Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Adaptive and augmented active anomaly detection on dynamic network traffic streams*

Bin LI[1], Yijie WANG‡[1], Li CHENG[2]

[1]*National Key Laboratory of Parallel and Distributed Computing, College of Computer, National University of Defense Technology, Changsha 410073, China*

[2]*College of System Engineering, National University of Defense Technology, Changsha 410073, China*

E-mail: libin16a@nudt.edu.cn; wangyijie@nudt.edu.cn; chengli09@nudt.edu.cn

Received Apr. 8, 2023; Revision accepted July 4, 2023; Crosschecked Feb. 4, 2024

Abstract: Active anomaly detection queries labels of sampled instances and uses them to incrementally update the detection model, and has been widely adopted in detecting network attacks. However, existing methods cannot achieve desirable performance on dynamic network traffic streams because (1) their query strategies cannot sample informative instances to make the detection model adapt to the evolving stream and (2) their model updating relies on limited query instances only and fails to leverage the enormous unlabeled instances on streams. To address these issues, we propose an active tree based model, adaptive and augmented active prior-knowledge forest (A^3PF), for anomaly detection on network traffic streams. A prior-knowledge forest is constructed using prior knowledge of network attacks to find feature subspaces that better distinguish network anomalies from normal traffic. On one hand, to make the model adapt to the evolving stream, a novel adaptive query strategy is designed to sample informative instances from two aspects: the changes in dynamic data distribution and the uncertainty of anomalies. On the other hand, based on the similarity of instances in the neighborhood, we devise an augmented update method to generate pseudo labels for the unlabeled neighbors of query instances, which enables usage of the enormous unlabeled instances during model updating. Extensive experiments on two benchmarks, CIC-IDS2017 and UNSW-NB15, demonstrate that A^3PF achieves significant improvements over previous active methods in terms of the area under the receiver operating characteristic curve (AUC-ROC) (20.9% and 21.5%) and the area under the precision-recall curve (AUC-PR) (44.6% and 64.1%).

Key words: Active anomaly detection; Network traffic streams; Pseudo labels; Prior knowledge of network attacks
https://doi.org/10.1631/FITEE.2300244 **CLC number:** TP309

1 Introduction

In the past few years, the growth of various emerging technologies has resulted in a massive accumulation of network traffic. Meanwhile, serious network attacks are increasing rapidly, targeting industrial infrastructures, computer networks, and personal end devices. Compared with the network traffic of normal users, these network attacks often show anomalous behaviors. Active anomaly detection in network traffic (Beaugnon et al., 2017; Liu TL et al., 2019; Siddiqui et al., 2019; Dong, 2021; Guerra-Manzanares and Bahsi, 2023) has become increasingly critical over the years, and considers the practical problem that annotating a large network traffic data set with intrusion labels is costly and requires significant domain knowledge from experts.

‡ Corresponding author

* Project supported by the National Science and Technology Major Project (No. 2022ZD0115302), the National Natural Science Foundation of China (No. 61379052), the Science Foundation of Ministry of Education of China (No. 2018A02002), and the Natural Science Foundation for Distinguished Young Scholars of Hunan Province, China (No. 14JJ1026)

ORCID: Bin LI, https://orcid.org/0000-0003-0876-2694; Yijie WANG, https://orcid.org/0000-0002-2913-4016

These methods sample as few informative network traffic instances as possible for manual labels and use them to train the detection model. Several active anomaly detection methods have recently been proposed to detect attacks on infinite network traffic streams, but they still face the following challenges:

First, with the proliferation of heterogeneous network systems, more users are involved in the network (Roshan et al., 2018) and attack modes are evolving constantly (Bilge and Dumitras, 2012). Existing active anomaly detection methods have low self-adaptability on network traffic streams because typical query strategies, such as anomaly-based strategies (Veeramachaneni et al., 2016; Kathareios et al., 2017), uncertainty-based strategies (Viegas et al., 2019), and random sampling based strategies (Shan et al., 2019), cannot effectively learn the dynamic distribution of network traffic streams by sampling informative instances.

Second, most existing active learning based methods update models only based on limited query instances (Veeramachaneni et al., 2016; Viegas et al., 2019; Li et al., 2022), and fail to exploit the enormous unlabeled instances. In practical scenarios, labeled network traffic instances are very limited due to the high costs of annotation, while abundant unlabeled network traffic data are typically more easily obtained. As studied by Apruzzese et al. (2022), these unlabeled instances can help improve the generalization performance of the model by capturing the underlying distribution of the data more accurately.

To address these problems, in this paper we introduce an active tree based model named adaptive and augmented active prior-knowledge forest (A^3PF) for anomaly detection on network traffic streams. We first construct a prior-knowledge forest using prior knowledge of network attacks to find feature subspaces that better distinguish network anomalies from normal traffic. Next, a novel adaptive query strategy is proposed to sample the most informative instances for manual labeling and model updating. Informativeness is measured in terms of the dynamic change of data distribution and the degree of uncertainty anomaly. Based on these informative instances, this strategy can make the model better adapt to network traffic changes. Finally, we further devise an augmented update mechanism to leverage the enormous unlabeled instances in model updating. Specifically, A^3PF generates pseudo labels for the unlabeled neighbors of query instances, assuming that the neighboring instances tend to belong to the same class. The instances with pseudo labels are used to incrementally update the model together with the query instances, which makes the detection model more generalizable. Comprehensive experiments using two intrusion benchmarks, CIC-IDS2017 (Sharafaldin et al., 2018) and UNSW-NB15 (Moustafa and Slay, 2015b), validate the performance of A^3PF. Compared with existing active methods, in terms of the area under the receiver operating characteristic curve (AUC-ROC) and the area under the precision-recall curve (AUC-PR), A^3PF achieves significant improvements. The contributions of this work are summarized as follows:

1. We propose an active model, A^3PF, which uses prior knowledge of attacks to detect anomalies in network traffic streams.

2. A novel adaptive query strategy is proposed in A^3PF to sample informative instances, which makes the model effectively adapt to dynamic network traffic streams.

3. To leverage unlabeled instances in achieving generalized model updates, some unlabeled instances are assigned pseudo labels based on these query instances.

4. Comprehensive experiments using two benchmarks demonstrate the effectiveness of A^3PF.

2 Related works

The related works in the field can be categorized as stream-based active network anomaly detection and stream-based active learning.

2.1 Stream-based active network anomaly detection

Active network anomaly detection samples as few network traffic instances as possible to maximize the informativeness of limited labeled instances. These active network anomaly detection methods query the labels of sampled instances and train the detection model on these query instances, and have been widely employed in network intrusion detection (Liu TL et al., 2019; Siddiqui et al., 2019; Dong, 2021; Wu et al., 2021; Guerra-Manzanares and Bahsi, 2023). However, because the models in these methods are retrained from the beginning in every query,

the efficiency will be degraded when these methods are directly used on the infinite dynamic network traffic streams (Shahraki et al., 2022). Recently, several stream-based active network anomaly detection methods have been proposed using different detection models and query strategies to achieve the model update. Although these methods demonstrate their effectiveness in detecting network anomalies, they still face the following challenges:

First, these stream-based active network anomaly detection methods cannot sample informative instances, which results in ineffective learning of the dynamic data distribution. The query strategy could be categorized into the anomaly-based query strategy, uncertainty-based query strategy, and random sampling based query strategy. The anomaly-based query strategy has been widely adopted to find potential network attacks. Veeramachaneni et al. (2016) proposed an analyst-in-the-loop security system, named AI^2, which includes three unsupervised anomaly detection models (autoencoder, principal component analysis, and copula density function) and a supervised random forest to detect anomalies based on querying the labels of the most anomalous instances. Kathareios et al. (2017) selected the most anomalous instances based on a shallow autoencoder model, and then used a custom nearest-neighbor classifier to filter the false positives. They sampled the instances most different from the known data distribution, which cannot help the model acquire knowledge of all dynamic changes on streams, especially when the number of query instances is limited. The uncertainty-based query strategy focuses on selecting the instances that are most difficult for the model to predict based on these known classes. Viegas et al. (2019) proposed BigFlow, composed of several Hoeffding trees (Hulten et al., 2001), to select the least confident instances based on the prediction probability for manual labels. This strategy helps the model describe the boundaries between the known classes clearly, but it is unable to find dynamic changes in network traffic streams. In addition, the random sampling based query strategy samples random instances to cover the whole data space and finds the changes in the data distribution. However, the unnecessary labeling cost increases because most random samples are valueless.

