Anomaly Detection in the Open World: Normality Shift Detection, Explanation, and Adaptation

Dongqi Han*†, Zhiliang Wang *†‡, Wenqi Chen*†, Kai Wang*†, Rui Yu§, Su Wang†¶, Han Zhang*†‡,
Zhihua Wang‖, Minghui Jin‖, Jiahai Yang*†‡, Xingang Shi*† and Xia Yin†¶
*Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing, China
†Zhongguancun Laboratory, Beijing, China
‡Quan Cheng Laboratory, Jinan, Shandong, China
§Tsinghua Shenzhen International Graduate School, Tsinghua University, Beijing, China
¶Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing, China
‖State Grid Shanghai Municipal Electric Power Company, Shanghai, China
{handq19, chenwq19, k-wang20, yur20, wangsu17}@mails.tsinghua.edu.cn, {wzl, yang, shixg}@cernet.edu.cn

***Abstract*—Concept drift is one of the most frustrating challenges for learning-based security applications built on the close-world assumption of identical distribution between training and deployment. Anomaly detection, one of the most important tasks in security domains, is instead immune to the drift of *abnormal* behavior due to the training without any abnormal data (known as *zero-positive*), which however comes at the cost of more severe impacts when *normality shifts*. However, existing studies mainly focus on concept drift of abnormal behaviour and/or supervised learning, leaving the normality shift for zero-positive anomaly detection largely unexplored.**

In this work, we are the first to explore the normality shift for deep learning-based anomaly detection in security applications, and propose OWAD, a general framework to *detect*, *explain*, and *adapt to* normality shift in practice. In particular, OWAD outperforms prior work by detecting shift in an *unsupervised* fashion, reducing the overhead of manual labeling, and providing better adaptation performance through distribution-level tackling. We demonstrate the effectiveness of OWAD through several realistic experiments on three security-related anomaly detection applications with long-term practical data. Results show that OWAD can provide better adaptation performance of normality shift with less labeling overhead. We provide case studies to analyze the normality shift and provide operational recommendations for security applications. We also conduct an initial real-world deployment on a SCADA security system.

I. Introduction

Anomaly detection is one of the most important tasks in security domains [13], trained with normal data and detecting anomalies that deviates from the distribution of *normality*. Recently, the adoption of Deep Learning (DL) enables anomaly detection to extract more complex features from massive data [12], [76], as well as detect unforeseen threats such as zero-day attacks through learning with only normal data, known as *zero-positive learning* [20]. Heretofore, researchers have applied DL-based anomaly detection for various security applications, such as detecting network intrusions [54], [72], finding threats from system logs [21], [53], tracing advanced persistent threats (APT) [9], [77], which all achieved satisfactory performance.

TABLE I: Comparison of representative related works.

Features	CADE[81]	TRANSCENDENT[37], [5]	UNLEARN[20]	OWAD
Support Timeseries	○	●	●	●
Unsupervised*	○	◐	●	●
Label-efficient†	◐	◐	○	●
Distribution-level‡	○	◐	○	●

(● = true, ◐ = partially true, ○ = false);
∗ TRANSCENDENT *can be used but requires non-trivial adjustments for unsupervised (zero-positive) cases;* † *Measured in* §V-B;
‡ TRANSCENDENT *statistically considers distributional information, but tackles drift in a sample-level fashion (i.e., rejection).*

Unfortunately, the superior performance of learning-based applications is built on the *close-world* assumption of independent and identically distributed (i.i.d.) between training and test samples [68]. Such assumption often does not hold in *open-world* settings due to the divergence of incoming test distribution from the original one, known as *concept drift*. In security domains, concept drift is pervasive as the malicious patterns are switched suddenly and dramatically over time in the hostile environment [4].

In this context, *anomaly detection* is instead immune to the drift of *malicious/abnormal* behavior due to zero-positive learning, which however comes at the price of more severe impact when the distribution of normality shifts. In real-world deployment, the way users interact with systems under monitoring can differ and evolve over time, so do the systems themselves. For example, the involvement of new patches, devices, and protocols all have the potential to shift the normal pattern. Such *normality shift*[1], if not detected and adapted, will induce a large number of false positives (FP) and false negatives (FN), suggested by anecdotal evidence in practice.

In recent years, several studies have been proposed to tackle concept drift for learning-based security applications in the community, which can be divided into two approaches: The first is to periodically retrain the models in the dynamic environment [14], [15], [38], [61], [36], [30], [57], regard-

[1]Normality shift intuitively refers to the change of distribution of normal data (detailed definition is in §II-C). In this paper, we interchangeably use terms "drift" and "shift". We tend to use "normality shift" as a whole term.

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.24830
www.ndss-symposium.org

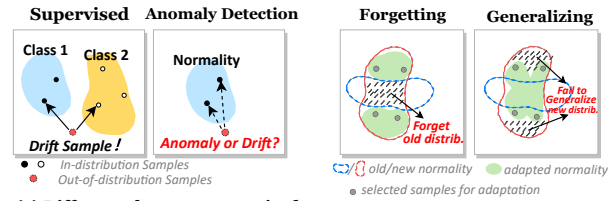
less of *whether*, *when*, and *how* drift occurs. However, this approach is inappropriate in security domains as continuous training is labor-intensive for labeling, as demonstrated in related security applications [78], [52], [81]. It is also hard to determine when to update the model. Late update will expose the model to new threats or overwhelming false positives. Moreover, security practitioners need to obtain evidence and explanation of drifting behind the black-box model [81].

The second solution is to *determine* and then *adapt* to the drift. However, as listed in Table I, most studies explore concept drift under the setting of supervised learning [37], [81], [3], [5], instead of normality shift in zero-positive anomaly detection. Moreover, most of them focus on the detection instead of adaptation, resulting in high labeling overhead when preparing samples used for adaptation. Besides, prior studies [20], [81], [3] prefer a *sample-level* approach by detecting whether a certain sample is out of distribution. Such approach fails to understand the drift in *distribution level*. This leads to poor generalization ability for drift after adaptation, as validated by our experiments in §V-B.

Challenges. In a nutshell, there are three major challenges for tackling normality shift for anomaly detection in security applications: (1) The first is how to detect shift in an *unsupervised* manner (i.e., without any prior knowledge of anomaly)? (2) As shown in Fig. 1a, different from supervised classification, we have to label drifting samples since a normal drift and real anomaly is not distinguishable for anomaly detection. Therefore, the second challenge is how to effectively select samples to reduce labeling overhead? (3) Since only a part of samples are labeled and used to update the model, the last challenge, as illustrated in Fig. 1b, is how to prevent the adapted model from forgetting the valid knowledge in the old distribution while can generalize to the new distribution.

Our Work. In light of the above challenges, we develop OWAD, a general distribution-level framework to tackle normality shift for security-related anomaly detection in practice, which consists of shift detection, explanation, and adaptation. (1) To solve the first challenge, we transform model outputs in an unsupervised way to better represent the distribution of normality, and then detect shift statistically through hypothesis testing on the distribution of model outputs. (2) To solve the second challenge, we construct an optimization problem that finds as few drifting samples as possible. By doing so, we only label the most influential samples inducing the shift to reduce labeling overhead. (3) To solve the last challenge, we assign different importance weights to model parameters to represent whether they contains valid distribution (left dashed region of Fig. 1b). During shift adaptation, we restrict updating important model parameters to avoid forgetting old valid distribution, while favor updating non-important parameters to generalize to new distribution (right dashed region of Fig. 1b).

Evaluation and Deployment. We conduct several experiments using practical long-term datasets (including a benchmark) on three security-related applications with two representative anomaly detection models. We compare OWAD with state-of-the-art lifelong learning [20] and drift detection approaches [37], [5], [81] w.r.t end-to-end adaptation performance, and conduct ablation experiments of each step in OWAD. We provide case studies to analyze the normality shift and provide operational recommendations for security applications.



(a) Difference between supervised classification and anomaly detection. **(b) Two challenges of adaptation.**

Fig. 1: Illustration of challenges in this study.

We also conduct an initial real-world deployment of OWAD on electric power supervisory and control system (SCADA) logs. Results show that OWAD can provide better adaptation performance of normality shift with extremely fewer labels.

Contributions. This study makes the following contributions:

- We are the first to examine the problem of normality shift (defined in §II-C) for deep learning-based anomaly detection. To this aim, we propose OWAD, a general framework to tackle normality shift for security-related anomaly detection in practice, which can be applied to any DL-based zero-positive anomaly detection models. We provide the prototype implementation[2] of OWAD over two types of DL-based anomaly detection used in time-insensitive and time-series applications (introduced in §II).
- We propose novel techniques in OWAD (§IV) to solve three challenges unique to anomaly detection in security applications, including a *Calibrator* dedicated for anomaly detection models to solve the challenge of *unsupervised shift detection* (§IV-A, §IV-B), a well-formalized *Explainer* to solve the challenge of *reducing labeling cost* (§IV-C), and a *distributional-level* adaptation to solve the challenge of *ensuring not forgetting and generalization* (§IV-D).
- We conduct several experiments on three security applications to demonstrate the performance and reveal several guiding insights (in §V). Besides, our initial deployment of OWAD on a real-world SCADA log anomaly detection system corroborates the effectiveness of our method (§VI).

II. Background and Problem Scope

This section introduces the background of DL-based anomaly detection (§II-A) and its representative security applications (§II-B), as well as the preliminary of concept drift and the definition of our problem scope—normality shift (§II-C).

A. Deep Learning Approaches for Anomaly Detection

As widely introduced in prior works [20], [29], DL-based anomaly detection, also called "zero-positive" learning, is trained with purely normal data (i.e., without any anomaly). Existing learning approaches can be generally divided into the two categories according to the model output probabilities (whether indicating normal or abnormal). Henceforth, let $\boldsymbol{f}(\boldsymbol{x})$ denote the anomaly detection model's outputs of sample $\boldsymbol{x}$.

Abnormal-confidence Models—*the larger the $\boldsymbol{f}(\boldsymbol{x})$, the more abnormal $\boldsymbol{x}$ is*. The first category, also called *reconstruction-based* learning, is to learn by minimize the *reconstruction*

[2]The implementation of OWAD: **https://github.com/dongtsi/OWAD**

error of normal data during training. In this case, generative DL models, such as Autoencoder [54], [79] and Generative Adversarial Networks (GAN) [82] are widely used. In the detection/testing phase, the output of abnormal-confidence models (i.e., $\boldsymbol{f}(\boldsymbol{x})$) is the reconstruction error between inputs and outputs. An anomaly is reported if the output is larger than a pre-defined detection threshold denoted with T.

Normal-confidence Models—*the larger the $\boldsymbol{f}(\boldsymbol{x})$, the more normal $\boldsymbol{x}$ is*. The second one, also called *prediction-based* learning, is to learn by maximizing the probability of predicting the point *next to* the input (purely normal) sequence. In this case, sequential DL models, such as RNNs or Long short-term memory (LSTM) [21], [20] are widely used. In the detection/testing phase, normal-confidence models output the predictive probabilities of next-to-appear points and report an anomaly if the probability of actual next point is lower than the pre-defined detection threshold T.

B. Representative Security Applications

We introduce three representative security applications using the aforementioned DL-based anomaly detection (in §II-A) and respective systems evaluated in this study (in §V).

Network Intrusion Detection. DL-based anomaly detection empowers Network intrusion detection systems (NIDS) to detect unforeseen attacks and advanced variants [1]. An important step before anomaly detection is to extract features from traffic in two ways generally. Packet-based ones such as [54] extract informative features from each network packet, which however is difficult to handle high-volume traffic in practice. Flow-based methods such as [19] aggregate traffic with the same source and destination and then retrieve features from each "flow", which is widely used in practice [71]. As for DL anomaly detectors, `KitNET`[54] is a state-of-the-art DL-based NIDS detector, which consists of ensemble Autoencoders to conduct abnormal-confidence detection (§II-A).

Log Anomaly Detection. Logs are important for ensuring the reliability and availability of software-intensive systems. DL-based models have been widely proposed to automatically detect anomalies in system logs [21], [53], [30], [45]. These studies follow the general pipeline that firstly parses raw unstructured logs into a sequence of structured events [85], and then leverages normal-confidence methods for anomaly detection (§II-A). For example, `DeepLog` [21] maps each type of logs into a discrete value (named "log key") and thus parses logs into sequences of log keys. Subsequently, LSTM is leveraged for learning the distribution of normal log keys.

Advanced Persistent Threat Detection. The adoption of anomaly detection for detecting and tracing APT is an emerging research direction [9], [31], [27]. For example, `GLGV` [9] is developed to detect lateral movements within enterprise networks. The training data is collected by first constructing graphs from purely benign authentication logs and then leveraging Deepwalk [62] to generate embedding vectors. Then, reconstruction-based learning is used to detect abnormal authentications (i.e., links in the constructed graph).

C. Problem Scope and Definition

Preliminary – Concept Drift. In machine learning community, concept drift has been widely studied mainly in the setting of *supervised* learning [25]. Specifically, concept drift is defined as the change of joint probability of samples $\boldsymbol{x}$ (from feature space $\mathcal{X}$) and labels y (from label space $\mathcal{Y}$), denoted as $\mathbb{P}(\boldsymbol{x} \in \mathcal{X}, y \in \mathcal{Y}) = \mathbb{P}(\boldsymbol{x}, y)$. As $\mathbb{P}(\boldsymbol{x}, y) = \mathbb{P}(\boldsymbol{x}) \times \mathbb{P}(y|\boldsymbol{x})$, there can be three sources of concept drift: (1) change of $\mathbb{P}(\boldsymbol{x})$, known as *virtual* shift; (2) change of $\mathbb{P}(y|\boldsymbol{x})$, know as *actual* shift; (3) a mixture of the above two changes.

TABLE II: Important notations in this study.

