Mateen: Adaptive Ensemble Learning for Network Anomaly Detection

Fahad Alotaibi
f.alotaibi21@imperial.ac.uk
Department of Computing, Imperial College London
UK

Sergio Maffeis
sergio.maffeis@imperial.ac.uk
Department of Computing, Imperial College London
UK

ABSTRACT

Anomaly-based intrusion detection systems are tasked with identifying deviations from established benign network behaviors, assuming such deviations to be indicators of malicious intent. Deep AutoEncoders (DAEs) have become increasingly popular in these systems due to their exceptional ability to model benign behavior with high accuracy, particularly in static, offline settings where the network's benign activity pattern is presumed to remain constant. However, this static approach becomes less effective as network behavior naturally evolves, leading to challenges in distinguishing new, benign activities from genuine threats. This evolution raises a critical question: How can we enhance offline DAEs to accurately identify threats while avoiding false alarms caused by benign behavior changes?

To address this question, we propose *Mateen*, an online learning framework designed to augment the capabilities of offline DAEs, enabling them to recognize and adapt to changing benign network behaviors efficiently and with minimal overhead. Mateen leverages an ensemble of DAEs to monitor and adjust to these changes. It optimizes resource usage by selecting only a few representative samples for updates and reduces the overall framework's complexity by retaining only the relevant models.

We evaluate the effectiveness of Mateen on five network intrusion datasets, each exhibiting different types of benign behavior evolution. The results demonstrate that Mateen consistently enhances offline DAE performance across various evolution types. For instance, Mateen boosts the F1-score on the IDS17 dataset, which exhibits light change, by 4.13%, and on the Kitsune dataset, characterized by heavy change, by 72.6%, while only necessitating labeling for 1% of the incoming samples.

CCS CONCEPTS

• **Security and privacy** → **Intrusion detection systems**; • **Computing methodologies** → *Neural networks*; Unsupervised learning settings.

KEYWORDS

Network Security, NIDS, Shift Detection and Adaptation

RAID 2024, September 30–October 02, 2024, Padua, Italy

ACM ISBN 979-8-4007-0959-3/24/09
https://doi.org/10.1145/3678890.3678901

ACM Reference Format:
Fahad Alotaibi and Sergio Maffeis. 2024. Mateen: Adaptive Ensemble Learning for Network Anomaly Detection. In *The 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2024), September 30–October 02, 2024, Padua, Italy*. ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3678890.3678901

1 INTRODUCTION

Securing computer networks requires balancing security with usability, a complex task managed by various strategies. One such strategy is using Network Intrusion Detection Systems (NIDS), based on whitelisting principles, which flag deviations from predefined rules as suspicious. However, formulating these rules is resource-intensive and lacks scalability, as it necessitates manual analysis of the network traffic's benign behavior. Fortunately, breakthroughs in unsupervised machine learning, notably with DAEs, offer a scalable alternative by automating benign traffic modeling. DAEs refine network data analysis through compression and reconstruction, effortlessly learning benign behavior patterns, thus easing the deployment of whitelisting-based NIDS across diverse networks. Recent research into whitelisting-based security frameworks, often termed one-class anomaly detection, predominantly harnesses DAEs, leading to substantial enhancements in detection performance [17, 27, 40, 42, 60, 73, 77, 85, 86, 91].

However, these promising results arise from an offline context in which models are fixed post-training, predicated on the belief that the distribution of testing data will mirror that of training data. This premise falls short in security scenarios where defense applications are deployed in dynamic, hostile environments [7, 16, 39]. Here, they confront testing data potentially at odds with the training data, giving rise to what is known as distribution shifts [7, 8, 81, 90]. Such shifts can stem from various factors, including changes in adversary tactics or in legitimate user behaviors. In this context, offline DAEs are naturally less sensitive to changes in malicious traffic patterns because their training is based exclusively on benign traffic samples [39]. However, DAEs may suffer when faced with shifts in benign traffic distribution, henceforth referred to as *benign shift*, since this no longer matches the training set distribution. As a result, an offline DAE model may begin to mistakenly flag instances of benign shift as malicious, thereby becoming outdated.

Updating an outdated DAE presents three main challenges. The first challenge is detecting benign shifts in a whitelisting setting. In this context, the DAE, trained exclusively on benign instances (Figure 1a), interprets deviations as potential malicious behaviors. This introduces complexity, as illustrated in Figure 1b, where deviations may indicate malicious activities, benign shifts, or a combination of both. The second challenge is selecting from the shifted distribution a small set of representative samples, to be labelled manually

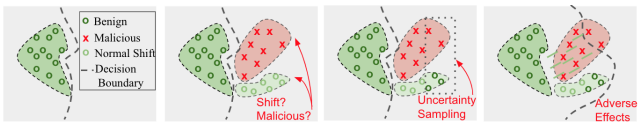


Figure 1: Key challenges in adapting to benign shifts.

and used for updating the DAE. Figure 1d illustrates the potential adverse effects of updating the model with non-representative samples. Uncertain samples located far from the decision boundary fail to capture the entire distribution of the benign shifts (Figure 1c). A model updated with such samples might not effectively capture the full distribution, risking the misclassification of malicious samples as benign (Figure 1d), or incorrectly identifying benign shift samples, especially those significantly different from the ones chosen for manual labelling, as malicious. The third challenge lies in effectively adapting the DAE to the shifted distribution without losing knowledge of the old distribution, given that only a limited number of samples will be selected and labeled.

