Few Shot Learning Approaches for Classifying Rare Mobile-App Encrypted Traffic Samples

Giampaolo Bovenzi, Davide Di Monda, Antonio Montieri, Valerio Persico, Antonio Pescapé
University of Napoli Federico II, Napoli (Italy) and IMT School for Advanced Studies, Lucca (Italy)
{giampaolo.bovenzi, antonio.montieri, valerio.persico, pescape}@unina.it, davide.dimonda@imtlucca.it

***Abstract*—Deep Learning (DL) is effective for classifying encrypted network traffic. However, it requires large amounts of labeled data to feed typical data-hungry training processes. Unfortunately, collecting and labeling rich network-traffic datasets is a complex and costly procedure not always affordable in practice, possibly hindering DL solutions. Few Shot Learning (FSL) aims at tackling this shortcoming, providing means to leverage non-few knowledge to support classification tasks related to traffic with few labeled data available. Although FSL has been largely investigated in other domains (e.g., computer vision), it has been only preliminarily adopted for the classification of encrypted traffic.**

In this work, we provide a first attempt in adopting FSL for classifying mobile-app encrypted traffic. Specifically, we consider the two most popular FSL paradigms: *meta learning* (learn to learn) and *transfer learning* (knowledge transfer from related tasks). We consider a number of variants for each (namely *MatchingNet*, *ProtoNet*, *RelationNet*, *MetaOptNet*, *fo-MAML*, *ANIL*, *Fine-Tuning*, and *Freezing*) and provide an empirical assessment of these approaches when adopted for mobile-app traffic classification considering the `Mirage-2019` dataset as a test bench. Results show that FSL in mobile-app traffic classification is feasible, reaching satisfactory results (up to 80% F1-score), but leaving room for improvement.

***Index Terms*—Traffic Classification; Mobile Apps; Android Apps; Encrypted Traffic; Deep Learning; Few Shot Learning.**

I. Introduction and Background

The diffusion of mobile devices has dramatically modified the landscape of network traffic, which is rapidly growing in volume and changing in nature. In this context, traffic classification has acquired a more and more fundamental role as it supports a number of activities related to network management, such as resource provisioning, billing, accounting, security (e.g., intrusion detection and prevention) as well as user-activity identification and user profiling.

Indeed, traffic classification is an active research field, with the available approaches to be constantly adapted to the ever-evolving nature of the networks and of the traffic traversing them. For instance, simple approaches relying on port numbers are hindered by port-independent applications or applications using standard ports to disguise their traffic. Similarly, encryption (e.g., via TLS and HTTPS protocols) critically compromises the effectiveness of approaches based on Deep Packet Inspection (DPI) that look for signatures or patterns within cipher-text (that are further challenged by the will of safeguarding user privacy) [1]. To face these shortcomings, traffic classification approaches often take advantage of statistical or time-series features, which are leveraged to train data-driven approaches based on *Machine Learning*. Unfortunately, these approaches may not scale well with traffic classes rapidly evolving over time, as they require domain experts guiding the feature-extraction process. Thus, recent traffic-classification solutions today resort to Deep Learning (DL) approaches that capitalize on large amounts of labeled data to automatically train supervised models to perform proper feature extraction from raw data.

However, collecting large labeled network-traffic datasets is a cumbersome and time-consuming process. In fact, capturing network-traffic traces, splitting them in traffic objects (e.g., biflows), and associating labels to them, often requires dedicated setups [2] and introduces user-privacy and business-sensitivity concerns [3]. In addition, traffic rapidly evolving over time makes things even worse: new apps and new app versions are continuously deployed and introduce drift in traffic. This requires the above process to be run in a continuous way in order to provide large, rich, and up-to-date datasets that are mandatory to feed the data-hungry DL training process. As a result, it is not uncommon for available labeled data to be unbalanced, containing limited examples for some classes, thus hindering the adoption of DL that falls short when only limited knowledge is available for some of the classes.