Notation	Description
$\boldsymbol{x}^c, \boldsymbol{x}^t$	**c**ontrol (old) and **t**reatment (new) features/samples (DEFINITION 2)
$\{\mathcal{X}_A^c, \mathcal{X}_N^c\}$	**A**bnormal or **N**ormal feature space of **c**ontrol (old) samples
$\{\mathcal{X}_A^t, \mathcal{X}_N^t\}$	**A**bnormal or **N**ormal feature space of **t**reatment (new) samples
$\boldsymbol{f}(\cdot), T$	abnormal-/normal-confidence anomaly detection models; detection threshold
$\mathcal{C}(\cdot)$	Calibrator of model outputs (§IV-A)
N_c, N_t	# **c**ontrol and **t**reatment features (in each shift measurement)
K, M	# bins of frequency histogram for shift detection and explanation
$\mathbb{H}_i(\cdot)$	transform $\cdot$ into i-bin vectors with relative frequencies (for histogram)
$\boldsymbol{m}^c, \boldsymbol{m}^t$	mask vectors corresponding to $\boldsymbol{x}^c$ and $\boldsymbol{x}^t$ (for shift explanation&adaptation)

Our Problem Scope — Normality Shift. Unlike general concept drift in supervised learning, for zero-positive anomaly detection, we focus on the shift of normal data as the models are trained with purely normal data and without any knowledge of anomaly. We refer to such shift as "*normality shift*". Formally, let $\mathcal{X} = \{\mathcal{X}_A, \mathcal{X}_N\}$ (*Abnormal* and *Normal* feature space), then we define normality shift as:

DEFINITION 1 (NORMALITY SHIFT). Normality shift is the change of $\mathbb{P}(\boldsymbol{x} \in \mathcal{X}_N, y) = \mathbb{P}(\boldsymbol{x}) \times \mathbb{P}(y|\boldsymbol{x} \in \mathcal{X}_N)$.

Note that, we do not specifically distinguish between *normality* and *abnormality* drift as abnormality is exactly the opposite of normality for zero-positive anomaly detection. In other words, if normality shift is well captured and handled, then abnormality drift is also solved, theoretically.

Since virtual shift of $\mathbb{P}(\boldsymbol{x})$ does not decrease model performance, we mainly focus on the actual shift, the key reason for model aging. However, the observation of actual shift for normality distribution, i.e., $\mathbb{P}(y|\boldsymbol{x} \in \mathcal{X}_N)$, is intractable as we cannot obtain the actual prior distribution $\mathbb{P}(\boldsymbol{x})$ or $\mathbb{P}(\boldsymbol{x} \in \mathcal{X}_N, y)$ in practice. Besides, it is worth noticing that there is an inevitable deviation between the normality learned by the anomaly detection model and the ground-truth one. Therefore, in this study we detect the shift of normality *learned by the anomaly detection model*. Specifically, we detect the change of $\mathbb{P}(\boldsymbol{f}(\boldsymbol{x})|\boldsymbol{x} \in \mathcal{X}_N)$. Note that, such method is high-fidelity for models as the ultimate purpose of tackling normality shift is to avoid the aging of model performance. To detect the shift of probabilistic distribution, we need to collect samples from the *old* and *new* distribution, which are named *control* and *treatment* in this paper, as the following definition:

DEFINITION 2 (CONTROL AND TREATMENT FEATURE SPACES). Control feature space denoted by $\mathcal{X}^c$ represents the old feature space that has been learnt/adapted by the anomaly detection model, while Treatment feature space denoted by $\mathcal{X}^t$ the new feature space.

Therefore, the shift detection in this study is to determine whether normality of *treatment* features $\boldsymbol{x}^t \in \mathcal{X}_N^t$ is the same as *control* features $\boldsymbol{x}^c \in \mathcal{X}_N^c$. Further, the detection problem is transformed into compare $\mathbb{P}(\boldsymbol{f}(\boldsymbol{x}^c)|\boldsymbol{x}^c \in \mathcal{X}_N^c)$ and $\mathbb{P}(\boldsymbol{f}(\boldsymbol{x}^t)|\boldsymbol{x}^t \in \mathcal{X}_N^t)$.

III. Overview and Motivation

In this paper, we propose a general distribution-level framework OWAD (short for **O**pen-**W**orld **A**nomaly **D**etection) to tackle normality shift for anomaly detection in security domains, which consists of four sub-modules. In this section, We provide the overview of OWAD workflow with a simplified example and introduce the motivation of each sub-module.

A. Overview and Simplified Example

In Fig. 2, we provide a simplified example to explain the workflow of OWAD. The whole procedure works in an online fashion to continuously detect normality shift by collecting new samples and examining the normality shift. In practice, the timing of detecting shift requires a case-by-case analysis according to the tolerance of shift and overhead budget. A common approach is to periodically collect data, while can also detect at any suspicious time (such as after system update and known faults). In this motivation example, suppose that at the very beginning, we collect samples $\{\boldsymbol{x}_1^c, \boldsymbol{x}_2^c, \boldsymbol{x}_3^c, \boldsymbol{x}_4^c, \boldsymbol{x}_5^c\}$ from the same distribution as the training data. After some time, another samples $\{\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \boldsymbol{x}_3^t, \boldsymbol{x}_4^t, \boldsymbol{x}_5^t\}$ are collected from the new distribution to check whether normality shift occurs.

Then, OWAD works with four key steps. In step ❶, we propose a novel unsupervised calibration method to enforce the calibrated outputs to provide information of predictive confidence. As shown in the figure (red probabilities), the distributions of calibrated $\boldsymbol{f}(\boldsymbol{x}_i^c)$ and $\boldsymbol{f}(\boldsymbol{x}_i^t)$ become more distinguishable. In step ❷, we leverage hypothesis testing to statistically determine whether calibrated outputs of $\boldsymbol{x}^c$ and $\boldsymbol{x}^t$ follow similar distributions. Obviously, they are different in this example. Therefore, in step ❸, we propose an optimization-based *Explainer* to track the most influential samples for explaining the normality shift. Explainer assigns and optimizes weights (denoted with $\boldsymbol{m}^c$ and $\boldsymbol{m}^t$) for each $\boldsymbol{x}_i^c$ and $\boldsymbol{x}_i^t$ indicating the importance w.r.t the shift. The intuition is to "*reconstruct*" the shifted (new) distribution with as *many* $\boldsymbol{x}_i^c$ and as *few* $\boldsymbol{x}_i^t$ as possible since $\boldsymbol{x}^c$ have already been checked as normal while $\boldsymbol{x}^t$ are newly collected and requires manual labeling to filter anomalies (since we only focus on the shift of *normality*). As shown in Fig. 2, we keep the samples with high weights, including $\boldsymbol{x}_1^c, \boldsymbol{x}_2^c, \boldsymbol{x}_4^c$ and $\boldsymbol{x}_4^t, \boldsymbol{x}_5^t$. $\boldsymbol{x}_4^t, \boldsymbol{x}_5^t$ are manually labeled and filter out anomalies (i.e., not included for the following step, no one is filtered in this example). Subsequently, security analysts can obtain insights from two aspects of explanation with representative samples, including samples $\{\boldsymbol{x}_3^c, \boldsymbol{x}_5^c\}$ that are *no longer* normal and samples $\{\boldsymbol{x}_4^t, \boldsymbol{x}_5^t\}$ that newly *become* normal. Then in step ❹, for each parameter in the anomaly detection model, we estimate its importance w.r.t. adapting to the new normality while preventing forgetting important knowledge not included in the updated samples. Then, the anomaly detection model is updated by the newly normal samples $(\boldsymbol{x}_4^t, \boldsymbol{x}_5^t)$ with a regularization term penalizing the update of parameters according to their importance. Finally, as the procedure works continuously, the remaining samples $(\boldsymbol{x}_1^c, \boldsymbol{x}_2^c, \boldsymbol{x}_4^c$ and $\boldsymbol{x}_4^t, \boldsymbol{x}_5^t)$ become a new $\boldsymbol{x}^c$ for the next round.

B. Motivation of OWAD Design

Why Calibration before Detection? As mentioned in §II-C, the idea to detect shift is to compare the distribution of model outputs (i.e., $\boldsymbol{f}(\boldsymbol{x}_i^c)$ with $\boldsymbol{f}(\boldsymbol{x}_i^t)$). However, a key observation of this study is the outputs of anomaly detectors in security applications are highly *under-calibrated*, that is, the model outputs fail to provide probabilistic information. For example, $\boldsymbol{f}(\boldsymbol{x}_1^c) = 0.99$ and $\boldsymbol{f}(\boldsymbol{x}_5^c) = 0.92$ only provide the relationship of normal confidence $\boldsymbol{x}_1^c > \boldsymbol{x}_5^c$ instead of exact meaning of 0.99 or 0.92. In this context, uncalibrated outputs will hamper statistical detection of shift and especially identifying the drift samples. In Fig. 2, the original outputs of anomaly detection models are over-confidence and indiscriminately concentrated within 0.9-1.0 for both $\boldsymbol{x}^c$ and $\boldsymbol{x}^t$. Therefore, the model outputs need to be calibrated to provide meaningful probabilistic information. We provides more empirical evidence of (un)calibrated distributions in security applications in Fig. 6.

Why Shift Explanation? A further purpose of detecting normality shift is to make the model adaptable to normality changes and avoid performance degradation. However, security practitioners will not easily update models unless there is sufficient evidence or motivation due to the high concentration on reliability. Therefore, we need to first explain how normality shift happens between control and treatment spaces, which is exactly why we introduced the shift explanation module.

Why not Sample-level Explanation? Existing studies [81], [3] detect and explain concept drift in a *sample-level* way, which essentially solve whether and why a certain sample is out of distribution. Specifically, CADE [81] finds important dimensions that drive a certain sample away from existing distributions for explanation. Although such approaches provide finer-grained insights but are unfortunately failed to understand the shift of distribution *holistically* and unsuitable for selecting samples of adaptation. By contrast, OWAD is built in a *distribution-level* way. Our goal of explanation is to explain how the entire normality distribution changes. Specifically, the explanation result is the most important samples that result in normality shift. These selected samples can help security practitioners explain and understand normality shift by answering two questions: (1) *which part (in treatment space) is the newly emerging normality*? (2) *which part (in control space) is no longer normality*?

Why not existing Adaptation? Existing approaches to adapt to concept drift can be divided into two types: (1) The first is to retrain the model with old training samples together with newly detected drift samples [81], [37], [5]. However, this approach is memory-intensive and time-consuming as maintaining and retraining with an increasing number of samples. More importantly, it fails to capture the distributional shift as the medley samples cannot represent the new distribution. (2) The second is to incrementally learn new samples. Despite with no need of old samples, knowledge learned before is erased when learning new samples. This problem is called *catastrophic forgetting* [40]. To solve this problem, UNLEARN [20] leverages weight consolidation [40] to compute the importance of the parameters in deep learning models and add an regularization term to restrict the update of parameters. However, UNLEARN cannot distinguish valid and out-of-date knowledge in the old distribution, or pick out representative samples for generalizing to the new distribution. In light of the above concerns, we need to propose an adaptation approach to prevent catastrophic forgetting valid knowledge in old distribution while ensuring generalization to the new distribution.

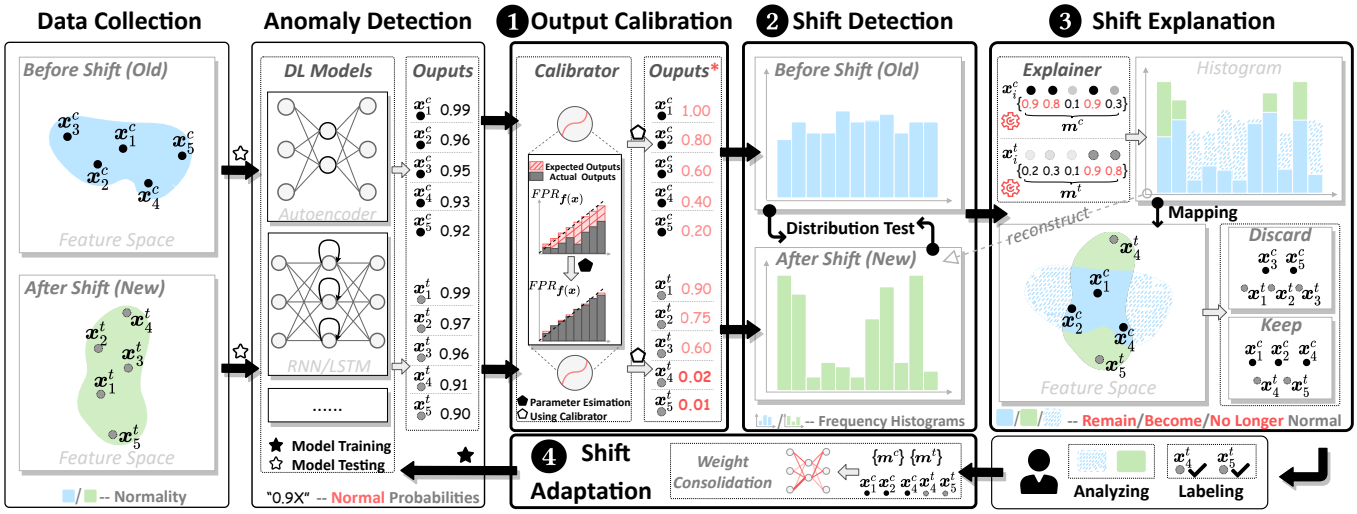


Fig. 2: The overview and example of OWAD.

IV. Methodology of OWAD

This section presents the technical details of four sub-modules in OWAD. Table II lists some important notations.