In response to the abovementioned challenges, this paper introduces *Mateen*, a framework designed for DAEs-based NIDS to detect and adapt to intricate and varied changes in benign traffic patterns. Mateen addresses the first challenge by employing a statistical analysis method to detect shift that demonstrates lower sensitivity to the distribution's tail. This approach allows Mateen to focus on deviations near the center of the distribution, where benign shifts are likely to occur. To address the second challenge, Mateen leverages a new sample selection technique designed to construct a compact coreset from the new data distribution in an unsupervised fashion. This coreset is carefully selected to encapsulate the entirety of the new distribution, thereby significantly enhancing the model's capacity to generalize across it. To address the third challenge, Mateen implements an ensemble of DAEs updated collaboratively and selectively to adapt to shifts. In this process, a long-lived DAE is updated with newly selected benign samples and historical data. Concurrently, a temporary DAE, derived from the long-lived DAE, is trained exclusively on the new benign samples. By doing so, Mateen ensures that the ensemble models remain diverse and relevant to both historical and recently captured distributions. Finally, Mateen keeps the ensemble as compact as possible by merging high-performing DAE(s) and discarding underperforming ones. In this way, Mateen streamlines ensemble complexity without sacrificing performance.

We carried out a comprehensive evaluation of Mateen, using five NIDS datasets—namely CICIDS2017 (IDS17) [29, 89], CSE-CIC-IDS2018 (IDS18) [64, 89], the IoT-focused Kitsune dataset [73], and two variants of Kitsune—under various shift conditions. [1] We compared Mateen against recent state-of-the-art methods in NIDS distribution shift management: INSOMNIA [4] for supervised learning adaptation, CADE [101] for out-of-distribution detection and sample selection, and OWAD [39] for unsupervised detection and adaptation to benign shifts. The results reveal that Mateen consistently outperforms these methods across a range of detection metrics, effectively handling various types of shifts. Notably, Mateen achieves AUC-ROC scores of 96.79% for IDS17, 98.17% for IDS18, and 94.76% for a modified version of Kitsune (mKitsune), exceeding the high-performing baseline by 2.23%, 4.36%, and 14.78% respectively. We also conducted an ablation study to analyze the contributions of each sub-component within Mateen, in addition to a hyperparameter sensitivity analysis to assess how different configurations affect its performance. Our results reveal that every element within Mateen plays a crucial role in its overall effectiveness. Furthermore, Mateen demonstrates robust performance, maintaining high levels of effectiveness even when operating with sub-optimal hyperparameters.

In summary, the contributions of this paper are:

- We designed a new adaptive ensemble learning approach, explicitly crafted to address the challenge of various forms of benign shift effectively.
- We enhanced the adaptive ensemble learning approach with three techniques focused on benign shift detection, sample selection, and ensemble complexity reduction, each contributing to a more robust and effective system.
- We incorporated these components into *Mateen*, a unified framework designed to allow an offline DAE to detect and adapt to benign shifts at the lowest possible cost. We have made the code publicly available[2], facilitating future research in this field.
- To the best of our knowledge, we provide the first evaluation of drift-aware NIDS on different scenarios of benign shift. We demonstrate the ability of Mateen to adapt to varying distributions, and explore the impact of its key components.

2 PRELIMINARIES

In this section, we provide an overview of anomaly detection and distribution shift for computer networks, coupled with a concise summary of ensemble learning and sample selection techniques.

2.1 Anomaly Detection

Anomaly Detection (AD) is the process of finding deviations from a defined pattern. The pattern can be only benign behaviors where the aim is to detect malicious ones (one-class AD), as seen in whitelisting-based NIDS [39, 73, 95], or can be a combination of both benign and malicious behaviors where the aim is to identify emerging malicious behaviors (multi-class AD), which is common in the detection of new types of network attacks [16, 18, 78, 101].

Reconstruction-based methods are the prevalent approach for one-class AD, typically involving the training of a DAE model, to minimize the reconstruction error of benign instances. The underlying rationale is that the model, honed to accurately reconstruct benign instances, will exhibit increased reconstruction errors when encountering malicious data [39, 73, 95]. Distance-based methods are the primary approach for multi-class AD, with contrastive learning-based strategies being especially prevalent

[1] Operational network data with a meaningful number of security events is rarely available for public research. Our use of synthetic datasets is consistent with recent work on NIDS [4, 22, 33, 36, 42, 56, 59] that uses exclusively synthetic datasets, most frequently IDS17 and IDS18.

