constituting the FSCIL classification steps are colored in red. Each block includes the parameters that characterize it along with its output.

10) are available for each class in $\mathcal{C}_{new}$. Figure 1 depicts the process just discussed. The *training phase* involves the initial training of the NIDS using the large dataset containing samples of classes in $\mathcal{C}_{old}$. Once trained, the NIDS is ready for deployment and *operational* use. However, to detect new attacks (i.e., those in $\mathcal{C}_{new}$), the NIDS requires to be updated. Hence, the FSCIL paradigm involves an *adaptation phase*. Such an adaptation phase allows the aforementioned practical issues of *training the model from scratch* to be avoided.

We can summarize the dual objective of the proposed approach as follows:

i) *Integrating knowledge about new, unforeseen attacks* (i.e., $\mathcal{C}_{new}$) into a model that has already been trained on a set of benign traffic and known attacks. This integration must be fast to minimize periods of ineffectiveness and ensure that it does not erase the model's prior knowledge about the training classes (i.e., $\mathcal{C}_{old}$).

ii) *Effectively learning from a small number of samples* belonging to $\mathcal{C}_{new}$. This capability enables the avoidance of time-consuming campaigns to collect attack-traffic data related to $\mathcal{C}_{new}$.

B. FSCIL NIDS WORKFLOW

The workflow of our NIDS based on FSCIL is depicted in Figure 2. Hereinafter, we detail the workflow's four-step process that a NIDS follows to classify the traffic of attacks. The first two steps are *traffic processing* operations (Section III-B1) which involve (*i*) the segmentation of traffic into relevant aggregates and (*ii*) the extraction of key features essential for characterizing potential attack traffic. Such input data are fed to the *FSCIL classification* part of the workflow (Section III-B2), which encompasses (*iii*) the embedding of extracted features via a proper embedding function (responsible for dimensionality and complexity reduction) and (*iv*) the actual attack-traffic classification (depending on the specific FSCIL approach utilized).

1) FSCIL TRAFFIC PROCESSING

The *FSCIL traffic processing* consists of two key operations specific to the attack-traffic classification domain, performed before feeding the training or the operational phase of the NIDS. The first operation involves segmenting the traffic into units being the objects of the attack-traffic classification task. Such traffic objects are processed in the second operation, where relevant features are extracted to characterize benign and malicious traffic samples. These features are the input of the data-driven core of the NIDS.

Traffic Object Segmentation. Network traffic is initially segmented into traffic objects (i.e., aggregates of network packets). Specifically, the packets are aggregated into *bidirectional flows* (*biflows*), with each biflow gathering all

TABLE 3. Per-packet input features extracted for each biflow.

Feature	Description
PL	Number of bytes in the network packet.
DIR	Packet direction: -1 for downstream or 1 for upstream packets.
WIN	Value contained in the window size field of a TCP packet; set to 0 for UDP packets.
IAT	Inter-arrival time: it indicates the time elapsed since the arrival of the previous packet.
TTL	Time-to-Live value in the IP header.
FLG	TCP flags: it is encoded as the base-10 representation of the 8-bit TCP flags field; set to 0 for UDP packets.

packets that have identical 5-tuple attributes (comprising `srcIP`, `dstIP`, `srcPort`, `dstPort`, and `L4Proto`). Note that such traffic segmentation does not take into account the direction of the communication (i.e., whether swapping source and destination endpoints, the packets still belong to the same biflow).

Input-Data Extraction. Each biflow is then processed to extract features crucial for attack-traffic classification. In line with the state of the art [8], [18], [20], [21], only the initial N_p packets of each biflow are considered. Such a choice enables *early traffic classification*, i.e., it allows the classifier to provide a verdict based only on the initial part of the biflow and thus enhances the responsiveness of the NIDS. In line with this goal, the designed NIDS also enforces an additional cut based on a temporal threshold such that only the packets of a biflow that arrive before T_s seconds are taken into account. Enforcing such a cut is particularly important for promptly detecting *slow flooding attacks* (cf. Section V-E). Indeed, these attacks involve the attacker gradually exhausting the limited resources of a victim machine by establishing and maintaining a large number of concurrent connections over an extended period (e.g., the attacker could slowly send partial HTTP headers forcing the server to keep several connections open) [42]. Given such long disrupting consequences, their early detection is paramount. Relying solely on the first packets or bytes—as in the case of related literature [8], [12], [18], [20], [21]—is therefore not sufficient. The attackers can spread packets over time, slowing down the classification process, especially in the case of slow flooding attacks. Particularly, this delay increases the *time-to-insight*, namely the period from when traffic-data collection begins to when actionable insights can be drawn from the model (i.e., based on the NIDS verdict).

After collecting N_p packets of a biflow or once T_s has expired, 6 per-packet features are extracted, as also suggested in [18], [21], [26], [40]. These features include: *(i)* the packet size, *(ii)* the packet direction, *(iii)* the TCP window size, *(iv)* the inter-arrival time, *(v)* the Time-to-Live (TTL), and *(vi)* the TCP flags. Table 3 summarizes the aforementioned input features. At the end of this process, an $N_p \times 6$ input matrix x is obtained for each biflow. If due to the effect of the T_s threshold, fewer than N_p packets are collected, appropriate zero-padding is added to ensure $x \in \mathbb{R}^{N_p \times 6}$. Lastly, x is normalized in the range $[0, 1]$ via a Min-Max function.

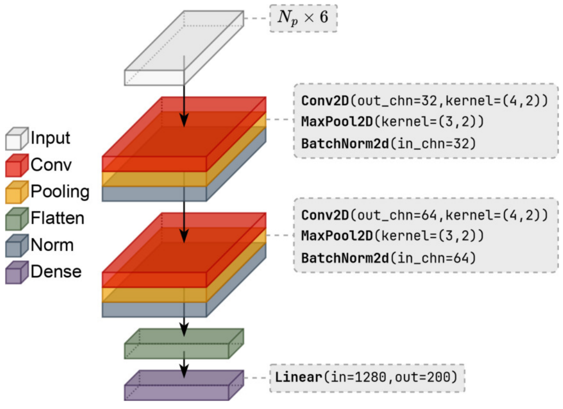


FIGURE 3. Architecture of the embedding function used in this work. Hyperparameter meaning: `in_chn`=number of input channels, `out_chn`=number of output channels, `kernel`=width and height of the kernel as a tuple, `in`=dimension of the input tensor, `out`=dimension of the output tensor. The `in` parameter is dependent from the chosen N_p value (i.e., $N_p = 20$).

2) FSCIL CLASSIFICATION

The `FSCIL` *classification* part of the workflow consists of the steps carried out via the data-driven *embedding function* (θ) and the *attack-traffic classifier* (ϕ) employed in our NIDS.

Feature Embedding. The embedding function $\theta(x)$ performs a transformation that maps high-dimensional input $x \in \mathbb{R}^{N_p \times 6}$ into a richer d-dimensional embedded space while preserving a meaningful structure (such a process is called feature embedding); more formally: $\theta(x) : \mathbb{R}^{N_p \times 6} \to \mathbb{R}^d$ and $\upsilon \in \mathbb{R}^d$ is the resulting feature vector. Concerning the specific embedding function, we adopt a well-known DL architecture originally proposed for IoT-traffic classification in [40] and also exploited for attack-traffic classification [8], [21], [26]. It encompasses two bidimensional convolutional layers interleaved with max-pooling and batch normalization. The output of the convolutional layers is activated by a *ReLU* function. Lastly, such output tensor is flattened and fed into a fully connected (viz. dense) layer that produces a 200-dimensional feature vector. Additional details about the embedding function layers and related hyperparameters are shown in Figure 3.

Attack-Traffic Classification. The attack-traffic classifier $\phi(\upsilon)$ takes the embedded feature vector υ as input and outputs a vector of probabilities $\hat{y}$ over $\mathcal{C}$ classes, more formally: $\phi(\upsilon) : \mathbb{R}^d \to \mathbb{R}^{\mathcal{C}}$. The specific implementation of the classifier varies depending on the particular `FSCIL` approach adopted. The most common implementations typically utilize a fully-connected layer or a metric-based function. We provide more details on these technical aspects in the next Section III-C.

C. TAILORING FSL APPROACHES TO FSCIL

Here we detail the `FSCIL` approaches at the core of the NIDS we have designed. Specifically, we revise state-of-the-art `FSL` approaches, tailoring them for `FSCIL`, namely when retaining previous knowledge is also required. In detail, we consider the adoption of two learning paradigms: *meta-learning* and *transfer learning*.

1) META-LEARNING

According to episodic learning, meta-learning approaches are trained on $\mathcal{C}_{old}$ using N-way K-shot tasks extracted from a training set in the *meta-training* phase. Subsequently, they are validated via tasks from a validation set (again on $\mathcal{C}_{old}$) in the *meta-validation* phase; this allows us to select the model that achieves the highest accuracy. In the adaptation phase—i.e., *meta-testing*—a new N-way K-shot task is drawn from a dataset containing both $\mathcal{C}_{old}$ and $\mathcal{C}_{new}$ classes (see *Evaluation Procedure* in Section IV-B for further details on the datasets setup).

We underline that the number of classes N selected for each task can differ between the different phases—and this is actually the case. Going into detail, the task in the adaptation phase features a *support set* with $|\mathcal{C}_{all}| \times K_s$ samples, where $\mathcal{C}_{all} = \mathcal{C}_{old} \cup \mathcal{C}_{new}$ (i.e., the number of ways N for meta-testing is equal to $|\mathcal{C}_{all}|$) and $K_s (= K)$ is the number of labeled biflows of $\mathcal{C}_{new}$ (viz. the number of shots, cf. Section II-A). Hence, a portion of the support set encompasses new and few data (i.e., $|\mathcal{C}_{new}| \times K_s$ samples) used for the model adaptation to $\mathcal{C}_{new}$. Whereas, $|\mathcal{C}_{old}| \times K_s$ samples constitute the model memory.[1] Notably, such memory samples of $\mathcal{C}_{old}$ are not employed in standard FSL approaches that do not require keeping old knowledge. Finally, after learning from the support set, the adapted model is tested on unseen biflows of $\mathcal{C}_{new}$ and $\mathcal{C}_{old}$ from the *query set* using $|\mathcal{C}_{all}| \times K_q$ samples.

When tailoring meta-learning from FSL to FSCIL, some specific concerns must be taken into account. In particular, meta-learning approaches need the learning objectives to coincide during both meta-training and meta-testing to perform well, which is the common setup of standard FSL.[2] Consequently, since the number of ways for the meta-testing tasks is fixed and equal to $|\mathcal{C}_{all}|$, the number of meta-training ways has been set to the closest possible value which is $|\mathcal{C}_{old}|$. Accordingly, meta-training and meta-validation are carried out using $|\mathcal{C}_{old}|$*-way* K*-shot* tasks, while meta-testing exploits $|\mathcal{C}_{all}|$*-way* K*-shot* tasks. It is also worth noticing that classification performance and computational cost per episode are in trade-off as the number of ways increases. In fact, using a higher number of ways leads to better performance thanks to the correspondence between the learning objective at training and adaptation time. On the other hand, the number of samples increases significantly with a higher number of ways, which causes a rise in computational cost that typically does not happen in standard meta-learning FSL setups characterized by a reduced number of ways and shots. Therefore, in complex cases, namely when N is higher, it is preferable to use FSCIL approaches based on the transfer-learning paradigm.