A. Calibration of Model Outputs

Challenge of Calibrating Anomaly Detectors. Confidence calibration is a well-studied technique in machine learning and deep learning community [56], [55], [28], which enables the calibrated model outputs to represent the true probability. For example, if we have some predictions all with output 0.8 after calibration, then 80% of them are expected to be correctly classified. Intuitively, calibration is to find a *function* (denoted with $\mathcal{C}(\cdot) : \mathbb{R} \to \mathbb{R}$) mapping original outputs to expect confidence. However, existing studies are calibrated for *supervised* classifiers as fully-labeled two-class samples are required to calculate the true probability such as accuracy (ACC), which is not computable for anomaly detection with only normal samples. Therefore, we need an *unsupervised* calibration method without any prior knowledge of anomaly. Besides, there are three requirements for the calibration:

1) **Non-linearity**: linear transformation is meaningless as it cannot change the density distribution of original outputs.
2) **Legality**: calibrated outputs must within the range of $[0, 1]$ to represent probability; formally, $\forall \boldsymbol{x}, \mathcal{C}(\boldsymbol{f}(\boldsymbol{x})) \in [0, 1]$.
3) **Monotonicity**: calibrated outputs cannot change the original order, otherwise will change the performance; formally, $\forall \boldsymbol{x}_1, \boldsymbol{x}_2$, if $\boldsymbol{f}(\boldsymbol{x}_1) < \boldsymbol{f}(\boldsymbol{x}_2)$, then $\mathcal{C}(\boldsymbol{f}(\boldsymbol{x}_1)) < \mathcal{C}(\boldsymbol{f}(\boldsymbol{x}_2))$.

Anomaly Detection Calibrator. In light of the above considerations, we propose a novel *Calibrator* which can transform the model output into expected confidence with only normal data. Without loss of generality, we primarily use normal-confidence anomaly detection models (recall §II-A) for illustrating our method (also the example in Fig. 2). The abnormal-confidence case is exactly the opposite and is discussed in Appendix A. For normal-confidence models, we leverage *false positive rate (FPR)* to define the expected normal confidence after calibration. As FPR is the ratio of false positives (FP) to negatives (TN+FP), not any anomaly is required. Recall §II-A, the anomaly is determined by comparing model outputs and detection threshold T. Therefore, T need to be determined to compute TN and FP. With a slight abuse of $\mathcal{X}_N$ to be all normal data used for calibration, we set T as $f(\boldsymbol{x})$ when calibrating $f(\boldsymbol{x})$. For example, if the calibrated output of a certain normal sample is 0.8, then ideally 80% of normal samples are FPs when the detection threshold $T = 0.8$. In other words, 80% of calibrated normal-confidence outputs are lower than this sample (i.e., with high confidence to be normal). Formally, we give the following definition:

Definition 3 (Perfect Calibration of Normal-confidence Anomaly Detection Models). The *perfect calibration* for a normal-confidence output is defined as FPR among normal samples after setting itself as the detection threshold. Namely,

$$\mathcal{C}(\boldsymbol{f}(\boldsymbol{x})) = FPR_{T=\boldsymbol{f}(\boldsymbol{x})}(\mathcal{X}_N), \quad \forall \boldsymbol{x} \in \mathcal{X}_N. \tag{1}$$

After defining the expected confidence, we need to select a parameterized function and perform parameter estimation to reduce the error between calibrated outputs with the expected confidences. In prior works [56], [28], exponential functions such as Sigmoid are preferred based on the assumption of long-tailed distribution of original outputs. However, we empirically find that the distribution of normality in security applications are more complicated. Therefore, we relax this assumption and leverage *piece-wise linear function* (PWLF) as a more general base function for security applications. PWLF has been widely studied in machine learning applications [22], which simply consists of several conterminous linear functions to together describe a complex function.

To satisfy the three requirements of calibration, we leverage isotonic regression [6] to fit the piecewise function. It is the technique of fitting a monotonic free-form line (i.e., piecewise function) given a sequence of observations. Therefore, the fitted function satisfies the requirements of non-linearity and monotonicity. To satisfy legality, we set the lower and upper bound of the fitted function as 0 and 1. In other words, predictions will be clipped to the nearest fitting interval endpoint. In a nutshell, isotonic regression solves the following quadratic

program given a sequence of observations $\{a_i, b_i\}$:

$$\min_{\mathcal{C}} \sum_{i=1}^{n} \big(\mathcal{C}(a_i) - b_i\big)^2, \quad \text{s.t. } \mathcal{C}(a_i) < \mathcal{C}(a_j) \text{ if } a_i < a_j. \quad (2)$$

According to DEFINITION 2, the observations to fit the calibrator are as follows. We sort the uncalibrated output $\boldsymbol{f}(\boldsymbol{x}_i^c)$ in increasing order for each *control* feature $\boldsymbol{x}_i^c \in \mathcal{X}_N^c$, and treat them as a_i. Here $\{\boldsymbol{x}^c\}$ means the set of $\boldsymbol{x}^c$, as a representation of $\mathcal{X}_N^c$. The corresponding b_i is computed as the expected confidence $FPR_{\boldsymbol{f}(\boldsymbol{x}_i^c)}(\{\boldsymbol{x}^c\})$. As isotonic regression is not our contribution, we refer readers to [17] for solution of (2).

B. Shift Detection

As mentioned in §II-C, the detection of normality shift in this work is to compare $\mathbb{P}(\boldsymbol{f}(\boldsymbol{x})|\boldsymbol{x} \in \mathcal{X}_N)$ between *control* ($\boldsymbol{x} = \boldsymbol{x}^c$) and *treatment* ($\boldsymbol{x} = \boldsymbol{x}^t$) features. With Calibrator, model outputs intrinsically contain probabilistic information to facilitate statistical detection of distribution shift. Therefore, we further transform the shift detection problem into comparing $\mathbb{P}(\mathcal{C}(\boldsymbol{f}(\boldsymbol{x})))$ between $\boldsymbol{x} = \boldsymbol{x}^c \in \mathcal{X}_N^c$ and $\boldsymbol{x} = \boldsymbol{x}^t \in \mathcal{X}_N^t$. Below, we introduce how to determine the normality shift based on the Calibrator.

Design Considerations. To compare the distributions, N_c control samples $\boldsymbol{x}^c$ and N_t treatment samples $\boldsymbol{x}^t$ are collected as discussed in §III. Generally, N_c is set to be the same size as the training data of the anomaly detection model in this study and $N_t \leq N_c$, considering the overhead of collection and labeling. Note that, $\boldsymbol{x}^c$ are supposed to be normal as they has been labeled in previous round of tackling shift. However, we cannot ensure that $\boldsymbol{x}^t$ are all normal as they are newly sampled from the current environment. In fact, our method initially detects shift with $\boldsymbol{x}^t \in \mathcal{X}^t$ (not $\mathcal{X}_N^t$). That is, it is allowed that there may be some anomalies in $\boldsymbol{x}^t$ as the further process will involve human investigation to filter anomalies.

To represent the distributions, discrete distributions of $\mathcal{C}(\boldsymbol{f}(\boldsymbol{x}^c))$ and $\mathcal{C}(\boldsymbol{f}(\boldsymbol{x}^t))$ are computed through frequency histogram with K bins (denoted with $\mathbb{H}_K$). K is a hyper-parameter and will be discussed and evaluated later. Subsequently, we statistically compare the two discrete distributions through hypothetical testing. We leverage *permutation tests*, a well-known and powerful methodology in statistics [32]. Compared with traditional tests (such as t-tests), permutation tests is distribution-free and support any test statistic without unverifiable assumptions about data, which is practical for security scenarios. Besides, permutation tests perform more exact results facing small sample sets through testing with resampling, compared with other non-parametric approaches (such as chi-square tests). This property is important especially considering that N_t is small for limited labeling overhead.

The null hypothesis ($\mathcal{H}_0$) is that $\mathcal{C}(\boldsymbol{f}(\boldsymbol{x}^c))$ and $\mathcal{C}(\boldsymbol{f}(\boldsymbol{x}^t))$ are from the same distribution (not shift) while the alternative hypothesis ($\mathcal{H}_1$) is the opposite (shift). The test statistic is the Kullback–Leibler (KL) divergence between two distributions, which is a common practice to measure the difference between probability distributions. For two discrete probability distributions P and Q defined on the same probability space, KL divergence is defined as $\mathcal{D}_{KL}(P||Q) = \sum_{p \in \mathcal{P}} P(p) \log\big(\frac{P(p)}{Q(p)}\big)$.

We leave the detailed algorithm procedure of shift detection in Appendix B.

C. Shift Explanation

As introduced in §III, we propose a distribution-level shift explanation method (called *Explainer*) in OWAD to find important samples inducing the normality shift to help security practitioners understand and adapt to shift. Specifically, we formulate an *optimization* problem of finding important samples for the normality shift. To provide better explanation results, the optimization needs to consider three aspects:

1) **Explanation Accuracy**: the selected samples for explanation should accurately represent the shifted distribution.
2) **Labeling Overhead**: samples selected from treatment space are expected to be few as manual labeling is required.
3) **Explanation Determinism**: the explanation is expected to be deterministic (i.e., not to be ambiguous for selection).

Formulation of Explainer. Note that the above three requirements must be considered at the same time. And the first two are contradictory (e.g., introducing more treatment samples will increase explanation accuracy but also increase labeling overhead). Therefore, we simultaneously consider them in the objective function to find the best trade-off. We introduce two *mask* vectors $\boldsymbol{m}^c$ and $\boldsymbol{m}^t$ corresponding to $\boldsymbol{x}^c$ and $\boldsymbol{x}^t$. The i-th of $\boldsymbol{m}^c$/$\boldsymbol{m}^t$ denoted with $\boldsymbol{m}_i^c$/$\boldsymbol{m}_i^t$ is the indicator of whether or not (1 or 0) to select $\boldsymbol{x}_i^c$/$\boldsymbol{x}_i^t$. For ease of optimization, we relax the value of $\boldsymbol{m}^c$/$\boldsymbol{m}^t$ to be within $[0, 1]$, and binarize them to 0/1 for sample selection. Thus, we formulate the problem as:

$$\min_{\boldsymbol{m}^c \oplus \boldsymbol{m}^t} \quad \mathcal{L}_{acc} + \lambda_1 \mathcal{L}_{lab} + \lambda_2 \mathcal{L}_{det}, \quad \text{s.t. } \boldsymbol{m}_i^c, \boldsymbol{m}_i^t \in [0, 1], \quad (3)$$

where:

$$\mathcal{L}_{acc} = \mathcal{D}_{KL}\big\{\mathbb{H}_M\big(\boldsymbol{p}^t\big) \;||\; \mathbb{H}_M\big((\boldsymbol{m}^c \odot \boldsymbol{p}^c) \oplus (\boldsymbol{m}^t \odot \boldsymbol{p}^t)\big)\big\}, \quad (4)$$

$$\mathcal{L}_{lab} = \frac{1}{N_c + N_t} \left\| (\boldsymbol{1} - \boldsymbol{m}^c) \oplus \boldsymbol{m}^t \right\|, \quad (5)$$

$$\mathcal{L}_{det} = \mathop{\mathbb{E}}_{m \in \boldsymbol{m}^c \oplus \boldsymbol{m}^t} [m \log m + (1 - m) \log(1 - m)]. \quad (6)$$

Note that $\oplus$ is the (row-by-row) concatenation of two (row) vectors and $\odot$ is Hadamard (element-wise) product.

As shown in (3), the optimization objective contains the aforementioned three terms, weighted by two customizable hyper parameters λ_1 and λ_2 for security operators with different requirements in various applications. For example, operator can increase λ_1 when the labeling ability is limited, while can decrease it for error-sensitive applications.

Accuracy Term—$\mathcal{L}_{acc}$ in (4). Here $\boldsymbol{p}^c = \mathcal{C}(\boldsymbol{f}(\boldsymbol{x}^c))$ and $\boldsymbol{p}^t = \mathcal{C}(\boldsymbol{f}(\boldsymbol{x}^t))$. The intuition is to "*reconstructs*" the new distribution after shift with selected $\boldsymbol{x}^c$ and $\boldsymbol{x}^t$, as shown in the motivation example of Fig. 2. Like shift detection, we still use $\mathcal{D}_{KL}$ to measure the distance between real treatment distribution and reconstructed distribution. In (4), the number of bins for computing relative frequencies is M, and $\mathbb{H}_M(\cdot)$ transforms calibrate outputs into M-bin vectors of relative frequencies in histogram. Compared with K used for shift detection, we need to ensure $M >> K$ to better explain and reconstruct the distribution at a fine-grained level.

Overhead Term—$\mathcal{L}_{lab}$ in (5). Note that $\boldsymbol{x}^c$ have already been labeled as normal previously while $\boldsymbol{x}_i^t$ require human investigation to filter anomalies. Therefore, we select as *many* $\boldsymbol{x}_i^c$ and as *few* $\boldsymbol{x}_i^t$ as possible to reduce labeling overhead. Specifically, we measure the L_2-norm of $\mathbf{1}-\boldsymbol{m}^c$ together with $\boldsymbol{m}^t$, and divide it by the number of samples (i.e., $N_c + N_t$).

Determinism Term—$\mathcal{L}_{det}$ in (6). To avoid ambiguity for selecting samples after optimization, we expect $\boldsymbol{m}^c$ and $\boldsymbol{m}^t$ to be deterministic (i.e., either close to 0 or close to 1). In (6), we measure the entropy (known as uncertainty) of $\boldsymbol{m}^c$ or $\boldsymbol{m}^t$.

We use gradient descent approach to solve the above optimization and randomly initialize $\boldsymbol{m}^c$/$\boldsymbol{m}^t$ in $[0, 1]$. Note that, although $\mathbb{H}_M(\cdot)$ in (4) is non-differentiable, we only use this operation once before optimization since the binning of $\boldsymbol{p}^c$ or $\boldsymbol{p}^t$ (computed from $\boldsymbol{x}^c$ or $\boldsymbol{x}^t$) is fixed during optimization. In other words, elements in $\boldsymbol{p}^c$ or $\boldsymbol{p}^t$ can only be selected or not (decided by $\boldsymbol{m}^c$ or $\boldsymbol{m}^t$) but cannot appear in other bins.