[2] https://github.com/ICL-ml4csec/Mateen/

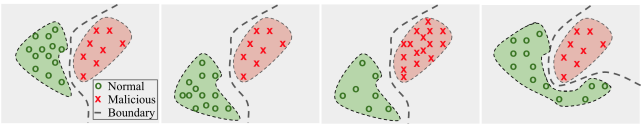


Figure 2: Distribution shift examples: (a) Training Distribution - Data on which the model was initially trained. (b) Covariate Shift - benign data distribution changes, but the decision boundary remains the same. (c) Ratio Shift - Reduction in benign data and increase in malicious data. (d) Concept Shift - a new concept that changes the decision boundary.

[16, 18, 23, 55, 101]. These strategies minimize the distance between samples of the same class while maximizing the separation between distinct classes. The core concept is that newly emerging malicious samples, such as those from new classes of attacks, will exhibit a significant distance from the samples in the established training classes.

2.2 Deep AutoEncoders

A Deep AutoEncoder (DAE) [9] is fundamentally composed of two interconnected multi-layer perceptrons (MLPs). The first part, the encoder, is responsible for mapping input data into a latent space, representing a compressed version of the data. The second part, the decoder, aims to reconstruct the original data from these latent representations. The optimization of the DAE focuses on minimizing the reconstruction error, a measure of the difference between the original data and its reconstructed form. Commonly used loss functions for this purpose include Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). In security applications, DAEs are frequently employed for AD, particularly for identifying security incidents in a one-class fashion where the model is trained exclusively on benign data. For example, OWAD [39] utilizes a single DAE to identify malicious activities, Kitsune [73] uses an ensemble of DAEs to detect IoT network attacks, and Enidrift [95] employs a dynamic pool of DAEs for detecting network attacks.

2.3 Distribution Shift

Distribution shift refers to the change in the joint probability distribution $P(x, y)$, where $x \in X$ and $y \in Y$, between the training and testing phases [35, 66, 75]. This means that the probability distribution during training, denoted as $P_{\text{train}}(x, y)$, differs from the distribution during testing, $P_{\text{test}}(x, y)$. The shift can be categorized into *covariate shift*, *ratio shift*, and *concept shift* (see Figure 2).

Covariate Shift occurs when there is a change in the probability distribution of the input variables, $p(x)$, but the conditional probability distribution $P(y|x)$ remains the same. For example, in NIDS that are trained on packet payloads to identify attacks, a covariate shift might occur when attackers change these payloads to achieve the same objective, such as employing alternative syntax or hex encoding in SQL injection attacks. This leads to a distribution shift, illustrated in Figure 2b, without altering the decision boundary. Mathematically, it is represented as a change in $P_{\text{train}}(x) \neq P_{\text{test}}(x)$, while $P_{\text{train}}(y|x) = P_{\text{test}}(y|x)$.

Ratio Shift is the converse of covariate shift, where the change occurs in the probability distribution of the output variable, $p(y)$, while the conditional probability distribution $P(x|y)$ remains the same. An example is shown in Figure 2c, demonstrating a change in the ratio of malicious to benign instances, such as a rise in Denial of Service (DoS) attacks during inference. It is described as $P_{\text{train}}(y) \neq P_{\text{test}}(y)$, but $P_{\text{train}}(x|y) = P_{\text{test}}(x|y)$.

Concept Shift represents a shift in the joint probability distribution $P(x, y)$, leading to modifications in the decision boundary, as depicted in Figure 2d. This shift may arise from the introduction of new classes during inference. In binary classifications, if such new classes are correctly classified (for example, recognizing a new attack as malicious), the model's performance may not significantly deteriorate [10, 45]. Conversely, in multi-class settings, identifying, labeling, and adjusting the classifier to accommodate the new class becomes essential [101]. Concept shift is thus a change where $P_{\text{train}}(x, y) \neq P_{\text{test}}(x, y)$ in a more general sense.

