Few-shot Anomaly Detection and Classification Through Reinforced Data Selection

Xiao Han*, Depeng Xu†, Shuhan Yuan*, Xintao Wu‡
* Utah State University, USA, † University of North Carolina at Charlotte, USA,
‡ University of Arkansas, USA
Email: {xiao.han,shuhan.yuan}@usu.edu, depeng.xu@uncc.edu, xintaowu@uark.edu

2022 IEEE International Conference on Data Mining (ICDM) | 978-1-6654-5099-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDM54844.2022.00115

***Abstract*—Due to the scarcity of anomalies, deep anomaly detection models are predominately trained in an unsupervised or semi-supervised manner depending on the availability of a small number of labeled samples. Currently, most unsupervised approaches detect anomalies by identifying the deviate patterns, and some semi-supervised studies also use labeled anomalies to improve performance. However, few studies have focused on how to take advantage of potential anomalies in an easily obtained and large-scale unlabeled dataset. Meanwhile, in a semi-supervised setting, although we assume having a small number of labeled anomalies, the task of anomaly classification is under-exploited. In this work, considering the problem of anomaly detection and classification by giving limited labeled samples as well as a large number of unlabeled samples, we propose a few-shot anomaly detection and classification model through reinforced data selection (FADS), a novel framework that iteratively improves the performance of anomaly detection and classification by exploring the unlabeled dataset to augment the training set. Experimental results show that FADS is able to improve the performance of anomaly detection and classification with only a few labeled samples initially.**

***Index Terms*—anomaly detection, few-shot learning, reinforcement learning**

I. Introduction

Anomaly detection indicates the detection of data samples that significantly deviate from the majority of data [1], [2]. Due to the extensive demand in a wide spectrum of applications, such as external and internal threats in cyberspace, anomaly detection has become an increasingly important research task.

Due to the small number of anomalies, classical supervised learning algorithms cannot be employed. Currently, the majority of approaches are trained in the unsupervised or semi-supervised learning manner [1]–[3]. However, one limitation of existing anomaly detection approaches is that existing approaches cannot further classify anomalies into specific anomaly categories. In many real-world scenarios, understanding the types of anomalies is critical.

However, due to the scarcity of anomalies, the number of anomalies of each class is even smaller. Hence, it is extremely hard to train a multiclass classifier in a traditional supervised manner. Recently, to learn from a limited number of labeled samples, few-shot learning as a special type of machine learning has become an emerging research topic, which aims to learn classifiers given only a few labeled samples of each class [4]. One common strategy of few-shot learning is to project limited samples into a smaller embedding space so that similar samples are grouped together while dissimilar samples are separated [4]–[7]. However, in the anomaly detection scenario, such a cluster assumption only holds for normal samples since normal data are similar. For anomalies, by only having a few samples, it is hard to build a cluster to represent a class of anomalies, especially considering that anomalies are much more diverse. As a result, current few-shot anomaly detection approaches only work on distinguishing anomalies from normal samples and cannot further divide the anomalies into fine-grained classes [8]–[11]. Hence, how to leverage the powerful few-shot learning models for anomaly detection and classification is still under-exploited.

In this work, by following real-world scenarios, we assume that we have plenty of labeled normal samples and limited abnormal samples in each anomaly class, as well as large-scale unlabeled samples. To achieve fine-grained anomaly detection based on few-shot learning, we propose a framework, called the Few-shot Anomaly detection and classification model with reinforced Data Selection (FADS), to leverage the large-scale unlabeled dataset to progressively improve the few-shot learner for anomaly detection and classification.

We assume that the unlabeled dataset follows the data distribution in the real world that consists of a large number of normal samples and a few anomalies. Initially, we train a few-shot learning model on a few labeled anomalies. Then, we iteratively update the few-shot model by selecting potential anomalies from unlabeled samples to augment the labeled training set. Especially, each iteration consists of two steps, data selection and model retraining. In the first step, we apply the current few-shot model to predict anomalies on the unlabeled dataset. The prediction outcomes can be considered as weakly-labeled samples as their predicted labels can be either accurate or inaccurate. We then apply our proposed reinforcement learning-based data selection to identify the weakly-labeled samples with high quality. The chosen samples have a high chance to be correctly labeled and hence are used to compose an augmentation set. After that, we build a new augmented training dataset by combing the existing labeled dataset with the newly generated augmentation set. In the second step, we retrain the few-shot learning model on the augmented training dataset to improve its performance. We expect the performance of FADS keeps improving as the training procedure moves forward.

The contributions of this work can be summarized as follow.