Meta-Learning Approaches. We adopt three meta-learning approaches initially introduced in the field of FSL for computer vision. These approaches reduce the risk of overfitting by narrowing the hypothesis space, which encompasses all possible solutions for a learning problem. Such a result is achieved by meta-training the embedding function using similarity metrics. In detail:

i) *Matching Networks* (MatchingNet) [28] use a classifier $\phi(\cdot)$, which is a generalized form of the nearest-neighbor matching. Specifically, after the feature embedding, $\phi(\cdot)$ calculates the prediction $\hat{y}$ with the formula: $\hat{y} = \sum_{i=1}^{K_s} a(\upsilon(x_{query}), \upsilon(x_i))y_i$. Here, an attention mechanism $a(\cdot, \cdot)$ is used to measure the similarity between the embedding of the query instance to be labeled (i.e., $\upsilon(x_{query})$) and the embedded samples of the support set (i.e., $\upsilon(x_i)$).[3] Finally, x_{query} is assigned the class of the most similar instance of the support set.

ii) *Prototypical Networks* (ProtoNet) [29] also rely on a similarity metric as the classifier $\phi(\cdot)$, but with a fundamental distinction from MatchingNet: the similarity is calculated between an embedded query sample and the centroid of each class, which is derived from the support set and referred to as *prototype*. Formally, $p_c = \frac{1}{K_s}\sum_{i=1}^{K_s} \upsilon(x_i)$, where p_c is the prototype for a generic class c in the support set.

iii) Conversely, MetaOptNet [33] leverages a classifier $\phi(\cdot)$, which consists of an SVM trained on embedded support set samples.

In these three approaches, the embedding function $\theta(\cdot)$ is frozen after meta-training. This means that the knowledge acquired on the train classes is fixed, which helps to *prevent forgetting phenomena* on $\mathcal{C}_{old}$.

2) TRANSFER LEARNING

According to the transfer-learning paradigm (cf. Section II-A), we consider a base model $\langle \theta(\cdot), \phi_{old}(\cdot) \rangle$, where $\theta(\cdot)$ is the embedding function and $\phi_{old}(\cdot)$ is a fully-connected classifier consisting of $|\mathcal{C}_{old}|$ neurons. During the *pre-training* phase ($\mathcal{T}_0$), the base model is trained, validated, and tested on the respective disjointed sets with the same $\mathcal{C}_{old}$ label space. The model with the highest validation accuracy is chosen for the following *fine-tuning* ($\mathcal{T}_1$), which coincides with the *adaptation* phase. In the latter phase, the model is further refined using the $|\mathcal{C}_{all}| \times K_s$ samples from the support set in a similar way to the meta-testing of meta-learning approaches—hence, leveraging a memory of $|\mathcal{C}_{old}| \times K_s$ biflows and K_s unseen biflows of $\mathcal{C}_{new}$. Finally,

[1] In the context of CIL and FSCIL, *memory* refers to a mechanism for storing information about previously learned classes (i.e., $\mathcal{C}_{old}$). This information is useful to overcome the *catastrophic forgetting problem*, where a model forgets $\mathcal{C}_{old}$ as it learns new classes (i.e., $\mathcal{C}_{new}$). Here, we use an *exemplar-based memory*, where K_s samples of $\mathcal{C}_{old}$ are stored.

[2] This outcome is both demonstrated in [28] and further confirmed via preliminary experiments not reported for brevity.

[3] For instance, the attention mechanism could be a distance metric, such as the cosine or the Euclidean distance.

the performance is assessed on $\mathcal{C}_{all}$ via the query set (i.e., using $|\mathcal{C}_{all}| \times K_q$ samples).

Transfer-Learning Approaches. We consider three transfer-learning approaches that share a similar $\mathcal{T}_0$ learning methodology. They mainly differ on the specific strategy applied during $\mathcal{T}_1$. In more detail:

i) *Fine-Tuning* (FT) appends a new fully-connected classifier of $|\mathcal{C}_{new}|$ neurons (i.e., $\phi_{new}(\cdot)$) to $\phi_{old}(\cdot)$. The weights of the whole model $\langle\theta(\cdot), \phi_{old}(\cdot), \phi_{new}(\cdot)\rangle$ are optimized during the adaptation phase.

ii) *Freezing* (FZ) also appends a $\phi_{new}(\cdot)$ classifier similarly to FT. However, it solely updates the weights of this newly added classifier, while the rest of the model (i.e., $\langle\theta(\cdot), \phi_{old}(\cdot)\rangle$) remains fixed.

iii) *Rethinking Few-Shot* (RFS) [35] uses *sequential knowledge distillation* to build an effective base model during $\mathcal{T}_0$. In particular, sequential knowledge distillation involves exploiting the knowledge of a teacher. The training proceeds over n learning cycles (also called self-distillation cycles). In the generic i^{th} learning cycle (where $i \leq n$), the teacher is the student from the previous $(i-1)^{th}$ cycle. The student is then trained to minimize a weighted sum (through α and β) of the cross-entropy loss ($\mathcal{L}^{ce}(\cdot)$) between the predictions ($\hat{y}_s$) and the ground-truth labels (y), and the *Kullback-Leibler divergence* ($KL(\cdot)$) between the student predictions ($\hat{y}_s$) and the soft targets predicted by the teacher ($\hat{y}_t$). Formally:

$$(\theta, \phi_{old})' = \underset{(\theta,\phi_{old})}{\arg\min}\big(\alpha \cdot \mathcal{L}^{ce}(\hat{y}_s, y) \; + \beta \cdot KL(\hat{y}_s, \hat{y}_t)\big)$$

where $(\theta, \phi_{old})'$ are the updated weights of the whole student model. In $\mathcal{T}_1$, $\theta(\cdot)$ is frozen, and RFS employs either the *logistic regression* or the *nearest-neighbor classifier* as $\phi(\cdot)$—replacing the $\phi_{old}(\cdot)$ trained in $\mathcal{T}_0$. When using the nearest-neighbor classifier, the model weights remain completely unchanged.

For FZ and RFS, the *fixed representation* helps to limit forgetting, similar to meta-learning approaches. Conversely, this problem is more acute for FT, since the embedding function $\theta(\cdot)$ is also updated. However, the usage of $\mathcal{C}_{old}$ data during $\mathcal{T}_1$ (i.e., the model memory) helps to mitigate forgetting.

IV. EXPERIMENTAL SETUP

This section details the setup adopted in our study for the experimental evaluation. Section IV-A introduces the threat categories we consider and the public datasets we adopt. Then, we delve into the specifics of the FSCIL setup in Section IV-B.

A. THREAT SCENARIO AND DATASETS

The threat scenario addressed in this paper includes *flooding*, *trojan*, *brute force*, *injection*, *information gathering*, and *man-in-the-middle* attacks.

- *Flooding* (i.e., DoS and DDoS attacks) is a targeted effort to render a resource—be it a website, application, or server—unavailable for its intended purpose. Several TCP/UDP packets overwhelm the server processing and network capabilities, leading to system paralysis. DDoS poses a greater challenge than DoS due to their coordinated assault from multiple locations (e.g., botnets managed by a *command-and-control* server), resulting in faster attack speed and more traffic volume. Such differences are also reflected in intrusion detection datasets [43], [44], [45], which commonly separate DoS and DDoS attack labels.
- *Trojan* is a type of malware that deceives users by disguising itself as a legitimate program. Once on a victim device, Trojans give attackers remote access, allowing them to steal sensitive information, download or upload files, install ransomware, spy on keystrokes, etc.
- *Brute force* consists of an attacker systematically trying to gain access to a system using predefined combinations for access credentials. For the sake of efficiency, an attacker may try to use common user names and passwords or may vary the credentials within a large range.
- *Injection* consists of an attacker trying to insert malicious code into a system. This malicious code can then compromise both the targeted system (e.g., the database) and potentially other connected users, leading to data breaches, unauthorized access, or even system manipulation.
- *Information gathering* attacks (e.g., port scanning, OS detection, and host discovery) represent the initial steps to map network vulnerabilities. These techniques involve sending several probes and analyzing the responses to identify active devices, open ports, and the type of operating systems running on those devices.
- *Man-in-the-middle (MITM)* attacks are deceptive tactics to intercept and manipulate communication channels between two parties. The attacker acts as a legitimate intermediary between the victim and the intended recipient.

We use CSE-CIC-IDS2018 and IoT-NID for assessing the performance of the proposed FSCIL NIDS. A series of reasons lead us to choose these datasets. *(i)* They accurately reflect the threat landscape introduced above. *(ii)* They capture diverse network scenarios; specifically, CSE-CIC-IDS2018 collects the traffic of various types of malicious activities, such as incoming attacks from external networks and PCs infected with malware (e.g., keylogging and ransomware); in contrast, IoT-NID focuses on the IoT environment, encompassing attacks that target and exploit compromised IoT devices (i.e., IoT botnets). *(iii)* Finally, these datasets have been collected recently and are widely used in the scientific literature, particularly CSE-CIC-IDS2018.

The FSCIL NIDS is evaluated against attackers from outside and inside the local network it monitors, which we refer to as *external attackers* and *internal attackers*, respectively. Following this nomenclature, CSE-CIC-IDS2018 collects the traffic from both external and internal attackers, whereas IoT-NID only from internal attackers.

CSE-CIC-IDS2018. CSE-CIC-IDS2018 [43] is the outcome of a collaborative project between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). The experimental campaign has been carried over 10 non-consecutive days, during which both benign and 14 types of malicious traffic are generated, including attack classes such as bruteforce, DoS, DDoS, injection, infiltration, and botnets.[4] The generated traffic is directed towards a LAN hosted on Amazon Web Services (AWS). The LAN consists of 6 subnets with 450 machines running different operating systems, applications, and vulnerabilities. The attack traffic is generated from another external network of 30 machines using the concept of *profiles* to emulate the behavior of real users (via the so called *B(enign)-profile*) and attackers (via the *M(alicious)-profile*). Captured traffic is released by the authors of the dataset in CSV format and as PCAP file—viz. raw traffic traces. In this paper, we leverage the PCAP files and apply to it the traffic processing techniques presented in Section III.

Given its large size ($\approx 3.5M$ biflows), we downsample the original dataset by capping the classes (including the Benign class) at their median size.[5] Such a downsampling procedure has been extensively validated in our previous works [8], [21], [26] and (being random) it does not impact the original distribution of (attack and benign) traffic samples. Furthermore, we carry out a careful *dataset cleaning*. We discard *Bruteforce-FTP* and *DoS SlowHTTPTest* traffic due to the sole presence of SYN packets followed by RST+ACK packets, which suggests misconfiguration at capture time (i.e., the server under-attack does not have the probed ports open, thus refusing to open the connection). We remove the last packets in biflows when appearing after an unexpectedly long silence span (i.e., with an average IAT of $\approx 650s$ compared to $\approx 7s$ of the other packets). This characteristic appears in 12% of *DoSGoldenEye* biflows. Lastly, we split the biflows of *DoS GoldenEye* when a new TCP three-way handshake is observed in the packet sequence ($\approx$ 19% of the biflows), suggesting that the same 5-tuple is reused, possibly due to the design of the attack automation process. The pre-processing phase yields a dataset with $109k$ biflows and 12 classes—see Figure 4 (top).

IoT-NID. IoT-NID [45] was collected in 2019 by a group of researchers from the Hacking and Countermeasure Research Lab. It consists of network traffic captured during various types of attacks on two smart home devices: *SKT NUGU* and *EZVIZ Wi-Fi Camera*. Such IoT devices—alongside laptops and smartphones—are interconnected within the same wireless network, simulating a realistic smart-home environment. The dataset consists of 42 PCAP files, each captured at different time intervals. IoT-NID encompasses 5 macro-categories (i.e., normal traffic, MITM attacks, Mirai botnet, DoS, and scanning). From these macro-categories, it is possible to derive 10 sub-categories concerning the particular attack technique (e.g., HTTP and ACK flooding). Except for Mirai—where the attacks are simulated through packets generated on a laptop and manipulated to appear as if originating from the IoT device—all other attacks involve packets captured during the simulation of the attack using tools like *Nmap*. IoT-NID is released as PCAP files. Labeling is performed by applying filter rules defined by the authors using *Wireshark*. Following the phases of data extraction from the PCAP files and pre-processing, we end up with a dataset that comprises $196k$ biflows and 10 classes—see Figure 4 (bottom).

[4]See https://www.unb.ca/cic/datasets/ids-2018.html for more details on labeling.

[5]We also remove the class *Infiltration*, as it only includes 5 biflows, i.e., an insufficient number of samples to rigorously adapt and test the FSCIL approaches.