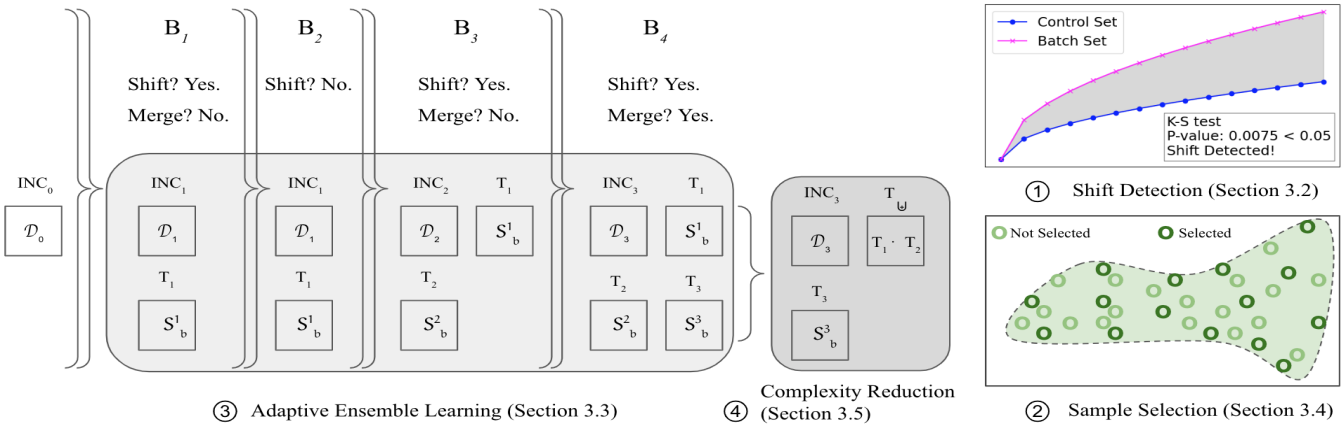
These types can also be classified by their temporal patterns, including *gradual shifts*, where changes occur progressively over time; *sudden shifts*, where the distribution changes abruptly and unpredictably; and *recurring shifts*, characterized by brief, repetitive changes throughout time [35, 66, 75].

2.4 Ensemble Learning

Ensemble learning refers to the use of multiple base models, often referred to as *learners*, to form one predictive model. The one predictive model is derived by aggregating the outputs of individual learners through methods like weighted voting, majority voting, or selecting the best performer. Typically, these learners are developed in an offline setting, trained once on a substantial dataset, and then deployed without subsequent updates. However, in environments characterized by frequent distribution shifts, such static learners quickly become outdated. Adaptive ensemble learning addresses this issue by updating the learners to adapt to new data distributions. Updates can be implemented in various ways, including incremental updates (enriching the training set with data from the new distribution before retraining) [3, 48] or through batch-based learning, where a new learner is trained for each data batch that meets certain criteria [25, 28, 68, 95, 99, 103], such as evidence of distribution shifts [99, 103] or a decline in detection performance [25, 28]. The new learner is added to a model pool. Subsequently, this pool leverages techniques such as weighted voting to predict incoming data [68, 95, 99, 103].

2.5 Sample Selection

Sample selection consists of selecting particular examples from a dataset to boost a model's accuracy, and is applicable to both labeled and unlabeled datasets. For labeled datasets, sample selection aims to identify a subset of samples that trains a model to match or exceed the performance of one trained on the full dataset, similarly to coreset selection or training-set reduction [5, 19, 50, 51, 62, 74, 102]. To achieve this, the selected subset must accurately represent the full dataset characteristics. With unlabeled datasets, the focus is on selecting data points that, once labeled, significantly enhance



Dataset	Description	Model	Description
B_i	The incoming batch of data at step i.	INC_i	The *incremental model* trained on $\mathcal{D}_i$.
$\mathcal{S}_b^i$	Samples selected from B_i and manually labelled as *benign*.	T_i	A *temporary model* exclusively trained on $\mathcal{S}_b^i$.
$\mathcal{D}_i$	Incremental dataset at step i: $\mathcal{D}_i = \mathcal{D}_{i-1} \cup \mathcal{S}_b^i$.	$T_\uplus$	The *merged model* averaging T_1 and T_2.

Figure 3: Overview of Mateen.

model performance, balancing informativeness, representativeness, and diversity [84]. This involves evaluating the inherent qualities of data or using the model feedback on these points [4, 16, 84].

Active learning, also often associated with selection from unlabeled datasets, involves querying a pool of unlabeled samples to identify those that require labeling [84, 87]. This selection process can occur once or iteratively on an n-by-n basis, where n represents either one sample or a batch of samples. This iterative approach integrates model feedback at each step, allowing for continuous refinement of the selection process. The most common active learning technique in security applications is uncertainty sampling [4, 16, 101]. It focuses on choosing samples that present the highest degree of uncertainty to the model. Such uncertainty is typically derived from various metrics, including low probability scores [4], low fitness scores (e.g., high distance to established training classes) [16, 101], and high reconstruction errors [39].

3 THE MATEEN FRAMEWORK

This section presents Mateen, a framework that detects and adapts to benign shifts in DAE-based NIDS. Mateen comprises four submodules: the shift detection module, responsible for detecting benign shifts; the sample selection module, which selects relevant samples for labeling and updates; the adaptive ensemble learning module, responsible for shift adaptation; and a complexity reduction module, designed to decrease Mateen's overall complexity.