978-1-6654-5099-7/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDM54844.2022.00115

First, we propose a novel framework to progressively improve the few-shot model for anomaly detection and classification through reinforced data selection from an unlabeled dataset. The few-shot model can achieve fine-grained prediction given a small number of labeled anomalies in each class. Second, we propose a reinforcement learning-based data selection strategy that can select correctly-labeled samples from a weakly-labeled set to augment the training set and further improve the performance of the few-shot model. Third, experimental results show that FADS can achieve state-of-the-art performance on anomaly detection and few-shot anomaly classification.

II. Proposed Approach

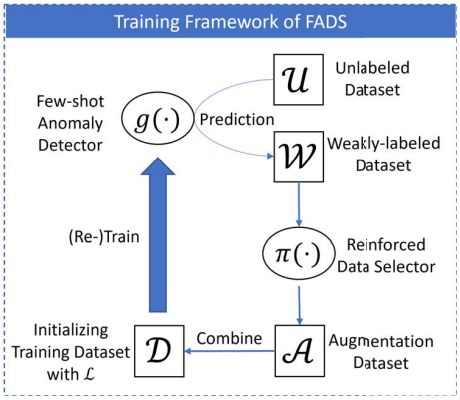

Fig. 1: The training framework of FADS

A. Problem Definition

Let $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{N_L}$ be a labeled dataset, where x_i indicates the i-th sample, while $y_i \in \{0, 1, ..., K\}$ indicates the corresponding label. Particularly, $y_i = 0$ indicates a normal sample while $y_i \in \{1, ..., K\}$ denotes a class of anomalies. Considering that it is usually feasible to have a large number of normal samples in the anomaly detection scenario, the labeled dataset can be decomposed as $\mathcal{L} = \mathcal{L}_N \bigcup \mathcal{L}_A$ with $|\mathcal{L}_N| > |\mathcal{L}_A|$, where $\mathcal{L}_N$ only consists of normal samples and $\mathcal{L}_A$ includes few samples in each anomaly class.

With such a small amount of abnormal samples, it is hard to train an accurate anomaly detection and classification model. To tackle this challenge, in this work, besides a labeled dataset $\mathcal{L}$, we further leverage an unlabeled dataset $\mathcal{U} = \{x_j\}_{j=1}^{N_U}$. The goal is to learn a few-shot anomaly detection and classification model that can leverage the knowledge of the unlabeled dataset $\mathcal{U}$, especially the potential anomalies in $\mathcal{U}$, to maximally improve the performance of the model on anomaly detection and classification.

B. Framework Overview

In this work, we propose a few-shot anomaly detection and classification model through reinforced data selection (FADS), which is able to gradually enhance the performance of the few-shot learning model by exploiting the unlabeled dataset $\mathcal{U}$.

FADS first trains a few-shot learning model on an initial training dataset with only labeled samples $\mathcal{D} = \mathcal{L}$. Then, FADS iteratively improves the model by selecting samples from unlabeled dataset $\mathcal{U}$ to augment the training dataset. We employ the reinforcement learning technique for data selection. In each training iteration, we first use the current few-shot model to predict the label $\hat{y}_j$ for each sample x_j in $\mathcal{U}$, thus producing a weakly-labeled dataset $\mathcal{W} = \{(x_j, \hat{y}_j)\}_{j=1}^{N_U}$. Then, we train a reinforcement learning agent to select the weakly-labeled samples that have high chances to be accurate into an augmentation set $\mathcal{A} = \{(x_j, \hat{y}_j) | a_j = 1\}_{j=1}^{N_U}$, where $a_j \in \{0, 1\}$ indicates whether the sample $(x_j, \hat{y}_j)$ is selected by the agent or not. We combine the augmentation dataset $\mathcal{A}$ with the existing training dataset to compose the new training dataset, i.e., $\mathcal{D} = \mathcal{D} \cup \mathcal{A}$, and then re-train the few-shot model on the new training dataset. We expect that the performance of the few-shot model will be improved with those augmented samples. As the training iteration moves forward, the few-shot anomaly detection model can be improved progressively. The overview of FADS is shown in Figure 1.

C. Prototypical Network

In this work, we adopt the prototypical network [5] as our base few-shot learning model. Following the typical process for training the few-shot learning model, we first randomly select N_S and N_Q samples from the training dataset $\mathcal{D}$ to compose the support set $\mathcal{S}$ and query set $\mathcal{Q}$, respectively.

The prototypical network learns an embedding function $g(\cdot)$ to map each sample x_i to an embedding space, denoted as $\mathbf{x}_i = g(x_i)$. Based on the support set, the prototype representation for each class can be derived by a mean operation:

$$\mathbf{c}_k = \frac{1}{|\mathcal{S}_k|} \sum_{(x_i, y_i) \in \mathcal{S}_k} g(x_i), \tag{1}$$

where $\mathcal{S}_k$ indicates the subset of samples in the support set $\mathcal{S}$ with the class k. Then, given a distance function $d(\cdot)$, the prototypical network predicts the distribution of classes for a query point $x \in \mathcal{Q}$ based on a softmax over distances to the prototypes:

$$p(y = k|\mathbf{x}) = \frac{exp(-d(g(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} exp(-d(g(\mathbf{x}), \mathbf{c}_{k'}))}. \tag{2}$$

The objective function of the prototypical network is the negative log-probability $L = -\log p(y = k|\mathbf{x})$ of the true class k. The training objective is to make the samples from the same class have small distances to the class prototype $\mathbf{c}_k$. A test sample can be labeled based on the label of its nearest prototype in the embedding space.