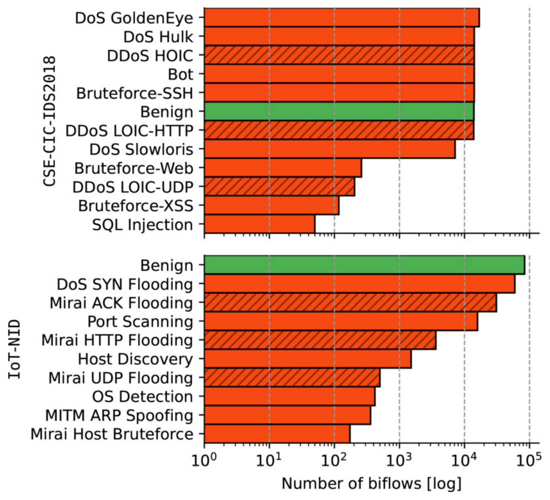


FIGURE 4. Number of per-class biflows (in *log* scale) of pre-processed CSE-CIC-IDS2018 (top) and IoT-NID (bottom) datasets. Bars with diagonal hatching represent DDoS attacks (i.e., $\mathcal{C}_{new}$).

Table 4 highlights how the attacks of the two datasets are distributed across the threat categories introduced at the beginning of the present section.

B. FSCIL SETUP

In the following, we detail the FSCIL setup to foster the reproducibility of our study. The main aspects are summarized in Table 5.

Evaluation Procedure. In line with the alarming emergence of (D)DoS threats (see Section I), $\mathcal{C}_{new}$ includes different types of DDoS attacks for evaluation purposes. Such a choice enables a thorough analysis of DDoS classification: we investigate both the ability of the approaches to differentiate among DDoS attacks and their capability to discern them from other known attacks and normal traffic (we recall that all the attack classes are considered in our evaluation process). Therefore, the traffic classes

TABLE 4. Categorization of the attacks present in `CSE-CIC-IDS2018` and `IoT-NID`.

Threat Category	`CSE-CIC-IDS2018`	`IoT-NID`
Flooding	*DDoS HOIC, DDoS LOIC-HTTP, DDoS LOIC-UDP, DoS GoldenEye, DoS Slowloris*	*Mirai ACK F†, Mirai HTTP F†, Mirai UDP F†, DoS SYN F†*
Trojan	*Bot (Ares and Zeus Trojan)*	–
Brute force	*B*-SSH, B*-Web, B*-XSS*	*Mirai Host B**
Injection	*SQL Injection*	–
Info gathering	–	*Port Scanning, Host Discovery, OS Detection*
MITM	–	*ARP Spoofing*

TABLE 5. Parameters characterizing the episodes for both `CSE-CIC-IDS2018` and `IoT-NID`.

Episode Parameter	`CSE-CIC-IDS2018`	`IoT-NID`
N (meta-training and meta-validation)	9	7
N (meta-testing and fine-tuning)	12	10
K_s	10	10
K_q	5	5

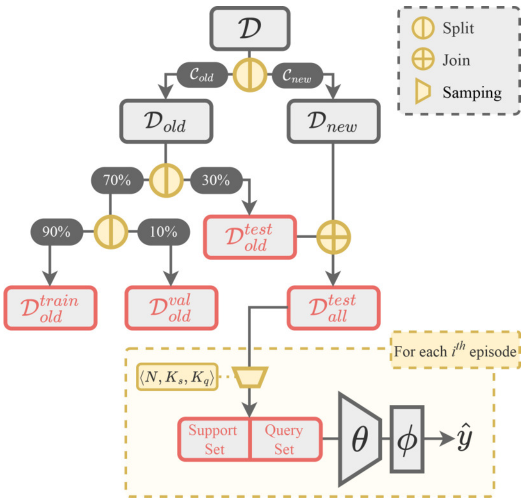


FIGURE 5. `FSCIL` evaluation procedure.

in the considered datasets are divided into $\mathcal{C}_{new}$ and $\mathcal{C}_{old}$ as follows. Concerning `CSE-CIC-IDS2018`, $\mathcal{C}_{new}$ contains *DDoS LOIC-UDP*, *DDoS LOIC-HTTP*, and *DDoS HOIC*. For `IoT-NID`, $\mathcal{C}_{new}$ includes *Mirai ACK Flooding*, *Mirai HTTP Flooding*, and *Mirai UDP Flooding*. For both datasets, classes in $\mathcal{C}_{new}$ are depicted with diagonal hatching in Figure 4. Remaining attacks and benign traffic are in $\mathcal{C}_{old}$.

After defining $\mathcal{C}_{new}$ and $\mathcal{C}_{old}$, we split the datasets for selecting the portions used for each `FSCIL` phase to perform a fair evaluation between the meta-learning and transfer-learning approaches. For brevity, we refer to Figure 5 for the particular evaluation procedure. Notably, the last two splits to obtain $\mathcal{D}_{old}^{train}$, $\mathcal{D}_{old}^{val}$, and $\mathcal{D}_{old}^{test}$ are performed via *stratified hold-out*.[6] The dataset partitions obtained in this way are utilized as follows:

i) Regarding *meta-learning approaches*: $\mathcal{D}_{old}^{train}$ is used for meta-training, $\mathcal{D}_{old}^{val}$ for meta-validation, and $\mathcal{D}_{all}^{test}$ ($= \mathcal{D}_{new} \cup \mathcal{D}_{old}^{test}$, see Figure 5) for meta-testing (viz. adaptation phase).

ii) Regarding *transfer-learning approaches*: $\mathcal{D}_{old}^{train}$ is employed for training, $\mathcal{D}_{old}^{val}$ for validation, and $\mathcal{D}_{old}^{test}$ for testing during pre-training; $\mathcal{D}_{all}^{test}$ is used during fine-tuning (viz. adaptation phase).

As mentioned in Section III-C, we draw a set of adaption episodes from $\mathcal{D}_{all}^{test}$. Each episode comprises two data partitions: one used to learn novel knowledge during the adaptation phase (i.e., the support set) and one to test the model on both new ($\mathcal{C}_{new}$) and previously encountered ($\mathcal{C}_{old}$) classes (i.e., the query set). Further details on episode setup are provided in the next paragraph.

Episode Setup. Herein, we elaborate on the triplet $\langle N, K_s, K_q \rangle$, which characterizes the episode sampling. During the meta-testing phase (resp. fine-tuning) for meta-learning (resp. transfer-learning) approaches, we leverage $|\mathcal{C}_{all}|$-*way* K-*shot* tasks when sampling from $\mathcal{D}_{all}^{test}$. This results in $N = 12$ for `CSE-CIC-IDS2018` and $N = 10$ for `IoT-NID`. We keep $K_s = K = 10$ and $K_q = 5$ for both datasets. Regarding meta-training and meta-validation of meta-learning approaches, we fix the triplet to $\langle N = |\mathcal{C}^{old}|, K_s = 10, K_q = 5 \rangle$, where $|\mathcal{C}^{old}| = 9$ for `CSE-CIC-IDS2018` and $|\mathcal{C}^{old}| = 7$ for `IoT-NID`. Such information is also reported in Table 5 for reader convenience. The approaches from both learning paradigms are evaluated across 100 episodes.

Hyperparameter Configuration. We set the hyperparameters determining the configuration of the `FSCIL` approaches based on preliminary tuning experiments (not reported for the sake of brevity), results from state-of-the-art papers proposing the original approaches, or analyses performed in our previous works [8], [21], [26].

In terms of **approach-specific hyperparameters**, `MetaOptNet` is trained with a *regularization parameter* set to 0.1 and a maximum of 15 *SVM iterations*. Both `MatchingNet` and `ProtoNet` resort to the Euclidean distance as *similarity metric*. `RFS` is characterized by $\alpha = \beta = 0.5$, and employs a nearest-neighbor classifier during the fine-tuning phase and one *self-distillation cycle* (see Section III-C2 for the meaning of hyperparameters).

Concerning the **general hyperparameters**, the *cross-entropy loss* is used to measure the classification error, except for `RFS` which adds the *Kullback-Leibler divergence* as an additional loss term during pre-training. A custom *early-stopping* mechanism is employed to address overfitting during the meta-training and pre-training phases. It monitors both *accuracy and loss* on $\mathcal{D}_{old}^{val}$, with a *minimum delta*

[6]To sample episodes with an adequate number of samples, 15 is chosen as the minimum number of biflows per class when performing the hold-out technique.

of 10^{-3} and *patience* of 20 epochs. Below, we report the remaining general hyperparameters according to the learning phases to which they refer:

- *During the meta-training and pre-training phases* of meta-learning and transfer-learning approaches, respectively, we set the maximum number of *epochs* to 200 and use the *Adam* optimizer with 10^{-4} *learning rate*. The transfer-learning approaches use a *batch size* of 64 samples.
- *During the fine-tuning phase*—performed only for transfer-learning approaches, except `RFS` leveraging the nearest-neighbor classifier—the *batch size* is 4 samples, the maximum number of *epochs* is 200, and the *learning rate* of the *Adam* optimizer is set to 10^{-3}.

Regarding **traffic segmentation**, we set $N_p = 20$ packets.[7] In our initial analyses, we empirically set T_s (cf. Section III-B1) to 330*s*. This choice guarantees sufficiently fast classification in the worst cases, while allowing the NIDS to collect enough packets also for slow DoS attacks, such as *DoS Slowloris* and *DoS GoldenEye*. In addition, in Section V-E, we showcase the NIDS performance for various values of T_s.

Non-FSCIL State-of-the-Art Approach. We evaluate our `FSCIL`-based NIDS against the non-`FSCIL` state-of-the-art way of handling novel attacks, that is *training the model from scratch* whenever new attack knowledge becomes available. This approach is referred to as $\texttt{SOTA}_{\texttt{nonFSCIL}}$ in the following. The model is trained using the *maximum number* of available $\mathcal{C}_{new}$ samples (i.e., there is no K-shot constraint). $\texttt{SOTA}_{\texttt{nonFSCIL}}$ comprises an embedding function $\theta(\cdot)$ having the same architecture presented in Section III-B2, specifically designed for IoT-traffic classification [40] and attack-traffic classification [8], [21], [26]. The embedding function is followed by a fully-connected layer $\phi(\cdot)$, with $|\mathcal{C}_{all}|$ neurons and *Softmax* activation function.

Regarding the evaluation scenario for $\texttt{SOTA}_{\texttt{nonFSCIL}}$, we partition the full dataset $\mathcal{D}$ into 3 subsets: training, test, and validation, using a stratified hold-out technique. The test set comprises 30% of $\mathcal{D}$, while the 70% of $\mathcal{D}$ is further split for the training set (90%) and the validation set (10%). $\texttt{SOTA}_{\texttt{nonFSCIL}}$ is trained, validated, and tested using the respective aforementioned subsets. As for the hyperparameters (i.e., epochs, learning rate, etc.), $\texttt{SOTA}_{\texttt{nonFSCIL}}$ shares the same configuration of transfer-learning approaches during the pre-training phase.

Performance Metrics. We employ the *F1-score* and *confusion matrices* for evaluating the performance of our `FSCIL`-based NIDS and the *Accuracy*, *False Positive Rate (FPR)*, and *True Positive Rate* (*TPR* or *Recall*) for in-detail insights. Specifically, for each metric, we compute the *mean per-episode score*—i.e., the average of the scores achieved on the 100 episodes sampled during the adaptation phase (see Sections II-A and IV-B for details)—and the corresponding *confidence interval* (at the 95% confidence level) on the query sets extracted from $\mathcal{D}_{all}^{test}$.

To compute the FPR and Recall, we combine all attack classes into a single class—turning the problem into a *binary* classification task (benign vs. malicious). In this way, we can use the FPR and Recall to validate the NIDS' ability to determine whether a biflow is malicious or not.

The *FPR* indicates the probability that a NIDS will generate false alarms, and thus it needs to be minimized. The FPR is formally computed as:

$$\texttt{FPR} = \frac{\texttt{FP}}{\texttt{FP} + \texttt{TN}}$$

where `TP` (*True Positive*) represents the number of correctly predicted benign biflows, `FP` (*False Positive*) the number of malicious biflows incorrectly predicted as benign, `TN` (*True Negative*) the number of correctly predicted malicious biflows, and `FN` (*False Negative*) the number of benign biflows incorrectly predicted as malicious.