3.1 Overview and Simplified Example

Figure 3 illustrates the high-level workflow of Mateen. Mateen operates in an online manner, predicting incoming samples as they arrive and adapting to benign shifts. Mateen classifies incoming samples by using the model that showed the best performance during the previous update round. Simultaneously, these samples are aggregated into a count-based batch B_i. When B_i reaches full capacity, Mateen performs hypothesis testing to determine whether a shift has occurred statistically, checking if the distribution of B_i aligns with that of the previous training data $\mathcal{D}_{i-1}$ (Figure 3, ①). If the distributions are aligned, no update is needed. If instead a distribution shift is detected, Mateen selects a stratified subset from B_i that accurately represents the overall distribution, as illustrated in Figure 3 ②, for annotation by human experts. This labeled subset is then employed to update the models.

Mateen performs two types of updates on its ensemble of models: a major update and a minor update (Figure 3, ③). In the major update, a new temporary model, T_i, is created and trained on the benign samples from the labeled subset, denoted as $\mathcal{S}_b^i$. In the minor update, Mateen augments the training data to include $\mathcal{S}_b^i$, resulting in $\mathcal{D}_i$, and retrains the previous incremental model, INC_{i-1}, on $\mathcal{D}_i$, to produce INC_i. Finally, Mateen optimizes the complexity of the ensemble by enforcing a maximum size limit (Figure 3, ④). Upon reaching this limit, high-performing models are combined in a merged model $T_\uplus$, and low-performing ones are discarded.

3.2 Shift Detection

The Mateen shift detection module is designed to identify occurrences of benign shift. In contrast to existing methods that operate in a supervised setting—where an instance is compared against both malicious and benign baselines [16, 55, 101]—Mateen functions within a one-class AD framework. Hence, we need to identify shift occurrences without relying on a direct comparison to a malicious baseline. Our strategy is to compare the distribution of two sets of equal size: the incoming batch of data B_i and a control set $\mathcal{D}_c$. The control set comprises the most recent samples from $\mathcal{D}_{i-1}$, ensuring it has the same number of samples as B_i. However, it is hard to derive a strong distribution comparison using the sample features, because of the high dimensionality. Instead, we leverage the current DAE (the highest-performing model from the most

recent update cycle) to map each sample to a single dimension. We calculate the MSE score for each sample x_i as follows:

$$\text{MSE} = \frac{1}{d}\sum_{j=1}^{d}(x_{ij} - \text{DAE}(x_i)_j)^2 \quad (1)$$

With the one-dimensional scores for each set in hand, our objective is to statistically compare their distributions. The null hypothesis is that $\mathcal{D}_c$ and B_i originate from the same distribution, indicating no shift. Note that, $\mathcal{D}_c$ is presumed to be benign, as it comprises instances that have been collected, labeled, and filtered to ensure they represent only benign behavior. In contrast, we cannot assert that B_i contains only benign instances. Indeed, if B_i comprises a mix of malicious and non-shifted benign instances (i.e., benign samples drawn from the same distribution as $\mathcal{D}_c$), the comparative analysis of the distributions for $\mathcal{D}_c$ and B_i might reveal a significant divergence, indicating that shift has occurred.

To compare distributions, we use the two-sample Kolmogorov-Smirnov (KS) test. Although it has been criticized for being less sensitive to outlier values at the tails of a distribution, such as extreme values [31, 70, 71], this characteristic is actually advantageous in our specific context. Typically, malicious instances exhibit high MSE scores, but occur less frequently than benign instances. Thus, the KS test's lower sensitivity to such extreme values allows for an equitable comparison across the full spectrum of both distributions, ensuring that rare, extreme malicious samples do not bias the results.

Let $F_{\mathcal{D}_c}(r)$ and $F_{B_i}(r)$ represent the empirical cumulative distribution functions (ECDFs) corresponding to the MSE scores derived from the control set $\mathcal{D}_c$ and the incoming batch B_i, respectively, where r denotes the MSE score values for which the ECDFs are calculated. We calculate the KS test as follows:

$$KS_{\mathcal{D}_c,B_i} = \sup_{r\in\mathbb{R}} \left|F_{\mathcal{D}_c}(r) - F_{B_i}(r)\right| \quad (2)$$

The term $\sup_{r\in\mathbb{R}}$ indicates the supremum, or the maximum deviation, between the two ECDFs across all possible values of MSE scores observed in both $\mathcal{D}_c$ and B_i. The resulting p-value from the KS test is then compared against the conventional significance threshold of 0.05. In statistical terms, a p-value below this threshold typically suggests a significant difference [24, 46, 63, 79, 104], hence supporting the alternative hypothesis that a distributional shift has occurred. Conversely, a p-value above this threshold would support the null hypothesis, indicating no significant shift between the distributions. Analysts may adjust the threshold value to better reflect the malicious-to-benign ratio in the specific network being secured, thus tailoring the sensitivity of the test to the context of their network. For instance, by setting a higher significance threshold (e.g. 0.1), the test becomes more permissive, thereby increasing its sensitivity to smaller deviations between distributions.

