Few-Shot Class-Adaptive Anomaly Detection with Model-Agnostic Meta-Learning

Tongtong Feng, Qi Qi, Jingyu Wang, Jianxin Liao
Beijing University of Posts and Telecommunications
Email: {ftt, qiqi8266, wangjingyu}@bupt.edu.cn, liaojx@gmail.com

***Abstract*—Anomaly detection in encrypted traffic is a growing problem, and many approaches have been proposed to solve it. However, those approaches need to be trained in the massive of normal traffic and specific-class abnormal traffic, so to achieve good results in that specific-class. For a new anomaly class with few labeled samples, the effectiveness of existing approaches will decline sharply. How to train a model using only a few anomaly samples to detect unseen new anomaly classes in training is a huge challenge. In this paper, we propose a Few-shot Class-adaptive Anomaly Detection framework (FCAD) with model-agnostic meta-learning (MAML) to meet this challenge. Given an input network flow, FCAD first extracts statistical features by feature extractor and feature selector, and time-series features using LSTM-based AutoEncoder. Then, FCAD designs a MAML-based few-shot anomaly detection model, relying on the episodic training paradigm and learning from the collection of K-way-M-shot classification tasks, which can mimic the few-shot regime faced at test time during training. Finally, FCAD uses the pre-trained model to adapt the new class by a few iterations steps. Our goal is to detect anomaly traffic in a before unseen anomaly class with only a few samples. A reliable solution to few-shot anomaly detection will have huge potential for real-world applications since it is expensive and arduous to collect a massive amount of data onto the new anomaly class; extensive experimental results demonstrate the effectiveness of our proposed approach.**

***Index Terms*—Anomaly Detection, Few-Shot Learning, Meta-Learning**

I. Introduction

Anomaly detection in encrypted traffic is a growing problem, especially in mobile platforms. An application containing a vulnerability might be exploited by the attackers, consequently can jeopardize the confidentiality, integrity, and availability of the user's crucial information. For example, since users usually download and register apps according to their personal interests, users' privacy and account information would be compromised if an attacker attacks these vulnerable apps. This might result in an immeasurable financial loss as well as unrecoverable damage to a person. Most importantly, the class of anomaly traffic is not static but is constantly updated and iterated, which will pose a tremendous threat to most state-of-the-art defense systems. To stop malicious traffic, the first step is to detect anomaly traffic as soon as possible by analyzing network traffic at the gateways, at edge servers, or in a scrubbing center.

Jingyu Wang is the corresponding author.
ISBN 978-3-903176-39-3© 2021 IFIP.

The goal of anomaly detection is to identify malicious behaviors automatically. Many existing approaches have been proposed and can be divided into four classes: port-based, deep packet inspection (DPI) based, Machine Learning (ML) based [1], [2], Deep Learning (DL) based [3]–[6] approaches. Port-based approaches detect anomaly traffic based on transport-layer port numbers. However, many applications today have port numbers assigned dynamically, which means the port-based approaches are no longer reliable. DPI-based approaches detect anomaly traffic by comparing the traffic payload with known signatures. Since most Internet traffic is now encrypted, this approach will also fail. ML-based approaches can realize high-precision classification by summarizing the rules that distinguish abnormal traffic from normal traffic. However, it is heavily dependent on the distribution of the dataset studied and have poor generalization performance to other datasets. DL-based approaches can train a neural network to learn the features of anomaly traffic and have a high accuracy in anomaly detection tasks.

However, the key limitation of existing approaches is they have poor generalization abilities to a new anomaly class with few labeled samples. They need to be trained in a large number of normal traffic and specific-class abnormal traffic, so to achieve good results in that specific-class. Due to that the class of anomaly traffic is not static but is in the continuous update iteration, for the new anomaly class, which has few labeled samples, the effectiveness of existing approaches will decline sharply. This can be demonstrated in the evaluation section: if we learn an anomaly detection model from existing anomaly classes and directly test the model in a completely new anomaly class, the performance will drop sharply. The reason is that existing approaches are impossible to collect training data that cover all possible anomaly classes; existing approaches are usually over-fitting, for the new anomaly class with few labeled samples, which is hard to generalize. How to train a model from only a few anomaly samples to detect unseen new anomaly classes in training is a huge challenge.

In this paper, we propose a Few-shot Class-adaptive Anomaly Detection framework (FCAD) with model-agnostic meta-learning (MAML) [7] to meet this challenge, as shown in Fig, 1. Given an input network traffic, FCAD first divides network traffic into flows sharing the same 5-tuple information. Then, FCAD extracts flow features using feature extractor and feature selector. Among them, flow features include statistical

2021 IFIP Networking Conference (IFIP Networking) | 978-3-9031-7639-3/21/$31.00 ©2021110.23919/IFIPNETWORKING52078.2021.9472814

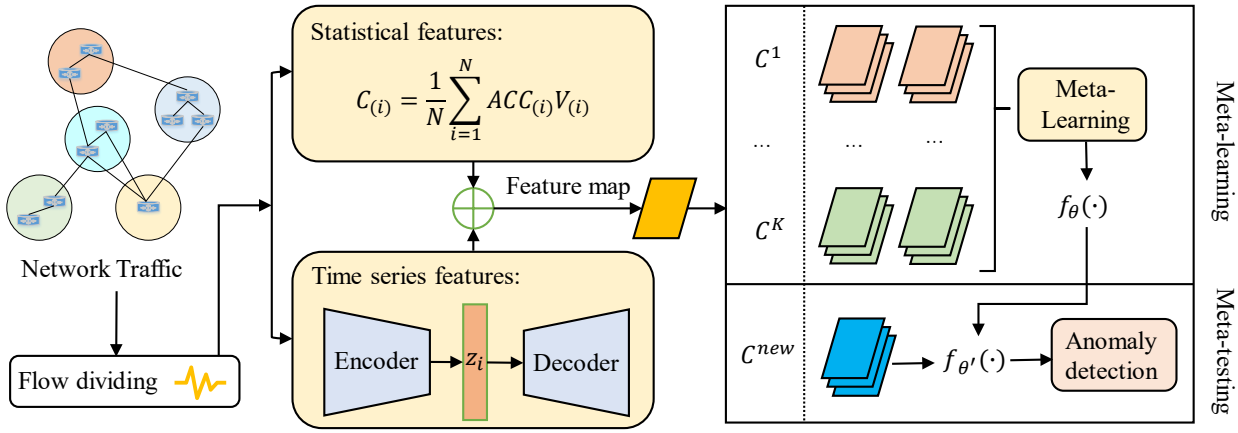

Fig. 1. An overview of the few-shot class-adaptive anomaly detection framework.