Second, most existing methods update detection models only on queried instances (Veeramachaneni et al., 2016; Viegas et al., 2019; Li et al., 2022), and fail to exploit unlabeled network traffic instances. Complex data distribution makes these methods ineffective and unstable in detecting network anomalies, especially when the number of query instances is extremely limited.

Finally, few methods leverage the useful intrinsic characteristics of these attacks to help the model achieve better performance in anomaly detection. Wang et al. (2021) used the rules to assign labels on flow instances based on the characteristics of the data set, but they did not explain how to extract the domain rules with specific feature values. Moreover, their prior knowledge may be overfit to the specific data sets and lacks generalizability in different network environments. Li et al. (2022) adopted the relationship between source Internet protocol (IP) addresses and destination IP addresses to exploit potential attack flows, but attackers can use IP spoofing techniques (Hafeez and Khalil, 2023) to randomly change the IP addresses during the attack phase to hide the attacker's identity. That is, the pair of addresses may not disclose the true relationships of attackers and victims.

2.2 Stream-based active learning

Stream-based active learning methods are also related to our work. Shan et al. (2019) proposed a new online active learning ensemble (OALE) framework for drifting data streams based on a long-term stable classifier and multiple dynamic classifiers. It takes a non-fixed labeling budget, supports on-demand request labeling, and adopts an uncertainty-based or random strategy to label instances. Similar ideas can be seen in Korycki et al. (2019), who proposed a dynamic ensemble classifier for binary classification to handle dynamic, sparsely labeled, and imbalanced streams. Semi-supervised adaptive classification over data stream (SACCOS) (Gao et al., 2022) is an active learning based cluster method that analyzes data streams. It leverages an ensemble of graph-based clusters in local regions to query the labels of instances outside clusters. An individual supervised classifier is also retrained with these query instances. Yan et al. (2021) proposed a clustering-based data stream classification framework based on a density-based stream clustering procedure. An active strategy was proposed to query anomaly instances according to the three-sigma

principle of the Gaussian distribution. The subcluster was modeled using the query instances and new instances classified using the K-nearest neighbors (KNN) method to address the overlap between classes. Das et al. (2019, 2020) proposed a stream-based active learning method, active anomaly detection on streams (AADS), to maximize the number of true anomalies discovered. To handle streaming data settings, they presented a novel data drift detection algorithm that not only detects the drift robustly but also allows taking corrective actions to adapt to the detector in a principled manner.

3 Methodologies

3.1 Problem statement

The network traffic stream $\mathcal{S}$ is shown in Eq. (1), where $\mathcal{X}^b = \{\boldsymbol{x}_0^b, \boldsymbol{x}_1^b, \cdots, \boldsymbol{x}_{|\mathcal{X}^b|-1}^b\}$ is the b^{th} batch of the stream. Each batch has the same size of flows, denoted as $|\mathcal{X}^b|$. Each flow is represented by a statistical flow feature vector with d dimensions and analyzed as a flow instance in later discussion. Let $\boldsymbol{x}_i^b \in \mathbb{R}^d$ be the i^{th} network flow of the b^{th} batch. To simplify, we use $\boldsymbol{x}_i$ to take the place of $\boldsymbol{x}_i^b$ to represent the i^{th} instance in the latest batch $\mathcal{X}^b$ if no additional instruction is shown.

$$\mathcal{S} = \{\cdots, \mathcal{X}^{b-1}, \mathcal{X}^b, \mathcal{X}^{b+1}, \cdots\}. \tag{1}$$

In our paper, all network flow instances include two categories: normal traffic and anomalies. The labels of these flow instances are denoted as y=1 for anomalies and y=0 for normal traffic. The active anomaly detection in network traffic streams includes two stages. First, static unlabeled network traffic $\mathcal{D}$ is stored offline and leveraged to initialize our model. Then, with the arriving network traffic stream $\mathcal{S}$, the model scores each flow instance in the arriving batch $\mathcal{X}^b$. According to the query ratio θ, $\lfloor |\mathcal{X}^b| \cdot \theta \rfloor$ instances are queried to obtain manual labels in each batch and used to update the model incrementally.

3.2 Preliminaries

In this subsection we introduce the prior knowledge of network attacks used in our paper. Network attacks are highly related to features. For example, in the well-known intrusion benchmark UNSW-NB15 (Moustafa and Slay, 2015b), the denial of service (DoS) attack (a malicious attempt to make a server or a network resource unavailable to users, usually by interrupting or suspending the services of a host connected to the Internet) is highly related to the feature ct_srv_dst (the number of flows of the same service and destination IP address in the last 100 flows). The higher number of connections to the same destination IP addresses with the same service means that a DoS attack is more likely to happen. Consequently, the feature subspace where anomalies deviate more from normal traffic could be generated if these features related to attacks could be assigned higher weights in anomaly detection. Thus, attacks could be distinguished from normal traffic more reasonably and effectively.

Prior knowledge of network attacks is an important supplement to the model because domain experts carefully craft it. Few methods leverage it to assist the machine learning based model in the active anomaly detection on network traffic streams. We use the relation between features and attacks as the generalized prior knowledge. The prior-knowledge weight vector $\boldsymbol{w}$ can be defined as $\boldsymbol{w} = [w(0), w(1), \cdots, w(d-1)]$, where $w(j)$ $(j = 0, 1, ..., d-1)$ is the weight assigned to the j^{th} feature. The weights of features weakly related to these attacks are set to 1, while the weights of features strongly related to attacks are set to higher values. Compared to the related method (Wang et al., 2021), the prior knowledge used in our paper is more generalized and does not target a specific data set. This is because we extract the important features of attacks rather than specific feature values used in Wang et al. (2021). In other words, the generalized prior knowledge we use could make the detection model useful in network environments. The important features come from the analysis in Sharafaldin et al. (2018) and Moustafa and Slay (2015a) for CIC-IDS2017 and UNSW-NB15, respectively. We list these important features in each benchmark in Table 1.

3.3 Overview

We first summarize the notations used in our paper as shown in Table 2. Then, the entire active anomaly detection model on network traffic streams is shown in Fig. 1. In the initialization stage, A^{3}PF first ensembles a set of tree models to construct the prior-knowledge forest based on the prior knowledge of network attacks. A tree with eight leaf nodes

Table 1 Attack-related features in CIC-IDS2017 and UNSW-NB15

Benchmark	Attack-related features
CIC-IDS2017	Subflow F.Bytes, Flow Duration, B.Packet Len Std, Total Len F.Packets, Init Win F.Bytes, F.IAT Min, B.Packet Len Min, F.IAT Min, Active Min, Active Mean, B.IAT Mean, F.PSH Flags, SYN Flag Count, Init Win B.Bytes, F.Packet Len Mean, B.Packets/s, PSH Flag Count, Avg Packet Size
UNSW-NB15	dttl, ct_state_ttl, state, sttl, sload, sload, service, ct_srv_dst, ct_srv_src, is_sm_ips_ports, dloss, dbytes, spkts, ct_src_tm, ct_src_dport_ltm, sloss

Table 2 Notations used in this paper

Notation	Description	Notation	Description
$\mathcal{S}$	Network traffic stream	$\mathcal{D}$	Static network traffic
d	Dimension of flow instances	$\mathcal{X}^b$	The b^{th} batch of network traffic stream
$\lvert\mathcal{X}^b\rvert$	Size of the b^{th} batch	$\boldsymbol{x}_i^b$	The i^{th} flow instance in the b^{th} batch
θ	Query ratio	r	Ratio of pseudo label instances
$\boldsymbol{w}$	Weights of prior knowledge	t	Number of trees
s	Laplacian smoothing parameter	$\mathcal{Z}$	Set of query and pseudo label instances
m_l	Number of instances in leaf node l	e	Tree depth
a	Anomaly score	q	Query score
α	Dynamic score	β	Uncertain anomaly score
$\mathcal{P}_l$	Nodes along the tree path from root to leaf node l	γ	Augmentation score of the unlabeled instance
$\mathcal{L}_{\text{a}}$ ($\mathcal{L}_{\text{n}}$)	Leaves with a majority of attacks (normal traffic)	δ	Rectifying term
u_{a} (u_{n})	Number of queried attacks (normal traffic) of each batch	$\mathcal{L}_i^k$	Leaf node in which $\boldsymbol{x}_i$ is located in the k^{th} tree