Few Shot Learning (FSL) aims at tackling this situation, by leveraging non-few knowledge (about tasks related to the few one) in order to build a model capable of generalizing enough on new tasks (i.e. those with few samples available). Indeed, the problem of learning with few samples (viz. *shots*) has strongly attracted the interest of the research community of several research domains, with the way the prior knowledge is exploited defining different families of approaches [4]. FSL was initially tackled and in-depth explored in the computer vision domain. Accordingly, the taxonomy of the available solutions is very rich [4], including: (*i*) *Algorithm-based approaches*, using prior knowledge to alter the search strategy for new parameters, providing a baseline that may be useful for the specific few-shot task [5]; (*ii*) *Model-based approaches*, allowing to jointly learn a set of related initial-tasks in order to reduce the searching space of best parameters [6–8]; (*iii*) *Data-based approaches*, leveraging prior knowledge to augment data of the few-shot tasks.

Given that such approaches proved promising in other domains, recently research efforts have been made to apply FSL also in networking to address network-traffic classification. Hence, most networking-related works dealing with FSL are

typically based on previous computer-vision solutions, with minor variants being proposed [9–11]. In fact, most of the proposals leveraging FSL in networking tackle the problem of attack traffic classification [9–14]. To the best of our knowledge, *we provide the first attempt in applying FSL to mobile-app encrypted traffic classification*. More in general, although the literature on FSL is rich, the studies that apply these solutions to network-traffic classification are limited and do not explore the whole range of possibilities that FSL provides. Often, the evaluation of the solutions investigated in the traffic-classification domain is also limited, as proposals are only compared against others not tailored for FSL [9, 11, 13, 14] (thus, emphasizing the benefits of FSL but not investigating those related to specific FSL families or approaches).

In this work, we provide an *extensive empirical assessment* of two popular FSL paradigms: *meta learning* and *transfer learning*. Meta learning improves the performance of a new task given the meta-knowledge extracted across tasks by a meta-learner. In a few-shot context, meta learning is used jointly with *episodic learning*, which consists in organizing the training phase in a series of learning problems—also called *episodes*—in order to allow a model (i) to quickly learn novel knowledge by generalizing from previously encountered learning tasks and (ii) to distinguish unknown classes given few samples. Differently, *transfer learning* aims to transfer knowledge from a task to a related one with the objective of fast adaptation, reduced complexity, and performance improvements. In the few-shot scenario, the knowledge acquired from abundant classes is used to train an embedding function that is subsequently adapted to a new task leveraging a limited number of samples. Herein, we consider a number of solutions, including `MatchingNet`, `ProtoNet`, `RelationNet`, and `MetaOptNet` from the model-based family, and `fo-MAML`, `ANIL`, *Fine Tuning*, and *Freezing* from the algorithm-based one. We underline that data-based approaches are not considered at all because training a generation model with few real samples may result in introducing biased synthetic samples.

Accordingly, in this paper: (*i*) we investigate *eight FSL approaches that are properly adapted to the traffic classification*, dealing with the problem of scarcity of data in *mobile-app traffic*; (*ii*) we introduce a *meta-learning procedure* that takes advantage of an extra portion of classes (viz. validation classes) to enforce early-stopping in meta-training; (*iii*) we explore *several directions of analysis* by inspecting the impact of using different FSL setups in terms of number of training classes (viz. N) and number of shots (viz. K); we also show the impact of having more classes available for training and deepen the link between embedding effectiveness and classification performance; (*iv*) we leveraged the *publicly available* mobile traffic dataset `Mirage-2019` to foster reproducibility; (*v*) we discuss several *future directions* of improvement.

II. Classifying Mobile-App Traffic with FSL

In this section, we provide a brief description of methodological aspects related to the adaptation of FSL to the mobile-app traffic classification task. *Design choices* involve the definition of the *traffic object* and related *input data* fed to the *embedding function* used to extract relevant features. Then, the specific FSL approach must be selected by identifying both the *FSL paradigm* and the *FSL family*.

A. Input Data and Embedding Function

We consider the *bidirectional flow* (*biflow*) as the relevant traffic object of our analysis. A *biflow* is defined as an aggregation of all network packets sharing the same 5-tuple (i.e. source IP and port, destination IP and port, and transport-level protocol) including both directions of communication.

To feed the FSL methods exploited herein, we extract a set of informative fields from the sequence of the first N_p packets of each biflow: (*i*) the number of bytes in transport layer payload; (*ii*) the packet direction (can be -1 or 1); (*iii*) the TCP windows size (equal to 0 for UDP packets); (*iv*) and the elapsed time since the arrival of the previous packet (i.e. inter-arrival time). The input data are normalized within $[0, 1]$ using a Min-Max normalization.