We manually involve out-of-bound samples (compared with the control set) into the final explanation. That is, we pick out samples in the treatment set whose model output probability is lower or higher than the minimum and maximum values of the control set (the output of such samples is 0 or 1 after calibration, as mentioned in §IV-A), and add these samples to the final explanation if not in the solution of (3).

D. Shift Adaptation

After filtering anomalies via human investigation and providing important samples, operators can obtain insights from our explanation. If normality shift is confirmed, we need to update anomaly detection models for *adapting* to normality changes, in order to avoid performance degradation.

We propose an adaptation approach to prevent catastrophic forgetting valid knowledge while ensuring generalization to the new distribution. Motivated by existing incremental adaptation methods such as `UNLEARN`, we add a special regularization term on the original loss function during model updating Different from L_2-norm regularization, we assign different importance weight denoted with Ω_i for each model parameter denoted with θ_i before updating. The intuition of Ω_i is to assess how important each parameter is to knowledge in the old distribution. By doing so, we restrict the updating of important parameters to prevent catastrophic forgetting, while relax the regularization of unimportant parameters to allow the model adapt to new distribution. Specifically, the new loss function of model adaptation $\mathcal{L}_{\theta^*}^{'}$ is:

$$\mathcal{L}_{\theta^*}^{'} = \mathcal{L}_\theta + \lambda_3 \sum_i \Omega_i \left(\theta_i^* - \theta_i\right)^2, \tag{7}$$

where $\mathcal{L}_\theta$ is the original loss function of anomaly detection. The regularization term is weighted by a hyper-parameter λ_3.

The core of (7) is how to assign Ω_i. In `UNLEARN` [20], Ω_i is the non-negative gradients $\partial\mathcal{L}_\theta/\partial\theta_i$ assessed with labeled samples from the old distribution. In such methods, old samples are randomly selected and equally considered to assess Ω_i. However, some old samples are out-of-date in the new distribution. Considering such samples will hinder the model from forgetting out-of-date knowledge in the old distribution, thus cannot effectively adapt to the new distribution. In this study, we assign different weights to different samples for assign Ω_i. Here the importance weights are exactly the mask $\boldsymbol{m}^c$ (before binarization) obtained from OWAD Explainer. Recall that lower mask value in control set indicates that the sample is not selected for representing the new distribution. Therefore, the intuition is that model will forget such out-of-date samples with lower weights/masks but memorize those with higher weights/masks. Specifically, Ω_i is assessed with control features $\boldsymbol{x}^c$ weighted by $\boldsymbol{m}^c$ from Explainer. Formally,

$$\Omega_i = \mathop{\mathbb{E}}_{\substack{x\in\boldsymbol{x}^c \\ m\in\boldsymbol{m}^c}} m \cdot \left(\frac{\partial l_\theta(x)}{\partial\theta_i}\right)^2, \tag{8}$$

where $l_\theta(x)$ represents the squared L_2-norm of model output logits [2]. Here we do not use the original loss function $\mathcal{L}_\theta$ to compute gradients like `UNLEARN` as it requires labels.

According to (7), we update the model with $\boldsymbol{x}^c$ and $\boldsymbol{x}^t$ selected after shift explanation and human investigation.

V. Evaluation

In this section, we first provide the experimental design considering the concerns in prior work and introduce several setups in §V-A. We compare OWAD with several baselines w.r.t. end-to-end performance after adaptation in §V-B. We conduct ablation experiments to demonstrate the effectiveness of sub-modules in OWAD in §V-C. We perform use-case analysis and conclude the experimental results and insights with operational recommendations in §V-D.

A. Experimental Design and Setups

1) Datasets in Three Security Applications: Our work aims to tackle normality shift for security-related anomaly detection in the *open world*. Therefore, our experiments are conducted in more *practical* settings. We find two flaws in prior studies: (1) The *time span* of evaluation data is too short to sufficiently observe and evaluate normality shift. For example, only 1.5 days of log data for evaluation in `UNLEARN` [20] and one week traffic flows in [3]. (2) Evaluation data is *simulated* in small testbeds which cannot represent the real-world shift (e.g., traffic is collected in several devices and simulated in [3]). In light of this, our experiments are evaluated on *long-term real-world* datasets. To demonstrate the universality, we evaluate OWAD under three security applications (introduced in §II-B), including two learning paradigms (§II-A) and three types of source data (tabular, time-series, and graph). We refer to Anoshift [18], a recent distribution shift benchmark for anomaly detection in NIDS to design our experiments. We primarily use public datasets to evaluate OWAD and baselines as they have ground truth and can be easily accessed by future work. However, the reason of shift is unavailable in public datasets. Therefore, we also conduct real-world deployment and analysis in §VI. Below, we introduce the respective datasets in three applications.

NID (Network Intrusion Detection). Here `KitNET` [54] is used as the anomaly detection model. We use the Kyoto 2006+ dataset [69] introduced by Anoshift benchmark [18]. It archives daily traffic from various honeypots inside and outside of Kyoto university network in 10 years from 2006 to 2015. It collects flow-level statistics and feedback from security products (e.g., IDS, Antivirus) as the feature set. Here

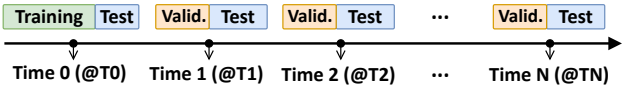


Fig. 3: Split of training, test and validation data.

we use the original experiment setting and pre-processed data in Anoshift benchmark to conduct fair evaluation.

LogAD (Log Anomaly Detection). `DeepLog` [21] is used as the log parser and anomaly detection model. We use a well-known dataset BGL [59] as the dataset, which is collected from a BlueGene/L supercomputer group in *214 days* with fully-labeled ground truth.

APT (Advanced Persistent Threat Detection). `GLGV` [9] is used as the graph embedding and the anomaly detection model. We use the same dataset LANL-CMSCSE [39] evaluated in `GLGV`, which consists of login events of *58 days* from internal computer network of LANL's Corporate with tens of thousands users and devices.

2) Data Selection and Split: Similar to the data split approach in Anoshift, our split of data is illustrated in Fig. 3. At the very beginning (at Time 0, abbreviated as @T0), we train anomaly detection models (mentioned in §II-B) with the *training* set (filtering out anomalies for zero-positive training, and the size of training set refer to their original works), and evaluate the original performance on the *test* set @T0. To measure the performance of different approaches facing with normality shift, we periodically collect samples N times (@T1, @T2, ..., @TN) and randomly split the collected data at each time point into *validation* and *test* sets according to the ratio of 1:1. The validation set is used by OWAD and other baseline approaches to determine drift and adapt anomaly detection models, while the corresponding test set is used to evaluate the model performance against drift after adaptation. For OWAD, the initial control set is the training set @T0 and the treatment set at each time is exactly the corresponding validation set.

Concretely, in NID case, we refer to the original setting in Anoshift. That is, we set N=5 (i.e., collect once a year from 2011 to 2015 for shifting evaluation). In LogAD and APT case, we set N=10 (collect 10 times in total) and the collection time interval is based on the overall span of each dataset. We set the ratio of the number of samples in control set to treatment set (i.e., N_c/N_t) as 1:1 by default. The intuition is that we collect the same number of samples as the training phase, while the difference is that the treatment set does not need to be fully labeled. Appendix C-1 provides more details of data split and experiment setup.

3) Baseline Approaches: We compare OWAD with several baselines (some of them are listed in Table I): (1) `No-Update` refers to original anomaly detection model without any remedy for shift; (2) `Retrain` refers to retraining models with old and new samples together; (3) `UNLEARN` in [20] is not originally designed to detect concept drift, but is selected for sharing the similar goal of updating zero-positive anomaly detection models. Note that, there is no mechanism of selecting important samples for labeling and adaptation in `Retrain` and `UNLEARN`. For fair comparison with same labeling overhead, we leverage *uncertainty sampling* for both of them to select important samples. Uncertainty sampling is to preferentially select samples with low-confidence model outputs (close to the detection threshold), which is widely-used in related works [3], [80] and proved to be more effective than random sampling.

As there is little work investigating concept drift for zero-positive setting, two SOTA approaches in security domains originally under supervised settings are adjusted and evaluated: (4) `CADE` [81] is designed for supervised classification models. As `CADE` inevitably requires all-class data to build its contrastive model, we additionally use anomalies in the training set (@T0), thus converting it to the binary classification drift detection. As the contrastive model can measure the distance of drifting samples and give anomaly (drift) scores, we preferentially select samples with far distances (high scores) for labeling and adaptation. (5) `TRANS` (abbreviated from `TRANSCENDENT`) [37], [5] requires non-trivial adjustments for zero-positive setting: the non-conformity measures (NCM) are the same as the metrics used for anomaly detection; as for types of conformal evaluator (CE), we use Inductive Conformal Evaluator (ICE) due to less runtime overhead. As for NCM threshold, we use quartile and credibility only (as the confidence is unavailable for the one-class case). The searching (random/grid) method is not used as the threshold optimization metrics require two-class labeled data (e.g., TPR). As for selecting the drifting samples (for labeling and adaptation), we preferentially choose/label the ones lower P-values. Detailed setting of hyper-parameters of baselines are in Appendix C-3.

4) Metrics: The metrics for evaluating adaptation performance are similar to those used for evaluating anomaly detection models in their works as our work is to recover the original performance after adapting to shift. We use the same metrics as the original works of three anomaly detection applications to evaluate the end-to-end adaptation performance. Specifically, TPR/FPR is used in original papers of NID and APT (i.e., `KitNET` [54] and `GLGV` [9]). However, such *single-threshold* metrics (i.e., computed with a pre-defined detection threshold, as introduced in §II-A) like TPR/FPR is sensitive to the selection of detection threshold. Thus, they cannot fully reflect the adaptation to the change of the entire normality distribution, but only for samples near the detection threshold. Therefore, we introduce AUC (the entire area underneath the ROC curve, derived from TPR and FPR of multiple thresholds) as a more reasonable metric for NID and APT. As for LogAD, `DeepLog` is a little different from normal-confidence models introduced in §II-A. It determines anomalies by observing whether the actual next point is in top g predictive ones. The original work [21] specifies $g = 9$ and measures F-Score for evaluation. Thus, we also measure F-Score in LogAD case.

B. End-to-end Performance

In this section, we measure the *end-to-end* performance—the performance of anomaly detection models after adaptation to the normality shift, which is a common practice of evaluation in related works [37], [81], [20] as ultimate goal of handling concept drift is to prevent performance degradation. To compare the end-to-end performance of OWAD with several baselines, we conduct various types of adaptation experiments under three applications. The results are shown in Fig. 4.

1) Performance of Single Adaptation: Here we conduct the comparison of OWAD with baselines by performing drift

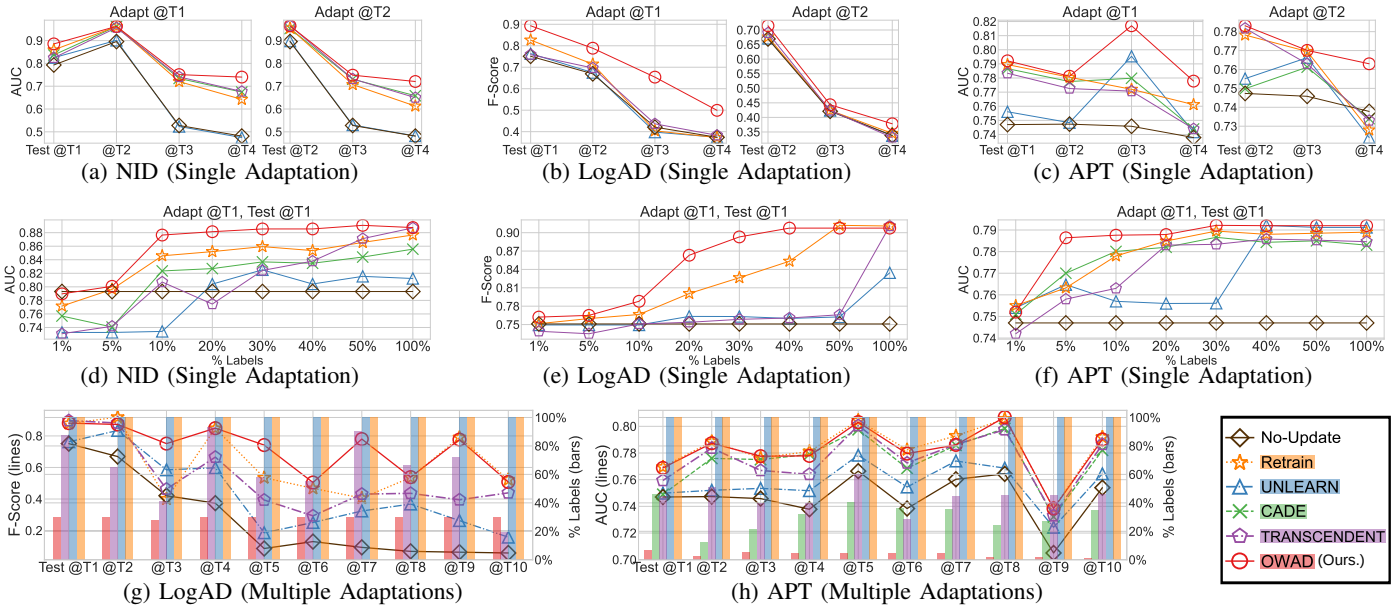


Fig. 4: End-to-end performance comparison under three security applications. Approaches are evaluated with 30% labels in (a)-(c). *Higher is better* for all line metrics in (a)-(h), *Lower is better* for bar metrics (right axis) in (g) and (h).

adaptation once on a specified time (e.g., Adapt @T1) for all approaches, referred to as "Single Adaptation". To perform a fair comparison, the budget of labeling overhead (i.e., the amount of available labels) for all approaches is set to be the same. To label prioritized samples, each baseline has own methods to select important samples, as introduced in §V-A3. As for OWAD, we preferentially select samples with higher mask values of $\boldsymbol{m}^t$ to label. We conduct two parts of experiments under "Single Adaptation". In Fig. 4a-4c, we fix the labeling ratio to be (at most) 30% of the validation set for all approaches, to evaluate the adaptation performance over time. In Fig. 4d-4f, we compare them under different labeling overhead budgets to measure the trade-off between labeling overhead and adaptation performance.