D. Reinforced Data Selection

As the labeled dataset $\mathcal{L}$ only has a very small number of anomalies, the performance of the initial prototypical network trained on $\mathcal{L}$ still has room for improvement if more training samples can be incorporated. To this end, we propose to exploit the unlabeled dataset $\mathcal{U}$. We first use the current prototypical network to predict the labels of samples in $\mathcal{U}$ and get

a weakly-labeled dataset $\mathcal{W}$, which consists of both correctly and incorrectly labeled samples. If we simply adopt $\mathcal{W}$ to augment the training set, due to the noisy supervised signals, the performance of the prototypical network could be worse. Hence, we propose a reinforcement learning-based data selection method to select samples from $\mathcal{W}$ based on their reliability to compose an augmentation set $\mathcal{A}$. Once the reinforced data selection agent is capable of selecting the correctly labeled samples to augment the training set, the performance of the few-shot learner can be improved. Therefore, the reward to the data selection agent is designed based on the performance of the few-shot learner on an unobserved validation set. The key components of the reinforcement learning framework are described below.

State. For each state s_j with weakly-labeled sample $(x_j, \hat{y}_j) \in \mathcal{W}$, its state representation $\mathbf{s}_j$ is defined as the concatenation of the embedding vector $\mathbf{x}_j = g(x_j)$ and the distance value d_j to the closest prototype, i.e., $\mathbf{s}_j = [\mathbf{x}_j, d_j]$.

Action. The data selection agent then needs to take an action on whether to select this sample $(x_j, \hat{y}_j)$ into the augmentation set $\mathcal{A}$ based on the state representation $\mathbf{s}_j$. In specific, $a_j = 0$ means the sample $(x_j, \hat{y}_j)$ will be rejected, while $a_j = 1$ indicates the sample will be selected and added to the augmentation set $\mathcal{A}$.

Policy Network. The data selection agent makes decisions about whether to select a weakly-labeled sample based on a policy network $\pi_\theta(\cdot)$. We adopt a neural network to parameterize the policy network that takes the state representation as input and outputs the probability of the action, $p(a_j|s_j) = \pi_\theta(\mathbf{s}_j)$. The action a_j is then sampled based on $p(a_j|s_j)$.

Rewards. The policy network is trained with guidance from the reward function. We define the reward based on the performance of the few-shot learning model on an unseen validation set. As in our task, we aim to detect anomalies as well as classify abnormal samples into different classes. Therefore we design the reward function based on the performance of anomaly detection and classification, i.e., an anomaly detection reward r^d and a classification reward r^c. Specifically, we adopt the F1 score as the metric to evaluate the performance of anomaly detection and the macro F1 score over all classes to evaluate the performance of classification.

In our scenario, if wrongly-labeled samples are selected to compose the training dataset, we can expect that the performance of the few-shot learning model will be damaged. On the other hand, if the samples with correct labels are selected, the performance of the model can be improved. Hence, the reward is designed based on the performance change after the model is trained on an augmentation set.

Specifically, in each episode, we first calculate the F1 and macro F1 scores of the current prototypical network. We denote the prototypical network at the beginning of each episode as $g_0(\cdot)$ and the corresponding F1 and macro F1 scores achieved by $g_0(\cdot)$ as $F1_0$ and $MarcoF1_0$. Then, we work on improving $g_0(\cdot)$ by composing an augmentation set. To this end, first, we randomly sample a set of batches $\mathcal{B} = \{B_l\}_{l=1}^{L}$ from $\mathcal{W}$, where each batch B_l consists of a number of samples. Given a batch B_l, for each sample in B_l, we sample an action a_j (select to the augmentation set or not) based on the policy $p(a|s_j) = \pi_\theta(\mathbf{s}_j)$. Then, we can compose the augmentation set $\mathcal{A}_l$ from the batch B_l. After combining $\mathcal{A}_l$ with $\mathcal{D}$, we update the prototypical network, denoted as $g_l(\cdot)$, and further evaluate $g_l(\cdot)$ on a validation set to derive F1 and macro F1 scores, denoted as $F1_l$ and $MacroF1_l$. Finally, the reward for the agent in a batch can be calculated as the difference of F1 scores, $r_l^d = F1_l - F1_0$ and $r_l^c = MacroF1_l - MacroF1_0$. The overall reward function in a batch is formulated as:

$$r_l = r_l^d + \alpha \cdot r_l^c, \tag{3}$$

where α is a hyperparameter to balance anomaly detection and classification tasks. In the experiments, we set $\alpha = 1$.