Additionally, all FSL models embed the input data into a lower-dimensional space using an *embedding function*. The specific embedding function used herein is a state-of-the-art DL network widely used in and suited for the traffic classification domain [15, 16].[1] Specifically, the embedding function is a *single-modal bidimensional convolutional neural network* (2D-CNN) whose architecture and hyperparameters are those originally proposed for traffic classification [16].

B. Few-Shot Learning Paradigms

Meta Learning. Meta-learning approaches require a preliminary phase to manage the generation of *N-way K-shot episodes*. In detail, given a dataset $D = \{D_{nf_1}, D_{nf_2}, D_f\}$, the three subsets have a disjoint label space. D_f includes the samples from the less populated (i.e. few-shot) classes, while D_{nf_1} and D_{nf_2} contain non-few classes. Detailing, each episode is formed by randomly sampling N classes (*N-way*). D_{nf_1}, D_{nf_2}, and D_f are used during training, validation, and testing phases, respectively. Defining an episode consists in constructing two (non overlapping) partitions: (i) the *support set* having $N \times K_s$ samples (where K_s defines the K-*shot* setup) and (ii) the *query set* having $N \times K_q$ samples.

Based on this episode creation, (i) during a *meta-training* phase, the model learns from a set of T tasks—i.e. *N-way K-shot classification tasks*—using samples from the support set and measures the error on the query set. Then, (ii) the generalization ability of the classifier is tested with a *meta-testing* phase, using an analogous episode-based procedure. Unlike the common meta-learning procedure, we perform an additional (iii) *meta-validation* phase leveraging D_{nf_2}.

More specifically, meta-training is performed for a certain number of epochs in an episodic manner. On the other hand, D_{nf_2} is exploited to enforce an early-stopping procedure based on the accuracy attained on it (in a way similar to [17]), and

[1] We have selected both input data and embedding function based on state-of-the-art outcomes [15, 16] and a preliminary experimental campaign whose results are not reported for the sake of brevity.

to select the model showing the best performance on it after the training procedure is completed.[2] Finally, meta-testing is performed on this best-performing model. We remark that during meta-testing the performance achieved by the selected model is properly evaluated on D_f.

Transfer Learning. Transfer learning aims at learning generic features from a set of non-few classes (T_{nf} task) and then specializes in the few-shot context (T_f task). Accordingly, similar to the meta-learning, $D = \{D_{nf_1}, D_f\}$, with the latter encompassing the *few-shot* classes. However, transfer learning needs to further split D_{nf_1} and D_f (e.g., via hold-out) into a *training set* and a *test set*, which are analogous to the support and the query sets in meta learning, respectively. Specifically, T_{nf} learns (resp. evaluates the error) with the training (resp. test) data from D_{nf_1}.[3] Similarly, T_f enriches the knowledge obtained on T_{nf} with a set of data obtained from the samples belonging to D_f. To mimic the support set of the meta-learning procedure, during T_f we sampled from the training set K shots per class.

C. Few-Shot Learning Approaches

The FSL approaches we deal with are described hereinafter. They belong to the model- and algorithm-based families. All (but transfer learning ones) implement meta learning.

Model-based Approaches. Model-based methods learn by constraining the hypothesis space to a smaller one through the use of prior knowledge, with the aim of reducing the risk of overfitting. More specifically, they apply a dedicated embedding function—learned based on prior knowledge extracted from the meta-training tasks—to support and query samples mapping them in a lower-dimensional space. In such a way, similar samples are closer to each other, whereas dissimilar samples are more easily differentiable (i.e. the hypothesis space is reduced). Then, the embeddings of the support set are used by a *comparator* to classify the embedded query set by measuring their similarity.

Model-based approaches differ by the comparator and then the similarity measure/mechanism leveraged. Herein, we consider the following approaches: (*i*) *Matching Networks* (`MatchingNet`) [6] performs a generalized form of nearest-neighbors classification based on Euclidean distance; (*ii*) *Prototypical Network* (`ProtoNet`) [7] classifies a sample based on a Euclidean distance function calculated between its embedding and a representative class centroid—named *prototype*; (*iii*) *Relation Network* (`RelationNet`) [8] exploits a *relation module*—based on a convolutional network—that measures the similarity between the embeddings of query and support samples of each class; (*iv*) `MetaOptNet` [18] employs a *linear Support Vector Machine (SVM)*—trained on labeled support set samples—as the comparator and measures the generalization error on query set samples (all samples are embedded via a common embedding function).