features and time-series features. Since the number of packets in different flows is different, we use an LSTM-based AutoEncoder to obtain fixed-length time-series features of each flow. We apply the MAML algorithm to learn a few-shot class-adaptive anomaly detection model, relying on the episodic training paradigm and learning from the collection of K-way-M-shot classification tasks, which can mimic the few-shot regime faced at test time during training. This model consists of a meta-training phase and a meta-testing phase. The meta-training phase will learn to adapt to a new class by training in a large number of few-shot class-adaptive anomaly detection tasks. The meta-testing phase will use the pre-trained model to adapt the new class by a few iterations steps. Our goal is to detect anomaly traffic in a previously unseen anomaly class of only a few samples by using a few iterations steps. To evaluate the efficiency of the FCAD, we carry out four comparative experiments in the public network traffic dataset and show the effectiveness of our proposed approach. The results further demonstrate the superiority of FCAD.

A reliable solution to few-shot class-adaptive anomaly detection will have huge potential for real-world applications since it is expensive and arduous to collect a massive amount of data onto the new anomaly class. To the best of our knowledge, we are the first to cast network traffic anomaly detection using the meta-learning approach. In summary, this paper makes the following key contributions:

- We propose a Few-shot Class-adaptive Anomaly Detection framework (FCAD) with model-agnostic meta-learning (MAML), which can detect anomaly flows in a previously unseen anomaly class with only a few samples. It consists of a meta-training phase and a meta-testing phase. The meta-training phase will learn to adapt to a new class by training in a large number of few-shot class-adaptive anomaly detection tasks. The meta-testing phase will use the pre-trained model to adapt the new class by a few iterations steps.
- We extract 33 statistical features by feature extractor and feature selector. Since the number of packets in different flows is different, we use an LSTM-based AutoEncoder to obtain fixed-length time-series features of each flow.
- We implement the FCAD framework and collect two real-world datasets for trace-driven experiments. We compare FCAD with the state-of-the-art solutions and the results further demonstrate the superiority of FCAD.

The rest of the paper is organized as follows. Section II briefly reviews the related works. Section III describes the design details of the FCAD framework. Section IV presents the implementation details of the FCAD framework. Section V presents evaluation results with comparison to the state-of-the-art solutions. Conclusions and future works are discussed in Section VI.

II. Related Work

In this section, we review existing work on the network traffic anomaly detection approaches, the few-shot learning approaches, and the meta-learning approaches.

A. Anomaly Detection in Network Traffic

The goal of anomaly detection in network traffic is to identify malicious behaviors automatically by learning exclusively from normal traffic. In general, existing networking traffic anomaly detection approaches can be divided into four classes: port-based, deep packet inspection (DPI) based, Machine Learning (ML) based, Deep Learning (DL) based approaches. *Port-based* approaches detect anomaly traffic based on transport-layer port numbers. However, many applications today have port numbers assigned dynamically, which means the port-based approaches are no longer reliable. *DPI-based* approaches detect anomaly traffic by comparing the traffic payload with known signatures. Since most Internet traffic is now encrypted, this approach will also fail. *ML-based* approaches can identify abnormal traffic by summarizing the characteristics of abnormal traffic, including C4.5 decision tree (DT) [8], Naive Bayesian (NB) [9], K-means [10], Support Vector Machine (SVM) [11], Random Forest (RF) [1], [2], Xgboost [12] approaches and so on. ML-based techniques have demonstrated higher classification accuracy than other

approaches. However, those approaches are to realize high-precision classification by summarizing the rules that distinguish abnormal traffic from normal traffic, which is heavily dependent on the distribution of the dataset studied and is difficult to exploit the input of the model to mine comprehensive information. So that the ML-based approaches have poor generalization performance to other datasets. *DL-based* approaches can train a neural network to learn the characteristics of anomaly traffic and then classify it. Most existing works are in favor of using convolutional neural network (CNN) [1], [2], [13], AutoEncoder [3]–[6], and Long Shot-Term Memory (LSTM) [2] to build the traffic classifier for anomaly traffic detection. The DL-based approaches can automatically extract features of network traffic and support to vary due to different tasks.

B. Few-Shot Learning

To mimic the fast and flexible learning ability of humans, few-shot learning aims to learn representations that generalize well to the novel classes where only a few samples are available. The research in few-shot learning can be categorized into three common classes: meta-learning based, metric-learning based, generative, and augmentation based approaches. *Meta-learning based* approaches [14]–[16] search for models which can transfer well to novel few-shot tasks. Typically, at training time, these approaches will learn to adapt to a new class by training from a large number of few-shot tasks; at testing time, simple fine-tuning is used. *Metric learning based* approaches [17]–[19], a non-linear metric is optimized base on classes and applied to unseen few-shot test tasks. In [20], distance to class prototype is replaced by distance to a class sub-space. As opposed to [20] and TAFSSL [21] that try to optimize a sub-space for each class, which seek a single sub-space optimally adapted to the entire data onto the few-shot task. In [17], non-meta-learning pre-training was used in combination with large backbones and a nearest-neighbor classifier to achieve state-of-the-art results. *Generative and augmentation based* approaches [22], [23], it generates more samples from the one or a few training examples to a given few-shot learning task. Delta-encoder [24] based on a modified auto-encoder learns to synthesize new samples for an unseen category just by seeing a few examples from it. In [25], this approach learns to map a novel sample instance to a concept, relates that concept to the existing ones in the concept space and, using these relationships, generates new instances, by interpolating among the concepts, to help learning.