Conversely, the *Recall* is the proportion of actual positives (i.e., benign biflows) correctly identified; formally:

$$\texttt{Recall} = \frac{\texttt{TP}}{\texttt{TP} + \texttt{FN}}$$

Regarding the evaluation of *multi-class* performance, the *(macro) F1-score* is calculated as the per-class average of the harmonic mean of Precision and Recall, formally:

$$\texttt{F1-score} = \frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{C}|} \frac{2 \times \texttt{Precision}_i \times \texttt{Recall}_i}{\texttt{Precision}_i + \texttt{Recall}_i}$$

Here, $|\mathcal{C}|$ denotes the number of classes (i.e., the cardinality of $\mathcal{C}_{new}$, $\mathcal{C}_{old}$, or $\mathcal{C}_{all}$), $\texttt{Precision}_\texttt{i}$ measures the ratio of predictions of class C_i being correct, $\texttt{Recall}_\texttt{i}$ measures the ratio of biflows actually belonging to class C_i correctly classified, and the F1-score takes into account both these aspects via their harmonic mean.

The *Accuracy* is the ratio of correctly classified biflows to the total number of biflows:

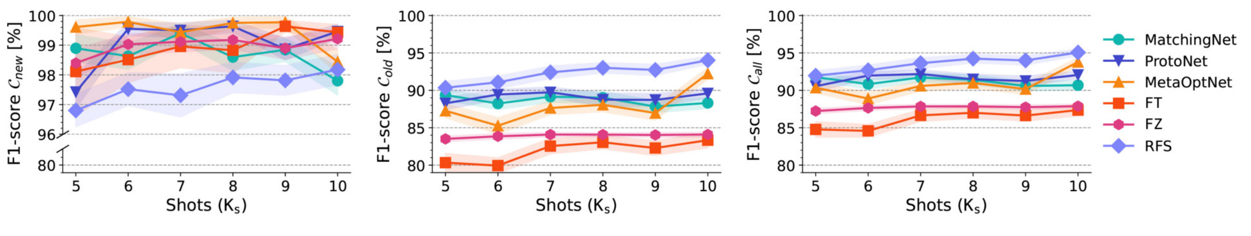
$$\texttt{Accuracy} = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} \mathbb{1}\left(\hat{y}_i = y_i\right)$$

where $|\mathcal{D}|$ denotes the number of biflows in the dataset $\mathcal{D}$, $\mathbb{1}(\cdot)$ is a function that returns 1 if the predicted label equals the ground truth (i.e., the condition is true) and 0 otherwise, y_i is the ground truth, and $\hat{y}_i$ is the prediction for the i^{th} biflow.

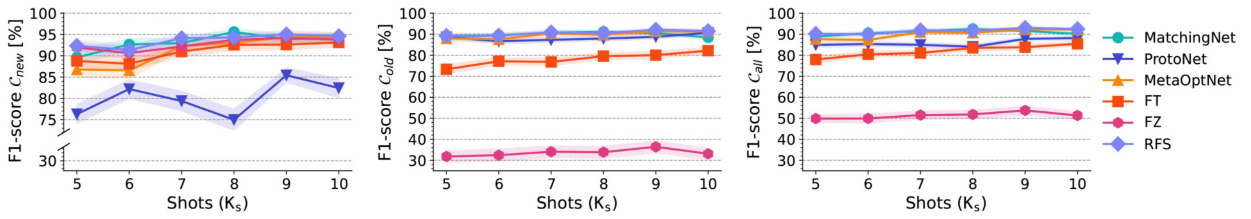
Lastly, we leverage the *confusion matrices* to identify the most frequent misclassification patterns. Indeed, they offer an intuitive visual representation of actual vs. predicted biflows for each class, where the diagonal of the matrix represents the correct predictions.

Implementation Details. All the analyses in this paper are executed on a machine with 12 Intel Xeon CPU E5-2430 v2 @ 2.50GHz and 62GB of memory. We tailored the *PyTorch* implementations of `ProtoNet` and `MetaOptNet` from the

[7]This choice is based on sensitivity analyses conducted in our previous works [21], [26] and on `IoT-NID`. In the latter case, we have varied N_p from 10 to 50 packets with a step of 10. The results show a growth of the F1-score up to $N_p = 20$ after which a plateau is reached.



(a) CSE-CIC-IDS2018.



(b) IoT-NID.

FIGURE 6. Sensitivity analysis to the number of shots on CSE-CIC-IDS2018 (a) and IoT-NID (b). Note that the figures related to the F1-score on the $\mathcal{C}_{new}$ classes have the upper part zoomed in to highlight the differences between the various approaches.

learn2learn framework [46], MatchingNet from the code provided in [28], while RFS from the *LibFewShot* framework [47]. We implemented FT, FZ, and SOTA$_{\mathrm{nonFSCIL}}$ according to their commonly known designs [48].

V. EXPERIMENTAL EVALUATION

In the following, we report the results of the experimental analysis. First, in Section V-A, we evaluate various FSCIL approaches (belonging to both the meta-learning and transfer-learning paradigms) based on their ability to classify new attacks with a varying (though limited) number of samples (i.e., biflows). The goal is to select the best-performing approach as the core for our FSCIL-based NIDS. In Section V-B, we assess the efficacy of this FSCIL-based proposal versus the non-FSCIL state of the art, which (re)trains the model from scratch when dealing with new attack classes. Afterward, we validate different properties of the proposed FSCIL-based NIDS. In Section V-C, we investigate its performance in classifying both malicious and benign traffic. We also cross-validate its robustness when operating on traffic from a dissimilar network context compared to the traffic seen during training. In Section V-D, we evaluate the usefulness of the NIDS' input data through a feature-ablation study. Finally, in Section V-E, we analyze the performance of our proposed NIDS in a tight early-classification scenario, aiming to find a good trade-off between the time-to-insight and classification effectiveness.

A. SENSITIVITY TO THE NUMBER OF SHOTS

The goal of this first analysis is to evaluate the impact of the number of shots composing episodes on the performance of the FSCIL approaches in classifying traffic biflows (viz. the effect of adapting the NIDS with few samples). Additionally, this analysis serves to select the best approach to be used as the core of the NIDS. This assessment is conducted after integrating a limited number of samples from previously unseen (i.e., during training) DDoS attacks, where the quantity is determined by the value of $K(= K_s)$. Therefore, we show the F1-score achieved on $\mathcal{C}_{new}$, $\mathcal{C}_{old}$, and $\mathcal{C}_{all}$ while varying the number of shots in the tasks used during the meta-testing and fine-tuning phase. In particular, we consider 6 possible values for K, namely $K \in [5, 10]$. In fact, with $K < 5$, the approaches exploiting metric-based classifiers (e.g., MatchingNet, ProtoNet, RFS) show a high variability given the low representativeness of the support set (having only one sample per class when $K = 1$). Results on CSE-CIC-IDS2018 and IoT-NID are summarized in Figure 6.

Performance on CSE − CIC − IDS2018. Figure 6(a) reveals satisfactory performance across all FSCIL approaches when considering the F1-score obtained on $\mathcal{C}_{new}$ for CSE-CIC-IDS2018: all of them achieve F1-score values greater than 97%, showing high effectiveness in classifying novel DDoS attacks with few samples. MetaOptNet is the best performing, with a near-perfect trend consistently close to 99.5% for values of K from 5 to 9. ProtoNet is following closely and also exhibits significant values around 99.5% starting from $K = 6$. In contrast, RFS exhibits a poorer (although still satisfactory) F1-score but with a loss always lower than 3%. Similar to MetaOptNet, MatchingNet shows a fairly stable trend with negligible degradation with $K = 10$. The remaining approaches exhibit a positive upward trend as K increases.

The performance on $\mathcal{C}_{old}$ is switched: RFS achieves the highest results even when employing an extremely limited number of $K = 5$ memory samples. Additionally, it gains +4% from $K = 5$ to $K = 10$ reaching up to 95% F1-score. This result likely originates from its sequential knowledge distillation strategy. By effectively transferring knowledge from the teacher model, RFS strengthens its representation of $\mathcal{C}_{old}$, thus building a grounded base model for FSCIL. MetaOptNet also shows a similarly high F1-score but only with $K = 10$. Although the remaining approaches do not reach the peaks of RFS and MetaOptNet, they still obtain significant F1-score values (up to 90%).

Finally, for $\mathcal{C}_{all}$, we can draw conclusions similar to $\mathcal{C}_{old}$, with RFS outperforming the other approaches for all K values. This outcome demonstrates that the 9 classes in $\mathcal{C}_{old}$ weigh more than the 3 classes in $\mathcal{C}_{new}$.

Performance on IoT-NID. Figure 6(b) depicts the F1-score on $\mathcal{C}_{new}$, $\mathcal{C}_{old}$, and $\mathcal{C}_{all}$ for IoT-NID. Starting with the values obtained on $\mathcal{C}_{new}$, the F1-score reveals noteworthy trends among the different approaches. Except for ProtoNet, all exhibit good performance for the considered shot values. In detail, ProtoNet shows lower F1-scores, ranging between 75% and 85%. This could be attributed to the shape of its feature-vector clusters in the embedded space, which hinders the effective utilization of the prototypes especially for lower shots. On the other hand, both RFS and FZ achieve satisfactory F1-scores (i.e., from 92% with $K = 5$ to 95% with $K = 10$). The remaining approaches (i.e., MatchingNet, FT, and MetaOptNet) report F1-score values between 85% and 90% with $K = 5$. As the number of shots increases, the gap among the approaches narrows, converging around 95% F1-score.

Regarding $\mathcal{C}_{old}$, all the approaches but FT and FZ achieve satisfactory results, reaching F1-scores up to 92%. As the value of K increases, a slight upward trend is observed, with FT experiencing the most substantial increase. FZ exhibits notably poor performance ranging between 30% and 40%, indicating that the sole fine-tuning of ϕ_{new} is insufficient for this dataset. Similar considerations can be brought also for $\mathcal{C}_{all}$ where all the approaches present slightly better performance than $\mathcal{C}_{old}$.

Take-Home Messages. All FSCIL approaches (except FZ on IoT-NID) exhibit excellent performance in classifying attacks whose samples were absent in the training set ($\mathcal{C}_{new}$). Regarding the performance achieved when classifying the attacks observed during the training phase ($\mathcal{C}_{old}$), RFS stands out due to its knowledge distillation strategy, which enables a highly effective model. This underscores that incorporating novel knowledge about DDoS attacks does not compromise the representation learned for $\mathcal{C}_{old}$. This is further confirmed by the fact that on $\mathcal{C}_{all}$, RFS is the overall-best-performing approach, and then it is chosen as the core component of our NIDS.

B. COMPARISON WITH NON-FSCIL STATE OF THE ART

Building on the previous analysis, in this section *we evaluate whether using an NIDS based on FSCIL results in greater effectiveness compared to the traditional SOTA$_{nonFSCIL}$ method.* As aforementioned, *we consider RFS as the base approach for our FSCIL-based NIDS.* RFS is evaluated on different K_s shots, namely $K_s \in [5, 10]$ with step 1. On the other hand, as described in Section IV-B, SOTA$_{\text{nonFSCIL}}$ is trained from scratch by leveraging *all the available samples of $\mathcal{C}_{new}$* (i.e., there is no few-shot constraint).

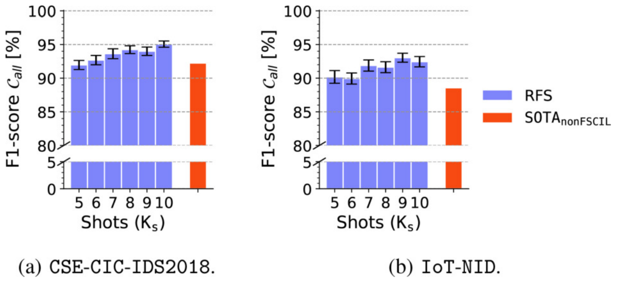


FIGURE 7. Comparison between RFS and SOTA$_{\text{nonFSCIL}}$ on $\mathcal{C}_{all}$. RFS is evaluated by varying K_s, while SOTA$_{\text{nonFSCIL}}$ uses the maximum number of samples available for $\mathcal{C}_{new}$.