Algorithm-based Approaches. Learning models belonging to the algorithm-based category search in the hypothesis space for the parameter set corresponding to the best hypothesis in such a space. Unfortunately, in few-shot scenarios, the samples available for training are limited to properly update the parameter set, thus resulting in an unreliable risk minimizer. To deal with this issue, algorithm-based methods exploit the prior knowledge to influence how the parameter set is obtained.

In the present paper, we exploit `MAML` [5] and `ANIL` [19], which are meta-learning approaches that aim at *refining meta-learned parameters* by learning an initial parameter set via a meta-learning procedure and further refining it using D_f. `MAML`—short for *Model-Agnostic Meta-Learning*—continuously updates the initial meta-learned parameter set based on the performance attained on the episodic tasks (viz. *inner-loop adaptation*) with the aim of finding a highly adaptable set of parameters. To mitigate the well-known computational burden of `MAML`, herein we employ a simpler but almost equally well-performing version named *First-Order* `MAML` (`fo-MAML`) [5] that only uses first-order gradients during parameter optimization. In addition to `fo-MAML`, we also evaluate `ANIL`—short for *Almost No Inner Loop*—a simplified version of `MAML` being equally effective but computationally faster. `ANIL` removes the inner-loop updates for the embedding function during meta-training and meta-testing and applies them only to the model head.

We also employ two *transfer learning* approaches (i.e. *Fine-Tuning* and *Freezing*) that allow the model to learn an initial parameter set from other tasks (D_{nf_1}) and then refine it using D_f. More specifically, *Fine-Tuning* ($\texttt{TL}_{\texttt{FT}}$) involves: ($i$) learning the initial model parameters through task T_{nf} and, (ii) refining the weights—during the training phase of T_f—starting from the values computed in the preceding task using only samples from few-shot classes. This results in significantly faster execution. On the other hand, $\texttt{TL}_{\texttt{FT}}$ is affected by the problem of *forgetting* the old classes. *Freezing* ($\texttt{TL}_{\texttt{FZ}}$) freezes the weights of the embedding function when a new task is presented and allows the update of the weights associated with the model head (differently from $\texttt{TL}_{\texttt{FT}}$ where both the embedding function and head are updated). The expected result is that the model retains the previously learned knowledge, but also adapts to few-shot classes in case of proper generalization capability.

III. Experimental Setup and Evaluation

A. Experimental Setup

Hereinafter, we describe the experimental setup in terms of (*i*) dataset, (*ii*) FSL setup, and (*iii*) evaluation metrics.

1) Dataset: In this study, we leverage the publicly released `Mirage-2019` dataset [2] containing traffic of 40 Android apps.[4] It was collected at the University of Napoli "Federico

[2]We compute the validation accuracy for each episode, then the per-episode accuracy values are averaged to obtain the mean for each epoch, and finally the best-performing model is selected based on the latter.

[3]A validation set extracted from the training set can be exploited to enforce early-stopping.

[4]https://traffic.comics.unina.it/mirage/mirage-2019.html

II" in 2017–2019 by involving 300 voluntary students mimicking typical app use cases. Ground truth is obtained by using metadata log files collected during each capture: each biflow is labeled with the related Android-package name. The number of biflows per app depends on the specific app and ranges from 361 to 8246, despite the time that apps are used was roughly the same. This denotes a real-world and challenging scenario for the mobile-app traffic classification task considered.

2) Few-Shot Learning Setup: The FSL setup is described by detailing the common meta-learning setup and the configuration of FSL models.