Adaptation Performance Over Time under the same Labeling Overhead. In Fig. 4a-4c, by performing adaptation for all approaches with fixed 30% labeling overhead, we update the the anomaly detection models only once and test the performance on current/subsequent time points (i.e., Adapt @Ti and Test @Ti, @Ti+1, @Ti+2, ...). We can observe that OWAD outperforms other approaches at the adaptation time (e.g., 10% more than other methods Adapt @T1 w.r.t. F-Score for LogAD in Fig. 4b). More importantly, OWAD can mitigate the performance degradation in subsequent shifts over time. As shown in the results, the advantage of OWAD becomes most apparent when testing at Time 4 (@T4). This indicates OWAD is the most robust method against shift over time.

An interesting finding is that the earlier the shift is adapted, the more robust is against subsequent shifts. As shown in Fig. 4b, the performance of Adapt @T1 at the corresponding time point is generally better than that of Adapt @T2 (e.g., in Test @T3 of LogAD, F-Score is $\sim$0.65 Adapt @T1 while $<$0.45 Adapt @T2 for OWAD). The probable reason is that the degree of shift increases over time, so it is more difficult to recover model performance for Adapt @T2 than @T1. This inspires us to detect drift as early and accurately as possible.

Trade-off between Labeling Overhead and Adaptation Performance. In Fig. 4d-4f, we evaluate the impact of the ratio of available labels to the validation set (i.e., budget of labeling overhead) and compare the adaptation performance of different approaches. As shown in the results, OWAD enables to better balance the trade-off between labeling overhead and adaptation performance. Compared to baselines, OWAD achieves better performance with less required labels. However, baseline methods mostly require large amount of labels to achieve a satisfactory performance and may perform unfortunately worse than `No-Update` with few labels (e.g., 5%/20%/40% labels for `UNLEARN` in Fig. 4d, 10% labels for `CADE` in Fig. 4d, and 5% labels for `TRANS` in Fig. 4e). Although some baselines can surpass OWAD with nearly 100% labels, we deem this is impractical and labor-intensive for security applications.

Security practitioners may care about the specific overhead for certain applications. As shown in Fig. 4d-4f, 10%, 30%, and 5% is enough for NID, LogAD, and APT. Specifically, 10% for NID is 5,000 flows only need to be labeled once a year. For LogAD, 30% seems like a lot, but we find that there are many identical logs, which significantly reduced the workload (e.g., less than a thousand logs need to be labeled for @T1). Moreover, we provide more specific overhead in the real-world deployment in §VI.

Analyzing Superiority to Baselines. We briefly analyze the superiority of OWAD compared with baselines from the above results here and perform more reasonable comparative experiments of each step in OWAD in the sub-component evaluation (§V-C). `Retrain` is widely used in practice as a benchmark method. However, we surprisingly find OWAD outperforms `Retrain` in most cases of limited labeling overhead (See Adapt @T1 in Fig. 4a and 4b). This indicates that OWAD can effectively select important samples w.r.t. the new normality distribution (through our shift explanation).

TABLE III: # FPs and # FNs of single adaptation in LogAD case.

Methods	# FPs (Lower is Better)			# FNs (Lower is Better)			F-Score (Higher is Better)		
	@T1	@T2	@T3	@T1	@T2	@T3	@T1	@T2	@T3
No-Update	2404	903	6585	135	34	39	0.74	0.66	0.42
Retrain	2238	933	6213	233	32	28	0.74	0.66	0.43
UNLEARN	3350	1293	7369	**105**	**27**	**26**	0.68	0.59	0.39
TRANS.	1508	849	3237	552	197	106	0.76	0.59	0.58
OWAD	**1491**	**701**	**2519**	120	34	35	**0.82**	**0.72**	**0.65**

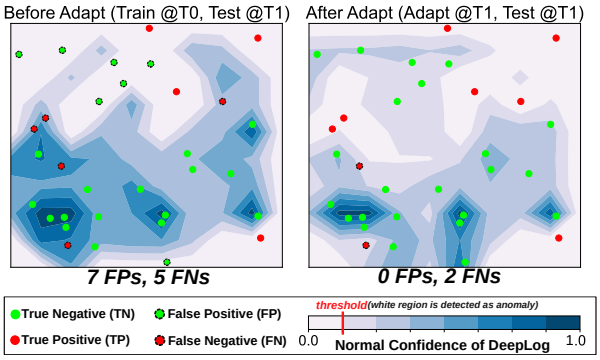


Fig. 5: Case study of FP/FNs before and after OWAD adaptation.

Ans our shift adaptation ensures not forgetting old valid normality while generalizing to new normality. By contrast, Retrain cannot remove out-of-date samples in old normality and focus on learning samples representing new normality. As for UNLEARN, we find FP/FNs tends to solve similar errors but is not suitable to deal with distributional shift as FP/FNs cannot represent the entire new normality distribution. For example, in Adapt @T1 of Fig. 4a, UNLEARN>No-Update when Test @T1 but <No-Update when Test @T3. Besides, the uncertainty sampling used by Retrain and UNLEARN can only capture samples close to the boundary (anomaly detection threshold), but not the shift of overall normality distribution. Like UNLEARN, CADE also uses sample-level adaptation. Their performance with limited labeling overhead can be unstable due to not well capturing overall distribution changes (e.g., see Fig. 4a-4c, their performance is sometimes better than No-Update, sometimes worse). CADE and TRANS have their own drift detection methods to select important samples to be labeled. However, their performance under less labeling overhead is unsatisfactory in Fig. 4d-4f. This is because their detected drifting samples may be significant relative to the old distribution but can not cover the entire drifted distribution. In other words, they are good at detecting drifting samples, but not suitable for directly adapting to these samples.

Case study of FPs and FNs. We present the analysis of FP/FNs in LogAD case. Firstly, we evaluate the number of FP/FNs before (No-Update) and after adaptation of baseline approaches and OWAD, and the results are listed in Table III. The experiment setting is the same as that in Fig. 4b. We conclude that OWAD is the only approach that is able to reduce both FPs and FNs (compared with No-Update) at the same time. Although UNLEARN has lower FNs, the number of FPs is significantly increased. We also provide an intuitive case study of how OWAD reduce FPs and FNs in Fig. 5. Here we depict the normal confidence of detection model (i.e., DeepLog) with its decision on some representative samples before (left) and after (right) OWAD adaptation. As the ground-truth normality of model is unavailable, we simulate and depict the 2-D contour with samples and outputs from @T0 to @T10. The samples in LogAD cases are time series. For depicting, we use LSTM hidden states and reduce them to 2 dimensions through PCA. Samples (green/red circles) in two sub-figures are the same and equally selected from the test set of @T1. As shown in Fig. 5, OWAD explanation can pick out important samples in the new normality distribution to update model (this is why FP is eliminated), and OWAD can remember important regions (this is why previous TN/TP remains) while forget unimportant areas (this is why FN is eliminated) by assigning importance weights for model parameters (§IV-D).

2) Performance of Multiple Adaptations: Apart from "Single Adaptation", here we also introduce "Multiple Adaptations", a more practical evaluation: In each time from @T1 to @T10, all approaches adapt to the shift with the validation sets and are evaluated with the corresponding test sets. Here we do not restrict the labeling overhead for all approaches and treat labeling overhead as another evaluation metric in addition to adaptation performance. For Retrain and UNLEARN, we label all the validation set (i.e. 100% labeling overhead). More specifically, Retrain retrains the anomaly detection model with control set together the entire validation set (after filtering anomalies) at each time point. UNLEARN incrementally trains the model with all FP/FNs in the validation set. For CADE and TRANS, we label all the drifting samples derived from their detection methods and use them (after filtering anomalies) for adaptation. For OWAD, different from that in Single Adapt, we binarize the mask $\boldsymbol{m}^c/\boldsymbol{m}^t$ to $0/1$ with a threshold of 0.5 and then label treatment samples with $\boldsymbol{m}^t = 1$. The results are shown in Fig. 4g and 4h. We measure the adaptive performance (lines) and required labels (bars) at each time point. The performance of baselines is significantly better compared to the previous results (with limited overhead) when satisfying labeling all requires samples, while mostly not as good as OWAD. Retrain is sightly better than OWAD in some cases of APT case but at the cost of dozens of times the required labels. In summary, OWAD can achieve better results with significantly less required labels.

C. Sub-components Evaluation

In this section, we conduct *ablation experiments* to illustrate the effectiveness of each of four steps in OWAD.

Calibration and Shift Detection (❶❷). We evaluate the effectiveness of OWAD Calibrator (§IV-A) by comparing the detection and adaptation performance between uncalibrated and calibrated outputs. We also compare PWLF used by OWAD Calibrator with other calibration functions. We compare PWLF with other typical interpolation/approximation methods that satisfy the three requirements (especially for monotonicity) in §IV-A. Thus, we use Sigmoid functions [56], [28] mentioned in §IV-A as an approximation method. As for interpolations, we use Linear spline interpolation and Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) [23].

We evaluate NID and logAD as two representative cases (normal-confidence and abnormal-confidence in §II-A). For uncalibrated outputs in NID, we normalize the reconstruction errors to $[0, 1]$. The results are listed in Table IV. (1) As for

TABLE IV: Comparison of Calibration Methods.

(a) NID

Methods	P-value*		ΔAUC ↑	
	Shift ↓	Unshift ↑	@T1	@T3
Uncalibrated	✗(0.13)	✓(0.21)	0.02	0.10
Sigmoid	✓(0.02)	✓(0.15)	0.08	0.19
PCHIP	✓(**0.00**)	✓(0.34)	0.07	0.15
Linear	✓(**0.00**)	✓(0.36)	**0.09**	0.21
PWLF	✓(**0.00**)	✓(**0.49**)	**0.09**	**0.22**

(b) LogAD

P-value*		ΔF-Score ↑	
Shift ↓	Unshift ↑	@T1	@T3
✗(0.06)	✓(0.23)	0.05	0.13
✗(0.76)	✓(0.73)	0.04	0.12
✗(0.74)	✓(0.36)	0.05	0.14
✗(0.69)	✓(0.78)	0.08	0.14
✓(**0.00**)	✓(**0.93**)	**0.09**	**0.17**

∗ P-value should close to 0 for "Shift" and close to 1 for "Unshift". ✓means test result is correct and ✗ is wrong (the cutoff threshold of p-value is 0.05).

TABLE V: Comparison of retraining-based adaptations.

Methods	**NID** (AUC)		**LogAD** (F1)		**APT** (AUC)	
	@T1	@T3	@T1	@T3	@T1	@T3
`Retrain`	0.85	0.72	0.82	0.40	0.78	0.77
`TRANS` (+Retraining)	0.83	0.74	0.75	0.43	0.78	0.77
`CADE` (+Retraining)	0.82	0.73	N/A.	N/A.	0.78	0.77
❶❷❸ + Retraining	**0.87**	**0.74**	**0.83**	**0.50**	**0.79**	**0.78**

TABLE VI: Comparison of lifelong-based adaptations.

Methods	**NID** (AUC)		**LogAD** (F1)		**APT** (AUC)	
	@T1	@T3	@T1	@T3	@T1	@T3
(FP/FNs+) `UNLEARN`	0.82	0.52	0.76	0.58	0.75	0.79
❶❷❸ + `UNLEARN`	0.84	0.52	0.83	0.60	0.76	0.80
❶❷❸ + ❹ (OWAD)	**0.88**	**0.75**	**0.89**	**0.65**	**0.79**	**0.81**

detection, we respectively treat @T1 with @T2 and @T1 with its down-sampling set as *shift* and *unshift* cases. A higher P-value means more likely to be unshift after detection (generally 0.05 as the threshold in statistics). As shown in the results, uncalibrated outputs fail to distinguish shift and unshift especially in two cases. OWAD is the only one can correctly detect shift/unshift with higher confidences for both cases. In NID case, Sigmoid, PCHIP, Linear all correctly detect shift/unshift but with lower confidence. In LogAD case, all three baselines fail to detect shift. Compared with PWLF, Sigmoid is only suitable for long-tail distribution, and interpolations may suffer "overfitting" and are unable to handle a large number of identical/similar dependent variables. This is why they fail on logAD. (2) As for *adaptation*, we evaluate the increase of AUC (for NID) or F-Score (for LogAD) compared to the original model (`No-Update`). Results corroborate that our calibration can effectively help to explain and adapt to shift compared with uncalibrated outputs and three interpolation/approximation methods.

Shift Explanation (❸). We evaluate the effectiveness of OWAD Explainer (§IV-C) by answering two questions:

- **RQ1**: Whether samples selected by OWAD Explainer are more effective than those by baselines (`CADE` and `TRANS`)?
- **RQ2**: Whether selected samples are more effective than FN/FPs (`UNLEARN`) and uncertainty sampling (`Retrain`)?

To answer **RQ1**, we force OWAD, `CADE`, and `TRANS` to all use the *retraining* method after selecting their respective shift samples (`CADE` and `TRANS` originally use retraining in their works), under the same setting as in "Single Adapt" with 30% labeling budget. We intuitively express our method as "❶❷❸ + Retraining". We also evaluate `Retrain` and the results are shown in Table V. The superior performance in all cases demonstrates the effectiveness of our shift explanation method. As for **RQ2**, we compare adaptation approach in `UNLEARN` using our explained samples (denoted with "❶❷❸ + `UNLEARN`") and the original `UNLEARN` using FP/FNs. As listed in the first two lines of Table VI, our selected samples outperforms FP/FNs w.r.t adaptation performance, thanks to the overall understanding of shifted normality provided by our distribution-level explanation. Results also demonstrate the superiority of OWAD compared with uncertainty sampling.