C. Meta-Learning

Meta-learning (also known as learning to learn) has been shown to be an effective solution to the few-shot learning problem in the form of episodic training. The research in meta-learning can be categorized into three common classes: metric-based, model-based, and optimization-based approaches. *Metric-based* approaches [26]–[29] learn a metric with the intent of reducing the intra-class variations while training on base categories. For example, Siamese [26] explores a method of learning siamese neural networks which employ a unique structure to naturally rank similarity between inputs; MMN [27] based on deep neural features and augment neural networks with external memories; RN [28] learns to learn a deep distance metric to compare a small number of samples within episodes and can classify samples of new classes by computing relation scores between query samples and the few examples of each new class without further updating the network; prototypical networks [29] is proposed to learn a feature space where instances of a given class are located close to the corresponding prototype (centroid), allowing accurate distance-based classification. *Model-based* approaches [30], [31] are devised for fast learning from the model architecture perspective, where rapid parameter updating during training steps is usually achieved by the architecture itself. Lastly, *optimization-based* approaches [7], [32], [33] modify the optimization algorithm for quick adaptation. These methods can quickly adapt to a new task through the meta-update scheme among multiple tasks during parameter optimization. In network traffic anomaly detection, we follow a similar optimization-based meta-learning approach MAML [7] and apply it to the much more challenging task of anomaly detection with new classes. To the best of our knowledge, we are the first to cast anomaly detection as meta-learning with new classes.

III. Methodology

In this section, we introduce a Few-shot Class-adaptive Anomaly Detection framework (FCAD) with model-agnostic meta-learning (MAML), as shown in Fig 1. Given an input network traffic, FCAD first divides network traffic into flows sharing the same 5-tuple information. Then, FCAD extracts flow features using feature extractor and feature selector. Among them, flow features include statistical features (describe in III-A) and time-series features (describe in III-B). Since the number of packets in different flows is different, we use an LSTM-based AutoEncoder to obtain fixed-length time-series features of each flow. We apply the MAML [7] algorithm to learn a few-shot class-adaptive anomaly detection model (describe in III-C), which consists of a meta-training phase and a meta-testing phase. The meta-training phase will learn to adapt to a new class by training in a large number of few-shot class-adaptive anomaly detection tasks. The meta-testing phase will use the pre-trained model to adapt the new class by a few iterations steps. Our goal is to detect anomaly traffic in a previously unseen anomaly class of only a few samples by using a few iterations steps.

A. Statistical Features Extraction

In this subsection, we first introduce the basic unit of feature extraction: flow. Then, we describe how to extract the statistical features and design a feature selection algorithm to select valid features.

Flow. We define flow as a series of packets sharing the same 5-tuple information: source IP address, destination IP address,

TABLE I
STATISTICAL FEATURES.

	Up_flow	Down_flow	Flow
#1	max,mean,std,sum	max,mean,std,sum	max,mean,std,sum,ratio
#2	-	mean,sum	mean,ratio
#3	max,min,mean,std	max,mean	max,mean,sum
#4	sum	sum	sum,ratio
#5	-	-	sum
#6	pst_cnt	-	Ack_cnt

[1] #1-6: Packet Length, Packet Number, Packet Interval, Packet Header Length, Packet Window Size, Packet Control Character.

source port numbers, destination port numbers, and transport layer protocol. We first divide the network traffic into flows according to the 5-tuple information.

Feature extractor. We select 78 flow-level statistical features to represent each flow. The extraction process is as follows: for each flow, we first divide it into up_flow and down_flow. Among them, the up_flow refers to the collection of packets from the source IP address to the destination IP address for data transmission and the down_flow is the rest. Then for up_flow, down_flow, and flow, we calculate the maximum, minimum, mean, variance, and sum of six types of features, respectively. These six types features are packet length, packet number, packet interval, packet header length, packet window size, and packet control character. We also calculate the numerical ratio between the up_flow and down_flow for these six types of features in terms of the mean. Finally, a total of 78 statistical features are extracted.

Feature selector. We adopt three strategies for feature selection: the proportion of missing values, the entropy, and the cumulative importance. When the missing proportion of the feature f is greater than 50% or the entropy of the feature f is equal to 0, the feature f is deleted. Because different classification algorithms have a large difference in the order of feature importance, we use the comprehensive cumulative importance ranking of five different algorithms. First, the accuracy of each algorithm $Acc_{(i)}$ is obtained by running the dataset separately in five different algorithms, including RF, Extraboost, Admboost, GBDT, and Lightgbm algorithm. Then the feature importance value of each feature $V_{(j)}$ is calculated in five different algorithms. Next, the cumulative feature importance of each feature $C_{(j)}$ is calculated as follows:

$$C_{(j)} = \frac{1}{5}\sum_{i=1}^{5} Acc_{(i)} V_{(j)} \tag{1}$$

where i represents the different algorithms, $i \in \{1, 2, .., 5\}$ and j represents different features, $j \in \{1, 2, ..., 78\}$. Finally, by sorting the importance of features $C_{(j)}$, the last 30% of features are deleted. Through three strategies, 33 features were selected as the basic statistical features. These features are shown in TABLE I.

B. Time-series Features Extraction

Since the number of packets in different flows is different, in this subsection, we introduce how to use an LSTM-based AutoEncoder to obtain fixed-length time-series features for each flow. We first describe the construction of the feature matrix. We then describe the design detail of the LSTM-based AutoEncoder.

Feature matrix. In order to obtain the time-series features of each flow, we first need to construct a flow feature matrix $\mathbb{X}$ as the input of the AutoEncoder, which can represent the time relationship between the package and the package in each flow:

$$\mathbb{X}_i = \{\vec{x_{i,j}}\}, j \in \{1, 2, ..., n\} \tag{2}$$

where (i, j) represents the j-th packet of the flow i; and $\vec{x_{i,j}}$ represents the feature vector of the j-th packet in the flow i. Among them, $\vec{x_{i,j}}$ contains 8 packet-level features: header length, payload length, packet interval, window size, ack_cnt, pst_cnt, direction. The reason why we do not select any of the 5-tuple values as the packet feature is that the IP address and port number will change frequently. Since the number of packets between different flows is not equal, so we select n as the maximum number of packets contained in the feature matrix $\mathbb{X}_i$. If the number of packets in the flow i is greater than n, we will select the first n packets to construct the feature matrix. In contrast, we will fill the feature matrix by increasing the $\vec{\mathbf{0}}$ vector.