Meta-Learning Setup. We explain the meta-learning setup in terms of dataset partitioning and episode definition. The classes (viz. apps) of `Mirage-2019` are **partitioned** into three disjoint sets corresponding to the apps considered to build D_{nf_1}, D_{nf_2}, and D_f, respectively (Sec. II-B describes their use). More specifically, D_{nf_1} includes the 24 most populous apps of `Mirage-2019`, D_{nf_2} consists of the most populous 8 apps besides those in D_{nf_1}, and the last subset D_f includes the 8 remaining least-populated apps.[5]

Once the dataset partition is set, the definition of **meta-learning episodes** is based on how N (*ways*) and K (*shots*) are selected (see Sec. II-B). For meta-training, we set N and K according to the goals of our analyses (always considering $K = K_s = K_q$). For meta-validation and meta-testing, we set $N = 8$ and $K_q = 100$, to classify the samples belonging to all classes in D_{nf_2} and D_f and improve the coverage and the stability of obtained results.

Configuration of FLS Models. The configuration of FSL models is specified in terms of model-specific and common hyperparameters.[6] Different **model-specific hyperparameters** can be configured for each FSL approach. `ProtoNet` and `MatchingNet` both use the Euclidean distance. `RelationNet` utilizes the relation module proposed in [8]. `MetaOptNet` is tuned with a *regularization parameter* c equal to 0.1 and 15 maximum SVM iterations. `MAML` and `ANIL` are calibrated with an *inner/adaptation learning rate* equal to 0.01 and 8 *inner loop iterations*. The *Cross Entropy Loss* measures the error in every model, but in `RelationNet` and `MatchingNet`, which resort to the *Mean Squared Error* and the *Negative Log Likelihood Loss*, respectively.

Regarding the **common hyperparameters**, characterizing all FSL models, the best configuration has 200 *epochs*, each encompassing 100 *episodes*, and the *Adam* optimizer set with 10^{-4} *learning rate* and a *learning rate scheduler* having *step size* of 20 and *decay* of 1.0. To mitigate the overfitting, we exploit an *early-stopping* mechanism that monitors the *accuracy on* D_{nf_2} and has *minimum delta* of 0.01 and *patience* of 20 epochs. Finally, regarding input data (see Sec. II-A), we use $N_p = 10$. This choice is motivated by the outcome of the sensitivity analysis we conducted in [2] on `Mirage-2019` using the same embedding function considered herein.

[5]We selected the apps of D_f by considering those having less than 1000 samples. Then, we chose for D_{nf_2} the same number of apps.

[6]Hyperparameters choices are based on both state-of-the-art outcomes and a preliminary experimental campaign on D_{nf_2}, not reported for brevity.

3) Performance Metrics: To evaluate the performance of FSL methods, we use the macro F1-score and the silhouette score. For both metrics, we show the per-episode mean and related standard deviation attained on D_f.[7] The *macro F1-score* is the harmonic mean of per-class precision and recall arithmetically averaged over apps. We leverage the F1-score for both meta-learning (whose episodes are balanced by construction) and transfer-learning (suffering from data imbalance) approaches because it is more robust than the common accuracy when working with skewed data. The *silhouette score* quantifies how similar a sample is to its own cluster compared to the others, and it ranges from -1 (worst) to $+1$ (best). This is paramount since most meta-models behave like a nearest-neighbor classifier in the embedded space.

B. Experimental Evaluation

1) Sensitivity to N: This experimental campaign investigates the trend of the F1-score when varying the *number of train ways* N.[8] More specifically, N ranges from 2 to 8 with a step of 2, and the number of shots is kept constant at $K = 25$ for both query and support sets (i.e. N-ways 25-shots meta-training episodes).

Results in Fig. 1 show that *the number of train ways has little impact on the performance of FSL approaches*, with the best-performing ones exhibiting a constant trend when varying N. More specifically, `MatchingNet` reaches the highest F1-score of $\approx 70\%$, $\texttt{TL}_{\texttt{FZ}}$ and $\texttt{TL}_{\texttt{FT}}$ follow close behind, while `MetaOptNet` achieves $\approx 65\%$ F1-score. `fo-MAML` and `ANIL` perform worse than all model-based approaches but `RelationNet`, being the worst performing. Notably, only the performance of the latter significantly varies with N.

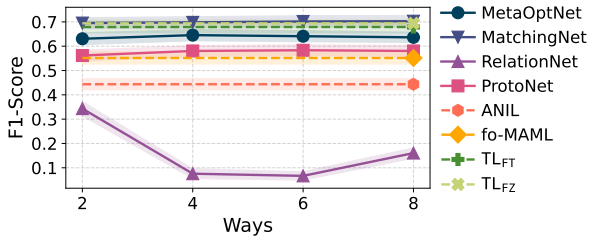

Figure 1: Sensitivity analysis to the number of ways (N) on D_f.