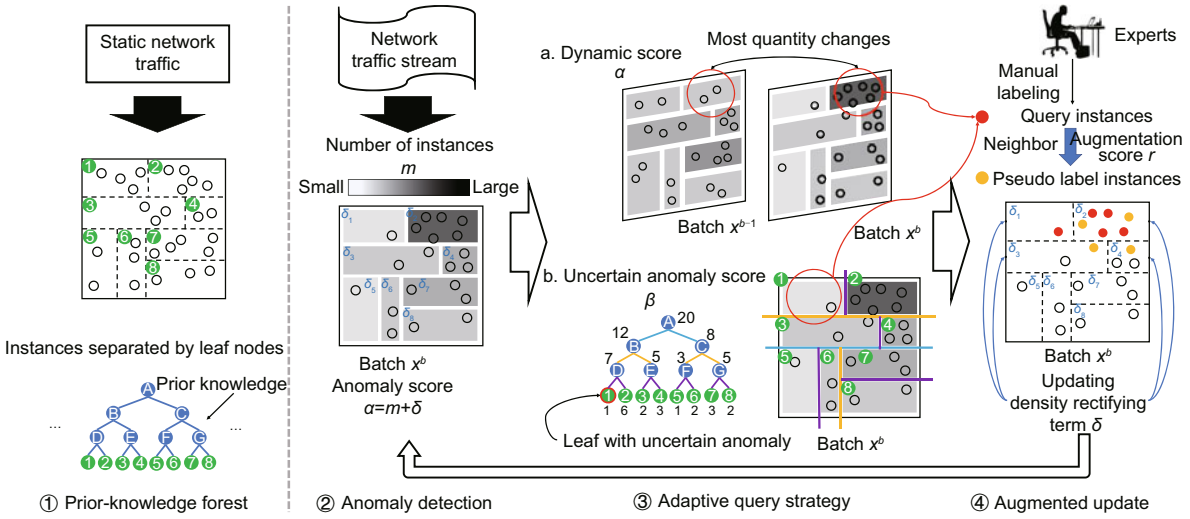

Fig. 1 Overview of A^{3}PF. The initialization stage, constructing prior-knowledge forest based on the static network traffic, is shown on the left of the gray dotted lines, while the testing stage on the network traffic stream is shown on the right. The white arrows show the relationship among anomaly detection, adaptive query strategy, and augmented update. The black arrows show that the network traffic is processed by the model. References to color refer to the online version of this figure

(green nodes) and a two-dimensional toy data set are used as examples. The forest is constructed on a static data set and the data space is divided into eight local regions (represented by each leaf node) according to the data distribution of the static network traffic in Section 3.4. Each black dashed line in Fig. 1 represents a division. In the testing stage, when receiving a new batch of flows $\mathcal{X}^b$ whose size is 20, A^{3}PF performs anomaly detection on each flow instance based on density information m and rectification information δ in each leaf node in Section 3.5. The local density is reflected by the color depth in the figure. Limited flow instances are sampled based on a novel adaptive query strategy by calculating dynamic score α and uncertain anomaly score β in Section 3.6. The dynamic score is measured by the changes in the number of instances between adjacent batches in the same leaf node. The uncertain

anomaly scores are calculated from two aspects: (1) the changes in the number of instances along a tree path, to measure the uncertainty of the model; (2) the number of instances in a leaf node, to measure the anomaly degree. Specifically, the instance in leaf node 1 is regarded as uncertainty anomaly in terms of these two aspects, and we will show the detailed calculation later. Finally, in Section 3.7, A^{3}PF updates the forest model. It augments the supervised information from queried instances (red points) and generates pseudo labels for their neighbors (yellow points) according to the augmentation score γ. All these labeled instances are used to incrementally update the density rectifying term δ in each node (blue lines).

3.4 Prior-knowledge forest

To initialize the model, A^{3}PF constructs a forest including t binary trees based on the static network traffic $\mathcal{D}$ and incorporates prior knowledge of network attacks. We define the tree as follows:

Definition 1 A prior-knowledge tree on a static data set $\mathcal{D}$ is generated as follows:

1. Choose a feature according to the probability assigned by the prior knowledge weights $\boldsymbol{w}$.

2. Let j be the index of the selected feature among the feature set. Calculate the mean value $v(j)=\frac{1}{|\mathcal{D}|}\sum_{i=0}^{|\mathcal{D}|-1}x_i(j)$ as the dividing value for all instances $\boldsymbol{x}_i$ in $\mathcal{D}$ on the selected feature.

3. Let $\mathcal{D}_1=\{\boldsymbol{x}_i|\boldsymbol{x}_i\in\mathcal{D},x_i(j)<v(j)\}$ and $\mathcal{D}_2=\mathcal{D}\backslash\mathcal{D}_1$. Then recurse on $\mathcal{D}_1$ and $\mathcal{D}_2$ until reaching the height limit of the tree.

The tree construction division process is similar to that in the well-known anomaly detection method iForest (Liu FT et al., 2008). However, we adopt dividing features and values based on prior knowledge and mean values, while iForest adopts a random strategy in the dividing feature and values. On one hand, the attack-related features have been assigned higher weights than other features. The prior knowledge weights are adopted as the probability of selecting features to construct the tree. The higher the weights, the higher the probability that the corresponding features are selected to construct the tree. Thus, it is more likely to generate a feature space in which attacks deviate more from normal traffic, which contributes to distinguishing anomalies from normal traffic. On the other hand, the advantage of the mean value dividing strategy is that it could separate these flow instances into different regions with their distribution information as soon as possible. In contrast, the random strategy may incur many meaningless divisions.

The forest includes t trees to maintain diversity. Each tree is different from other trees because the dividing features are selected with a probability based on the prior knowledge weights. Each tree consists of 2^{e-1} leaf nodes and the data space is divided into several local regions represented by each leaf node, where e is the tree depth. Instances of each batch traversing the tree path from the root to a leaf node are shown. During the traversing process, each child node has fewer instances than its parent node. In our paper, the leaf node in the k^{th} tree, in which the instance $\boldsymbol{x}_i$ of the batch $\mathcal{X}^b$ is located, is denoted as $\mathcal{L}_i^{k,b}$. To simplify, we use $\mathcal{L}_i^k$ to take the place of $\mathcal{L}_i^{k,b}$ to represent the leaf node in the latest batch $\mathcal{X}^b$ if no additional instruction is shown.

3.5 Anomaly detection

Anomaly detection in A^{3}PF assumes that normal network traffic shows a dense distribution with more neighbors, whereas anomalies are distributed sparsely with fewer neighbors in the local view. This assumption is commonly used in traditional anomaly detection methods, such as the randomized subspace hashing algorithm (RS-Hash) (Sathe and Aggarwal, 2016) and local outlier factor (LOF) (Breunig et al., 2000). We take the number of instances in leaf node l (denoted as m_l) to represent its local density. Thus, the number of instances in the leaf node in which $\boldsymbol{x}_i$ is located in the k^{th} tree could be denoted as $m_{\mathcal{L}_i^k}$. Similarly, the anomaly score of leaf node l of the k^{th} tree could be denoted as $a_{\mathcal{L}_i^k}$. In Eq. (2), the anomalous instance (lower density) is assigned a higher anomalous score a. We define the anomaly score of instance $\boldsymbol{x}_i$ (i.e., $a(\boldsymbol{x}_i)$) as the average of anomaly scores from each tree in the forest. Specifically, $\delta_{\mathcal{L}_i^k}$ is the density rectifying term of this leaf node, which will be introduced in Section 3.7.

$$a(\boldsymbol{x}_i)=\frac{1}{t}\sum_{k=0}^{t-1}a_{\mathcal{L}_i^k}=\frac{1}{t}\sum_{k=0}^{t-1}-(m_{\mathcal{L}_i^k}+\delta_{\mathcal{L}_i^k}). \tag{2}$$

3.6 Adaptive query strategy

To sample instances that contribute most to the detection model on network traffic streams, an adaptive query strategy to sample the informative

instances is proposed. It is considered from two aspects for each instance $\boldsymbol{x}_i$, i.e., the dynamic score α to capture the changes in data distribution and the uncertain anomaly score β to capture the uncertainty degree and anomaly degree. On one hand, it is more conducive to query the labels of instances whose distribution has changed, which makes the model learn the new distribution and adapt to the changes. On the other hand, querying the labels of the most uncertain anomalies could maximize the limited supervised information in labeled instances to distinguish anomalies from normal traffic. The adaptive query strategy samples instances for manual labels according to the query score $q(\boldsymbol{x}_i)$, as Eq. (3) shows. In each batch, $\lfloor |\mathcal{X}^b| \cdot \theta \rfloor$ instances with the largest query scores are sampled and delivered to domain experts for manual labels.