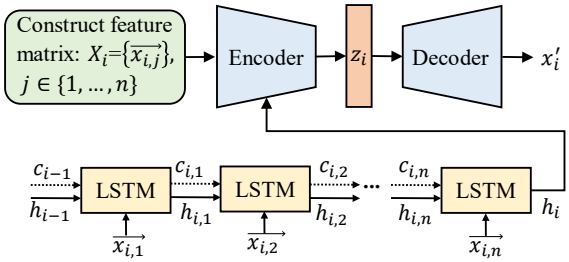

Fig. 2. Time-series features extraction LSTM-based AutoEncoder

The LSTM-based AutoEncoder. We use the outstanding LSTM-based AutoEncoder to extract the time-series features of each flow. The design details of it are as follows. Both the encoder and the decoder use the LSTM network. The input of the encoder is the flow feature matrix $\mathbb{X}_{(i)}$, and the output is a fixed-length 5-dimensional vector $\mathbb{Y}_{(i)}$. The input of decoder is a vector $\mathbb{Y}_{(i)}$, and the output of the decoder is a matrix $\mathbb{X}'_{(i)}$, which has the same matrix structure as the encoder input. The iterative mechanism of the AutoEncoder is to reduce the mean square error between the input of the encoder and the output of the decoder:

$$J_{(W,b)} = \frac{1}{n}\sum_{j=1}^{n}\sum_{k=1}^{m}(\mathbb{X}_{(i,j,k)} - \mathbb{X}'_{(i,j,k)})^2 \tag{3}$$

where W and b are the network hyperparameter; $\mathbb{X}_{(i,j,k)}$ represents the k-th feature of the j-th packet in the flow i; m represents the number of packet-level features, which is equal to 8.

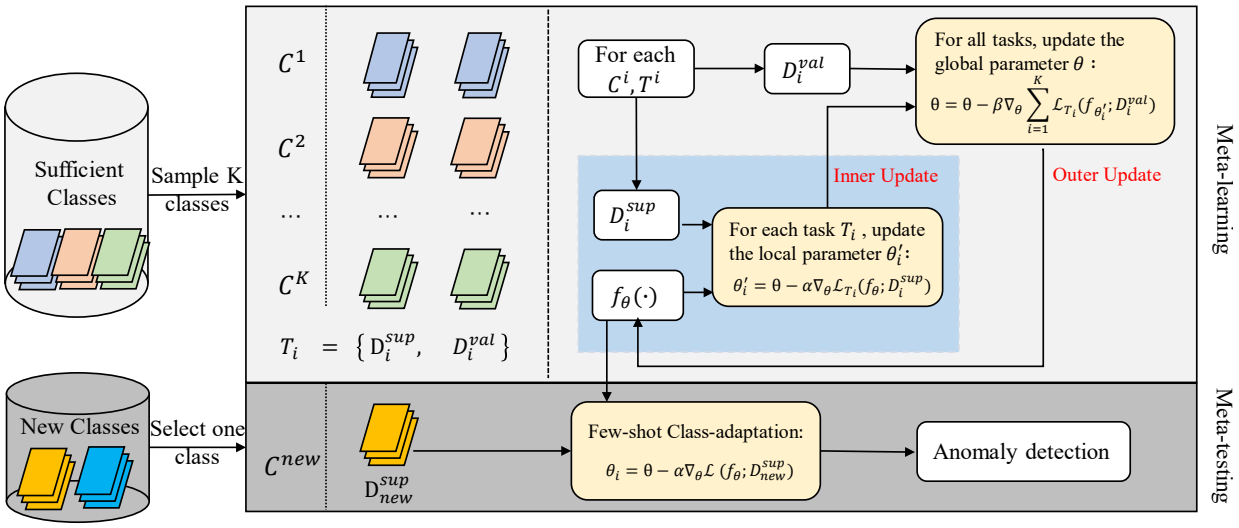

Fig. 3. An overview of class-adaptive anomaly detection model with MAML algorithm.

C. Class-Adaptive Anomaly Detection with MAML

We propose a few-shot class-adaptive anomaly detection model using the MAML algorithm. Figure 3 shows an overview of the proposed approach.

Formally, we have a large training dataset $\mathcal{D}^{tr}$ (typically anomaly classes of containing a large samples) and a test dataset $\mathcal{D}^{test}$ (new anomaly classes of containing one or a few samples), in which their respective class setscolon $\mathcal{C}_{tr} = \{1, ..., |\mathcal{C}_{tr}|\}$ and $\mathcal{C}_{test} = \{|\mathcal{C}_{tr}| + 1, ..., |\mathcal{C}_{tr}| + |\mathcal{C}_{test}|\}$ are disjoint. Our approach aim to learn a classification model on $\mathcal{C}^{tr}$ that can generalize to unseen classes $\mathcal{C}_{test}$ with one or few samples per class. Our approach is based on the MAML algorithm, relying on the episodic training paradigm and learning from collection of K-way-M-shot classification tasks, which can mimic the few-shot regime faced at test time during training on $\mathcal{D}^{tr}$. It consists of a meta-training phase and a meta-testing phase. In each episode of meta-training phase, we first sample K classes from $\mathcal{C}_{tr}$, $\mathcal{C}^K \sim \mathcal{C}_{tr}$. Then, we sample M and N labelled flows per class in $\mathcal{C}^K$ to construct the support set $\mathcal{D}^{sup} = \{(x_m, y_m)_m\}$ and the validation set $\mathcal{D}^{var} = \{(x_n, y_n)_n\}$, respectively. Meanwhile, we construct the task $\mathcal{T}_i = \{\mathcal{D}_i^{sup}, \mathcal{D}_i^{var}\}$ for each class $\mathcal{C}^i$ with a support set $\mathcal{D}_i^{sup}$ and a validation set $\mathcal{D}_i^{var}$. $\mathcal{D}_i^{sup}$ is used for inner update through gradient descent to obtain the updated parameters θ_i' for each task. Then $\mathcal{D}_i^{var}$ is used to measure the performance of θ_i'. An outer update procedure is used to update the model parameters θ by taking into account of all the sampled tasks. In meta-testing phase, given a new class $\mathcal{C}^{new} \sim \mathcal{C}_{test}$, we use only a few flows to get the adapted parameters θ' for this specific anomaly class $\mathcal{C}^{new}$. In the process of model construction, specific details such as task definition, meta-training, meta-testing, and backbone architecture, are as follows.