2) Sensitivity to K: This section presents the sensitivity analysis of FSL algorithms when considering a *variable number of shots* K. For meta-training episodes, we consider $K = K_q = K_s \in \{5, 15, 25, 50, 100\}$ in both query and support sets and $N = 8$. For meta-testing and meta-validation ones, K_s follows the values used in meta training.

Figure 2 exhibits that *all FSL approaches significantly increase their F1-score for higher values of train shots*. The best-performing approach (i.e. `MatchingNet`) achieves an F1-score $> 80\%$ when $K = 100$, despite starting from 50%

[7]For transfer-learning approaches we perform multiple T_f (starting from the same T_{nf}) by randomly sample K biflows over 10 runs.

[8]Note that the algorithm-based approaches (i.e. `fo-MAML` and `ANIL`) require that the same value of N is used for both meta-training and meta-testing, not allowing for this specific analysis.

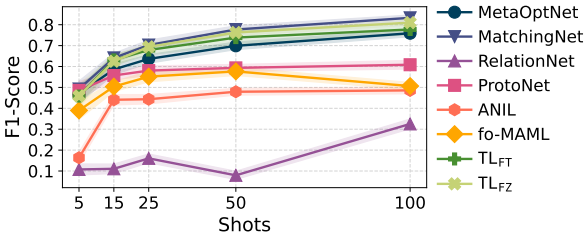

Figure 2: Sensitivity analysis to the number of shots (K) on D_f.

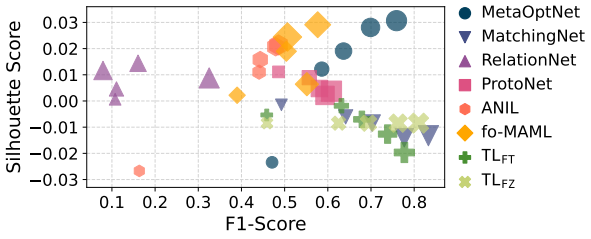

Figure 4: Scatter plot of F1-score and silhouette score of all FSL approaches considering 8-ways K-shots episodes with $K \in \{5, 15, 25, 50, 100\}$. The bigger K, the bigger the marker.

when $K = 5$. $\texttt{TL}_{\texttt{FT}}$, $\texttt{TL}_{\texttt{FZ}}$, and `MetaOptNet` similarly have satisfactory performance. `ProtoNet`, `ANIL`, and `fo-MAML` do not exceed 60% F1-score, despite a sharp rise when passing from $K = 5$ to $K = 15$ can be observed.

3) Using More Classes for Training: Hereinafter, we investigate if using a larger set of apps than that in D_{nf_1} (i.e. for meta training and for performing T_{nf} task in TL) may lead to better generalization and performance. Accordingly, we extend D_{nf_1} with apps of D_{nf_2}, passing from 24 to 32 apps. We analyze the best model-based approach (`MatchingNet`) and the best algorithm-based ones for both paradigms (`fo-MAML` for meta learning and $\texttt{TL}_{\texttt{FZ}}$ for transfer learning).

Figure 3 shows that the outcome depends on the specific FSL approach. `MatchingNet` performance improves when passing from 24 to 32 apps, $\texttt{TL}_{\texttt{FZ}}$ appears insensitive, whereas `fo-MAML` gets worse. We remark that this analysis is a proxy to evaluate the effectiveness of the proposed meta-learning procedure because merging D_{nf_1} and D_{nf_2} prevents the enforcement of early stopping. Indeed, we train for all the 200 epochs and select the best model based on the performance on $D_{nf_1} \cup D_{nf_2}$. Because increasing the number of training classes should lead to performance improvement, the worsening of `fo-MAML` performance is likely related to overfitting.

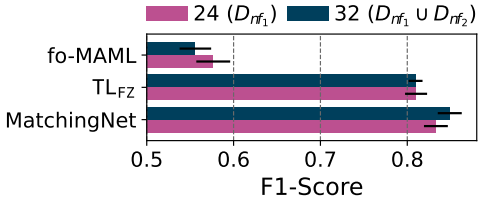

Figure 3: F1-score of the best approaches when a larger training set is used, i.e. D_{nf_1} has 32 classes.