3.3 Adaptive Ensemble Learning

To adapt to distribution shifts, Mateen employs adaptive ensemble learning, a method that generates new models to encompass emerging distributions. While various approaches have been devised to apply this concept across different scenarios, they often fall short in the context of DAE-based NIDS for a crucial reason.

Commonly, these approaches generate a new model using the entirety of the incoming data batch [25, 28, 68, 95, 99, 103]. This batch may be processed in two ways: either by being labeled and cleansed of malicious instances before training [68, 95], or by allowing the model to update itself in an unsupervised manner without any prior filtering [99, 103]. The challenge with the former is the impracticality of labeling every piece of incoming data, while the latter poses a risk of self-poisoning, potentially causing the model to incorrectly classify malicious activity as benign. Furthermore, limiting the model's training to a small set of labeled benign samples can cause overfitting, where the model fails to generalize well beyond the specific instances it was trained on. Our approach, detailed below and illustrated in Figure 4, mitigates these issues.

If data batch B_i exhibits a distribution shift, we employ the selection method outlined in Section 3.4 to select a subset $\mathcal{S}_i$ of representative samples for manual labeling. $\mathcal{S}_i$ is then used to assess the need for updating the ensemble. If the model's performance on $\mathcal{S}_i$, measured using the F1 score, surpasses a predefined threshold t, no update is necessary[3]. Otherwise, the samples from $\mathcal{S}_i$ are filtered to eliminate instances of attacks, obtaining the benign-only subset $\mathcal{S}_b^i$, which is utilized for two types of updates: minor and major.

For the minor update, we augment the benign data accumulated so far with $\mathcal{S}_b^i$, resulting in an updated dataset $\mathcal{D}_i = \mathcal{D}_{i-1} \cup \mathcal{S}_b^i$. We then update INC_{i-1} using $\mathcal{D}_i$ to obtain INC_i (Figure 4, ③). This incremental model is capable of adapting to minor distribution shifts, such as covariate shifts, because it continuously learns from a growing dataset and only needs to adjust to changes within the feature space. For instance, if a user's browsing behavior changes, INC_i can reassess the importance of the features to establish a more effective decision boundary.

For the major update, we clone INC_{i-1} and fine-tune it on $\mathcal{S}_b^i$ to obtain the new temporary model T_i, which we add to the ensemble (see ① and ② in Figure 4). Leveraging the weights from the incremental model, instead of initializing T_i with random weights, allows INC_{i-1} to transfer its deep understanding of benign traffic patterns to T_i, avoiding the pitfall of learning from scratch, which can lead to overfitting. This strategy presents multiple advantages. It prepares the temporary models to efficiently manage new data distributions, or concept shifts, despite having limited data for training. This capability starkly contrasts with the incremental model, which, due to empirical risk minimization principles, may display a bias towards the dominant patterns within the training data. Moreover, not every statistical representation benefits from fine-tuning, leaving some statistics unchanged and thus still reflective of prior distributions. As a result, the temporary models, even after adaptation, are expected to make fewer mistakes on previous distributions compared to models that begin with random weights.

After the updates, we select the model that achieves the highest F1 score on the $\mathcal{S}_i$ labeled samples for use as the reference model in the next prediction and shift detection cycle.

3.4 Selection Method

The adaptive ensemble learning module necessitates a labeled dataset for its update step. Due to the impracticality of labeling

[3]We set t at 99%, as this level of performance is typically observed when evaluating the model on network data from the same distribution (e.g., [1, 42, 85]).

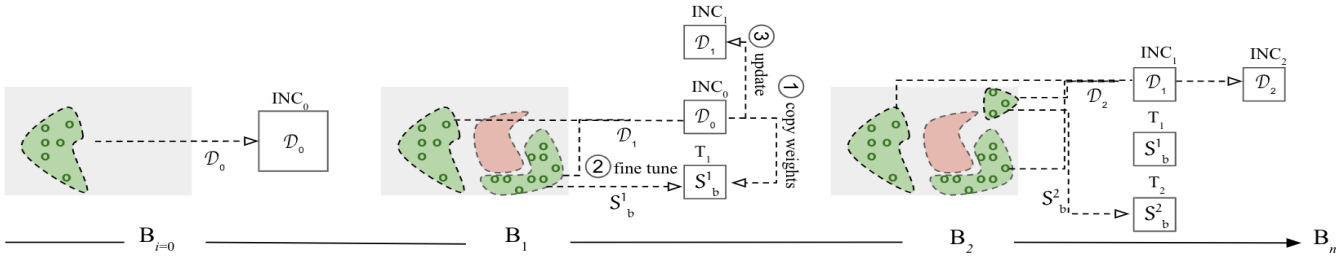


Figure 4: Mateen's adaptive ensemble learning module.