$$q(\boldsymbol{x}_i) = \alpha(\boldsymbol{x}_i) + \beta(\boldsymbol{x}_i). \tag{3}$$

3.6.1 Dynamic scores

We take the changes in the number of instances in the same leaf node between adjacent batches to measure the changes in data distribution, as shown in Fig. 1. During the dynamic score calculation, we find that the number of instances in the red region has changed the most between two batches. Thus, the dynamic score of this region is higher than those of other regions. We measure the change ratio of the number of instances in the same leaf node between batches as the dynamic score of $\boldsymbol{x}_i$ in Eq. (4):

$$\alpha^k(\boldsymbol{x}_i) = \left| \widetilde{m}_{\mathcal{L}_i^{k,b}} - \widetilde{m}_{\mathcal{L}_i^{k,b-1}} \right| \Big/ \widetilde{m}_{\mathcal{L}_i^{k,b-1}}, \tag{4}$$

where $\widetilde{m}_{\mathcal{L}_i}^{k,b} = \left(s + m_{\mathcal{L}_i^{k,b}}\right) \Big/ \left(s \cdot 2^{e-1} + |\mathcal{X}^b|\right)$. We adopt Laplacian smoothing (Field, 1988), a technique for parameter estimation that accounts for unobserved events in case none of the instances exists in the leaf node, and the denominator becomes 0. Because there are only two classes, i.e., normal class and anomaly class, the Laplacian smoothing parameter s is 0.5.

By averaging the dynamic scores of the leaf node in which $\boldsymbol{x}_i$ is located in each tree, the final dynamic score $\alpha(\boldsymbol{x}_i)$ can be calculated, as Eq. (5) shows. The higher $\alpha(\boldsymbol{x}_i)$ is, the higher the probability that $\boldsymbol{x}_i$ belongs to the new data distribution.

$$\alpha(\boldsymbol{x}_i) = \frac{1}{t} \sum_{k=0}^{t-1} \alpha^k(\boldsymbol{x}_i). \tag{5}$$

3.6.2 Uncertain anomaly scores

To effectively distinguish the limited anomalies from the normal traffic, the uncertain anomaly score measures the informativeness from two aspects: uncertainty degree and anomaly degree. On one hand, in terms of the uncertainty degree, we measure the changes in the number of instances along a tree path that starts from the root and ends in a leaf node, including all branch nodes and a leaf node. The size of the data space and the number of instances in each node decrease with increase in the depth of the tree. We take the separability of instances along the tree path as a measure of uncertainty for model classification. More precisely, the later the number of instances in nodes on the path changes significantly, the harder it is to separate the instances in the leaf node on that path. On the other hand, in terms of the anomaly degree, because limited anomalies exist in network traffic streams, we prioritize selecting the instances with a higher anomaly degree for manual labels. Consequently, we average the number of instances along a tree path as the uncertainty and count the number of instances in a leaf node to measure the anomaly degree. Specifically, let $\mathcal{P}_{\mathcal{L}_i^k}$ denote the set of all nodes along the tree path, from the root node to the leaf node in which $\boldsymbol{x}_i$ is located in the k^{th} tree. Let $o \in \mathcal{P}_{\mathcal{L}_i^k}$ represent each node along the tree path. Eq. (6) shows the calculation of uncertain anomaly scores of the corresponding leaf node:

$$\beta^k(\boldsymbol{x}_i) = \frac{\sum_{o \in \mathcal{P}_{\mathcal{L}_i^k}} m_o}{em_{\mathcal{L}_i^k}}. \tag{6}$$

To clearly show the calculation of the uncertain anomaly score for each leaf node, we introduce the calculation process of the example in Fig. 1. The number of instances in each node of the tree structure is marked in the figure. The uncertain anomaly scores for eight leaf nodes are calculated as

$$\beta^k(\boldsymbol{x}_i) = \begin{cases} (20+12+7+1)/4 = 10, & \text{leaf node 1,} \\ (20+12+7+6)/24 = 1.88, & \text{leaf node 2,} \\ (20+12+5+2)/8 = 4.88, & \text{leaf node 3,} \\ (20+12+5+3)/12 = 3.33, & \text{leaf node 4,} \\ (20+8+3+1)/4 = 8, & \text{leaf node 5,} \\ (20+8+3+2)/8 = 4.13, & \text{leaf node 6,} \\ (20+8+5+3)/12 = 3, & \text{leaf node 7,} \\ (20+8+5+2)/8 = 4.38, & \text{leaf node 8.} \end{cases} \tag{7}$$

Because $\beta^k(\boldsymbol{x}_i)$ of leaf node 1 is the largest among these cases, the instances that traverse the path along A-B-D-1 are most likely the uncertain anomalies. As shown in the figure, the number of instances along the tree path A-B-D-1 significantly changes later than that on the tree path A-C-F-5, although leaf nodes 1 and 5 show the same anomaly degree in density. Thus, the instances in leaf node 1 behave with more uncertainty for the tree model to separate them from other instances. We average $\beta^k(\boldsymbol{x}_i)$ in all trees to obtain the uncertain anomaly scores (i.e., $\beta(\boldsymbol{x}_i)$) as

$$\beta(\boldsymbol{x}_i)=\frac{1}{t}\sum_{k=0}^{t-1}\beta^k(\boldsymbol{x}_i). \tag{8}$$

The higher $\beta(\boldsymbol{x}_i)$ is, the more likely the instance $\boldsymbol{x}_i$ is an uncertain anomaly.

3.7 Augmented update

To leverage the unlabeled instances in model updating, A^{3}PF augments the supervised information of query instances by generating pseudo labels for the unlabeled neighbor instances. The detection model is updated with these query instances and pseudo label instances in the latest batch. The specific details are shown as follows.

3.7.1 Augmented score

The augmented update is based on the assumption that it is more likely that neighbor instances belong to the same category. Thus, we augment the supervised information from these queried instances to their neighbors in the feature space. The instances in the same leaf region are considered as neighbors. For the leaf node in which the query instances are located, if there are more query instances labeled as normal than query instances labeled as anomalies in the leaf node, this leaf node is denoted as $\mathcal{L}_{\mathrm{n}}^k$. Otherwise, the leaf node is denoted as $\mathcal{L}_{\mathrm{a}}^k$. In addition, we use $\gamma^k(\boldsymbol{x}_i)$ to show the augmentation score of unlabeled instance $\boldsymbol{x}_i$ in the k^{th} tree, as shown in Eq. (9). Let u_{a} and u_{n} be the numbers of query instances labeled as anomalies and normal ones in this batch, respectively; $u_{\mathrm{a}}+u_{\mathrm{n}}=\lfloor|\mathcal{X}^b|\cdot\theta\rfloor$.

$$\gamma^k(\boldsymbol{x}_i)=\begin{cases}1/u_{\mathrm{a}}, & \mathcal{L}_i^k\in\mathcal{L}_{\mathrm{a}}^k,\\ -1/u_{\mathrm{n}}, & \mathcal{L}_i^k\in\mathcal{L}_{\mathrm{n}}^k,\\ 0, & \text{otherwise.}\end{cases} \tag{9}$$

We average $\gamma^k(\boldsymbol{x}_i)$ in all trees to obtain the augmentation score $\gamma(\boldsymbol{x}_i)$ for each unlabeled instance $\boldsymbol{x}_i$, as shown in Eq. (10):

$$\gamma(\boldsymbol{x}_i)=\frac{1}{t}\sum_{k=0}^{t-1}\gamma^k(\boldsymbol{x}_i). \tag{10}$$

The higher $\gamma(\boldsymbol{x}_i)$ is, the more confident we are that $\boldsymbol{x}_i$ could be assigned a pseudo label of the anomaly class. Conversely, the lower $\gamma(\boldsymbol{x}_i)$ is, the more confident we are that $\boldsymbol{x}_i$ could be assigned a pseudo label of the normal class.

To determine the number of pseudo labels of the normal class and anomaly class, the list of non-query instances is ranked in descending order of the augmentation scores. We select the top $\left\lfloor\frac{r\cdot|\mathcal{X}^b|}{2}\frac{u_{\mathrm{a}}}{u_{\mathrm{a}}+u_{\mathrm{n}}}\right\rfloor$ instances to generate anomaly pseudo labels and bottom $\left\lfloor\frac{r\cdot|\mathcal{X}^b|}{2}\frac{u_{\mathrm{n}}}{u_{\mathrm{a}}+u_{\mathrm{n}}}\right\rfloor$ instances to generate normal pseudo labels, where $r\in[0,1-\theta]$ is the pseudo label ratio set in advance.