Task Definition. The first challenge of constructing the class-adaptive anomaly detection model is the implicit definition of the task, which is usually generated by randomly mixed sampling K classes in each episode. We definite the tasks for our application as follows. For a given anomaly class $\mathcal{C}^i$, we definite a task as $\mathcal{T}_i = \{\mathcal{D}_i^{sup}, \mathcal{D}_i^{var}\}$, where $\mathcal{D}_i^{sup}$ and $\mathcal{D}_i^{var}$ are the support and validation set in the task $\mathcal{T}_i$. In the meta-training phase, we randomly sample M and N input/output pairs from $\mathcal{T}_i$ as the support set $\mathcal{D}_i^{sup} = \{(x_1, y_1), (x_2, y_2), ..., (x_M, y_M)\}$ and the validation set $\mathcal{D}_i^{var} = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ (excluding those in $\mathcal{D}_i^{sup}$), respectively. In the testing phase, we need to distinguish the new anomaly class from the normal traffic, which is a two-class classification problem. So during the training phase, in order to mimic the few-shot regime faced at test time, we make sure that all the samples in task $\mathcal{T}_i$ come from the same anomaly class $\mathcal{C}^i$ rather than mixed classes

Meta-learning. The goal of meta-learning is to construct a pre-trained anomaly detection model $f_\theta : x \to y$ with parameters θ. It can learn the optimal initial model parameters θ, so that the model can quickly adapt to a new task through the meta-update scheme with a few fine-tuning steps. It includes two modules: the inner update and the outer update. Algorithm 1 shows the process of meta-learning by the pseudocode.

In the inner update. The goal of the inner update is to update the local parameters θ_i' for each task $\mathcal{T}_i$ through gradient descent. Following the MAML algorithm, for a given anomaly class $\mathcal{C}^i$ and task $\mathcal{T}_i$, we first definite a loss function on the support set $\mathcal{D}_i^{sup}$:

$$\mathcal{L}_{\mathcal{T}_i}(f_\theta; \mathcal{D}_i^{sup}) = \sum_{(x_i, y_i) \in \mathcal{D}_i^{sup}} L(f_\theta(x_i), y_i) \tag{4}$$

where $L(f_\theta(x_i), y_i)$ measure the difference between the prediction class $f_\theta(x_i)$ and the actual class y_i. We define $L(\cdot)$ is the cross entropy error function:

$$L(f_\theta(x_i), y_i) = y_i log(p_{x_i}) + (1 - y_i) log(1 - p_{x_i}) \tag{5}$$

Algorithm 1 Meta-learning for few-shot class-adaptive anomaly detection with MAML

Input: Hyper-parameters α, β;

Randomly initialize parameter θ with a meta model $f_\theta(\cdot)$;
while not done **do**
  Sample K classes from $\mathcal{C}_{tr}$, $\mathcal{C}^K = \{\mathcal{C}^i\}_{i=1}^K$, $\mathcal{C}^K \sim \mathcal{C}_{tr}$;
  for each C^i **do**
    Construct the support set $\mathcal{D}^{sup} = \{(x_m, y_m)_m\}$ by sampling M labelled flows from class $\mathcal{C}^i$;
    Construct the validation set $\mathcal{D}^{var} = \{(x_n, y_n)_n\}$ by sampling N labelled flows from class $\mathcal{C}^i$;
    Construct the task $\mathcal{T}_i = \{\mathcal{D}_i^{sup}, \mathcal{D}_i^{var}\}$ for class $\mathcal{C}^i$;
    Compute the loss function $\mathcal{L}_{\mathcal{T}_i}(f_\theta; \mathcal{D}_i^{sup})$;
    Update the local parameter: $\theta_i' = \theta - \alpha \bigtriangledown_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta; \mathcal{D}_i^{sup})$;
  end for
  Update the global parameter by taking into account of all the sampled tasks: $\theta = \theta - \beta \sum_{i=1}^K \bigtriangledown_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}; \mathcal{D}_i^{val})$;
end while

where p_i is the probability that sample x_i is predicted to be positive. Then, we use one gradient update to change the parameters from θ to θ_i':

$$\theta_i' = \theta - \alpha \bigtriangledown_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta; \mathcal{D}_i^{sup}) \tag{6}$$

where α is the step size. In the inner update, the updated parameters θ' are specifically adapted to the task $\mathcal{T}_i$.

In the outer update. The goal of the outer update is to update the global parameters θ by taking into account of all the sampled tasks. For a given anomaly class $\mathcal{C}^i$ and task $\mathcal{T}_i$, we also definite a loss function on the validation set $\mathcal{D}_i^{var}$:

$$\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}; \mathcal{D}_i^{val}) = \sum_{(x_i, y_i) \in \mathcal{D}_i^{val}} L(f_{\theta_i'}(x_i), y_i) \tag{7}$$

Then, we use one gradient update to change the parameters from θ_i' to θ by taking into account of all the sampled tasks:

$$\theta = \theta - \beta \sum_{i=1}^K \bigtriangledown_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}; \mathcal{D}_i^{val}) \tag{8}$$

where β is the step size in the outer update.

Meta-testing. After meta-training, we obtain the pre-trained model parameters θ. During meta-testing, for a given a new target anomaly class $\mathcal{C}^{new}$, we first obtain the adapted parameters θ' with a few fine-tuning steps by the gradient update:

$$\theta' = \theta - \alpha \bigtriangledown_\theta \mathcal{L}(f_\theta(\cdot)) \tag{9}$$

Then, we use this model $f_{\theta'}(\cdot)$ for anomaly detection in new class $\mathcal{C}^{new}$ by measuring the difference between the prediction class $f_{\theta'}(x_i)$ and the actual class y_i.

Backbone architecture. Our class-adaptive anomaly detection model is general. In theory, we can use any anomaly detection network as the backbone architecture. Because the data format of network traffic is irregular, the features of network traffic have been extracted preliminarily through the data preprocessing module. For the anomaly detection module, we only need a simple backbone architecture to learn good accuracy by using our approach. So we come up with the Deep Neural Networks (DNN [34]) as backbone architecture.