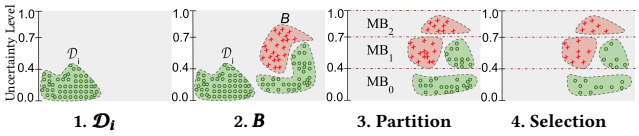


Figure 5: The proposed sample selection method.

an entire batch B, we need a method for selecting a representative subset $\mathcal{S}$ from B. For the update to be effective, this $\mathcal{S}$ needs to satisfy three key criteria: it must encompass samples from every segment of B's distribution, ensuring a comprehensive representation; the selected samples need to be highly informative to enhance the learning process; and the sample set must display adequate diversity to reduce any potential bias towards overly specific patterns in the data.

To construct $\mathcal{S}$, Mateen introduces a new selection method tailored to fulfill these requirements. The intuition is illustrated in Figure 5 and the procedure is sketched in Algorithm 1. As an initial step, we partition B into mini-batches MB_n of size ρ (step ③ in Figure 5). Specifically, we first calculate the MSE score for every individual sample in B. Following this, we arrange the samples in ascending order according to their MSE scores. Then, we distribute these ordered samples into the mini-batches MB_n in a sequential manner. By selecting samples from each mini-batch, we ensure comprehensive representation across the entire distribution of B (Figure 5, ④).

To perform selection from each mini-batch, we compute two key metrics: informativeness ($\hat{I}$) and uniqueness ($\hat{U}$), as outlined in Algorithm 1. Informativeness is designed to measure a sample's value to the overall learning process, indicating that a highly informative sample is expected to significantly decrease the model's error. Uniqueness assesses how distinct a sample is relative to others, with a unique sample potentially introducing novel and less common patterns from B. In the rest of this Section, we detail our approach to calculate $\hat{I}$ and $\hat{U}$, and the criteria for selecting samples based on these scores.

3.4.1 Informativeness. This metric is designed to meet the second criterion: the identification of informative samples that contribute to a reduction in model error. Since we lack access to the actual labels of the entire batch, we cannot directly measure model error. Instead, we assess the informativeness of samples through a distance-based measure. The underlying assumption here is that

Algorithm 1 Pseudocode for Mateen's selection method.

Input: a batch of data B, a model M, the size of minibatch ρ, the retention rate σ, and the selection rate δ.
Output: The subset $\mathcal{S}$ of B to be labeled.

function MateenSelector(B, M, ρ, σ, δ)
  ▷ Initialise $\mathcal{S}$
  $\mathcal{S} \leftarrow$ empty list
  ▷ Distribute B into bins of size ρ
  $IDX_{batch} \leftarrow$ DataToBins(M, B, ρ)
  ▷ Iterate over each bin index j in IDX_{batch}
  for $j \in IDX_{batch}$ **do**
    ▷ Get the minibatch MB_j from B using the index j
    $MB_j \leftarrow B[j]$
    ▷ Calculate the number of samples to select from MB_j
    $N \leftarrow \lfloor \delta \times \rho \rfloor$
    ▷ Get MB_j^- of MB_j based on a retention rate σ
    ▷ Calculate informativeness scores I for samples in MB_j^-
    $MB_j^-, I \leftarrow$ Informativeness(M, MB_j, σ)
    ▷ Calculate uniqueness scores U for samples in MB_j^-
    $U \leftarrow$ Uniqueness(M, MB_j^-)
    ▷ Normalize I and U using min-max scaling
    $\hat{I} \leftarrow \frac{I - \min(I)}{\max(I) - \min(I)}$
    $\hat{U} \leftarrow \frac{U - \min(U)}{\max(U) - \min(U)}$
    ▷ Calculate the combined weight W using $\hat{U}$ and $\hat{I}$
    $W \leftarrow \lambda_0 \cdot \hat{U} + \lambda_1 \cdot \hat{I}$
    ▷ Selects the samples with the highest values in W
    $IDX_{top} \leftarrow \{i \mid W[i] \in$ top N highest values in $W\}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{MB_j^-[i] \mid i \in IDX_{top}\}$
  end for
  return $\mathcal{S}$
end function

samples that are more similar to a larger group of other samples are more informative because they represent a greater portion of the data. However, relying solely on similarity can be problematic as it is sensitive to noise and may lead to over-selecting samples from similar patterns, thus missing out on diverse, yet informative samples from other patterns.

To address this, we use a selective method to ensure a broad representation of informative samples. We iteratively select the first sample from each distinct pattern, based on its temporal order,

while removing samples that are too similar to it beyond a specific threshold. Consequently, a sample is deemed informative if its selection results in the removal of many other similar samples.

Setting the threshold can be challenging due to the variability in similarity levels across mini-batches. Therefore, we employ a binary search algorithm to determine the threshold that allows us to select a specific percentage, σ, of samples from each mini-batch MB_j, thereby creating a condensed mini-batch MB_j^-. The choice of σ is flexible, tailored to the user's needs, and can be adjusted based on the desired balance between redundancy and the degree of similarity among the data in the given network.