Optimization. The data selection agent is trained based on the actor-critic algorithm [12]. The output of the actor is the probability of two actions (select or not), while the output of the critic is the predicted reward value based on the current state. Both actor and critic are parameterized by feed-forward neural networks. The goal of the agent is to maximize the rewards, which is defined as:

$$\mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[r_l]. \tag{4}$$

The parameter θ in policy network π_θ is trained based on policy gradient [13]:

$$\theta \leftarrow \theta + \eta \bigtriangledown_\theta \mathcal{J}(\theta), \tag{5}$$

where η indicates the learning rate. The gradient for a batch of weakly-labeled samples can be approximated by [12]

$$\bigtriangledown_\theta \mathcal{J}(\theta) = \frac{1}{|B_l|} \sum_{j=1}^{|B_l|} \bigtriangledown_\theta log \pi_\theta(\mathbf{s}_j)(r_l - V_\varphi(\mathbf{s}_j)), \tag{6}$$

where $|B_l|$ indicates the number of samples in a batch B_l; $V_\varphi(\mathbf{s}_j)$ is the expected reward from a critic network $V_\varphi(\cdot)$ parameterized by φ. The structure of the critic network is similar to the policy network with the last layer as a regression function. The critic network is designed to estimate the expected reward and hence updated according to the cumulative difference between the real reward r_l and the predicted value $V_\varphi(\mathbf{s}_j)$,

$$\mathcal{L}(\varphi) = \frac{1}{|B_l|} \sum_{j=1}^{|B_l|} |r_l - V_\varphi(\mathbf{s}_j)|. \tag{7}$$

The parameters φ in $V_\varphi(s_j)$ can be optimized by:

$$\varphi = \varphi - \eta' \bigtriangledown_\varphi \mathcal{L}(\varphi), \tag{8}$$

where η' indicates the learning rate for updating the critic network.

III. Experiments

A. Experimental Setup

Datasets. We apply FADS to detect and classify anomalies on the following three datasets.

- **UNSW-NB15** [14]. The UNSW-NB15 dataset was generated by the IXIA PerfectStorm tool in the Cyber Range

TABLE I: Statistics of datasets in experiments

Dataset	Class	Training		Testing
		$\mathcal{L}$	$\mathcal{U}$	
UNSW-NB15 (293 features)	Normal	100	30,000	20,000
	Fuzzers	10	3,000	2,000
	DoS	10	3,000	2,000
	Exploits	10	3,000	2,000
	Generic	10	3,000	2,000
	Reconnaissance	10	3,000	2,000
IDS2018 (77 features)	Normal	100	30,000	20,000
	Bot	10	3,000	2,000
	DoS	10	3,000	2,000
	Bruteforce	10	3,000	2,000
CERT (various length)	Normal	100	2,000	2,000
	DataUploading	10	27	28
	DataStealing	10	201	201
	MassEmail	10	10	10
	UnauthdLogin	10	21	21

Lab of UNSW Canberra. It consists of normal and five synthetic attacks including suspension caused by feeding random data (Fuzzers), denial-of-services by overloading the server or network (DoS), attacks with known security problems (Exploits), attack block-ciphers without information about the structure (Generic), and attacks for collecting information (Reconnaissance).

• **IDS2018** [15]. IDS2018 contains detailed network traffic and log files. It consists of normal network traffic and three different types of attacking traffic including automatically synthesized requests of upload, download, screenshots, and keylogging (Bot), denial-of-service by overloading the server or network (DoS), and password crack by brute-force (Bruteforce).

• **CERT** [16]. CERT is an insider threat dataset that contains a collection of synthetic normal and malicious user activities. We use CERT V5.2, which consists of four types of insider threats including unauthorized uploading data (DataUploading), using a thumb drive to steal data (DataStealing), sending out mass emails causing panic (MassEmail), unauthorized logging into other computers (UnauthdLogin). Due to the extremely small number of insiders, the number of samples for each anomaly class is small.

Data Preprocessing. Both UNSW-NB15 and IDS2018 are multidimensional datasets, where each instance consists of multiple features to describe the basic information. After typical preprocessing steps, each sample in UNSW-NB15 consists of 293 features, while each sample in IDS2018 consists of 77 features. We randomly select 100 normal samples as $\mathcal{L}_N$ and 10 samples from each anomaly class as $\mathcal{L}_A$. To construct an unlabeled dataset $\mathcal{U}$, we randomly select 30,000 normal samples and 3,000 abnormal samples from each anomaly class. The testing dataset contains 20,000 normal samples and 2,000 abnormal samples from each anomaly class.