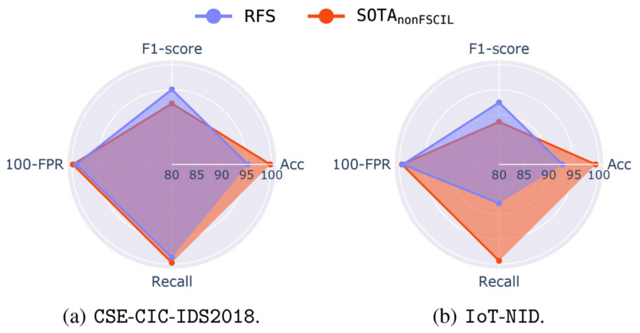


FIGURE 8. Comparison in terms of F1-score [%], Accuracy (Acc) [%], FPR [%], and Recall [%] between RFS (with $K_s = 10$) and SOTA$_{\text{nonFSCIL}}$ on $\mathcal{C}_{all}$. To better highlight the results, the radial axis is zoomed between 80% and 100%. We report the complement of FPR to 100% to make such a metric comparable to the others (i.e., the higher the better).

RFS vs. SOTA$_{\text{nonFSCIL}}$: Classification Performance. For both datasets, the performance of RFS in terms of F1-score is comparable to SOTA$_{\text{nonFSCIL}}$ already with $K_s = 5$ on $\mathcal{C}_{all}$ (see Figure 7). Specifically, on CSE-CIC-IDS2018, RFS equals SOTA$_{\text{nonFSCIL}}$ with $K_s = 5$. With a higher number of shots (i.e., $K_s = 10$), RFS reaches an F1-score of 95%, gaining +3% compared to SOTA$_{\text{nonFSCIL}}$. On IoT-NID, RFS achieves an F1-score of 90% with $K_s = 5$ versus the 88% of SOTA$_{\text{nonFSCIL}}$. The gap increases up to +5% in favor of RFS with 9 shots.

When considering the performance on $\mathcal{C}_{new}$ (figure omitted for the sake of brevity), SOTA$_{\text{nonFSCIL}}$ slightly surpasses RFS by +1% F1-score on CSE-CIC-IDS2018 and by +4% on IoT-NID with $K_s = 10$. Indeed, SOTA$_{\text{nonFSCIL}}$ is trained using all samples of $\mathcal{C}_{new}$, thereby providing an upper bound to the performance of RFS, which is limited to K_s samples.

Figure 8 provides further insights into the intrusion detection capability of the two approaches. Specifically, SOTA$_{\text{nonFSCIL}}$ achieves +5% Accuracy compared to RFS. Detailing, SOTA$_{\text{nonFSCIL}}$ achieves a higher Accuracy due to its bias towards majority classes, as it is trained on all available samples. This results in a class imbalance issue, evident when comparing the F1-score of SOTA$_{\text{nonFSCIL}}$ with that of RFS. On the other hand, RFS leverages a balanced number of $\mathcal{C}_{old}$ memory samples and $\mathcal{C}_{new}$ shots, thus mitigating such an issue. When merging all attack classes into a single malicious class (cf. Section IV-B), $\mathtt{SOTA}_{\mathtt{nonFSCIL}}$ shows superior performance in terms of Recall on IoT-NID. Conversely, there is no significant difference on CSE-CIC-IDS2018 and in terms of FPR.

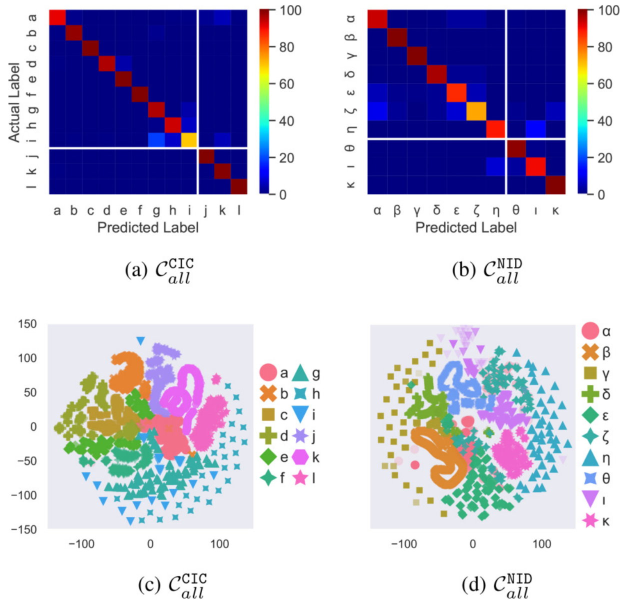
RFS vs. $\mathtt{SOTA}_{\mathtt{nonFSCIL}}$: Time Analysis. Another fundamental aspect to consider is the *time required to adapt the model to new classes*—during meta-testing for meta-learning approaches, $\mathcal{T}_1$ for transfer-learning approaches, and retraining from scratch for $\mathtt{SOTA}_{\mathtt{nonFSCIL}}$. In particular, the *adaptation time* is extremely short for RFS ($\approx$ 0.2 seconds with $K = 10$), while the full retrain of $\mathtt{SOTA}_{\mathtt{nonFSCIL}}$ is much longer ($\approx$ 620 seconds): namely, FSCIL guarantees a $\approx 3000\times$ adaptation-time reduction compared to $\mathtt{SOTA}_{\mathtt{nonFSCIL}}$.

Take-Home Messages. RFS achieves classification performance on par with $\mathtt{SOTA}_{\mathtt{nonFSCIL}}$ already with very few shots and surpasses it for higher shots. This demonstrates its high adaptability even with few available samples. Additionally, our RFS-based NIDS exhibits an adaptation time 3 orders of magnitude shorter than the time required by $\mathtt{SOTA}_{\mathtt{nonFSCIL}}$ for training the model as the few samples of previously unseen attacks become available.

C. MISUSE-DETECTION STUDY

This analysis delves into the behavior of our NIDS based on the best-performing RFS approach in the challenging scenario where only 10 samples for each previously unseen attack are available ($K_s = 10$). *The objective is to validate the RFS ability to distinguish between various (DDoS) attacks not available at the training phase, telling them apart from other malicious and benign traffic.*

Validation on Single Datasets. The confusion matrix on CSE-CIC-IDS2018 classes (denoted as $\mathcal{C}_{all}^{\mathtt{CIC}}$) is depicted in Figure 9(a). Overall, the separability among DDoS attacks is notably high, with negligible misclassifications. For the training classes ($\mathcal{C}_{old}^{\mathtt{CIC}}$), the maximum confusion arises between *SQL Injection* and *Bruteforce Web*: 18% of *SQL Injection (i)* instances are classified as *Bruteforce Web (g)*. This outcome suggests a certain similarity between these two attacks. Notably, 9% of *Benign (a)* biflows are confused with various attacks, including 5% as *DDoS LOIC-HTTP (k)* and 2% as *DDoS HOIC (j)*. Starting from these results, we aggregate the attack classes into a single malicious class (cf. Section IV-B). We observe that the FPR and Recall achieve satisfactory values, i.e., 0.8% and 98.7%, respectively. To deepen the investigation, Figure 9(c) presents the projection of the feature vectors υ produced by the embedding function trained with RFS into a two-dimensional space via the *t-Distributed Stochastic Neighbor Embedding (t-SNE)*. This method allows us to visualize the quality of the output produced by the embedding CNN [49]. In fact, given that t-SNE aims to preserve the local structure of υ—i.e., nearby (*resp.* distant) elements should also be close (*resp.* far) in the two-dimensional space—we can observe how the classes are organized into well-defined clusters. This outcome is particularly useful when the classification is carried out with a distance metric applied on feature vectors (as for RFS). Only minor overlapping can be observed between *Bruteforce Web (g)*, *Bruteforce XSS (h)*, and *SQL Injection (i)*, hence confirming the effectiveness of the Euclidean distance used by RFS.



(a) $\mathcal{C}_{all}^{\mathtt{CIC}}$ (b) $\mathcal{C}_{all}^{\mathtt{NID}}$ (c) $\mathcal{C}_{all}^{\mathtt{CIC}}$ (d) $\mathcal{C}_{all}^{\mathtt{NID}}$

FIGURE 9. Confusion matrices of RFS when tested with tasks having $K_s = 10$ on (a) $\mathcal{C}_{all}^{\mathtt{CIC}}$ and (b) $\mathcal{C}_{all}^{\mathtt{NID}}$. Feature vectors projected in a two-dimensional space via t-SNE on (d) $\mathcal{C}_{all}^{\mathtt{CIC}}$ and (c) $\mathcal{C}_{all}^{\mathtt{NID}}$. For (a) and (b), white lines separate $\mathcal{C}_{old}$ from $\mathcal{C}_{new}$, which are the last three classes.

CSE-CIC-IDS2018 label encoding:
a = Benign, b = Bot, c = DoS Hulk, d = DoS GoldenEye, e = DoS Slowloris, f = B-SSH*, g = B-Web*, h = B-XSS*, i = SQL Inj. j = DDoS HOIC, k = DDoS LOIC-HTTP, l = DDoS LOIC-UDP, * B stands for Bruteforce.

IoT-NID label encoding:
α = Benign, β = DoS SYN F*, γ = Port Scanning, δ = Host Discovery, ϵ = OS Detection, ζ = MITM ARP S†, η = Mirai Host B‡, θ = Mirai ACK F*, ι = Mirai HTTP F*, κ = Mirai UDP F*, * F stands for Flooding; † S stands for Spoofing; ‡ B stands for Bruteforce.

Figure 9(b) showcases the confusion matrix of the classes in IoT-NID ($\mathcal{C}_{all}^{\mathtt{NID}}$). The examination of $\mathcal{C}_{new}$ reveals a 7% misclassification rate where *Mirai HTTP Flooding* (ι) biflows are erroneously classified as *Mirai Host Bruteforce* (η). Conversely, 11% of *Mirai Host Bruteforce* (η) biflows are mistakenly classified as *Mirai HTTP Flooding* (ι). Regarding $\mathcal{C}_{old}$, 9% of *MITM ARP Spoofing* (ζ) biflows are confused with *Benign* (α) and *OS Detection* (ϵ). This confusion might stem from the similarity in ARP traffic among these three classes. On the other hand, a specular trend emerges: 2% of *Benign* (α) biflows are wrongly classified as *Os Detection* (ϵ) and another 2% as *MITM ARP Spoofing* (ζ). Differently from the results obtained on CSE-CIC-IDS2018, the Recall tends to be lower (87.8%), while the FPR is relatively the same (0.7%). In Figure 9(d), we depict the two-dimensional projection also for the samples of $\mathcal{C}_{all}^{\mathtt{NID}}$. It can be noticed that the previous observations on ARP traffic are reflected in this figure. Benign traffic is divided into three clusters, two of which are partially overlapped with the projections of *MITM ARP Spoofing* (ζ).