IV. Implementation

In this section, we delve in-depth into the implementation specifics of each component of FCAD and describe the best hyper-parameters of the LSTM-based AutoEncoder and the few-shot class-adaptive anomaly detection model.

The LSTM-based AutoEncoder. First, we select $n = 20$ as the maximum number of packets contained in the feature matrix $\mathbb{X}_i$. The network structure of the encoder contains two LSTM layers. The number of hidden cells in the first layer and the second layer is 256 and 128, respectively. Results from these layers are then aggregated in a hidden layer that uses 5 neurons to apply the softmax function. The network structure of the decoder also contains two LSTM layers. the first layer and the second layer includes 128 and 256 hidden cells, respectively. But its final output is a matrix. In training, we use mean square error as a loss function for model optimization. Finally, for each flow, we will use a 5-dimensional vector to represent the time-series features.

Backbone architecture. We use the DNN as backbone architecture for the few-shot class-adaptive anomaly detection model. The network structure contains three fully-connected layers. The number of hidden cells in the first layer is 256 and the number of hidden cells in other layers is 128. Outputs from these layers are then aggregated in a hidden layer that uses V ($V = 2$, which is equal to the number of action) neurons to apply the loss function. We fix the hyperparameters α and β in meta-learning at 0.001. During meta-training, we sample the batch size K of classes in each epoch to be 5; we sample the batch size M and N of flows in each inner update is the same (i.e. $M = N$).

V. Evaluation

In this section, we compare our approach against existing state-of-the-art solutions cover a broad set of realistic network traffic datasets. Experiment results answer the following questions:

1) With the standard anomaly detection, how does the FCAD compare to the existing state-of-the-art ML-based approaches and the DL-based approaches in terms of each evaluation metrics? We use the same train/test split as prior work (i.e. train/test set contains the same anomaly class and has large samples). TABLE II suggest our approach is comparable to the state-of-the-art.
2) With the M-shot class-adaptive anomaly detection, how does the FCAD compare to the existing state-of-the-art ML-based approaches and the DL-based approaches in terms of the AUC? We use 30 classes for training and the remaining classes for testing. TABLE III report results for $M = 5, 10, 20$; our approach outperforms others.

3) Under the cross-dataset testing setting with M-shot class-adaptive, how does the FCAD compare to the existing state-of-the-art ML-based approaches and the DL-based approaches in terms of the AUC? TABLE IV demonstrate the effectiveness of our approach on few-shot class-adaptive anomaly detection.
4) How much does the time-series features can improve overall performance in terms of each evaluation metrics? As shown in Fig. 4, time-series relationships between packets are also critical for anomaly detection.

A. Methodology

Datasets. We consider the following publicly available datasets of network traces to train and test our proposed approaches.

CICAndMal2017 [35]. This dataset is gathered through executing 5065 benign and 429 malware Apps on a real smartphone instead of any emulator. The benign APPs are collected from the Google play market published in 2015-2017. These APPs were collected based on their popularity and identified based on the detection results from VirusTotal. The malware Apps are collected from 4 main malware categories (i.e., adware, ransomware, scareware, and SMS malware), which consists of 43 malware families. Finally, 1700 benign and 429 malware network traffics were captured in the installation of APPs, before the restart and after the restart.

CICIDS2017 [36]. This dataset is provided by the Canadian Institute of Cybersecurity, which contains benign traffic and the most up-to-date common 8 attacks traffic include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. The data capturing period started at 9 a.m., Monday, July 3, 2017, and ended at 5 p.m. on Friday, July 7, 2017, for a total of 5 days, which is the latest labeled dataset.

Evaluation metrics. Following prior work [7], [32], we evaluate the performance of FCAD by using accuracy, recall, precision, F1-score, and the area under the ROC curve (AUC):

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$recall = \frac{TP}{TP + FN} \tag{11}$$

$$precision = \frac{TP}{TP + FP} \tag{12}$$

$$F1 - score = 2\frac{recall * precision}{recall + precision} \tag{13}$$

where TN is True Negative, TP is True Positive, FN is False Negative, and FP is False Positive.

Baselines. We select 3 types of ML-based approaches as comparison algorithms: Support Vector Machine (SVM) [11], C4.5 decision tree (DT) [8], and Random Forest (RF) [1], [2], which can be implemented by scikit-learn [37]. Similarly, we also choose 3 types of DL-based approaches as comparison algorithms: convolutional neural network (CNN) [1], AutoEncoder [3], and Long Shot-Term Memory (LSTM) [2], which can be implemented through parameter iteration.

TABLE II
FCAD ON STANDARD ANOMALY DETECTION.

Category	Approach	Metric		
		Accuracy	F1-score	AUC
ML-based	SVM	0.73	0.71	0.81
	DF	0.981	0.978	0.982
	RF	**0.985**	**0.989**	0.99
DL-based	LSTM	0.984	0.983	**0.993**
	CNN	0.926	0.933	0.941
	AutoEncoder	0.983	0.982	0.988
Ours	FCAD	0.983	**0.989**	0.991

To the best of our knowledge, this is the first work on the network traffic class-adaptive anomaly detection problem. Therefore, Prior works will have very poor performance on this problem. Nevertheless, we define an additional comparison baseline. *Pre-trained:* the model learning from the meta-learning phase is directly applied to the meta-test phase without any adaptation. *Fine-tuning:* the model learning from the meta-learning phase is applied to the meta-test phase by a few iterations steps, which is the standard FCAD approach.

B. FCAD on Standard Anomaly Detection.

The goal of the FCAD is to detect anomaly traffic in a previously unseen anomaly class of only a few samples. But in real scenarios, the traditional anomaly family still accounts for the majority of anomaly traffic. So in this subsection, we first need to verify that the FCAD can be applied to traditional anomaly detection problems. We compare it with the existing state-of-the-art approaches using the standard training/testing set dividing (training set and testing set are provided by the same anomaly classes and contain a large number of samples). Those outstanding existing approaches as listed in Section V-A, including SVM, DT, RF, LSTM, CNN, and AutoEncoder. We use three metrics to evaluate our approach: accuracy, F1-score, and AUC. For comparison, we continue to iterate the DL-based approaches until we get the optimal model of the considered evaluation metrics. The experimental results are collected on the CICAndMal2017 dataset and are shown in TABLE II.