CERT is a sequential dataset, where each entry indicates one user activity. For CERT V5.2, there are 23 different types of activities, such as logon, logoff, web visiting, and file editing. We group user activities into sessions as data samples, and each session consists of all user activities between logon and logoff operations on a computer. Similarly, we randomly select 100 normal samples as $\mathcal{L}_N$ and 10 samples from each anomaly class as $\mathcal{L}_A$. The unlabeled dataset $\mathcal{U}$ is composed of randomly selected 2,000 normal sessions and half of the remaining abnormal sessions (excluding the 10 sessions used in $\mathcal{L}_A$) from each anomaly class. The testing dataset consists of 2,000 normal sessions and the other half of the remaining data in each anomalous class. Table I summarizes the statistics of datasets used in our experiments.

Baselines. We use two sets of baselines to evaluate the performance of FADS for anomaly detection and classification, respectively.

Baselines for anomaly detection. We adopt the following six baselines to evaluate the performance of anomaly detection: 1) **One-Class SVM (OCSVM)** [17], is a one-class classification model; 2) **Isolation Forest (iForest)** [18] is a tree-based anomaly detection model; 3) **Label Random PU Learning (LRPU)** [19] is a positive-unlabeled learning approach; 4) **PU learning with a Selection Bias (PUSB)** [20] is an advanced positive-unlabeled learning approach; 5) **Deep Support Vector Data Description (DeepSVDD)** [21] is a deep learning-based one-class anomaly detection method; 6) **Deep Semi-Supervised Anomaly Detection (Deep SAD)** [22] is a semi-supervised anomaly detection model.

OCSVM, iForest, and DeepSVDD as one-class models are trained on the normal set $\mathcal{L}_N$. LRPU and PUSB as positive-unlabeled models are trained on the anomalous set $\mathcal{L}_A$ and unlabeled set $\mathcal{U}$. Deep SAD as a semi-supervised model is trained on the labeled set $\mathcal{L}$ and unlabeled set $\mathcal{U}$.

Baselines for anomaly classification. We adopt the following two baselines to evaluate the performance of anomaly classification, **Support Vector Machine (SVM)** [23] and **Multilayer Perceptron (MLP)** [24].

We train both baselines on the labeled set $\mathcal{L}$ and also adopt the synthetic minority oversampling technique (SMOTE) [25] for oversampling the data in anomaly classes.

Evaluation Metrics. For anomaly detection, we consider all anomaly classes as one anomaly class and adopt the Area Under Precision-Recall Curve (AUC-PR), the Area Under Receiver Operating Characteristic Curve (AUC-ROC), and False Positive Rate (FPR) at 95% True Positive Rate (TPR) to measure the performance of FADS. Especially, FPR at 95% TPR can be considered as the probability that an anomaly is misclassified as normal when the true positive rate is 95%. For anomaly classification, F1 score is used to evaluate the performance of FADS on each abnormal class. We also derive the Macro-F1 score to evaluate the performance over all classes. In our experiments, we report the results over 10 runs. The paired t-test is adopted to examine the statistical significance of FADS over the best baseline.

B. Implementation Details

For UNSW-NB15 and IDS2018 datasets, we adopt a feed-forward neural network with three fully connected layers as the implementation of the prototypical network $g(\cdot)$. For the CERT dataset, we adopt a GRU neural network [26] to map

sequences into an embedding space with the dimension of 64. For all three datasets, as we only have 10 labeled samples in each anomaly class in the initial training set $\mathcal{L}_A$, we initially assign 6 samples to $\mathcal{D}^{tr}$ to train the prototypical network with 3 samples as the support set $\mathcal{S}$ and 3 samples as the query set $\mathcal{Q}$ and the rest 4 samples to $\mathcal{D}^{val}$ for evaluation. With the training episode increasing, we assign more samples selected by the policy network to $\mathcal{D}^{tr}$ and $\mathcal{D}^{val}$ with a fixed ratio of 30% of samples in $\mathcal{S}$, 30% of samples in $\mathcal{Q}$, and 40% of samples in $\mathcal{D}^{val}$ to retrain the prototypical network.

In the implementation of actor-critic reinforcement learning, both actor and critic are three-layer feedforward neural networks. In the training stage, we set the number of training episodes $T = 20$, while in each episode, we generate a set of sample batches $\{B_l\}_{l=1}^{L}$ with $L = 50$. For each batch B_l, we randomly select samples from each predicted class in $\mathcal{W}$ and compose a balanced set for the actor to select. Especially, for UNSW-NB15 and IDS2018 datasets, we randomly select 10 samples from each class, so the sizes of each batch are $|B_l| = 60$ for the UNSW-NB15 dataset and $|B_l| = 40$ for the IDS2018 dataset. For the CERT dataset, we select 2 samples from each class with the batch size $|B_l| = 10$. The **code and datasets** used in the experiments are available online [1].