Cross-Validation on Different Datasets. We further enrich our analysis by cross-validating RFS on different

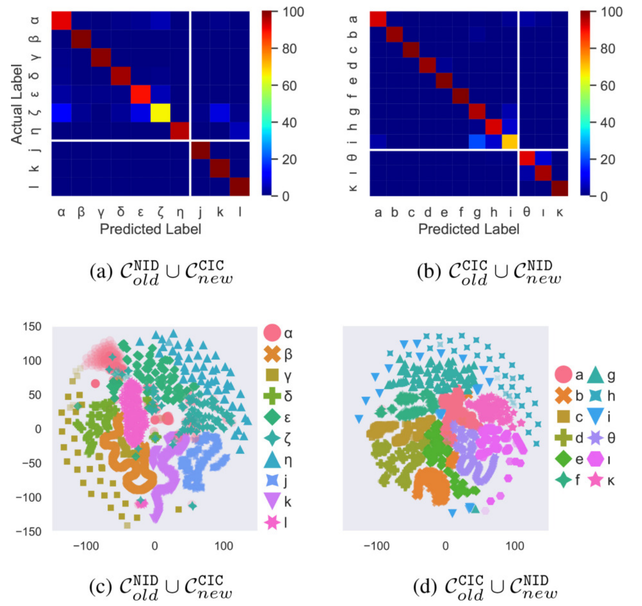


FIGURE 10. Confusion matrices of RFS when tested with tasks having $K_s = 10$ on (a) $\mathcal{C}_{old}^{\text{NID}} \cup \mathcal{C}_{new}^{\text{CIC}}$ and (b) $\mathcal{C}_{old}^{\text{CIC}} \cup \mathcal{C}_{new}^{\text{NID}}$. Feature vectors projected in a two-dimensional space via t-SNE on (c) $\mathcal{C}_{old}^{\text{NID}} \cup \mathcal{C}_{new}^{\text{CIC}}$ and (d) $\mathcal{C}_{old}^{\text{CIC}} \cup \mathcal{C}_{new}^{\text{NID}}$. Label encoding is the same as reported in Figure 9.

datasets, i.e., by evaluating the effectiveness in classifying previously-unseen attacks for which few labeled samples are available *when the model has been trained on the traffic of another dataset*. The objective is twofold: (*i*) *Assessing the resilience of an already trained NIDS in a context where it faces also attacks with a different nature*; this analysis allows for an evaluation of how well the existing model generalizes and adapts to diverse attack scenarios, shedding light on its robustness in varied landscapes. (*ii*) *Validating the possibility of training a NIDS in a simpler scenario* (e.g., a controlled laboratory environment or a scenario with only internal attackers) *and effortlessly adapting it to more complex cases* (e.g., an open-world environment or a scenario comprising external attackers) even with a very limited number of samples for new classes. In line with the previous analyses and the datasets considered (see Section IV-A for details), we discuss the performance when $\mathcal{C}_{old}$ are taken from one dataset and $\mathcal{C}_{new}$ from the other one: namely, $\mathcal{C}_{old}^{\text{NID}} \cup \mathcal{C}_{new}^{\text{CIC}}$ and $\mathcal{C}_{old}^{\text{CIC}} \cup \mathcal{C}_{new}^{\text{NID}}$.

Figure 10(a) assesses the performance for $\mathcal{C}_{old}^{\text{NID}} \cup \mathcal{C}_{new}^{\text{CIC}}$. The 3 DDoS attacks from `CSE-CIC-IDS2018` are perfectly classified. Such a behavior is also confirmed in Figure 10(c), where the projections of these classes are grouped in distinct clusters. However, concerning the training classes of `IoT-NID`, a misclassification pattern is evident: 8% of the biflows of *MITM ARP Spoofing* (ζ) are confused with *DDoS LOIC-HTTP (k)*. This discrepancy is likely due to the inadequate representation of these classes in the embedded space. There is a slight decline compared to the evaluation on $\mathcal{C}_{all}^{\text{NID}}$, with the FPR standing at 1% and the Recall at 86.5%.

On the other side, Figure 10(b) presents the confusion matrix obtained with $\mathcal{C}_{old}^{\text{CIC}} \cup \mathcal{C}_{new}^{\text{NID}}$. Differently than Figure 9(a), a better separation between $\mathcal{C}_{new}$ and $\mathcal{C}_{old}$ emerges. However, there is a slight confusion among $\mathcal{C}_{new}^{\text{NID}}$. Especially, 7% of the *Mirai ACK Flooding* (θ) biflows are misclassified with *Mirai HTTP Flooding* (ι). The remaining classes of $\mathcal{C}_{old}^{\text{CIC}}$ do not exhibit noteworthy differences from previously discussed observations. The lack of noteworthy shifts is also confirmed by the FPR (0.7%) and Recall (96.8%). These outcomes are further confirmed in Figure 10(d). Indeed, the clusters related to $\mathcal{C}_{old}^{\text{CIC}}$ remain relatively similar to their respective clusters in Figure 9(c), while a slight overlap can be noticed between *Mirai ACK Flooding* (θ) and *Mirai HTTP Flooding* (ι).

Take-Home Messages. *When tested on `CSE-CIC-IDS2018` or `IoT-NID`, the NIDS based on the best-performing `RFS` approach is able to distinguish previously-unseen attacks with just* 10 *biflows for each attack-traffic class not observed during training. Minor confusion exists between specific pairs of classes. The cross-validation on different datasets reveals the approach to be robust against DDoS attacks as captured in diverse scenarios compared to those considered in the training phase. However, misclassifications tend to be slightly higher, emphasizing the need for further refinement in handling diverse attack scenarios.*

D. FEATURE-ABLATION STUDY

In this section, we investigate the performance of the proposed `RFS`-based NIDS with $K_s = 10$ when ruling out the 6 input features one at a time (cf. Section III-B1), namely packet length (`PL`), inter-arrival time (`IAT`), TCP window size (`WIN`), time-to-live (`TTL`), TCP flags (`FLG`), and direction (`DIR`). *Such an ablation study aims to evaluate which features help `RFS` the most in classifying benign and attack traffic.*

Table 6 displays the values of the F1-score related to this analysis conducted on `CSE-CIC-IDS2018` and `IoT-NID`. Furthermore, the distributions of the values for the 6 features for each attack and benign traffic are illustrated in Figure 11 (left) via box plots. This provides a more comprehensive overview of the characteristics of malicious and benign traffic in both datasets, thus allowing us to provide an initial explanation of the attained results.

Ablation Analysis on `CSE-CIC-IDS2018`. The removal of any feature does not yield significant improvements in classifying $\mathcal{C}_{new}$ (being always lower than the confidence interval over the number of episodes). In detail, the ablation of `PL`, `WIN`, `TTL`, and `FLG` results in a small F1-score boost—up to +1.06% when removing `WIN`. Regarding the `WIN` feature, the observed improvement is not due to a better classification of $\mathcal{C}_{new}$ biflows (i.e., the true positives are almost unchanged compared to the case without ablation) but rather to confusion between *Benign* and $\mathcal{C}_{new}$. In fact, 7% of *Benign* biflows that were previously misclassified as *DDoS LOIC-HTTP* or *DDoS HOIC* have now decreased to 4%. This can be explained by observing how the distribution of `WIN` for the *Benign* class is similar to that of *DDoS HOIC*, as shown in Figure 11. A different trend emerges

TABLE 6. F1-scores [%] of RFS for different *ablated features* (*AF*) when $K_s = 10$. Results achieved with the complete set of features are highlighted in bold. Red and green colors indicate performance losses and gains, respectively, compared to the scenario without feature removal. The confidence interval is always < 2%.

	CSE-CIC-IDS2018			IoT-NID		
AF	$\mathcal{C}_{new}$	$\mathcal{C}_{old}$	$\mathcal{C}_{all}$	$\mathcal{C}_{new}$	$\mathcal{C}_{old}$	$\mathcal{C}_{all}$
None	**98.18**	**94.02**	**95.06**	**94.64**	**91.51**	**92.45**
PL	98.33 +0.15	87.78 −6.24	90.41 −4.65	95.36 +0.72	92.20 +0.69	93.15 +0.70
IAT	95.56 −2.62	91.37 −2.65	92.42 −2.64	94.38 −0.26	91.47 −0.04	92.35 −0.10
WIN	99.24 +1.06	94.29 +0.27	95.53 +0.47	93.89 −0.75	89.52 −1.99	90.83 −1.62
TTL	99.15 +0.97	93.78 −0.24	95.12 +0.06	82.70 −11.94	90.95 −0.56	88.48 −3.97
FLG	98.60 +0.42	93.53 −0.49	94.80 −0.26	92.24 −2.40	90.75 −0.76	91.20 −1.25
DIR	97.41 −0.77	95.58 +1.56	96.03 +0.97	93.43 −1.21	91.48 −0.03	92.07 −0.38

when ablating IAT and DIR. The removal of the former leads to a higher percentage (9%) of *DDoS LOIC-HTTP* biflows misclassified as *Benign*. This is likely because the IAT exhibits distributions substantially different between the classes (see Figure 11), easing their differentiation. A similar consideration holds when ablating DIR but involving fewer classes (e.g., *Bruteforce* and *SQL Injection* attacks), hence with less impact on results.

On the other hand, different outcomes can be observed on $\mathcal{C}_{old}$. While the ablation of WIN, TTL, and FLG has again a minor effect, removing the PL has a non-negligible negative impact ($\approx -6\%$) on the F1-score. Such feature plays indeed a relevant role in distinguishing between *Bruteforce Web* and *SQL Injection* attacks, which would otherwise be too similar when compared in terms of the remaining 5 features, as illustrated in Figure 11. The ablation of IAT affects $\mathcal{C}_{old}$ similarly to what is noticed on $\mathcal{C}_{new}$. On the contrary, the opposite behavior is observed when removing DIR (and to a lesser extent WIN): in this case, *Bruteforce-Web*/*Bruteforce-XSS* and *SQL Injection* share a distribution of DIR and WIN values extremely similar. Therefore, the elimination of these features leads to an improved separability between these attack classes.

Ablation Analysis on IoT-NID. The ablation study reveals two noteworthy outcomes regarding $\mathcal{C}_{new}$. First, when removing PL the F1-score gains +0.72%. Such an improvement is primarily attributed to a smaller percentage of *Mirai HTTP Flooding* biflows mistakenly classified as *Mirai Host Bruteforce*. This phenomenon can be explained by the analogous PL distribution between the two attacks, as depicted in the right part of Figure 11. Secondly, removing TTL leads to a substantial performance drop of $\approx -12\%$. The reason behind this result lies mainly in the confusion between *Mirai ACK* and *HTTP Flooding*. Specifically, the packets of the former attack mostly have a TTL set to 64, while those of the latter have it set to 128. This discrepancy in TTL values significantly influences the classification process, highlighting a relevant role in distinguishing between the two aforementioned DDoS attacks. However, this finding raises concerns about the vulnerability of the model to the manipulation of TTL values, potentially enabling evasion strategies. The removal of the remaining features results in relatively minor shifts in performance. Only FLG constitutes a notable exception: removing it produces a performance decrease of $\approx -2\%$. This drop can be attributed to a general degradation of the misclassification errors that were already present in the model trained without feature ablation.

Regarding the performance on $\mathcal{C}_{old}$, a slight improvement is observed upon the removal of PL. The cause can be ascribed to the improved separability between *OS Detection* and *MITM ARP Spoofing*. However, it should be noted that malicious samples classified as *Benign* slightly increase in number, highlighting the effectiveness of this feature in distinguishing malicious and benign traffic (which are characterized by distinct distributions of PL). The opposite behavior (i.e., −2% in F1-score) is observed when removing WIN. Specifically, this ablation leads to an increase in confusion between *MITM ARP Spoofing* and *Benign*. Similarly to previous cases, the rationale behind this shift becomes apparent when examining Figure 11, where it is detected that the distribution of WIN diverges slightly between the two classes.

Take-Home Messages. Experimental results show that removing any of the features leads to a drop or a negligible gain in the F1-score when classifying $\mathcal{C}_{new}$. This demonstrates that each selected feature is essential for the NIDS to effectively handle all the attacks of the threat scenario. The drop is more significant when considering the effect of ablation on $\mathcal{C}_{old}$. Notably, the absence of TTL leads to a significant $\approx -12\%$ F1-score decrease due to increased confusion between Mirai ACK and HTTP Flooding, thus raising challenges concerning the vulnerability of the NIDS to TTL manipulation. Nevertheless, this outcome also reveals insights into the importance of this feature along with the other ones chosen to feed our RFS-based NIDS.