Shift Adaptation (❹). We compare our adaptation method with `UNLEARN` to demonstrate the effectiveness of introducing sample explanation ($\boldsymbol{m}^c$) into the regularization term (§IV-D). The results in the last two lines of Table VI demonstrate the effectiveness of our adaptation method and better performance compared with retaining in the last line of Table V. In Appendix E, we also evaluate the impact of regularization term in (7) by comparing $\lambda_3 = 0$ and $\lambda_3 > 0$. The conclusion is that the regularization term improves F-Score by ~ 2 %.

D. Analysis and Recommendations

In this section, we provide case studies and investigate how normality changes in three applications with the help of OWAD. We also provide operational recommendations for real-world applications based on our analysis and experiments.

1) Case Studies: We analyze normality shift in three applications with the explanation results provided by OWAD Explainer (§IV-C). As mentioned before, we only analyze the feature-space changes here as the reason of shift is unavailable public datasets (as they are not collected by us) and raw information is sanitized (e.g., IPs in NID). The root-cause analysis of normality shift is analyzed in §VI. Fig. 6 (Appendix D) provides intuitive visualization of normality shift.

Normality Shift in NID. For Kyoto 2006+ dataset, we find that there are two sharp normality shifts happened in 2011 and 2014, while the shift is relatively slow between other years (in line with Anoshift). With the help of OWAD Explainer, a representative example is IDS alert (as a feature in dataset) "9-124-1": none in 2006-2010 and appeared in 2011 (about 15%), while disappeared in 2014 and 2015. Another example is the ratio of UDP flows: ∼1% in 2006-10 and became less (<0.2%) in 2011-13, while >85% in 2014-15.

Normality Shift in LogAD. We find that BGL dataset [59] used in LogAD is extremely irregular, which may be caused by the uncertainty of tasks running on the monitored supercomputer systems. After explanation and analysis based on OWAD, we find the main reason for normality shift in LogAD is the change of order of logs in time series. Besides, normality shift is also reflected in two other aspects: One is due to the emergence of new types of logs (e.g. "`iar<*>dear<*>`" only appears after the sixth month), and another is the disappearance of (normal) logs (e.g., "`generating core<*>`" appears abundantly in the first two months but disappears completely in the last six months).

Normality Shift in APT. We find the dataset in APT case is the most stable relatively, thanks to its robust feature extraction method (extracting the connection relationship in the authentication graph of user logins via graph embedding). After explanation, we find the normality shift is mostly due to the emergence of a new subgraph pattern indicating new users' normal login behavior. For example, User `U482@DOM1` is authorized to login with 45 destinations in the first week and the number becomes 362 in the third week, while then back to 42 in the fifth week.

TABLE VII: Real-world deployment and test of OWAD.

	Week 1	Week 9 (@T1)		Week 18 (@T2)		Test @T2 (Adapt@T1)
	#FP	#FP	P-value	#FP	P-value	#FP
Device A	14	25	0.999	79	0.257	Unshift
Device B	45	1,027	0.000	1,678	0.000	154
Device C	68	3,071	0.000	3,103	0.000	98

2) Operational Recommendations: Based on our empirical evaluation and analysis, we provide the following recommendations for the deployment of practical security applications:

- Normality shift in practical security applications is common and complicated (case-by-case). Therefore, timely and efficient adaptation is imperative. Besides, it is not appropriate to provide a binary result (yes or no) of shift but to explain and understand the complicated shift.
- Model outputs in security applications are highly under-calibrated. OWAD *Calibrator* can help analysts understand outputs of anomaly detection models and facilitate the selection of detection threshold by assigning a probabilistic meaning to the calibrated outputs.
- The labeling overhead and adaptation performance is a trade-off. Compared with baselines, OWAD can achieve similar or better performance with extremely lower labels.
- In practice, the timing of detecting shift requires case-by-case analysis. A common approach is to periodically collect data, while more recommended to detect at suspicious times or involve other intelligence/domain knowledge. Besides, empirical results demonstrate that the earlier the shift is adapted, the more robust is against subsequent shifts.
- OWAD is shown to effectively reduce both False Positives and False Negatives. Admittedly, OWAD is better at resolving FPs as it designed to directly tackle *normality* shift. Although zero-positive anomaly detection is theoretically independent of the anomaly distribution, the model performance indeed relies on the detection of anomalies.

VI. Real-World Test on SCADA

Task Description. We have worked with a large company responsible for supplying electricity transmission and unified control of power grid in China and performed an initial test of OWAD on their security monitoring device of electric power supervisory and control system (SCADA). Specifically, their security systems monitor logs of *periodic* and *trigger* events in critical devices (e.g., servers, workstations) in the grid, and then perform anomaly detection of these logs. We collect tens of millions logs in total from over 20 devices of diverse business or management platforms in four months (18 weeks) from October 19, 2021 to February 20, 2022. Similar to LogAD case in §V, we use `DeepLog` [21] as the base model to conduct per-device log anomaly detection. To evaluate the performance, we let the analyst label the alerts outputted by anomaly detection models (to determine FP/TPs). We choose three representative devices (others share similar results with one of the three devices due to the same role) and the results are shown in Table VII. Results of "Week 1" corroborate a good performance of anomaly detection without shift.

Effectiveness of OWAD. We only evaluate normality (#FPs) here as ground-truth anomalies are not available in practice. As for the detection interval, we choose 9-week as SCADA environment is relatively stable. We determine whether normality shift occurs in Week 9 and 18 via OWAD and validate with performance degradation. P-values show that shift happens in device B and C. We conduct shift adaptation in Week 9 and test the performance again in Week 18. Results show extraordinary adaptation performance of OWAD in reducing FPs. Visualization of normality shift in three devices is also depicted in Fig. 6 for reference.

Case Studies and Shift Analysis. As for the specific labeling overhead, OWAD Explainer locates about 200 logs (some of them are identical) for Device A and takes an operator about half an hour for labeling and explaining, which is totally acceptable. We analyze the root cause of normality shift with the help of OWAD. As for device B, we find the distributional change is similar to gradual drift. OWAD Explainer locates the periodic logs of received/sent network volume. After analysis, we find the shift reason of device B, as the business platform, is due to dramatic change of user number and the usage of FTP service. As for device C, Explainer locates the trigger logs of process/service sequence. Based on the track of OWAD shift detection, we find the shift suddenly happens in about week 9. The reason is that there was a system update at that time and the order of service execution of device C changed after reboot. Besides, we also find that other reasons including (legal) actions by operators such as starting certain services but forgetting to close. The analysis of normality shift can help build more robust models (training with more shifted normal data) and filter FPs caused by normality shift.

VII. Discussions

Complexity Analysis. For memory footprint, lifelong learning-based methods (OWAD and `UNLEARN`) are memory-efficient as they only maintain a constant number (N_c) of samples. For time complexity, as methods depend on their own parameters, we empirically evaluate the runtime of end-to-end adaptation. `UNLEARN` is omitted here as it does not have the process of detecting and finding drift samples. The average run time of default configured OWAD, `CADE` and `TRANS` with 10^5 validation samples are 116s, 975s, and 1,449s. OWAD is efficient as it does not build extra learning models (compared to `CADE`) or repeatedly traverse all samples (compared to `TRANS`). Note that, this excludes the labeling time, otherwise `CADE`/`TRANS` is more expensive.

Hyper-parameter sensitivity. We evaluate the sensitivity of hyper-parameters and provide the default configuration of OWAD in Appendix E. The conclusion is hyper-parameters are insensitive in individual ranges. Future work could focus on more systematic and general configuration method and/or in other security applications

Limitations and Future Work. First, the explanation of shift indeed requires domain knowledge. Future work can relax the requirements and provide analysis for more security applications. How to automatically mine the in-depth root causes (not only feature-space samples) of shifting is also worthy of future work research. Second, the real-world test in §VI is initiatory in this study. Future work will provide more in-depth analysis and long-term deployment. Third, the robustness of OWAD itself (e.g., against adversarial attacks

or noisy labels) will be studied in future work. Fourth, we periodically collect data for evaluating shift in this study and future work will investigate more advanced methods regarding shift detection timing and treatment data collection.

VIII. Related Work

Concept Drift in Security Applications. In security domains, prior works on tackling concept drift can be generally divided into two types. The first is to *detect* and then *adapt to* the new concept. However, existing studies are mostly in supervised settings, which are ill-suited for unsupervised (zero-positive) anomaly detection. In malware detection, CADE [81] is mainly designed for detecting unseen attack classes in the supervised classification, while Transcend [37] and TRANSCENDENT [5] can be transferred for zero-positive setting after non-trivial adjustments, but focus more on handling drift by rejection instead of adaptation. In network intrusion detection, INSOMNIA [3] detects uncertain model outputs as drifting samples with another classifier. However, we deem that uncertain predictions cannot reflect distributional change. Besides, the underlying DL model is also supervised binary classification.

Another promising direction is to design features with strong *robustness* against concept drift, which has been studied for malware detection [51], [83], [73], [10], network intrusion detection [24], and log anomaly detection [84], [47]. However, this approach requires a lot of domain knowledge in designing features and is difficult to generalize to other domains. Moreover, robust features can only mitigate aging but not essentially trickle the drift (especially for label drift). Nevertheless, this approach is orthogonal to our work and can be used to strengthen the base anomaly detection model.

Concept Drift in Streaming Data. There have been lots of studies on concept drift of time-insensitive streaming data over the past decade [7]. Data steams are continuously monitored with fixed or adaptive sliding windows [8], and the statistics between windows are compared to detect drift based on error rate [26], [58] and statistical likelihood [43], [32]. As for adaptation, self-adaptive learners are extensively studied for decision trees [34]. Ensemble learning is also leveraged via training multiple weak classifiers with time windows [70], [42] A recent study on network intrusion detection used the above methods to detect drift for SVM-based NIDS [35]. However, detection based on error rate is delayed after model aging and needs a lot of manual labeling, and statistical testing based on strong assumptions on testing data, which is infeasible for complicated practical data in security domains. Moreover, prior approaches in this domain focus on the data itself with statistical learning models. By contrast, our work aims to improve existing DL-based anomaly detection systems. A few works for time-series streaming data focus on drift on univariate time series which can be typically defined as a sudden spike, a jitter, or a dip in curves [50]. Obviously, the underlying anomaly detection models and definition of drift are thoroughly different from our scope.

Out of Distribution (OOD) Detection. Machine learning community has widely studied OOD detection recently [33], [46], [64]. The high-level idea is to identify test samples that are not from distribution of training set (in-distribution). However, OOD detection studies are under supervised settings and not directly applicable in our case. In zero-positive anomaly detection, we cannot distinguish OOD normal samples from real anomalies as they both deviate from normality (Fig. 1a).

Domain Adaptation. Domain adaptation (DA) is a important field associated with transfer learning. Great progress has been made in both statistical learning [16], [63] and deep learning community [49], [74], [75]. DA aims to adapt a model trained on a label-sufficient *source* domain (i.e., data distribution) to a label-scarce *target* domain. By contrast, OWAD focuses more on shift detection and explanation before adaptation. As mentioned in §I, direct and continuous adaptation is inappropriate for security applications for lack of interpretability and expensive. Besides, DA studies mostly focus on supervised classification and end-to-end learning in the domain of computer vision or nature language processing. Their insights and assumptions such as preserving high-level visual/linguistic patterns across domains may not hold in security tasks.

Benchmarks on Distribution Shift. There is a growing popularity for building benchmarks to evaluate distribution shift [41], [11], [48], [44], [67]. However, most of them are built with in non-security datasets (texts/images) and/or supervised tasks. Anoshift [18] is a recent benchmark for anomaly detection in NIDS. We have evaluated NID case on Anoshift and extend similar setting to LogAD and APT cases.

Lifelong Learning. Lifelong learning (also known as continuous/incremental learning) has been widely studied in the deep learning community to train the same model over multiple datasets and tasks [40], [2], [60], [66], [65]. `UNLEARN` [20] has applied related techniques to anomaly detection, which is evaluated as a baseline of this study. However, in OWAD, we focus more on the identification and understanding of normality shift and then adapt to new normality with shifted samples. Motivated by the unsupervised evaluation of parameter importance [2], we propose a similar method for shift detection by leveraging the explained importance. Therefore, we believe lifelong learning is orthogonal to our work, and more advanced approaches can be used to devise stronger adaptation approaches. Future work could focus on this.

IX. Conclusions

In this paper, we propose OWAD, a general framework to detect, explain, and adapt to normality shift of DL-based anomaly detection in security applications. We propose several novel techniques including (1) an unsupervised Calibrator dedicated to anomaly detection to help models provide meaningful probabilities and facilitate detection of distributional change, (2) a well-formalized Explainer to help security operators determine and understand shift with less labeling overhead, and (3) a distributional-level adaptation approach to ensure not forgetting old useful normality while generalizing to new normality. We conduct more realistic experiments on three representative security applications and anomaly detection systems with practical long-term datasets as well as real-world deployment on SCADA security monitoring systems. Results demonstrate the wide applicability and great effectiveness of OWAD for tackling normality shift of security applications in practice, compared with prior studies. We also conduct case studies to analyze normality shift in each application and provide operational recommendations for the deployment of OWAD on more security applications.

ACKNOWLEDGMENT

We are grateful to all the anonymous reviewers for their insightful comments. We also thank for the help of all the members from NMGroup and CNPT-Lab in Tsinghua University. This work is supported by the science and technology project of State Grid Corporation of China "Research on Vulnerability Analysis and Threat Detection Key Technology of Power Monitoring System in Cyberspace" (Grand No.5108-202117055A-0-0-00). Zhiliang Wang is the corresponding author of this paper.