C. Experimental Results

TABLE II: Results on anomaly detection (mean ± std.). ↑ indicates larger value is better; ↓ indicates lower value is better.

		UNSW-NB15	IDS2018	CERT
AUC-PR ↑	OCSVM	0.5001 ± 0.0020	0.3026 ± 0.0137	0.1233 ± 0.0199
	iForest	0.7564 ± 0.0321	0.2936 ± 0.0093	0.1331 ± 0.0218
	LRPU	0.6542 ± 0.0049	0.4393 ± 0.0689	0.1395 ± 0.0362
	PUSB	0.6108 ± 0.0372	0.3741 ± 0.0155	0.1147 ± 0.0000
	DeepSVDD	0.6334 ± 0.0163	0.3516 ± 0.0759	0.1149 ± 0.0087
	DeepSAD	0.8203 ± 0.1976	0.5959 ± 0.0905	0.1269 ± 0.0103
	FADS	$\mathbf{0.9178 \pm 0.0214}$	$\mathbf{0.6791 \pm 0.0279}^{*}$	$\mathbf{0.2480 \pm 0.0260}^{**}$
AUC-ROC ↑	OCSVM	0.7501 ± 0.0020	0.6447 ± 0.0202	0.5122 ± 0.0910
	iForest	0.9131 ± 0.0181	0.5513 ± 0.0233	0.5751 ± 0.0827
	LRPU	0.7407 ± 0.0037	0.6828 ± 0.0894	0.5404 ± 0.0623
	PUSB	0.7081 ± 0.0279	0.5932 ± 0.0101	0.5000 ± 0.0000
	DeepSVDD	0.7932 ± 0.0298	0.6235 ± 0.0709	0.5032 ± 0.0409
	DeepSAD	0.9042 ± 0.1219	0.8310 ± 0.0553	0.5510 ± 0.0419
	FADS	$\mathbf{0.9751 \pm 0.0080}^{*}$	$\mathbf{0.9249 \pm 0.0134}^{**}$	$\mathbf{0.7286 \pm 0.0286}^{**}$
FPR (95% TPR) ↓	OCSVM	0.9938 ± 0.0007	0.9819 ± 0.0031	0.9046 ± 0.0484
	iForest	0.1941 ± 0.0459	0.9796 ± 0.0228	0.7732 ± 0.1477
	LRPU	0.4557 ± 0.1765	0.6235 ± 0.0776	0.8872 ± 0.0489
	PUSB	0.9923 ± 0.0041	0.8291 ± 0.2636	0.9532 ± 0.0070
	DeepSVDD	0.7081 ± 0.1564	0.8515 ± 0.1457	0.8694 ± 0.0572
	DeepSAD	0.2884 ± 0.4027	0.4002 ± 0.3597	0.7645 ± 0.1141
	FADS	$\mathbf{0.0375 \pm 0.0180}^{*}$	$\mathbf{0.2380 \pm 0.1108}$	$\mathbf{0.7036 \pm 0.1119}$

Significantly outperforms DeepSAD at the: * 0.05 and ** 0.01 level, paired t-test.

Anomaly Detection. We first evaluate the performance of FADS for anomaly detection. In particular, after the prototypical network predicts each anomaly into a specific anomaly class, we consider all detected samples in anomaly classes as anomalies. We compare FADS for anomaly detection with two traditional one-class classification models (OCSVM and iForest), two positive-unlabeled learning models (LRPU and PUSB), and two advanced deep anomaly detection models (DeepSVDD and DeepSAD). As shown in Table II, FADS achieves better performance than all baselines with a large margin on all three datasets. Especially, although DeepSAD as a semi-supervised model also uses both labeled and unlabeled datasets, FADS significantly outperforms DeepSAD in most cases and has a much smaller standard deviation. It means actively identifying the potential anomalies in the unlabeled dataset can improve the performance of anomaly detection. Meanwhile, DeepSAD still outperforms other baselines, which shows the advantage of using a small set of labeled samples for anomaly detection. LRPU as a PU learner achieve better performance than three one-class models (OCSVM, iForest, and DeepSVDD) on IDS2018 and CERT datasets, meaning that in some cases, leveraging the unlabeled set can improve the performance. However, due to the limited anomalies (positive samples), both LRPU and PUSB underperform FADS.

[1]https://github.com/hanxiao0607/FADS

TABLE III: Results on anomaly classification (mean ± std.)