E. SENSITIVITY TO THE TEMPORAL THRESHOLD

This last analysis showcases the performance of our NIDS as the threshold T_s varies. The value chosen for T_s influences the *time-to-insight*, namely the time elapsed from the arrival of the first packet of a biflow to its classification (then driving the actions/countermeasures of the network administrator). In fact, the other terms that affect such time (i.e., the time needed for input-data extraction and feature embedding) are

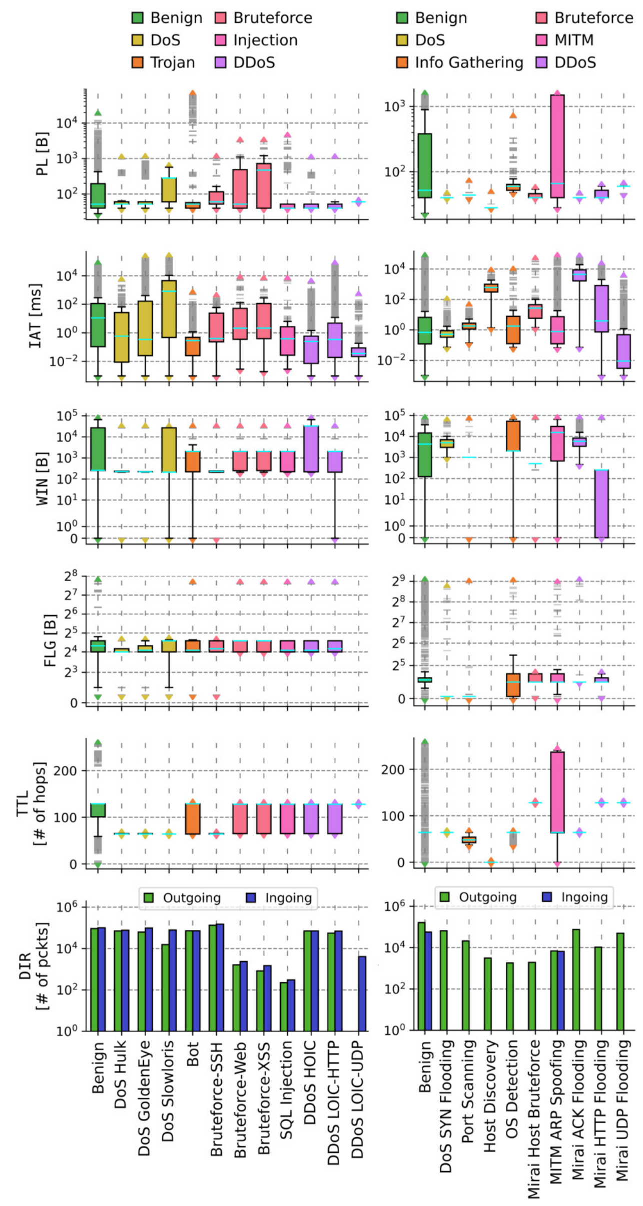


FIGURE 11. Data distribution of CSE-CIC-IDS2018 (left) and IoT-NID (right) classes. Regarding the box plots, the cyan line represents the median value, outliers are depicted with gray dashes, ▲ is the maximum value, whereas ▼ is the minimum. Traffic classes are grouped based on their threat category. PL, IAT, and DIR are represented in *log* scale, WIN and FLG in *symlog* for visualization purposes.

significantly lower compared to T_s values on the order of tens of seconds.

The value of T_s is expected to impact the performance of the NIDS as follows: when T_s is low (e.g., less than a minute), we expect degraded performance of the NIDS, albeit with a reduced time-to-insight; conversely, higher T_s values should lead to performance enhancement due to the availability of more data (i.e., more packets per biflow) but with a longer time-to-insight, thus potentially increasing vulnerability to slow flooding attacks. *Therefore, we aim at finding a good trade-off between classification effectiveness and time-to-insight.*

Based on the results of the previous investigations, for this analysis, we employ RFS with $K_s = 10$ and all the 6 input features. We focus on CSE-CIC-IDS2018, as it contains biflows of longer duration, e.g., slow flooding attacks. This analysis has no practical interest for the traffic in IoT-NID as 90% of its biflows complete within 0.95*s*.

Figure 12 presents the confusion matrices obtained by varying $T_s \in \{1s, 5s, 10s, 50s, 100s, \mathtt{MAX_b}\}$, where $\mathtt{MAX_b} = 1340s$ is the maximum biflow duration in CSE-CIC-IDS2018. As anticipated, the F1-scores in Figure 12 show a gain of +9% passing from $T_s = 1s$ to $T_s = 10s$. Specifically, with $T_s = 1s$, the percentage of biflows affected by the threshold equals 41.17%, while it decreases to 22.88% at $T_s = 10s$. Despite setting $T_s = 1s$ reduces the number of packets to be capitalized for a significant number of biflows (i.e., 41.17% of CSE-CIC-IDS2018 samples), the F1-score is partially affected down to 86%. This outcome is mainly due to the higher rate of false-positive misclassifications (16.6%): higher values for T_s are required to decrease the ratio of undetected intrusions. In fact, increasing T_s beyond 10*s* exhibits a substantial plateau with only negligible fluctuations. Setting T_s to $\mathtt{MAX_b}$ does not yield performance improvements, indicating that a short T_s (e.g., $T_s = 10s$) allows for an excellent trade-off between time-to-insight and classification performance.

The confusion matrices in Figure 12 allow us to delineate a fine-grained investigation of the benign and attack traffic which discloses three interesting misclassification patterns:

i) As discussed in Section V-C, a confusion between the three attack types *Bruteforce-Web (g)*, *Bruteforce-XSS (h)*, and *SQL Injection (i)* is evident. The pattern remains consistent regardless of the T_s value, suggesting a common profile for these attacks.

ii) *DoS Slowloris (e)* is confused with *DoS GoldenEye (d)* or *DoS HULK (c)*. This misclassification is particularly noticeable for short T_s values: to address this issue, a minimum observation time of $T_s \geq 10s$ is recommended. Indeed, the packets generated by this attack tend to have higher IAT values, as evident by the IAT distribution in Figure 11.

iii) Lastly, some *Benign* biflows *(a)* are misclassified as *DDoS LOIC-HTTP (k)*. The confusion decreases for higher T_s: from 9.5% of misclassified benign samples with $T_s = 1s$ to 2.8% with $T_s = 100s$. Nonetheless, a good compromise is reached at $T_s = 50s$, with 3% of benign samples erroneously classified.

Take-Home Messages. This analysis shows that properly adjusting the threshold T_s can improve both the time-to-insight and the accuracy of a NIDS. An appropriate trade-off is needed based also on the attack types to be detected—for instance, certain slow DoS attacks require more time (e.g., $T_s = 50s$) for accurate detection. Overall, by setting a short T_s (e.g., equal to 10s), the NIDS can achieve satisfactory performance without sacrificing a prompt time-to-insight.

VI. CONCLUSION AND FUTURE PERSPECTIVES

In order to deal with the constant evolution of the threat landscape, in this paper, we designed a NIDS based on the

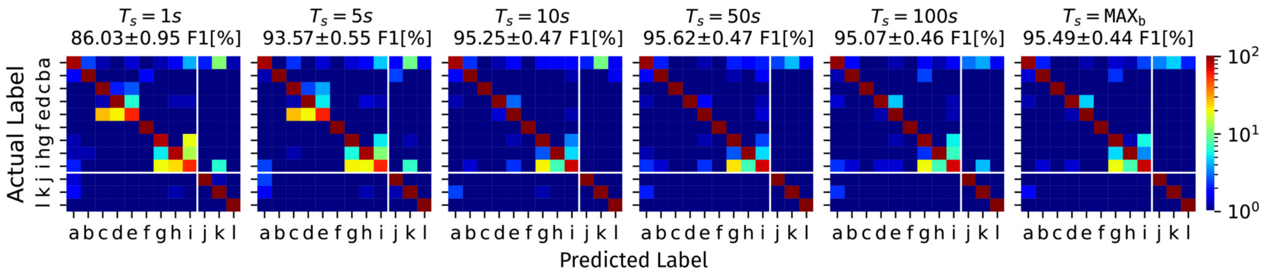


FIGURE 12. Confusion matrices of RFS when tested with $K_s = 10$ tasks and different T_s values. Results are in *log* scale to enhance small errors.

CSE-CIC-IDS2018 label encoding: a = Benign, b = Bot, c = DoS Hulk, d = DoS GoldenEye, e = DoS Slowloris, f = B-SSH*, g = B-Web*, h = B-XSS*, i = SQL Inj., j = DDoS HOIC, k = DDoS LOIC-HTTP, l = DDoS LOIC-UDP; * B stands for Bruteforce.

FSCIL paradigm. Our proposal can adapt to classify attacks not observed at the training phase when only few samples of each (i.e., less than 10) become available. It demonstrates excellent and rapid adaptability to *new attacks* ($\mathcal{C}_{new}$) while retaining knowledge of *old traffic classes* ($\mathcal{C}_{old}$). This work delved into the key aspects that define our proposed NIDS.

Our key findings are as follows: *(i)* a comparison between three meta-learning approaches and three transfer-learning approaches revealed that RFS offers excellent performance on both $\mathcal{C}_{new}$ and $\mathcal{C}_{old}$; *(ii)* RFS outperforms $\text{SOTA}_{\text{nonFSCIL}}$ on $\mathcal{C}_{all}$ (up to a +5% F1-score) and slightly underperforms (−4% F1-score, at most) on $\mathcal{C}_{new}$—with only 10 samples, unlike $\text{SOTA}_{\text{nonFSCIL}}$ not imposing constraints on the (limited available) number of $\mathcal{C}_{new}$ samples for training; *(iii)* FSCIL approaches adapt to $\mathcal{C}_{new}$ in a time 3 orders of magnitude shorter than $\text{SOTA}_{\text{nonFSCIL}}$; *(iv)* a cross-validation analysis confirmed that RFS is effective in detecting DDoS not seen during training and related to attack scenarios different from those already observed; *(v)* the whole set of features characterizing the input proved necessary for the classification of various considered cyberattacks; *(vi)* the cut to the input packets based on a temporal threshold enables the NIDS to output its verdict in a timely manner still guaranteeing satisfactory results (95% F1-score when the threshold is set to 10 seconds).

Limitations and Future Avenues. We envision several promising avenues for future research based on the findings presented in this paper, and that will allow us to also deal with possible *limitations* of the proposed approach: *(i)* addressing the *robustness of FSCIL-based NIDSs against the poisoning of input data* by exploring the use of *ad-hoc adversarial learning* for attack-traffic classification, therefore providing a deeper understanding of how FSCIL approaches behave in this context; *(ii)* coping with the *scarcity of available samples* of new attack classes via the integration of *generative techniques* to augment the support set—e.g., conditional variational autoencoders, generative adversarial networks, or generative pre-trained transformers; *(iii)* dealing with *forgetting and poor adaptation* to new classes through the adoption of advanced strategies borrowed from CIL approaches—e.g., regularization [37] and bias correction [38]; *(iv) shedding light on the "black box"* of DL-based FSCIL NIDSs and providing understandable insights into their decision-making process [50] adopting *eXplainable AI* techniques (e.g., SHAP, Integrated Gradients, GradCAM) in line with recent regulatory frameworks concerning this matter—e.g., the EU AI Act [51].

ACKNOWLEDGMENT

This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.

REFERENCES

[1] (World Economic Forum, Cologny, Switzerland). *How Does Your Industry Compare When It Comes to the Financial Loss Exposure of Cyber Threats?* (Mar. 2023). [Online]. Available: https://www.weforum.org/agenda/2023/03/how-does-your-industry-compare-when-it-comes-to-the-financial-impact-of-cyber-threats/

[2] A. Petrosyan. "Monetary Loss of Companies in the United States as a Result of Cyber Attacks as of 2022." Feb. 2024. [Online]. Available: https://www.statista.com/statistics/1334399/us-common-results-of-cyber-attacks/

[3] O. Yoachimik. "Cloudflare DDoS threat report 2022 Q3." Dec. 2022. [Online]. Available: https://blog.cloudflare.com/cloudflare-ddos-threat-report-2022-q3/

[4] D. Jones. "Novel zero-day exploits fuel Q3 surge in DDoS attacks." Oct. 2023. [Online]. Available: https://www.cybersecuritydive.com/news/zero-day-surge-ddos-attacks/697928/

[5] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescape, "AI-powered Internet traffic classification: Past, present, and future," *IEEE Commun. Mag.*, vol. 62, no. 9, pp. 168–175, Sep. 2024, doi: 10.1109/mcom.001.2300361.