REFERENCES

[1] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[2] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *European Conference on Computer Vision (ECCV)*, vol. 11207. Springer, 2018, pp. 144–161.

[3] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "INSOMNIA: towards concept-drift robustness in network intrusion detection," in *Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security (AISec)*. ACM, 2021, pp. 111–122.

[4] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security." USENIX Association, 2022.

[5] F. Barbero, F. Pendlebury, F. Pierazzi, and L. Cavallaro, "Transcending transcend: Revisiting malware classification with conformal evaluation," *IEEE Symposium on Security and Privacy (S&P)*, 2022.

[6] R. E. Barlow and H. D. Brunk, "The isotonic regression problem and its dual," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 140–147, 1972.

[7] R. S. M. Barros and S. G. T. C. Santos, "A large-scale comparison of concept drift detectors," *Information Sciences*, vol. 451, pp. 348–370, 2018.

[8] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[9] B. Bowman, C. Laprade, Y. Ji, and H. H. Huang, "Detecting lateral movement in enterprise computer networks with unsupervised graph ai," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2020, pp. 257–268.

[10] H. Cai, "Assessing and improving malware detection sustainability through app evolution studies," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 2, pp. 1–28, 2020.

[11] Z. Cai, O. Sener, and V. Koltun, "Online continual learning with natural distribution shifts: An empirical study with visual data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 8281–8290.

[12] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *CoRR*, vol. abs/1901.03407, 2019.

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

[14] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis, "Casting out demons: Sanitizing training data for anomaly sensors," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE Computer Society, 2008, pp. 81–95.

[15] G. F. Cretu-Ciocarlie, A. Stavrou, M. E. Locasto, and S. J. Stolfo, "Adaptive anomaly detection via self-calibration and dynamic updating," in *International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, vol. 5758. Springer, 2009, pp. 41–60.

[16] H. Daume III and D. Marcu, "Domain adaptation for statistical classifiers," *Journal of artificial Intelligence research*, vol. 26, pp. 101–126, 2006.

[17] J. De Leeuw, "Correctness of kruskal's algorithms for monotone regression with ties," *Psychometrika*, vol. 42, no. 1, pp. 141–144, 1977.

[18] M. Drăgoi, E. Burceanu, E. Haller, A. Manolache, and F. Brad, "Anoshift: A distribution shift benchmark for unsupervised anomaly detection," *Neural Information Processing Systems NeurIPS, Datasets and Benchmarks Track*, 2022.

[19] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP)*. SciTePress, 2016, pp. 407–414.

[20] M. Du, Z. Chen, C. Liu, R. Oak, and D. Song, "Lifelong anomaly detection through unlearning," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, pp. 1283–1297.

[21] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 1285–1298.

[22] J. G. Dunham, "Optimum uniform piecewise linear approximation of planar curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. PAMI-8, no. 1, pp. 67–75, 1986.

[23] F. N. Fritsch and J. Butland, "A method for constructing local monotone piecewise cubic interpolants," *SIAM journal on scientific and statistical computing*, vol. 5, no. 2, pp. 300–304, 1984.

[24] C. Fu, Q. Li, M. Shen, and K. Xu, "Realtime robust malicious traffic detection via frequency domain analysis," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2021, pp. 3431–3446.

[25] J. a. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Survey*, vol. 46, no. 4, 2014.

[26] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian symposium on artificial intelligence*. Springer, 2004, pp. 286–295.

[27] P. Gao, X. Xiao, D. Li, K. Jee, H. Chen, S. R. Kulkarni, and P. Mittal, "Querying streaming system monitoring data for enterprise system anomaly detection," in *IEEE International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1774–1777.

[28] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*. JMLR.org, 2017, p. 1321–1330.

[29] D. Han, Z. Wang, W. Chen, Y. Zhong, S. Wang, H. Zhang, J. Yang, X. Shi, and X. Yin, "Deepaid: Interpreting and improving deep learning-based anomaly detection in security applications," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021, p. 3197–3217.

[30] S. Han, Q. Wu, H. Zhang, B. Qin, J. Hu, X. Shi, L. Liu, and X. Yin, "Log-based anomaly detection with robust feature extraction and online learning," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2300–2311, 2021.

[31] X. Han, T. F. J. Pasquier, A. Bates, J. Mickens, and M. I. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," in *27th Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2020.

[32] M. Harel, S. Mannor, R. El-Yaniv, and K. Crammer, "Concept drift detection through resampling," in *Proceedings of the 31st International Conference on Machine Learning (ICML)*, vol. 32. JMLR.org, 2014, pp. 1009–1017.

[33] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[34] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2001, pp. 97–106.

[35] M. Jain, G. Kaur, and V. Saxena, "A k-means clustering and svm based hybrid concept drift detection technique for network anomaly detection," *Expert Systems with Applications*, p. 116510, 2022.

[36] S. T. K. Jan, Q. Hao, T. Hu, J. Pu, S. Oswal, G. Wang, and B. Viswanath, "Throwing darts in the dark? detecting bots with limited data using

neural data augmentation," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2020, pp. 1190–1206.

[37] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *26th USENIX Security Symposium (USENIX Security)*. USENIX Association, 2017, pp. 625–642.

[38] A. Kantchelian, S. Afroz, L. Huang, A. C. Islam, B. Miller, M. C. Tschantz, R. Greenstadt, A. D. Joseph, and J. D. Tygar, "Approaches to adversarial drift," in *Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security (AISec)*. ACM, 2013, pp. 99–110.

[39] A. D. Kent, "Comprehensive, Multi-Source Cyber-Security Events," Los Alamos National Laboratory, 2015.

[40] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[41] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao *et al.*, "Wilds: A benchmark of in-the-wild distribution shifts," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5637–5664.

[42] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *The Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

[43] L. I. Kuncheva, "Change detection in streaming multivariate data using likelihood detectors," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 5, pp. 1175–1180, 2011.

[44] A. Lazaridou, A. Kuncoro, E. Gribovskaya, D. Agrawal, A. Liska, T. Terzi, M. Gimenez, C. de Masson d'Autume, T. Kocisky, S. Ruder *et al.*, "Mind the gap: Assessing temporal generalization in neural language models," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 348–29 363, 2021.

[45] V. Le and H. Zhang, "Log-based anomaly detection with deep learning: How far are we?" *The 44th International Conference on Software Engineering (ICSE)*, 2022.

[46] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," in *International Conference on Learning Representations (ICLR)*, 2018.

[47] X. Li, P. Chen, L. Jing, Z. He, and G. Yu, "Swisslog: Robust and unified deep learning based log anomaly detection for diverse faults," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2020, pp. 92–103.

[48] Z. Lin, J. Shi, D. Pathak, and D. Ramanan, "The clear benchmark: Continual learning on real-world imagery," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[49] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning (ICML)*. PMLR, 2015, pp. 97–105.

[50] M. Ma, S. Zhang, D. Pei, X. Huang, and H. Dai, "Robust and rapid adaption for concept drift in software system anomaly detection," in *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2018, pp. 13–24.

[51] E. Mariconti, L. Onwuzurike, P. Andriotis, E. D. Cristofaro, G. J. Ross, and G. Stringhini, "Mamadroid: Detecting android malware by building markov chains of behavioral models," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2017.

[52] W. Meng, Y. Liu, S. Zhang, F. Zaiter, Y. Zhang, Y. Huang, Z. Yu, Y. Zhang, L. Song, M. Zhang *et al.*, "Logclass: Anomalous log identification and classification with partial labels," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1870–1884, 2021.

[53] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 7 2019, pp. 4739–4745.

[54] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Network and Distributed Systems Security (NDSS) Symposium*, 2018.

[55] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2015, p. 2901–2907.

[56] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proceedings of the 22nd International Conference on Machine Learning (ICML)*. ACM, 2005, p. 625–632.

[57] D. Nigenda, Z. Karnin, M. B. Zafar, R. Ramesha, A. Tan, M. Donini, and K. Kenthapadi, "Amazon sagemaker model monitor: A system for real-time insights into deployed machine learning models," *arXiv preprint arXiv:2111.13657*, 2021.

[58] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *International conference on discovery science*. Springer, 2007, pp. 264–269.

[59] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *37th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2007, pp. 575–584.

[60] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[61] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: eliminating experimental bias in malware classification across space and time," in *28th USENIX Security Symposium (USENIX Security)*. USENIX Association, 2019, pp. 729–746.

[62] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2014, pp. 701–710.

[63] J. Raghuram, D. J. Miller, and G. Kesidis, "Semisupervised domain adaptation for mixture model based classifiers," in *2012 46th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2012, pp. 1–6.

[64] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan, "Likelihood ratios for out-of-distribution detection," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[65] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[66] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 4548–4557.

[67] S. Smeu, E. Burceanu, A. L. Nicolicioiu, and E. Haller, "Env-aware anomaly detection: Ignore style changes, stay true to content!" *arXiv preprint arXiv:2210.03103*, 2022.

[68] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusio detection," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE Computer Society, 2010, pp. 305–316.

[69] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation," in *Proceedings of the first workshop on building analysis datasets and gathering experience returns for security*, 2011, pp. 29–36.

[70] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2001, pp. 377–382.

[71] C. Systems, "Cisco systems netflow services export version 9," RFC 3954, 2004.

[72] R. Tang, Z. Yang, Z. Li, W. Meng, H. Wang, Q. Li, Y. Sun, D. Pei, T. Wei, Y. Xu *et al.*, "Zerowall: Detecting zero-day web attacks through encoder-decoder recurrent neural networks," in *39th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2020, pp. 2479–2488.

[73] L. Tong, B. Li, C. Hajaj, C. Xiao, N. Zhang, and Y. Vorobeychik, "Improving robustness of ml classifiers against realizable evasion at-

tacks using conserved features," in *28th USENIX Security Symposium (USENIX Security)*, 2019, pp. 285–302.

[74] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 7167–7176.

[75] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[76] R. Wang, K. Nie, T. Wang, Y. Yang, and B. Long, "Deep learning for anomaly detection," in *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM)*, 2020, pp. 894–896.

[77] S. Wang, Z. Wang, T. Zhou, H. Sun, X. Yin, D. Han, H. Zhang, X. Shi, and J. Yang, "Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3972–3987, 2022.

[78] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3540–3552, 2020.

[79] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference (WWW)*, 2018, pp. 187–196.

[80] L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh, and G. Wang, "Bodmas: An open dataset for learning based temporal analysis of pe malware," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 78–84.

[81] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "Cade: Detecting and explaining concept drift samples for security applications," in *30th USENIX Security Symposium (USENIX Security)*, 2021.

[82] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially learned anomaly detection," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 727–736.

[83] X. Zhang, Y. Zhang, M. Zhong, D. Ding, Y. Cao, Y. Zhang, M. Zhang, and M. Yang, "Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security (CCS)*, 2020, pp. 757–770.

[84] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li *et al.*, "Robust log-based anomaly detection on unstable log data," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE/ESEC)*, 2019, pp. 807–817.

[85] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Tools and benchmarks for automated log parsing," in *The 41st International Conference on Software Engineering (ICSE)*. IEEE / ACM, 2019, pp. 121–130.

APPENDIX A
SUPPLEMENT OF MODEL CALIBRATION

Calibrator for Abnormal-confidence Models. In §IV-A, we use normal-confidence models to introduce our calibration method and the related definition for the sake of illustration. Without loss of generality, here we provide the supplementary definition of DEFINITION 3 in the case of abnormal-confidence models.

DEFINITION 4 (PERFECT CALIBRATION OF ABNORMAL-CONFIDENCE ANOMALY DETECTION MODELS). The *perfect calibration* for an abnormal-confidence output is defined as TNR among normal samples after setting itself as the detection threshold. Namely,

$$\mathcal{C}(\boldsymbol{f}(\boldsymbol{x})) = TNR_{T=\boldsymbol{f}(\boldsymbol{x})}(\mathcal{X}_N), \quad \forall \boldsymbol{x} \in \mathcal{X}_N. \tag{9}$$

Note that, here the calibrated outputs also represent abnormal probabilities to avoid violating monotonicity. For example, if the calibrated output of a certain normal sample is 0.2, then ideally 20% of normal samples are TNs when the threshold is exactly the output. In other words, 20% of calibrated outputs (anomaly scores) are less than this sample.

Transforming Abnormal Confidence into Normal. In the abnormal-confidence (`KitNET`/NID/APT) case of Fig. 6, we transform the abnormal outputs of original anomaly detection models to normal confidence for better illustration. The approach is straightforward. We firstly normalize the anomaly probabilities (reconstruction errors) into 0 to 1 using the min-max method and then subtract them from 1 (to transform them into normal probabilities).

APPENDIX B
DETAILED ALGORITHM OF SHIFT DETECTION

The procedure for shift detection introduced in §IV-B is presented in Algorithm 1. First, the test statistic of two distributions (calibrated outputs of $\boldsymbol{x}^c$ and $\boldsymbol{x}^t$) are computed on lines 1-2. Then, tested samples are jointed and randomly shuffled, followed by dividing them into two groups and obtaining two distributions (on line 3). The core idea of permutation test is that samples are exchangeable under $\mathcal{H}_0$. Therefore, we compute the p-value as the proportion of permutations where their test statistics are greater than the actual one (on line 4). Optimally, all possible permutations need to be tested but is intractable due to combinatorial explosion. A much less resampling number N_p is set in practice. Finally, we return the p-value p of permutation test indicating the confidence of non-shift. Generally, p is compared with a statistical threshold to determine whether shift or not. In this study, we set this threshold as 0.05 (i.e., $p < 0.05$ indicates shift).