3.7.2 Model update

Considering the bias in unsupervised anomaly detection based on density, the model needs to update itself on network traffic streams by incorporating the supervised information from these labeled instances. We introduce a rectifying term $\delta_l^{k,b}$ of batch $\mathcal{X}^b$ for each leaf node (initialized to 0 at the beginning of the detection), which rectifies the local density of leaf node l in the k^{th} tree by incorporating the latest supervised information. The density rectifying term is incrementally calculated by accumulating the rectifying degree of a set of instances with manual and pseudo labels from the same leaf node in the same batch (denoted as $\mathcal{Z}_l^{k,b}$). We show the definition of the density rectifying term in Eq. (11):

$$\begin{cases}\delta_l^{k,b}=\displaystyle\sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{Z}_l^{k,b}}(|\mathcal{X}^b|^{1-y_i}\cdot 1^{y_i}-\eta_l^k)\\ \qquad\qquad\cdot\left|y_i-\frac{|\mathcal{X}^b|-\eta_l^k}{|\mathcal{X}^b|-1}\right|,\\ \eta_l^k=\max(\min(m_l^{k,b}+\delta_l^{k,b-1},|\mathcal{X}^b|),1).\end{cases} \tag{11}$$

Specifically, $\delta_l^{k,b-1}$ is the rectifying term of this leaf node in the last batch $\mathcal{X}^{b-1}$. We denote the update degree and update ratio of each leaf region as $|\mathcal{X}^b|^{1-y_i}\cdot 1^{y_i}-\eta_l^k$ and $\left|y_i-\frac{|\mathcal{X}^b|-\eta_l^k}{|\mathcal{X}^b|-1}\right|$, respectively. Let η be a constraint of the update degree, which makes the value range in $[1,|\mathcal{X}^b|]$. As shown in Eqs. (2) and (11), for labeled anomaly instances

(y_i=1), A^{3}PF tries to rectify the density of the leaf node with $\delta_l^{k,b}<0$ to increase the anomaly score in the leaf node, while it tries to rectify the density with $\delta_l^{k,b}>0$ to decrease the anomaly score in the leaf node for normal instances (y_i=0).

4 Experiments

4.1 Benchmarks

Two publicly available network traffic benchmarks, CIC-IDS2017 and UNSW-NB15, were used in our experiments. CIC-IDS2017 spanned five days (from Monday to Friday) of normal and attack traffic data of the Canadian Institute of Cybersecurity. Each file recorded network traffic information for a while and could be regarded as a network traffic stream. These files included the network traffic analysis results obtained using CICFlowMeter with the labeled flow to generate 80 features with the class label. Specifically, because the first day included only normal traffic, we used network traffic data of the latter four days (Tuesday, Wednesday, Thursday, and Friday) from CIC-IDS2017 in our experiments. UNSW-NB15 was created by the IXIA PerfectStorm tool for generating a hybrid of modern normal activities and synthetic contemporary attack behaviors. Argus and Bro-IDS techniques and 12 procedures were executed in parallel to generate 45 features with the class label, including nine attack classes and one normal class. A total of 257 673 records were stored in the training set and testing set as CSV files. We split the network traffic into several setups to test the performance of A^{3}PF in adapting to the dynamic network stream. The normal network traffic was divided into two types according to the service. Different normal services represent different distributions of normal network traffic. We combined two kinds of normal network traffic with eight attack classes, excluding worms (only 137 attacks), into nine traffic chunks to construct the dynamic network traffic. The first chunk contained 10 000 normal flow instances from a random type of normal class. After the first chunk, each chunk contained 10 000 network flow instances, including a new attack class and a random type of normal network traffic. These data sets simulated dynamic network traffic streams, such as the concept drift of known classes and the emergence of novel attack classes. To compare the performance of methods under different anomaly ratios, we formed four setups where the anomaly ratio was specified as 0.5%, 1%, 2%, and 4%. They are denoted as $\text{UNSW}_{0.5}$, UNSW_1, UNSW_2, and UNSW_4, respectively. Furthermore, the categorical features in these two benchmarks were not used in A^{3}PF, and will be analyzed in our future work.

4.2 Comparative methods

A^{3}PF was compared with state-of-the-art methods, including:

Active anomaly detection on network traffic streams: AI^2 (Veeramachaneni et al., 2016) and BigFlow (Viegas et al., 2019);

Active anomaly detection on general data streams: AADS (Das et al., 2019);

Active classification on general data streams: OALE (Shan et al., 2019);

Online unsupervised network anomaly detection: Kitsune (Mirsky et al., 2018), which was taken as the baseline in our experiments.

The former four methods were introduced in Section 2. Kitsune is an unsupervised method for detecting anomalies efficiently, adopting an ensemble of neural networks to differentiate between normal and anomalous traffic patterns collectively.

4.3 Parameters and settings

For A^{3}PF, the number of trees t was 50, and the tree depth e was 8. We incorporated the results in Sharafaldin et al. (2018) and Moustafa and Slay (2015a) for CIC-IDS2017 and UNSW-NB15 as the prior knowledge about features, respectively. For AI^2, we used PyOD (Zhao et al., 2019) to implement the autoencoder, principal component analysis, and the copula model. Random forest was implemented by scikit-learn (Pedregosa et al., 2011). The number of epochs and batch size in the autoencoder were 20 and 128, respectively. The number of trees in the random forest was 50. For BigFlow and OALE, the Hoeffding tree was adopted as the base model implemented in scikit-multiflow (Montiel et al., 2018). The number of dynamic classifiers in OALE was 50. Additionally, we used the original implementations for AADS (https://github.com/shubhomoydas/ad_examples) and Kitsune (https://github.com/ymirsky/Kitsune-py) by their authors. For AADS, the number of

trees was 50. For Kitsune, the maximum size for any autoencoder in the ensemble layer was 10. Other parameters were set as default. All methods were initialized on the first 5000 flow instances, and the batch size $|\mathcal{X}^b|$ was set at 1000. All data sets were preprocessed by min-max standardization, and each experiment was run 10 times to obtain the average results.

4.4 Metrics

Two complementary performance metrics were adopted, the area under receiver operating characteristic curve (AUC-ROC) and the area under precision-recall curve (AUC-PR), to show the performance of anomaly detection. AUC-ROC summarizes the ROC curve of true positives against false positives, whereas AUC-PR summarizes the curve of precision against recalls. AUC-PR is more suitable than AUC-ROC in many anomaly detection applications that require excellent performance on the positive class and do not care much about the performance on the negative class. It is because AUC-ROC is affected by performance on both anomaly and normal classes, and the performance on the normal class can bias AUC-ROC due to the class-imbalance nature of anomaly detection data. By contrast, AUC-PR evaluates how many positive predictions are correct (precision) and how many of the positive predictions that are truly positive compose the positive class (recall).

4.5 Effectiveness on CIC-IDS2017

Here we examined the performance of A^{3}PF in real-world data sets with respect to (w.r.t.) different query ratios compared with state-of-the-art methods. By default, the prior-knowledge weights of attack-related features (denoted as w^*) were 7.5, while w^*=1 for other features. The ratio of instances with pseudo labels r was 25%. The list of query ratios θ's was {0.1%, 0.5%, 1%, 1.5%, 2%} in five setups.

The detailed and average changes w.r.t. query ratios are plotted in the first and second rows of Fig. 2, in terms of AUC-ROC and AUC-PR, respectively. Most active methods showed an increasing trend with the query ratio growing, whereas AADS and OALE exhibited unstable performance. The AUC-ROC and AUC-PR of Kitsune did not increase with an increase in the query ratio because it is an unsupervised detection method. A^{3}PF achieved a comparable result on each date for CIC-IDS2017 and obtained substantially better average improvements than AI^2 (15.6%, 42.5%), BigFlow (8.2%, 17.8%), AADS (48.3%, 62.7%), OALE (13.4%, 21.4%), and Kitsune (19.1%, 78.4%), in terms of (AUC-ROC, AUC-PR). This demonstrates the stability of A^{3}PF when faced with different network attacks.