[6] Y. Sun, H. Esaki, and H. Ochiai, "Adaptive intrusion detection in the networking of large-scale LANs with segmented federated learning," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 102–112, 2021, doi: 10.1109/OJCOMS.2020.3044323.

[7] J. Kirkpatrick et al, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci.*, vol. 114, no. 13, pp. 3521–3526, 2017. doi: 10.1073/pnas.1611835114.

[8] G. Bovenzi, D. Di Monda, A. Montieri, V. Persico, and A. Pescapè, "Classifying attack traffic in IoT environments via few-shot learning," *J. Inf. Security Appl.*, vol. 83, Jun. 2024, Art. no. 103762, doi: 10.1016/j.jisa.2024.103762.

[9] C. Constantinides, S. Shiaeles, B. Ghita, and N. Kolokotronis, "A novel online incremental learning intrusion prevention system," in *Proc. 10th IFIP Int. Conf. New Technol., Mobility Security (NTMS)*, 2019, pp. 1–6, doi: 10.1109/ntms.2019.8763842.

[10] S. Huang et al., "A gated few-shot learning model for anomaly detection," in *Proc. IEEE Int. Conf. Inf. Netw. (ICOIN)*, 2020, pp. 505–509, doi: 10.1109/icoin48656.2020.9016599.

[11] 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 *Proc. IEEE Int. Conf. Commun. (ICC)*, 2021, pp. 1–6, doi: 10.1109/icc42927.2021.9500667.

[12] C. Rong, G. Gou, C. Hou, Z. Li, G. Xiong, and L. Guo, "UMVD-FSL: Unseen malware variants detection using few-shot learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2021, pp. 1–8, doi: 10.1109/ijcnn52387.2021.9533759.

[13] T. Wang, Q. Lv, B. Hu, and D. Sun, "A few-shot class-incremental learning approach for intrusion detection," in *Proc. IEEE Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2021, pp. 1–8, doi: 10.1109/icccn52240.2021.9522260.

[14] W. Liang, Y. Hu, X. Zhou, Y. Pan, and K. I.-K. Wang, "Variational few-shot learning for microservice-oriented intrusion detection in distributed industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5087–5095, Aug. 2022, doi: 10.1109/tii.2021.3116085.

[15] M. Data and M. Aritsugi, "T-DFNN: An incremental learning algorithm for intrusion detection systems," *IEEE Access*, vol. 9, pp. 154156–154171, 2021, doi: 10.1109/access.2021.3127985.

[16] M. Data and M. Aritsugi, "AB-HT: An ensemble incremental learning algorithm for network intrusion detection systems," in *Proc. IEEE Int. Conf. Data Sci. Appl. (ICoDSA)*, 2022, pp. 47–52, doi: 10.1109/ICoDSA55874.2022.9862833.

[17] 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 Things J.*, vol. 10, no. 24, pp. 21309–21321, Dec. 2023, doi: 10.1109/jiot.2023.3283408.

[18] Z. Song et al., "I^2 RNN: An incremental and interpretable recurrent neural network for encrypted traffic classification," *IEEE Trans. Dependable Secure Comput.*, early access, Feb. 28, 2023, doi: 10.1109/tdsc.2023.3245411.

[19] M. Pawlicki, R. Kozik, and M. Choraś, "Improving siamese neural networks with border extraction sampling for use in real-time network intrusion detection," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2023, pp. 1–8, doi: 10.1109/ijcnn54540.2023.10191496.

[20] F. Cerasuolo, G. Bovenzi, C. Marescalco, F. Cirillo, D. Ciuonzo, and A. Pescapè, "Adaptive intrusion detection systems: Class incremental learning for IoT emerging threats," in *Proc. IEEE Int. Conf. Big Data (BigData)*, 2023, pp. 3547–3555, doi: 10.1109/BigData59044.2023.10386129.

[21] D. Di Monda, G. Bovenzi, A. Montieri, V. Persico, and A. Pescapè, "IoT botnet-traffic classification using few-shot learning," in *Proc. IEEE Int. Conf. Big Data (BigData)*, 2023, pp. 3284–3293, doi: 10.1109/BigData59044.2023.10386602.

[22] P.-F. Marteau, "Random partitioning forest for point-wise and collective anomaly detection—Application to network intrusion detection," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2157–2172, 2021, doi: 10.1109/tifs.2021.3050605.

[23] G. Bovenzi, G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "Network anomaly detection methods in IoT environments via deep learning: A fair comparison of performance and robustness," *Comput. Security*, vol. 128, May 2023, Art. no. 103167, doi: 10.1016/j.cose.2023.103167.

[24] P. Casas, J. Mazel, and P. Owezarski, "Unsupervised network intrusion detection systems: Detecting the unknown without knowledge," *Comput. Commun.*, vol. 35, no. 7, pp. 772–783, Apr. 2012, doi: 10.1016/j.comcom.2012.01.016.

[25] E. Paolini, L. Valcarenghi, L. Maggiani, and N. Andriolli, "Real-time network packet classification exploiting computer vision architectures," *IEEE Open J. Commun. Society*, vol. 5, pp. 1155–1166, 2024, doi: 10.1109/OJCOMS.2024.3363082.

[26] A. Nascita, F. Cerasuolo, D. Di Monda, J. T. A. Garcia, A. Montieri, and A. Pescapè, "Machine and deep learning approaches for IoT attack classification," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–6, doi: 10.1109/infocomwkshps54753.2022.9797971.

[27] R. Raman, A. K. Sahu, V. K. Nair, and P. Nedungadi, "Opposing agents evolve the research: A decade of digital forensics," *Multimedia Tools Appl.*, to be published, doi: 10.1007/s11042-024-19519-8.

[28] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, and K. Kavukcuoglu, "Matching networks for one-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–11, doi: 10.48550/arXiv.1606.04080.

[29] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–13, doi: 10.48550/arXiv.1703.05175.

[30] 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 *Proc. IEEE Comput. Vis. Pattern Recognit. Conf. (CVPR)*, 2018, pp. 1199–1208, doi: 10.1109/cvpr.2018.00131.

[31] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1–13, doi: 10.48550/arXiv.1703.03400.

[32] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? towards understanding the effectiveness of MAML," 2019, *arXiv:1909.09157*, doi: 10.48550/arXiv.1909.09157.

[33] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf. (CVPR)*, 2019, pp. 1–9.

[34] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," 2019, *arXiv:1904.04232*, doi: 10.48550/arXiv.1904.04232.

[35] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola, "Rethinking few-shot image classification: A good embedding is all you need?" in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 266–282, doi: 10.1007/978-3-030-58568-6_16.

[36] B. Liu et al., "Negative margin matters: Understanding margin in few-shot classification," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 438–455, doi: 10.1007/978-3-030-58548-8_26.

[37] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018, doi: 10.1109/tpami.2017.2773081.

[38] Y. Wu et al., "Large scale incremental learning," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf. (CVPR)*, 2019, pp. 1–9, doi: 10.48550/arXiv.1905.13260.

[39] G. Bovenzi et al., "A first look at class incremental learning in deep learning mobile traffic classification," 2021, *arXiv:2107.04464*, doi: 10.48550/arXiv.2107.04464.

[40] 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*, vol. 5, pp. 18042–18050, 2017, doi: 10.1109/access.2017.2747560.

[41] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolutional neural networks," in *Proc. IEEE Int. Conf. Intell. Security Inform. (ISI)*, 2017, pp. 43–48, doi: 10.1109/isi.2017.8004872.

[42] W. Zhijun, L. Wenjing, L. Liang, and Y. Meng, "Low-rate DoS attacks, detection, defense, and challenges: A survey," *IEEE Access*, vol. 8, pp. 43920–43943, 2020, doi: 10.1109/ACCESS.2020.2976609.

[43] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSp*, vol. 1, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[44] 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 Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019, doi: 10.1016/j.future.2019.05.041.

[45] K. Hyunjae, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. Kim, 2019, "IoT network intrusion dataset," IEEE Dataport. [Online]. Available: https://ieee-dataport.org/open-access/iot-network-intrusion-dataset

[46] S. M. Arnold, P. Mahajan, D. Datta, I. Bunner, and K. S. Zarkias, "Learn2learn: A library for meta-learning research," 2020, *arXiv:2008.12284*, doi: 10.48550/arXiv.2008.12284.

[47] W. Li et al, "LibFewShot: A comprehensive library for few-shot learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 12, pp. 14938–14955, Dec. 2023, doi: 10.1109/tpami.2023.3312125.

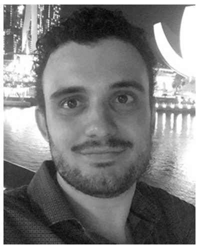
[48] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surveys*, vol. 53, no. 3, p. 63, 2020, doi: 10.1145/3386252.

[49] J. Xue, Y. Chen, O. Li, and F. Li, "Classification and identification of unknown network protocols based on CNN and T-SNE," in *Proc. J. Phys. Conf. Series*, 2020, Art. no. 12071, doi: 10.1088/1742-6596/1617/1/012071.

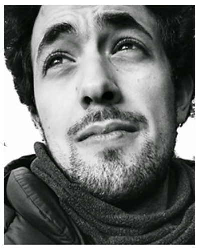
[50] Z. Abou El Houda, B. Brik, and L. Khoukhi, "Why should I trust your IDS?": An explainable deep learning framework for intrusion detection systems in Internet of Things networks," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 1164–1176, 2022, doi: 10.1109/OJCOMS.2022.3188750.

[51] "The EU artificial intelligence act (AI Act)," European Union, Jun. 2024. [Online]. Available: https://artificialintelligenceact.eu/

DAVIDE DI MONDA received the M.S. degree (summa cum laude) in computer engineering from the University of Napoli Federico II in July 2022. He is currently pursuing the Ph.D. degree with the IMT School for Advanced Studies Lucca and the University of Napoli Federico II. His research interests include cybersecurity, attack classification, and blockchain.



ANTONIO MONTIERI received the Ph.D. degree in information technology and electrical engineering in April 2020. He is an Assistant Professor with the DIETI, University of Napoli Federico II. He has co-authored 48 papers in international journals and conference proceedings. His work concerns network measurements, encrypted traffic classification, traffic modeling and prediction, and network security. He received the Elsevier Computer Networks 2020 Best Paper Award, the Best Paper Awards at IEEE ICCCS 2019 and IEEE ISCC 2022, and the Best Poster Award at the TMA Ph.D. School 2016.



VALERIO PERSICO received the Ph.D. degree in computer and automation engineering from the University of Napoli Federico II in 2016, where he is an Assistant Professor with DIETI. He has co-authored more than 50 papers in international journals and conference proceedings. His work concerns network measurements, cloud-network monitoring, and Internet path tracing and topology discovery.

PASQUALE VORIA received the bachelor's degree in computer engineering from the University of Napoli Federico II in December 2021, where he is currently pursuing the M.S. degree. His research interests include network and software security and secure system design.

MATTEO DE IESO received the bachelor's degree in computer engineering from the University of Sannio in February 2023. He is currently pursuing the M.S. degree with the University of Napoli Federico II. His studies focus on cybersecurity in particular network security.

ANTONIO PESCAPÈ (Senior Member, IEEE) is a Full Professor of Computer Engineering with the University of Napoli Federico II. Recently, he is working on bioinformatics and ICTs for a smarter health. He has co-authored more than 200 conference and journal papers. Also, he has served as an independent reviewer/evaluator of research projects/project proposals co-funded by several governments and agencies. His work focuses on Internet technologies and more precisely on measurement, monitoring, and analysis of the Internet. He is the recipient of a number of research awards, including the Elsevier Computer Networks 2020 Best Paper Award.

Open Access funding provided by 'Università degli Studi di Napoli "Federico II"' within the CRUI CARE Agreement