From TABLE II, we can find that our approach performs as well as the state-of-the-art. This is because, in a large number of training samples, the ML-based and DL-based approaches can comprehensively summarize and learn the rules of how to distinguish abnormal traffic from normal traffic. So in the testing set with the same anomaly distribution, those approaches can achieve extremely high anomaly detection accuracy. Therefore, in the traditional anomaly detection problem, our proposed approach can achieve good performance.

C. FCAD on M-shot Class-Adaptive Anomaly Detection.

Compared with the traditional anomaly detection problem (having large samples; the training set and the test set are equally distributed), the FCAD can also address the new challenge: detecting anomaly traffic in a previously unseen anomaly class with only a few samples. In this subsection, we compare it with the existing state-of-the-art ML-based

approaches and the DL-based approaches on the M-shot class-adaptive anomaly detection. In the CICAndMal2017 dataset, we randomly selected 30 classes from 43 malware families as the training set and the remaining classes as the testing set. For each iteration, the sample number of the train/test set is equal to $5, 10, 20$, respectively (i.e. $M = 5, 10, 20$). For comparison, we continue to iterate the DL-based approaches until we get the optimal model of the considered evaluation metrics. For ML-based approaches, because the sample number of the train/test is very small, we choose to take the mean value of multiple experiments (e.g. 10 times) to achieve the reliability of the results. The experimental results are shown in TABLE III.

TABLE III
M-SHOT CLASS-ADAPTIVE ANOMALY DETECTION.

Category	Approach	M		
		M=5	M=10	M=20
ML-based	SVM	0.541	0.587	0.659
	DF	0.617	0.633	0.724
	RF	0.664	0.727	0.782
DL-based	LSTM	0.749	0.823	0.903
	CNN	0.675	0.741	0.790
	AutoEncoder	0.782	0.836	0.894
Ours	Pre-training	0.882	0.903	0.935
	Fine-turning	**0.943**	**0.971**	**0.986**

From TABLE III, we can find our approach is far superior to the ML-based and DL-based approaches in terms of performance. This is because existing approaches realize anomaly detection by summarizing the rules that distinguish abnormal traffic from normal traffic, which is heavily dependent on the distribution of the dataset studied. They need to be trained in a large number of normal traffic and specific-class abnormal traffic, so to achieve good results in that specific-class. However, existing approaches are impossible to collect training data that cover all possible anomaly classes. Due to that the class of anomaly traffic is not static but is in the continuous update iteration, for the new anomaly class, which has few labeled samples, the effectiveness of existing approaches will decline sharply. The FCAD using a meta-learning method to learn a few-shot class-adaptive anomaly detection model, relying on the episodic training paradigm and learning from the collection of K-way-M-shot classification tasks, which can mimic the few-shot regime faced at test time during training. In the meta-training phase, we first learn a general few-shot class-adaptive anomaly detection model. In the meta-testing phase, we use the pre-trained model to adapt the new class by a few iterations steps. So that our approach can quickly adapt to the new anomaly class.

D. Generalization.

In this subsection, to verify the FCAD has a strong generalization, we compare it with existing state-of-the-art ML-based approaches and the DL-based approaches in cross-dataset in terms of AUC. We first use the CICAndMal2017 dataset to train a pre-trained model. Then, we use it to adapt the new class in the CICIDS2017 dataset by a few iterations steps. And conversely, the same operation is performed. In this experiment, we set the $M = 3$. TABLE IV demonstrate the effectiveness of our approach on few-shot class-adaptive anomaly detection. We can find that our approach can still achieve the optimal results in cross-dataset testing.

TABLE IV
CROSS-DATASET TESTING.

Category	Approach	Cross-testing set	
		CICAndMal2017	CICIDS2017
ML-based	SVM	0.542	0.535
	DF	0.621	0.598
	RF	0.669	0.615
DL-based	LSTM	0.882	0.853
	CNN	0.722	0.741
	AutoEncoder	0.794	0.835
Ours	Pre-training	0.915	0.893
	Fine-turning	**0.966**	**0.951**

E. Feature Selection.

In this subsection, we set three sets of features: *Feature_set_1* includes statistical features obtained through feature extractor; *Feature_set_2* includes 33 statistical features obtained through feature selector; *Feature_set_3* inlucde 33 statistical features and 5 time-series features. By running these feature sets separately in the FCAD, the experimental results are shown in Fig. 4.

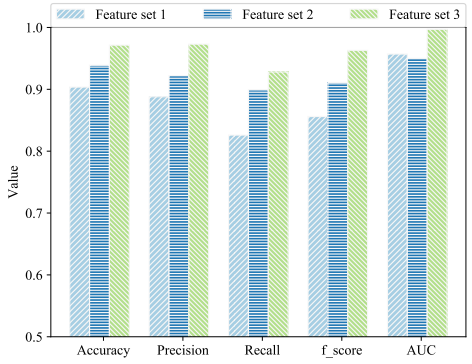

Fig. 4. Performance evaluation of different feature sets.

We can find that the performance of Feature_set_2 is slightly higher than that of Feature_set_1, but the performance of Feature_set_3 is significantly higher than that of Feature_set_1. This is because the feature selection algorithm can reduce the influence of the feature without discrimination on the classification result. Time-series features allow a more comprehensive and fine-grained evaluation of the entire flow to obtain a more accurate flow description. Therefore, time-series relationships between packets are also critical for anomaly detection.

VI. CONCLUSION

Anomaly detection in encrypted traffic is a growing problem, especially in mobile platforms. The goal of anomaly

detection is to identify malicious behaviors automatically by learning exclusively from normal traffic. Many existing approaches have been proposed to address this problem. However, most existing approaches are usually data-hungry and have limited generalization abilities to new anomaly classes. For a new anomaly class of few labeled samples, the effectiveness of existing methods will decline sharply. How to train a model from only a few anomaly samples to detect unseen new anomaly classes in training is a huge challenge. A reliable solution to this challenge will have huge potential for real-world applications since it is expensive to collect a massive amount of data onto a new anomaly class and is difficult to detect unseen new anomaly classes in training with few new anomaly samples. In this paper, we propose a Few-shot Class-adaptive Anomaly Detection framework of model-agnostic meta-learning to address the limitations of previous approaches and extensive experimental results demonstrate the effectiveness of our proposed approach.