4.6 Effectiveness on UNSW-NB15

In this subsection we examined the performance of A^{3}PF in adapting to the dynamic network traffic

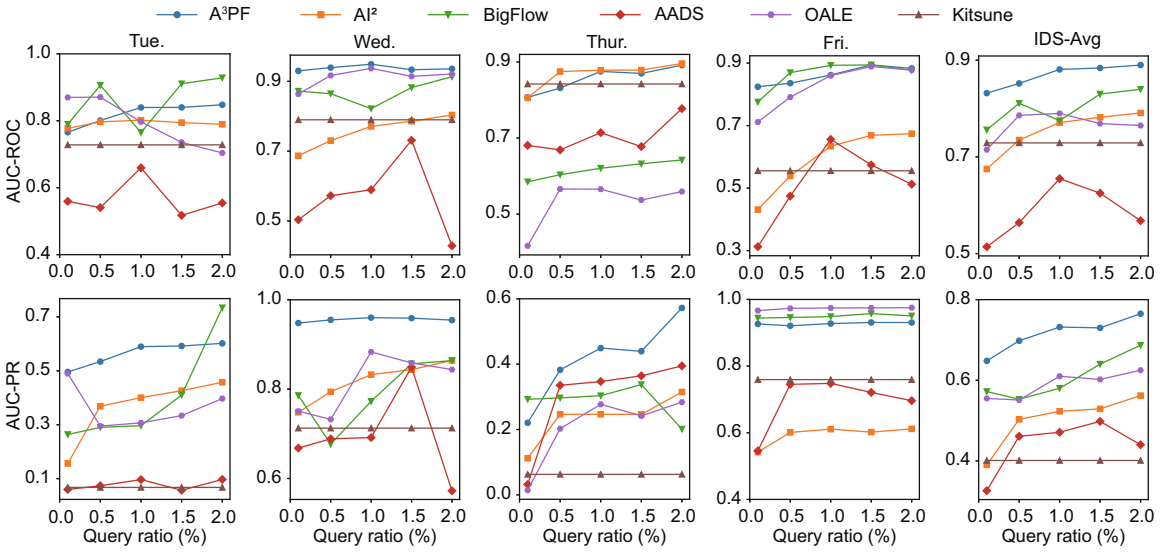

Fig. 2 AUC-ROC and AUC-PR w.r.t. query ratios and methods on CIC-IDS2017

stream. To show more obvious changes of the results, the query ratio θ in UNSW-NB15, different from that in CIC-IDS2017, was {1%, 2.5%, 5%, 7.5%, 10%} in five setups. As a default, the prior-knowledge weights of attack-related features w^* were 7.5, while w^*=1 for other features. The ratio of data with pseudo labels r was 25%.

Fig. 3 shows similar results to Fig. 2. The performances of these active methods were generally improved with an increase of the query ratio and anomaly ratio. Higher anomaly ratios and more query instances led these methods to select the anomalous instance with a higher probability and better learned their distributions. However, when the anomaly ratio was low, these compared methods achieved undesirable performance in learning the anomaly distribution. AADS and OALE showed unstable performance, even poorer than that of the unsupervised method Kitsune when the anomaly ratio was low. Compared to these methods, A^{3}PF performed the best on four UNSW-NB15 setups with different query ratios in terms of the respective AUC-ROC and AUC-PR, and its performance was comparable to that of the best performer, where it ranked second. On average, A^{3}PF obtained substantially better average improvements than AI^2 (23.5%, 28.5%), BigFlow (12.6%, 144.9%), AADS (32.2%, 16.0%), Kitsune (22.7%, 318.7%), and OALE (17.8%, 67.1%), in terms of (AUC-ROC, AUC-PR).

4.7 Parameter sensitivity analysis

In this subsection we examined the effectiveness of two key parameters in A^{3}PF, including the prior-knowledge weights of attack-related features w^* and the ratio of instances with pseudo labels r.

4.7.1 Prior-knowledge weights of attack-related features

We plotted the changes of detection performance with different weights of attack-related features in the two benchmarks in Fig. 4. The larger w^* was, the higher the probability that these attack-related features were selected to construct each tree. The query ratio was fixed as θ=1% for CIC-IDS2017 and θ=5% for UNSW-NB15. The ratio of pseudo label instances r was 25%. The list of weights of attack-related features was {1, 2.5, 5, 7.5, 10} in five setups.

In CIC-IDS2017, when w^*=7.5 (0.881, 0.655), A^{3}PF showed better average performance than A^{3}PF with w^*=1 (0.838, 0.625), w^*=2.5 (0.852, 0.643), w^*=5 (0.848, 0.634), and w^*=10 (0.837, 0.647) in terms of (AUC-ROC, AUC-PR). According to the experimental results, a lower weight (w^*=1) and a higher weight (w^*=10) both reduced performance. It is because a proper weight, such as w^*=7.5, focuses on selecting these attack-related features and leveraging the randomness of feature selection to improve the robustness of the model. Similar results can be seen from the changes in UNSW-NB15. On average, when the weight of attack-related features was w^*=7.5 (0.880, 0.363), A^{3}PF achieved better performance than w^*=1 (0.852, 0.288), w^*=2.5 (0.878, 0.363), w^*=5 (0.876, 0.365), and w^*=10 (0.868, 0.353).

4.7.2 Ratios of pseudo labels

We measured the change of detection performance w.r.t. the ratio of pseudo labels r in A^{3}PF. The query ratio was fixed as θ=1% for CIC-IDS2017 and θ=5% for UNSW-NB15. The weights of attack-related features were set as w^*=7.5. The lists of r were {0, 25%, 50%, 75%, 99%} and {0, 25%, 50%, 75%, 95%} for CIC-IDS2017 and UNSW-NB15, respectively, where 99% and 95% were the maximum proportions of non-query data accounting for all data in these two benchmarks due to different θ's.

As shown in Fig. 5, the changes showed first an increasing trend and then a decreasing trend, and especially when r was 25% the detection model could achieve the best performance. Specifically, in CIC-IDS2017, the performance when r=25% (0.881, 0.678) was better than those when r=0 (0.832, 0.637), r=50% (0.832, 0.604), and r=99% (0.799, 0.576) in terms of (AUC-ROC, AUC-PR). In UNSW-NB15, the performance when r=25% (0.880, 0.363) was better than those when r=0 (0.817, 0.347), r=50% (0.862, 0.299), and r=95% (0.852, 0.271) in terms of (AUC-ROC, AUC-PR). When $r \geq 25\%$, the detection performance decreased with the growth of r. This is because more pseudo labels are incorrect when increasing the pseudo label ratio r. These results demonstrated that a proper r to augment the limited supervised information is beneficial to improving the performance of A^{3}PF.