Dataset	Class	F1		
		SVM	MLP	FADS
UNSW-NB15	Normal	0.9356 ± 0.0219	0.9474 ± 0.0146	$\mathbf{0.9766 \pm 0.0070}$
	Fuzzers	0.3369 ± 0.1012	$\mathbf{0.4079 \pm 0.0471}$	0.3808 ± 0.0530
	DoS	$\mathbf{0.4996 \pm 0.1607}$	0.4797 ± 0.1040	0.4130 ± 0.1036
	Exploits	0.1174 ± 0.1338	$\mathbf{0.3809 \pm 0.0835}$	0.3720 ± 0.0809
	Generic	$\mathbf{0.9752 \pm 0.0049}$	0.7428 ± 0.0841	0.9478 ± 0.0362
	Reconnaissance	0.2236 ± 0.1037	0.2663 ± 0.0683	$\mathbf{0.3052 \pm 0.0585}$
	macro-average	0.5147 ± 0.0482	0.5375 ± 0.0353	$\mathbf{0.5659 \pm 0.0348}^{*}$
IDS2018	Normal	0.7150 ± 0.0331	0.8398 ± 0.0653	$\mathbf{0.9246 \pm 0.0102}$
	Bot	0.5972 ± 0.0373	$\mathbf{0.6298 \pm 0.0303}$	0.6269 ± 0.0191
	DoS	0.3126 ± 0.0525	0.4693 ± 0.1540	$\mathbf{0.7817 \pm 0.0668}$
	Bruteforce	0.6684 ± 0.0813	0.7053 ± 0.0952	$\mathbf{0.8371 \pm 0.0465}$
	macro-average	0.5733 ± 0.0292	0.6610 ± 0.0628	$\mathbf{0.7926 \pm 0.0197}^{**}$
CERT	Normal	$\mathbf{0.9392 \pm 0.0000}$	0.6397 ± 0.0653	0.8830 ± 0.0196
	DataUploading	0.0000 ± 0.0000	0.3196 ± 0.1032	$\mathbf{0.4440 \pm 0.1228}$
	DataStealing	0.0000 ± 0.0000	0.4023 ± 0.0382	$\mathbf{0.4477 \pm 0.0568}$
	MassEmail	0.0000 ± 0.0000	0.0428 ± 0.0083	$\mathbf{0.1179 \pm 0.0553}$
	UnauthdLogin	0.0000 ± 0.0000	0.1428 ± 0.0229	$\mathbf{0.4755 \pm 0.0860}$
	macro-average	0.1787 ± 0.0000	0.3095 ± 0.0331	$\mathbf{0.4736 \pm 0.0344}^{**}$

Significantly outperforms MLP in terms of macro-F1 at the: * 0.05 and ** 0.01 level, paired t-test.

Anomaly Classification. FADS can achieve fine-grained anomaly classification based on the prototypical networks, so we further evaluate FADS for the anomaly classification task. We compare FADS with two classification models, SVM and MLP. Table III shows the results on anomaly classification. In short, FADS achieves the highest macro-F1 score over all classes, meaning that FADS can achieve the best performance for anomaly classification. Meanwhile, on the UNSW-NB15 and IDS2018 datasets, FADS achieves the best performance in most classes in terms of F1 score compared with SVM and MLP. CERT is the most challenging dataset because of extremely limited labeled samples. On the CERT dataset, SVM predicts all anomalies as normal, while MLP has a low recall in the normal set indicating a high misclassification rate on normal samples. On the other hand, FADS achieves a significant improvement compared with baselines on CERT. It indicates for the challenging anomaly detection task with extremely limited samples, leveraging anomalies in the unlabeled dataset is critical to boosting performance.

FADS with and without Data Selection. We further evaluate the performance improvement of FADS over the simple prototypical networks without reinforced data selection. We consider three settings as baselines. First, we train a prototypical network on the initially labeled dataset, denoted as *ProtoNet*. Second, after we train the prototypical network on the initial dataset, we apply the network to label the unlabeled dataset $\mathcal{U}$ and use the whole weakly-labeled dataset $\mathcal{W}$ as the augmentation set ($\mathcal{A} \leftarrow \mathcal{W}$) to retrain the prototypical network. We evaluate the performance of the prototypical

TABLE IV: Performance of FADS with or without reinforced data selection (mean $\pm$ std.)