ACKNOWLEDGMENT

This work was supported in part by the National Key R&D Program of China 2020YFB1807805, in part by the National Natural Science Foundation of China under Grants 62071067, 61771068, 61872310.

REFERENCES

[1] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "TR-IDS: Anomaly-based Intrusion Detection through Text-convolutional Neural Network and Random Forest," *Security and Communication Networks*, vol. 2018, pp. 1–9, 2018.

[2] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile Encrypted Traffic Classification using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[3] E. Min, J. Long, Q. Liu, J. Cui, Z. Cai, and J. Ma, "SU-IDS: A Semi-supervised and Unsupervised Framework for Network Intrusion Detection," in *International Conference on Cloud Computing and Security*, 2018, pp. 322–334.

[4] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BAIOT: Network-based Detection of IOT Botnet Attacks using Deep Autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[5] J. Höchst, L. Baumgärtner, M. Hollick, and B. Freisleben, "Unsupervised traffic flow classification using a neural autoencoder," in *LCN*, 2017, pp. 523–526.

[6] Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated convolutional autoencoders," *Security and Communication Networks*, vol. 2017, pp. 1–10, 2017.

[7] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017, pp. 1126–1135.

[8] S. Wang, Z. Chen, L. Zhang, Q. Yan, B. Yang, L. Peng, and Z. Jia, "TrafficAV: An Effective and Explainable Detection of Mobile Malware Behavior using Network Traffic," in *IWQoS*, 2016, pp. 1–6.

[9] Q. Shang, L. Feng, and S. Gao, "A hybrid method for traffic incident detection using random forest-recursive feature elimination and long short-term memory network with bayesian optimization algorithm," *IEEE Access*, vol. 9, pp. 1219–1232, 2021.

[10] M. Alrowaily, F. Alenezi, and Z. Lu, "Effectiveness of Machine Learning Based Intrusion Detection Systems," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, 2019, pp. 277–288.

[11] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[12] Z. Zhang, P. Chirupphapa, H. Esaki, and H. Ochiai, "XGBoosted Misuse Detection in LAN-Internal Traffic Dataset," in *IEEE International Conference on Intelligence and Security Informatics*, 2020, pp. 1–6.

[13] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[14] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-Learning With Differentiable Convex Optimization," in *CVPR*, 2019, pp. 10 649–10 657.

[15] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," *arXiv preprint*, 2018.

[16] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical Networks for Few-shot Learning," in *NIPS*, 2017, pp. 4077–4087.

[17] Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten, "Simpleshot: Revisiting nearest-neighbor classification for few-shot learning," *arXiv preprint*, 2019.

[18] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-Learning with Memory-Augmented Neural Networks," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, 2016, p. 1842–1850.

[19] B. Oreshkin, P. Rodriguez, and A. Lacoste, "TADAM: Task dependent adaptive metric for improved few-shot learning," in *NIPS*, 2018, p. 7352.

[20] A. Devos and M. Grossglauser, "Subspace Networks for Few-shot Classification," *CoRR*, 05 2019.

[21] M. Lichtenstein, P. Sattigeri, R. Feris, R. Giryes, and L. Karlinsky, "Tafssl: Task-adaptive feature sub-space learning for few-shot classification," in *European Conference on Computer Vision*, 2020, pp. 522–539.

[22] A. Alfassy, L. Karlinsky, A. Aides, J. Shtok, S. Harary, R. Feris, R. Giryes, and A. Bronstein, "LaSO: Label-Set Operations Networks for Multi-Label Few-Shot Learning," in *CVPR*, 2019, pp. 6541–6550.

[23] B. Hariharan and R. Girshick, "Low-Shot Visual Recognition by Shrinking and Hallucinating Features," in *ICCV*, 2017, pp. 3037–3046.

[24] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, R. Feris, A. Kumar, R. Giryes, and A. M. Bronstein, "Delta-encoder: an effective sample synthesis method for few-shot object recognition," *arXiv preprint*, 2018.

[25] Z. Chen, Y. Fu, Y. Zhang, Y.-G. Jiang, X. Xue, and L. Sigal, "Multi-Level Semantic Feature Augmentation for One-Shot Learning," *IEEE Transactions on Image Processing*, vol. 28, pp. 4594–4605, 2019.

[26] Q. Cai, Y. Pan, T. Yao, C. Yan, and T. Mei, "Memory Matching Networks for One-Shot Image Recognition," in *CVPR*, 2018, pp. 4080–4088.

[27] O. Vinyals, C. Blundell, T. Lillicrap, koray kavukcuoglu, and D. Wierstra, "Matching Networks for One Shot Learning," in *NIPS*, 2016, pp. 3630–3638.

[28] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. Torr, and T. Hospedales, "Learning to Compare: Relation Network for Few-Shot Learning," in *CVPR*, 2018, pp. 1199–1208.

[29] J. Zhang, C. Zhao, B. Ni, M. Xu, and X. Yang, "Variational Few-Shot Learning," in *ICCV*, 2019, pp. 1685–1694.

[30] T. Munkhdalai and H. Yu, "Meta Networks," in *ICML*, 2017, p. 2554–2563.

[31] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-Learning with Memory-Augmented Neural Networks," in *ICML*, 2016, p. 1842–1850.

[32] Y.-X. Wang, D. Ramanan, and M. Hebert, "Meta-Learning to Detect Rare Objects," in *ICCV*, 2019, pp. 9924–9933.

[33] S. W. Yoon, J. Seo, and J. Moon, "Tapnet: Neural network augmented with task-adaptive projection for few-shot learning," in *ICML*, 2019, pp. 7115–7123.

[34] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[35] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification," in *2018 International Carnahan Conference on Security Technology (ICCST)*, 2018, pp. 1–7.

[36] I. Sharafaldin, A. H. Lashkari, and A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *ICISSP*, 2018.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.