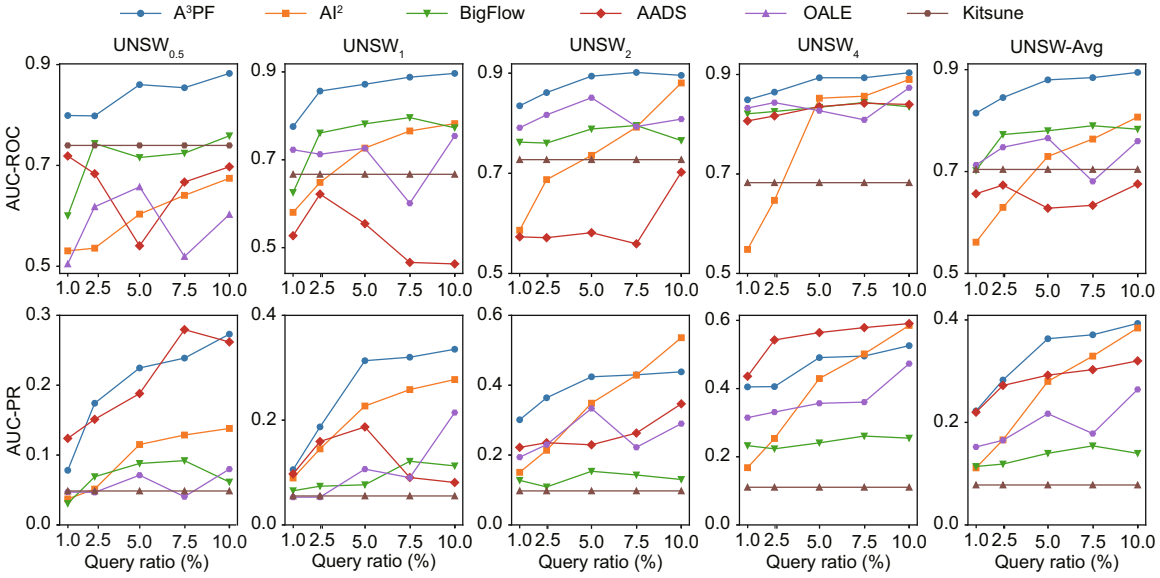

Fig. 3 AUC-ROC and AUC-PR w.r.t. query ratios and methods on UNSW-NB15

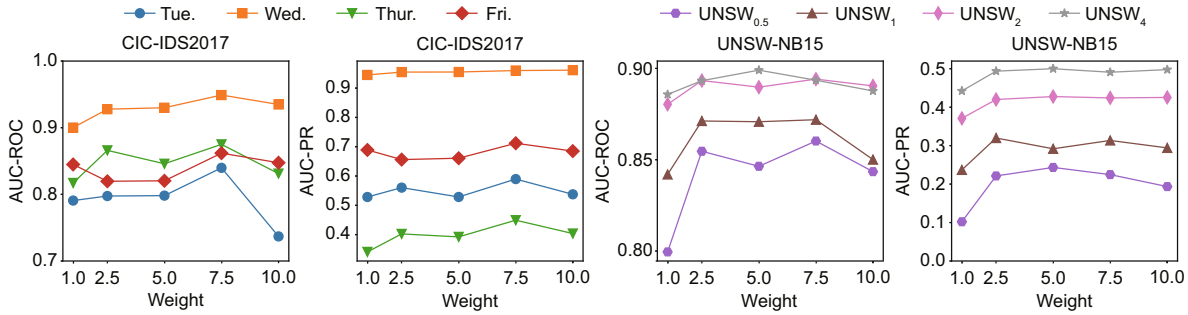

Fig. 4 AUC-ROC and AUC-PR w.r.t. weights of attack-related features on CIC-IDS2017 and UNSW-NB15

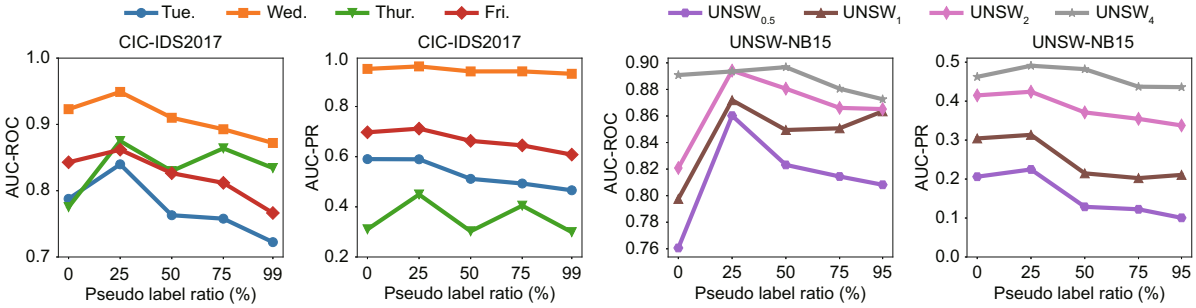

Fig. 5 AUC-ROC and AUC-PR w.r.t. pseudo label ratios on CIC-IDS2017 and UNSW-NB15

4.8 Ablation study

In this subsection we examined the importance of each key component in A^{3}PF by comparing A^{3}PF with several variants. Specifically, we focused on the following three key questions:

1. How does the incorporated prior knowledge affect A^{3}PF?

2. How does the augmentation affect A^{3}PF?

3. How does the adaptive query strategy affect A^{3}PF?

To answer these questions, we proposed several variants of A^{3}PF:

1. To answer question 1, the variant A^{3}PF-w/o-Prior Knowledge was leveraged to denote a variant

of A^{3}PF without prior knowledge (the weights of attack-related feature w^*=1).

2. To answer question 2, the variant A^{3}PF-w/o-Pseudo Label was proposed to update the model only based on the query instances without generating pseudo labels (the pseudo label ratio r=0).

3. To answer question 3, we proposed four variants with different query strategies: The variant A^{3}PF-q-Anomaly Score queries instances based on the anomaly strategy rather than the adaptive query strategy. The variant A^{3}PF-q-Inconsistency Score adopts the query strategy of query-by-committee to exploit the inconsistency of an ensemble of the committee member. It adopts the variance of the anomaly score in each tree as the inconsistency. Instances with higher variance will be sampled as the query instances. Two variants A^{3}PF-q-Dynamic Score and A^{3}PF-q-Uncertainty Anomaly Score were proposed to query instances based only on the dynamic scores and the uncertainty anomaly scores, respectively.

In our experiments, the query ratio was fixed as θ=1% for CIC-IDS2017 and θ=5% for UNSW-NB15. The weights of attack-related features and the pseudo label ratio were set at w^*=7.5 and r=25%, respectively, excluding w^*=1 for A^{3}PF-w/o-Prior Knowledge and r=0 for A^{3}PF-w/o-Pseudo Label according to questions 1 and 2, respectively. The average and specific results on CIC-IDS2017 and UNSW-NB15 are shown in Tables 3 and 4, respectively.

As we can see from Tables 3 and 4, A^{3}PF performed the best in most data sets in term of both AUC-ROC and AUC-PR, and its performance was comparable to the best in the rest. It indicated a significant role for prior knowledge of network attacks, adaptive query strategy, and augmented update in A^{3}PF. They could help the model achieve good performance with limited manual labels on network traffic streams. Specifically, A^{3}PF-q-Dynamic Score and A^{3}PF-q-Uncertainty Anomaly Score both had better performance than A^{3}PF-q-Anomaly Score and A^{3}PF-q-Inconsistency Score, which demonstrates the effectiveness of querying dynamic instances and querying uncertain anomalies, respectively.

5 Conclusions

We propose an adaptive and augmented active prior-knowledge forest (A^{3}PF) to detect network anomalies on network traffic streams. A^{3}PF builds the forest model with prior knowledge of network attacks to detect anomalies based on local density.

Table 3 Ablation study on CIC-IDS2017

Method	AUC-ROC					AUC-PR				
	Tue.	Wed.	Thur.	Fri.	Avg	Tue.	Wed.	Thur.	Fri.	Avg
A^{3}PF	0.840	**0.949**	**0.876**	**0.862**	**0.882**	0.590	**0.960**	**0.450**	**0.712**	**0.678**
A^{3}PF-w/o-Prior Knowledge	0.790	0.900	0.817	0.844	0.838	0.528	0.945	0.341	0.689	0.626
A^{3}PF-w/o-Pseudo Label	0.787	0.920	0.775	0.843	0.831	**0.591**	0.950	0.310	0.697	0.637
A^{3}PF-q-Anomaly Score	0.698	0.779	0.770	0.758	0.751	0.457	0.848	0.384	0.583	0.568
A^{3}PF-q-Inconsistency Score	0.703	0.752	0.695	0.743	0.723	0.404	0.772	0.207	0.607	0.497
A^{3}PF-q-Dynamic Score	**0.855**	0.932	0.872	0.852	0.878	0.586	0.938	0.448	0.678	0.662
A^{3}PF-q-Uncertainty Anomaly Score	0.785	0.902	0.842	0.833	0.841	0.562	0.930	0.441	0.698	0.658

The best and second best performances are boldfaced and underlined, respectively. w/o: without; q: query

Table 4 Ablation study on UNSW-NB15

Method	AUC-ROC					AUC-PR				
	UNSW$_{0.5}$	UNSW$_1$	UNSW$_2$	UNSW$_4$	Avg	UNSW$_{0.5}$	UNSW$_1$	UNSW$_2$	UNSW$_4$	Avg
A^{3}PF	**0.860**	**0.872**	**0.894**	0.894	**0.880**	**0.225**	**0.314**	**0.424**	0.491	**0.363**
A^{3}PF-w/o-Prior Knowledge	0.799	0.841	0.880	0.885	0.851	0.102	0.236	0.370	0.442	0.287
A^{3}PF-w/o-Pseudo Label	0.760	0.797	0.820	0.890	0.816	0.206	0.304	0.414	0.462	0.346
A^{3}PF-q-Anomaly Score	0.692	0.755	0.727	0.649	0.705	0.118	0.207	0.260	0.256	0.210
A^{3}PF-q-Inconsistency Score	0.733	0.762	0.710	0.718	0.731	0.155	0.182	0.239	0.289	0.216
A^{3}PF-q-Dynamic Score	0.816	0.847	0.874	**0.899**	0.859	0.135	0.226	0.369	**0.521**	0.313
A^{3}PF-q-Uncertainty Anomaly Score	0.846	0.865	0.871	0.887	0.867	0.164	0.264	0.379	0.456	0.316

The best and second best performances are boldfaced and underlined, respectively. w/o: without; q: query

To sample the most informative instances of network traffic streams for manual labels, an adaptive query strategy is proposed to annotate limited instances, which makes the detection model adapt to the dynamic streams. To leverage the unlabeled instances in model updating, pseudo labels are generated for unlabeled instances and together with these query instances, incrementally updating the detection model. Experimental results have demonstrated that the proposed approach improves the detection performance compared with related methods.