Dataset	Approach	Anomaly Detection	Anomaly Classification
		AUC-PR ↑	F1 ↑
UNSW-NB15	ProtoNet	0.8639 ± 0.0461	0.5307 ± 0.0388
	ProtoNet*	0.7868 ± 0.0222	0.4819 ± 0.0449
	ProtoNet*(10%)	0.7578 ± 0.0288	0.4384 ± 0.0355
	FADS	$\mathbf{0.9178 \pm 0.0214}$	$\mathbf{0.5659 \pm 0.0348}$
IDS2018	ProtoNet	0.6086 ± 0.0643	0.7349 ± 0.0431
	ProtoNet*	0.5751 ± 0.0713	0.7138 ± 0.0523
	ProtoNet*(10%)	0.4825 ± 0.0818	0.5474 ± 0.0880
	FADS	$\mathbf{0.6791 \pm 0.0279}$	$\mathbf{0.7926 \pm 0.0197}$
CERT	ProtoNet	0.1818 ± 0.0314	0.3595 ± 0.0606
	ProtoNet*	0.1747 ± 0.0194	0.3525 ± 0.0434
	ProtoNet*(10%)	0.1892 ± 0.0267	0.3905 ± 0.0589
	FADS	$\mathbf{0.2480 \pm 0.0260}$	$\mathbf{0.4736 \pm 0.0344}$

network after re-training on the test set, denoted as *ProtoNet**. The third one is similar to ProtoNet*, but we retrain the prototypical network based on samples in $\mathcal{W}$ having top 10% of the highest probabilities in each class, which means high confidence belonging to a class, denoted as *ProtoNet* (10%)*.

Table IV shows that FADS achieves unanimously better performances over baselines on three datasets for both anomaly detection and classification tasks. Meanwhile, we can notice that in general, both ProtoNet* and ProtoNet* (10%) cannot beat regular ProtoNet, which shows that simply using weakly labeled samples for training will hurt the performance due to incorrect labels. Furthermore, FADS still has the largest performance gains on the CERT dataset in terms of macro-F1 for classification, which shows the criticalness of augmenting the training set with reliable samples in the case of limited samples. Another advantage of FADS is that it can have a lower standard deviation compared with traditional prototypical networks that do not have the augmentation dataset from the reinforced data selection.

IV. Conclusion

In this paper, we have developed FADS, a few-shot anomaly detection and classification model through reinforced data selection. The core idea of FADS is to improve a few-shot learning model iteratively by selecting the potential anomalies from an unlabeled dataset to augment the training dataset through a reinforcement learning-based agent. Especially, FADS trains the prototypical network with extremely limited samples and applies the prototypical network to label an unlabeled dataset to get a weakly-labeled dataset. A reinforced data selection agent is then trained to identify the reliable weakly-labeled samples to augment the training set for the prototypical network. The training procedure is conducted as a loop to iteratively improve the performance of the prototypical network for anomaly detection and classification. The experimental results show the advantage of FADS.

Acknowledgment

This work was supported in part by NSF grants 1564250 and 2103829.

References

[1] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[2] G. Pang, C. Shen, L. Cao, and A. v. d. Hengel, "Deep learning for anomaly detection: A review," *arXiv preprint arXiv:2007.02500*, 2020.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[4] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[5] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *arXiv preprint arXiv:1703.05175*, 2017.

[6] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, pp. 3630–3638, 2016.

[7] A. Ayyad, Y. Li, R. Muaz, S. Albarqouni, and M. Elhoseiny, "Semi-supervised few-shot learning with prototypical random walks," in *AAAI Workshop on Meta-Learning and MetaDL Challenge*. PMLR, 2021, pp. 45–57.

[8] K. Ding, Q. Zhou, H. Tong, and H. Liu, "Few-shot network anomaly detection via cross-network meta-learning," in *Proceedings of the Web Conference 2021*, ser. WWW '21, 2021, p. 2448–2456.

[9] S. Yuan, P. Zheng, X. Wu, and H. Tong, "Few-shot insider threat detection," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, ser. CIKM '20, 2020.

[10] G. Pang, C. Ding, C. Shen, and A. v. d. Hengel, "Explainable deep few-shot anomaly detection with deviation networks," *arXiv preprint arXiv:2108.00462*, 2021.

[11] Y. Lu, F. Yu, M. K. K. Reddy, and Y. Wang, "Few-shot scene-adaptive anomaly detection," in *European Conference on Computer Vision*. Springer, 2020, pp. 125–141.

[12] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[13] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[14] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[15] A. collaborative project between the Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC), "A realistic cyber defense dataset (cse-cic-ids2018)."

[16] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *2013 IEEE Security and Privacy Workshops*, 2013, pp. 98–104.

[17] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt *et al.*, "Support vector method for novelty detection." in *NIPS*, vol. 12. Citeseer, 1999, pp. 582–588.

[18] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[19] C. Elkan and K. Noto, "Learning classifiers from only positive and unlabeled data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 213–220.

[20] M. Kato, T. Teshima, and J. Honda, "Learning from positive and unlabeled data with a selection bias," in *International conference on learning representations*, 2018.

[21] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 4393–4402.

[22] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, "Deep semi-supervised anomaly detection," in *International Conference on Learning Representations*, 2019.

[23] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, V. Vapnik *et al.*, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, pp. 155–161, 1997.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. Cambridge, MA, USA: MIT Press, 1986, p. 318–362.

[25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[26] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.