4) Embedding Effectiveness vs Performance: This analysis relates classification performance (measured via F1-score) to the capability of the embedding function to separate app-clusters into the latent space (expressed via silhouette score). Accordingly, Fig. 4 depicts the related scatter-plot by considering 8-ways K-shots episodes with $K \in \{5, 15, 25, 50, 100\}$. Three main trends emerge for different FSL approaches: (*i*) *correlated*, for `ANIL`, `fo-MAML`, and `MetaOptNet`; (*ii*) *not correlated*, for `RelationNet` and $\texttt{TL}_{\texttt{FZ}}$; (*iii*) *negatively correlated*, for `ProtoNet`, $\texttt{TL}_{\texttt{FT}}$, and `MatchingNet`. In detail, considering the peculiarities of each approach, these results can be summarized as follows. First, exploiting complex comparators (i.e. `ANIL`, `fo-MAML`, and `MetaOptNet`) leads to more separable clusters: the better accuracy, the higher number of shots used. Then, $\texttt{TL}_{\texttt{FZ}}$ (and also $\texttt{TL}_{\texttt{FT}}$) behavior clearly demonstrates that training a complex comparator (i.e. a fully-connected layer) on a poorly separable latent space still obtains a higher accuracy when the number of shots increases. Finally, using a simple comparator (i.e. `MatchingNet` and `ProtoNet` using Euclidean distance) does not lead to an easily-separable latent space—which is less separable with higher values of K—but it is increasingly capable of distinguishing classes when the number of shots grows.

From this analysis emerges that *the selection of the approach has an huge impact on the embedding function training*. In particular, the way the embeddings are manipulated by the comparator/classifier component has the major impact.

IV. Related Work

Because of the relevance of the problem FSL aims to solve, it has attracted the interest of research communities working in several domains, including networking and traffic classification. Because FSL originated in the field of computer vision [5–8, 18, 19], applications of FSL to the networking domain mostly exploit minor variants of solutions designed for computer vision. Hereinafter, we discuss the most relevant studies facing traffic classification in a few-shot context, underlining their key aspects and the changes made w.r.t. computer vision-tailored FSL approaches that inspired them. Notably, all the relevant studies [9–14, 20, 21] date to 2018–2022, witnessing the recent interest of the networking community in this topic. Additionally, we select these works mainly because they focus on the problem of classifying *unseen* traffic with just few samples. On the other hand, we do not consider other studies dealing with FSL that perform the model evaluation on seen classes (i.e. where the main goal is simply mitigating the class imbalance in data). Regarding the **specific task** addressed, a reduced number of works face (encrypted) traffic classification [11, 20, 21]. However, differently from the present paper, *none considers mobile-app traffic*. Conversely, a higher number of works leverage FSL in the (similar) context of attack traffic classification and intrusion detection [9, 10, 12–14]. The **FSL approaches** applied in these contexts belong to *model-based* and *algorithm-based* families. Most of them also exploit prior knowledge via *meta learning*, whereas only the oldest works [20, 21] leverage the *transfer-learning* paradigm. We underline that only two studies [10, 12] evaluate more than

one FSL approach. This aspect highlights the *lack of a wide comparison* among the various few-shot families in the traffic-classification literature.

More in detail, Rezaei and Liu [21] cope with the problem of few-shot encrypted traffic classification using a semi-supervised strategy and transfer learning. They pre-train a CNN using a large unlabeled dataset and use it as a pre-processing unit for a second CNN fine-tuned with a small number of labeled flows. Similarly, Xiao et al. [20] propose a supervised approach to simultaneously pre-train a common-knowledge base-layer and task-specific branches of a deep neural network. After pre-training, new branches are added to acquire further knowledge on few-shot tasks. Huang et al. [13] (inspired by the original `MatchingNet` [6]) tackle the problem of network anomaly detection by exploiting an FSL model that integrates a gating technique to the `MatchingNet` algorithm. Gates can be viewed as soft classifiers that provide a similarity score to assess the importance of known and unknown anomalies w.r.t. a test instance. Zheng et al. [9] also exploit meta learning by combining the `RelationNet` [8] with a data-based method. Specifically, authors augment the training set via a "hallucinator" that generates new samples by adding noise to the data. Zhao et al. [11] propose another mixed approach that jointly uses the `RelationNet` with an ensemble of CNNs. Unlike previous studies, Rong et al. [12] and Feng et al. [14] apply `ProtoNet` [7] and `MAML` [5] as-is tackling intrusion and anomaly detection, respectively.