Algorithm 1: Procedure for shift detection

Input: $\boldsymbol{x}^c \in \mathcal{X}_N^c$, $\boldsymbol{x}^t \in \mathcal{X}^t$; K; permutation number N_p
Output: P-value p indicating the probability of non-shift
$\triangledown$ *getting original discrete distributions (histograms)*
1 $P_{org} \leftarrow \mathbb{H}_K\big(\mathcal{C}(\boldsymbol{f}(\boldsymbol{x}^c))\big)$; $\quad Q_{org} \leftarrow \mathbb{H}_K\big(\mathcal{C}(\boldsymbol{f}(\boldsymbol{x}^t))\big)$;
2 $s_{org} \leftarrow \mathcal{D}_{KL}(P_{org}||Q_{org})$; $\quad \triangleright$ *original test statistics*
3 $\{\boldsymbol{P}'_i, Q'_i\}_{i=1}^{N_p} \leftarrow$ Permutating/Resampling and recomputing two histograms ($\mathbb{H}_K$) from $\{\mathcal{C}(\boldsymbol{f}(\boldsymbol{x}^c))\} \cup \{\mathcal{C}(\boldsymbol{f}(\boldsymbol{x}^t))\}$;
4 $p \leftarrow \frac{1+\sum_{i=1}^{N_p} \mathbb{1}[s_{org} \le D_{KL}(P'_i||Q'_i)]}{N_p+1}$; $\quad \triangleright$ *p-value of test*
5 **return** p $\quad \triangleright$ *confidence of non-shift*

APPENDIX C
DETAILS OF EXPERIMENTAL SETUP

This Appendix provides several details of the experimental setup omitted in the main body of §V, including the preprocessing, selection, and split of datasets (in C-1), as well as anomaly detection models (in C-2) in three applications , detailed settings of the baseline approaches for zero-positive anomaly detection (in C-3).

1) Data Selection and Preprocessing: As mentioned in §V-A, we choose real-world long-term datasets[1] to conduct more realistic experiments compared with prior works. As shown in Fig. 3, in addition to the training data collected at the initial moment (@ Time 0), we collect samples N times

[1] Anoshift: https://github.com/bit-ml/AnoShift
BGL: https://doi.org/10.5281/zenodo.1144100
LANL-CMSCSE: https://csr.lanl.gov/data/cyber1/

in chronological order and divide them into validation set and test set. The number of training data (@ Time 0) refers to the original papers of each anomaly detector and is set according to our setup (only normal data here for training the zero-positive anomaly detectors). For example, 50K normal samples for training KitNET in NID case [54].

Split of Validation and Test Sets. The data split of Kyoto 2006+ dataset in NID case refers to Anoshift benchmark [18]. That is, we use traffic in 2007 as Time 0 (@T0) to train anomaly detection model (KitNET) and test the original performance before any shift. For the following 5 years (2011-2015), we treat each of them as a time point (i.e., 2011 data is @T1, 2012 is @T2, so on and so force) and randomly split them into validation and test set. For LogAD and APT cases, we collect samples 10 times according to the data volume and time span of different cases. Since there is no ground truth or intelligence about normality shift in three public datasets, we collect data at the same time interval and set the interval considering the total time span. BGL dataset in LogAD case has 7 months of logs, thus we collect data about every 2-3 weeks. LANL-CMSCSE in APT cases have only 58 days of login logs, thus we collect data every week (5-6 days).

2) Anomaly Detection Models: We have introduced the anomaly detectors used in three applications in §V-A and their technical principles in §II-A. In NID case, we use the original implementation of KitNET[2] in Kitsune which is one of the state-of-the-art DL-based network intrusion detectors. KitNET consists of Autoencoders to conduct reconstruction-based learning (§II-A). In LogAD case, we use the well-known log anomaly detector DeepLog[3]. In APT case, we use GLGV as the anomaly detector proposed in [9]. Similar to [29], we tailor GLGV by replacing the anomaly detector with an Autoencoder as the original work use a supervised classifier in the last step which is not suitable for this study. And we use their implementation[4] which achieves similar performance to the original work. The above anomaly detectors are trained with samples collected @ Time 0 with original configurations introduced in their papers or implementations.

3) Baseline Approaches: For UNLEARN, we obtain the source code from the authors and primarily use the original configurations in the paper [20]. The original work has evaluated the performance of partial hyper-parameters and provides a suggestion of range which is insensitive within it. To conduct a fair comparison, we refer to the method of parameter tuning in the paper (only for the hyper-parameters with reference values provided in the paper, others use the default values), that is, to evaluate the effect of their values within the recommended range and select the best (in most cases). case, we have also confirmed that these ranges are indeed insensitive) Specfically, we set $BND = 10$ in the bounding loss, regularization weight as 10^3. For CADE, we use their public implementation[5] and use additional anomalies in the training set to construct the contrastive learning model to detect samples that are far away from non-shift samples. The tuning of hyper-parameters provided in CADE is similar to that of UNLEARN, so we also use the same method to tun m and λ, and set $m = 10$ and $\lambda = 0.1$ by default. For TRANS, we obtain the source code from their project website[6] and tailor it for zero-positive learning as introduced in §V-A3. It is worth noting that the configuration of TRANS is highly customizable. We chose choice of non-conformity measure (NCM), type of conformal evaluator (CE), and Transcendent thresholds' optimizations as described in the main body of paper according to the characteristics of zero-positive learning. There are also some hyper-parameters while we found that there are not sensitive to the effectiveness, so we used the default value (test rate=1/3 of ICE). TRANS provides an alpha assessment method to evaluate the quality of the configuration, and we also use this method to verify the effectiveness of our configuration. We do not deny that other configurations may have better results (but there is no existing research, we are the first to adapt it to zero-positive anomaly detection, and we leave other configurations and optimization of TRANS for zero-positive learning as future work.

[2]KitNET: https://github.com/ymirsky/Kitsune-py

[3]DeepLog: https://github.com/wuyifan18/DeepLog

[4]GLGV: https://github.com/dongtsi/DeepAID

[5]CADE: https://github.com/whyisyoung/CADE

[6]TRANS: https://s2lab.cs.ucl.ac.uk/projects/transcend/

APPENDIX D
VISUALIZATION OF NORMALITY SHIFT

Here we provide the (relative) frequency histograms constructed with calibrated (and uncalibrated) model outputs in three security applications and real-world deployment to help intuitively observe shift. The histogram is depicted in Fig. 6, where we represent the data @ Time 0 in blue (non-shift) and all subsequent moments in green (possible shift). For NID/LogAD/APT cases, we compare the calibrated (right blue) with uncalibrated (left blue) output distributions @ Time 0. For three public datasets, as analyzed in §V-D, LogAD has the most severe shift, and NID has two significant shifts in 2011 and 2014. APT has the most slight shift relatively. Meanwhile, the drift of APT occurs mostly in the middle of model outputs. That is to say, the shift cannot be obviously reflected in the model performance (since the performance is mainly related to the small or large outputs), which validates the statement in our paper that the normality shift cannot be fully understood only with the model performance degradation. As for the data in real-world deployment of SCADA security systems, We can clearly verify the analysis in §VI from the histograms, that is, Device A has no significant normality shift, Device B is similar to gradual drift, while Device C may have a sudden drift at @Time 1, and no significant drift after that.

APPENDIX E
HYPER-PARAMETER SENSITIVITY AND CONFIGURATION

In this section, we evaluate the sensitivity of hyper-parameters and provide the guideline for hyper-parameter configuration.

1) Hyper-parameter Sensitivity: In §V-C, we evaluate the impact of the hyper-parameters and give recommendations on the range of values. Below, we evaluate the sensitivity of hyper-parameters in OWAD.

Shift Detection. Here we evaluate the resampling number N_p of permutation test in the shift detection. As N_p can only affect the performance of shift detection, we evaluate its impact on P-value, as listed in the last row of Table VIII. The results show that N_p is insensitive to the shift detection in a considerable

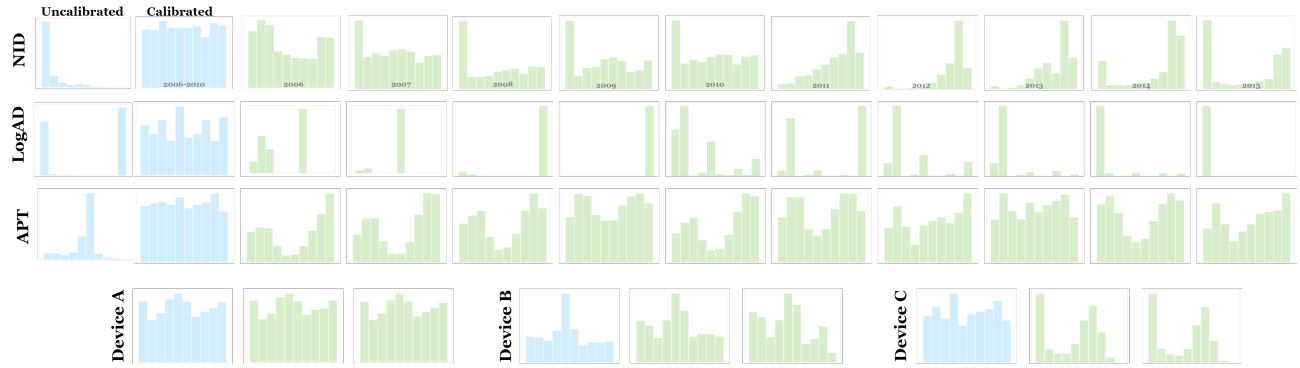


Fig. 6: The histograms of normality shift in three datasets (the first three lines) and real-world deployment (the last line). For each case, the control set (@T0) is shown with blue (the left is uncalibrated, the right is calibrated), while the treatment sets (@T1 to @TN) are shown with green in chronological order from left to right. In NID case, we re-split data in 2006-2015 as @T1-@T10 for better illustration.

TABLE VIII: Sensitivity of hyper-parameters on detection.

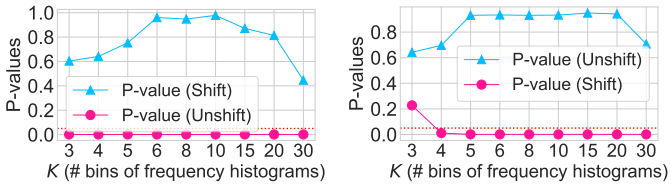
Parameters	Range	P-value of Unshift (Range)	P-value of Shift (Range)
N_p	$[10^3, 10^5]$	[0.502, 0.593]	[0.0000, 0.0000]

TABLE IX: Sensitivity of hyper-parameters on adaptation.

Parameters	Range	AUC of Test @T1 (Range)
λ_1	$[10^{-1}, 10]$	[0.867, 0.872]
λ_2	$[10^{-1}, 10]$	[0.870, 0.874]

range. We also evaluate the effect of K, the number of bins of histograms in shift detection. The exact number of bins is usually a judgment call in statistics. Intuitively, the larger the K, the finer the description of the discrete distribution. Fig. 7 shows the results of the impact of K on shift detection performance. We can observe that a very small K (≤ 3) may lead to inaccurate identification of shift, while a very large K (> 30) may lead to the sensitivity to subtle distributional changes. Nevertheless, results show that a wide range of K (4 to 30) suffices to provide robust and stable shift detection in both cases. We omit the impact on adaptation performance as we find K is extremely insensitive.

Shift Explanation. We evaluate λ_1 (weight of labeling overhead term) and λ_2 (weight of determinism term) in (3). Note that λ_1 and λ_2 here are the values after the three terms are reconciled to the same magnitude. We evaluate their impact on adaptation performance by each time changing one of them while fixing other parameters. Results are shown in Table IX. The results show that all the hyper-parameters are insensitive to adaptation performance in a large range. As for bin number M, we evaluate the ratio M/K in Fig. 8 as M is supposed



(a) NID (b) LogAD

Fig. 7: The impact of K on shift detection (P-values).

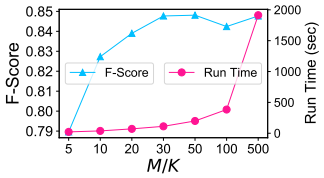


Fig. 8: The impact of M/K.

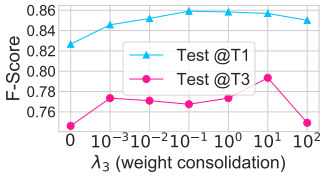


Fig. 9: The impact of λ_3.

to refer to K and $M>>K$ (see §IV-C). Results show that adaptation performance tends to be stable after M/K increases to a certain extent. Considering the linearly increasing run time, M/K can be selected within a moderate interval (which is large enough, 20-100 in this case).

Shift Adaptation We evaluate the impact of λ_3, the weight of parameter consolidation in (7). As shown in Fig. 9, a too small λ_3 may induce old knowledge to be forgotten, while a too large λ_3 will cause new knowledge to be hardly learned. Nevertheless, results shows that a wide range of λ_3 (10^{-3} to 10^1) suffices to improve adaptation performance.

2) Hyper-parameter Configuration: To facilitate deployment, we provide a default configuration of hyper-parameters in OWAD for reference. As for shift detection, we set resampling number $N_p = 10^3$ of permutation test, and the number of bins $K = 5$ in histograms and $M = 100$ for shift explanation. We have verified the insensitivity of them within relatively large ranges. Therefore, a convenient way is to select value from these ranges or directly use the default value. However, these ranges are empirically have not been verified in other security applications. Future work will focus on this. As for three weights λ_1 and λ_2 in (3) and λ_3 in (7), we use an automated method by forcing all weighted terms to be the same for optimization. For example, the three terms $\mathcal{L}_{acc}$, $\mathcal{L}_{lab}$ and $\mathcal{L}_{det}$ are 5, 0.1 and 0.5 before optimization. Here we set $\lambda_1 = 50$ and $\lambda_2 = 10$ to ensure three terms are all 5 before optimization. The intuition is we equally consider each term for optimization. Note that, operators can also adjust the weight ratio according to their requirements by multiple certain constant after normalization. Practical experience also suggests preferentially increasing the weight of accuracy and overhead term. This is because the subsequent binarization operation will weaken the effect of optimization of deterministic term.