Our future work is introduced as follows: First, we will combine the categorical features with these existing numerical features to detect network anomalies. Second, we will improve the robustness of our method to adversarial attacks that could manipulate the actual network traffic with knowledge of features used in machine learning based classifiers. Finally, the prior knowledge leveraged in A^{3}PF may lead the anomaly detection model towards known attacks, which results in inferior performance in detecting new types of attacks that are not based on the known attack features. Thus, how to use prior knowledge to improve detection performance without affecting the detection of new types of attacks is worthy of further study.

Contributors

Bin LI designed the research, processed the data, and drafted the paper. Yijie WANG and Li CHENG helped organize the paper. Bin LI, Yijie WANG, and Li CHENG revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Apruzzese G, Laskov P, Tastemirova A, 2022. SoK: the impact of unlabelled data in cyberthreat detection. IEEE 7th European Symp on Security and Privacy, p.20-42. https://doi.org/10.1109/EuroSP53844.2022.00010

Beaugnon A, Chifflier P, Bach F, 2017. ILAB: an interactive labelling strategy for intrusion detection. 20th Int Symp on Research in Attacks, Intrusions, and Defenses, p.120-140. https://doi.org/10.1007/978-3-319-66332-6_6

Bilge L, Dumitras T, 2012. Before we knew it: an empirical study of zero-day attacks in the real world. Proc ACM Conf on Computer and Communications Security, p.833-844. https://doi.org/10.1145/2382196.2382284

Breunig MM, Kriegel HP, Ng RT, et al., 2000. LOF: identifying density-based local outliers. Proc ACM SIGMOD Int Conf on Management of Data, p.93-104. https://doi.org/10.1145/342009.335388

Das S, Islam MR, Jayakodi NK, et al., 2019. Active anomaly detection via ensembles: insights, algorithms, and interpretability. https://arxiv.org/abs/1901.08930

Das S, Wong WK, Dietterich T, et al., 2020. Discovering anomalies by incorporating feedback from an expert. *ACM Trans Knowl Disc Data*, 14(4):1-32. https://doi.org/10.1145/3396608

Dong S, 2021. Multi class SVM algorithm with active learning for network traffic classification. *Expert Syst Appl*, 176:114885. https://doi.org/10.1016/j.eswa.2021.114885

Field DA, 1988. Laplacian smoothing and Delaunay triangulations. *Commun Appl Numer Methods*, 4(6):709-712. https://doi.org/10.1002/cnm.1630040603

Gao Y, Chandra S, Li YF, et al., 2022. SACCOS: a semi-supervised framework for emerging class detection and concept drift adaption over data streams. *IEEE Trans Knowl Data Eng*, 34(3):1416-1426. https://doi.org/10.1109/TKDE.2020.2993193

Guerra-Manzanares A, Bahsi H, 2023. On the application of active learning for efficient and effective IoT botnet detection. *Fut Gener Comput Syst*, 141:40-53. https://doi.org/10.1016/j.future.2022.10.024

Hafeez H, Khalil T, 2023. IP spoofing & its detection techniques for the prevention of DoS attacks. *Recent Prog Sci Technol*, 6:49-57. https://doi.org/10.9734/bpi/rpst/v6/4583C

Hulten G, Spencer L, Domingos P, 2001. Mining time-changing data streams. Proc 7th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining, p.97-106. https://doi.org/10.1145/502512.502529

Kathareios G, Anghel A, Mate A, et al., 2017. Catch it if you can: real-time network anomaly detection with low false alarm rates. 16th IEEE IEEE Int Conf on Machine Learning and Applications, p.924-929. https://doi.org/10.1109/ICMLA.2017.00-36

Korycki Ł, Cano A, Krawczyk B, 2019. Active learning with abstaining classifiers for imbalanced drifting data streams. IEEE Int Conf on Big Data, p.2334-2343. https://doi.org/10.1109/BigData47090.2019.9006453

Li B, Wang YJ, Xu KL, et al., 2022. DFAID: density-aware and feature-deviated active intrusion detection over network traffic streams. *Comput Secur*, 118:102719. https://doi.org/10.1016/j.cose.2022.102719

Liu FT, Ting KM, Zhou ZH, 2008. Isolation forest. Proc 8th IEEE IEEE Int Conf on Data Mining, p.413-422. https://doi.org/10.1109/ICDM.2008.17

Liu TL, Qi Y, Shi L, et al., 2019. Locate-then-detect: real-time web attack detection via attention-based deep neural networks. Proc 28th Int Joint Conf on Artificial Intelligence, p.4725-4731.

Mirsky Y, Doitshman T, Elovici Y, et al., 2018. Kitsune: an ensemble of autoencoders for online network intrusion detection. https://arxiv.org/abs/1802.09089

Montiel J, Read J, Bifet A, et al., 2018. Scikit-multiflow: a multi-output streaming framework. *J Mach Learn Res*, 19(72):1-5.

Moustafa N, Slay J, 2015a. The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems. 4th Int Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, p.25-31. https://doi.org/10.1109/BADGERS.2015.014

Moustafa N, Slay J, 2015b. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). Military Communications and Information Systems Conf, p.1-6. https://doi.org/10.1109/MilCIS.2015.7348942

Pedregosa F, Varoquaux G, Gramfort A, et al., 2011. Scikit-learn: machine learning in Python. *J Mach Learn Res*, 12:2825-2830.

Roshan S, Miche Y, Akusok A, et al., 2018. Adaptive and online network intrusion detection system using clustering and extreme learning machines. *J Frankl Inst*, 355(4):1752-1779. https://doi.org/10.1016/j.jfranklin.2017.06.006

Sathe S, Aggarwal CC, 2016. Subspace outlier detection in linear time with randomized hashing. IEEE 16th Int Conf on Data Mining, p.459-468. https://doi.org/10.1109/ICDM.2016.0057

Shahraki A, Abbasi M, Taherkordi A, et al., 2022. A comparative study on online machine learning techniques for network traffic streams analysis. *Comput Netw*, 207:108836. https://doi.org/10.1016/j.comnet.2022.108836

Shan JC, Zhang H, Liu WK, et al., 2019. Online active learning ensemble framework for drifted data streams. *IEEE Trans Neur Netw Learn Syst*, 30(2):486-498. https://doi.org/10.1109/TNNLS.2018.2844332

Sharafaldin I, Lashkari AH, Ghorbani AA, 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. Proc 4th Int Conf on Information Systems Security and Privacy, p.108-116. https://doi.org/10.5220/0006639801080116

Siddiqui MA, Stokes JW, Seifert C, et al., 2019. Detecting cyber attacks using anomaly detection with explanations and expert feedback. IEEE Int Conf on Acoustics, Speech and Signal Processing, p.2872-2876. https://doi.org/10.1109/ICASSP.2019.8683212

Veeramachaneni K, Arnaldo I, Korrapati V, et al., 2016. AI^2: training a big data machine to defend. IEEE 2nd Int Conf on Big Data Security on Cloud, IEEE Int Conf on High Performance and Smart Computing, and IEEE Int Conf on Intelligent Data and Security, p.49-54. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2016.79

Viegas E, Santin A, Bessani A, et al., 2019. BigFlow: real-time and reliable anomaly-based intrusion detection for high-speed networks. *Fut Gener Comput Syst*, 93:473-485. https://doi.org/10.1016/j.future.2018.09.051

Wang ZY, Wang YJ, Huang ZY, et al., 2021. Entropy and autoencoder-based outlier detection in mixed-type network traffic data. IEEE Int Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking, p.501-508. https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00075

Wu YH, Fang YZ, Shang SK, et al., 2021. A novel framework for detecting social bots with deep neural networks and active learning. *Knowl-Based Syst*, 211:106525. https://doi.org/10.1016/j.knosys.2020.106525

Yan XY, Homaifar A, Sarkar M, et al., 2021. A clustering-based framework for classifying data streams. https://arxiv.org/abs/2106.11823

Zhao Y, Nasrullah Z, Li Z, 2019. PyOD: a Python toolbox for scalable outlier detection. *J Mach Learn Res*, 20:1-7.