V. Conclusions and Future Perspectives

In this paper, an extensive empirical study of mobile traffic classification through different FSL approaches has been performed, as a means to cope with the problem of scarcity of training data for some of the classes. The evaluation relied on `Mirage-2019`, a publicly available mobile-app dataset. The analyses assessed: (*i*) the impact of changing the number of classes in episodic learning, (*ii*) the sensitivity to the number of samples available for each class, (*iii*) the effect of having a larger set of non-few classes for training, and (*iv*) the relation between the goodness of embedding space and the classification performance.

The performance figures of a number of meta-learning FSL algorithms have been assessed: `ProtoNet`, `MatchingNet`, `RelationNet`, `MetaOptNet`, `MAML`, and `ANIL`. Transfer-learning approaches were also considered, i.e. Fine-Tuning, and Freezing. We found that (*i*) the number of train ways has little impact on the performance of FSL approaches, (*ii*) all FSL approaches significantly increase their F1-score for higher values of train shots, (*iii*) using more classes for training enhances simple comparator-based approaches, and (*iv*) performance of FSL approaches with less (resp. more) complex comparators obtains high F1-score and bad (resp. good) embedding separation.

Based on the outcomes of the experimentation a number of avenues of improvement can be identified. These will explore novel FSL methods in multiple directions linked to (*i*) *the optimization of the learning objective* (using more complex loss functions to enhance the goodness of embeddings), (*ii*) *the adoption of different embedding functions* (e.g., multimodal architectures) to explore their benefits in this context, and (*iii*) *the implementation of a real prototype* that can scalably run over a real networking infrastructure.

Acknowledgements

This work has been funded in the framework of the Huawei Innovation Lab project on "Network Traffic and AI enabled Network Technologies" by Huawei Technologies France SASU at DIETI, University of Napoli Federico II.

References

[1] E. Papadogiannaki et al. A survey on encrypted network traffic analysis applications, techniques, and countermeasures. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

[2] G. Aceto, et al. Mirage: Mobile-app traffic capture and ground-truth creation. In *4th IEEE International Conference on Computing, Communications and Security (ICCCS)*, pages 1–8, 2019.

[3] C. Wang, et al. Appclassnet: A commercial-grade dataset for application identification research. *ACM SIGCOMM Computer Communication Review*, 52(3):19–27, 2022.

[4] Y. Wang, et al. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[5] C. Finn, et al. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1126–1135. PMLR, 2017.

[6] O. Vinyals, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[7] J. Snell, et al. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[8] F. Sung, et al. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[9] W. Zheng, et al. Learning to classify: A flow-based relation network for encrypted traffic classification. In *Proceedings of The Web Conference 2020*, pages 13–22, 2020.

[10] Z.-M. Wang, et al. A few-shot learning-based siamese capsule network for intrusion detection with imbalanced training data. *Computational Intelligence and Neuroscience*, 2021:7126913, 2021.

[11] Z. Zhao, et al. A few-shot learning based approach to iot traffic classification. *IEEE Communications Letters*, 26(3):537–541, 2022.

[12] C. Rong, et al. Umvd-fsl: Unseen malware variants detection using few-shot learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.

[13] S. Huang, et al. A gated few-shot learning model for anomaly detection. In *2020 International Conference on Information Networking (ICOIN)*, pages 505–509, 2020.

[14] T. Feng, et al. Few-shot class-adaptive anomaly detection with model-agnostic meta-learning. In *2021 IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2021.

[15] G. Aceto, et al. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Transactions on Network and Service Management*, 16:445–458, 2019.

[16] M. Lopez-Martin, et al. Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access*, 5: 18042–18050, 2017.

[17] S. Ravi et al. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[18] K. Lee, et al. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10657–10665, 2019.

[19] A. Raghu, et al. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.

[20] Y. Xiao, et al. Common knowledge based transfer learning for traffic classification. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pages 311–314, 2018.

[21] S. Rezaei et al. How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets. *arXiv preprint arXiv:1812.09761*, 2018.