For a sample s, let $\phi(s)$ be its reconstruction via $DAE(s)$, and let $\epsilon(s)$ be its feature error vector, where each component $\epsilon_j(s)$ is computed as $||s_j - \text{DAE}(s)_j||^2$ for each feature index j. Given two samples s_1 and s_2, our similarity measure is the Euclidean distance

$$||\phi(s_1)::\epsilon(s_1) - \phi(s_2)::\epsilon(s_2)||^2 \tag{3}$$

where :: denotes vector concatenation.

We include the feature error vector ϵ term because of the dynamics of our ensemble update process, where we repeatedly train our incremental model (INC) on a growing, large dataset, and train the temporary models (T) on small sets of samples. This could lead to the latent representations, or the reconstructions ϕ, of different inputs to be too similar to each other [20, 82]. In such cases, ϵ serves as a critical measure, capturing the genuine discrepancy between the feature space of a sample, s, and its reconstructed form via $\text{DAE}(s)$.

3.4.2 Uniqueness. This metric is crafted to fulfill the third criterion: identifying a diverse set of samples to ensure that the model trained on this set is not biased towards specific data patterns. Given the samples within MB_j^-, we compute a *uniqueness* value, u, for each sample, s, as follows:

$$u(s, MB_j^-) = \sum_{s' \in MB_j^- \setminus \{s\}} ||\epsilon(s) - \epsilon(s')|| \tag{4}$$

Here, u represents the cumulative distance of sample s to all other samples in MB_j^-. A higher u score indicates that the sample is distinct compared to others in the batch.

3.4.3 Selection. We derive a vector U, where each element corresponds to the u value for each s within MB_j^-. Similarly, we compute a vector I of *informativeness values*. Each value in I indicates the number of samples excluded by INFORMATIVENESS due to their similarity or redundancy with the corresponding sample in MB_j^-.

Both vectors U and I are normalized using min-max scaling to obtain $\hat{U}$ and $\hat{I}$. The final weights for each sample in a condensed mini-batch are calculated using the formula $\lambda_0\hat{U} + \lambda_1\hat{I}$, where λ_0 and λ_1 are hyperparameters that adjust the balance between uniqueness and informativeness. Samples with the highest weights from each minibatch are chosen based on a user-specified selection rate (δ), resulting in a subset $\mathcal{S}$ that represents the entire distribution of B.

3.5 Complexity Reduction

The proposed adaptive learning module in Mateen faces a key challenge: as time progresses, the number of models in the ensemble increases. On the one hand, discarding models risks losing valuable insights; on the other hand, retaining all models leads to increased complexity. Hence, a balanced approach is necessary. We, therefore, propose a dynamic scheme that balances the retention of valuable knowledge against the escalating complexity through selective model merging and elimination.

Suppose the number of the models reaches a critical number. In that case, we reduce the models' number to 3, keeping the incremental model INC_i, the temporary model T_i (trained on the most recent data $\mathcal{S}_b^i$), and adding a new merged model $T_\uplus$.

To obtain $T_\uplus$, we start by evaluating the F1 scores of the remaining temporary models on the latest subset $\mathcal{S}_i$, arranging these models in descending order based on their scores $(T^0, \ldots, T^k)$. We then proceed with a recursive merging process for those $n \leq k$ models whose performance is within 10% of the highest-scoring model, denoted T^0. This emphasis on F1 scores is due to the likelihood that models with similar performance metrics share akin parameter spaces and exhibit similar detection errors, rendering them nearly equivalent [76]. Thus, merging these models into a unified global model is expected to maintain overall performance on distributions learned from its constituent models and decrease redundancy within the pool [61, 72, 103].

The merging process employs the parameter averaging technique [72, 103], but adds a weighting mechanism. The merger is formalized by the recursive equations:

$$\begin{aligned} T_\uplus^0 &= T^0, \\ T_\uplus^{j+1} &= \alpha \cdot T_\uplus^j + (1-\alpha) \cdot T^{j+1}, \quad \text{for } j = 0, 1, \ldots, n-1 \end{aligned} \tag{5}$$

Here, α is the weighting coefficient, optimized to refine $T_\uplus$'s performance on the recent subset $\mathcal{S}_i$, ensuring the global model effectively encapsulates the strengths of its components while being specifically attuned to the latest distribution. To find the best α, we employ a direct empirical method: testing values from 0 to 1 in 0.01 increments. For each value, we calculate the F1 score against the subset $\mathcal{S}_i$. The α that delivers the highest F1 score is selected.

The models which fall outside the 10% performance margin are eliminated. While this might be perceived as a loss of valuable knowledge, the incremental model INC_i still retains insights gained from data corresponding to the removed models.

4 EVALUATION

We evaluated Mateen along with the related baselines using five network datasets in various distribution shift scenarios. In the following, we detail our experimental setup and discuss our results.

4.1 Experimental Setup

4.1.1 Datasets. Our dataset selection was guided by two key criteria: they must have been captured over time, and encompass a variety of both attack and benign sources. We chose three well-known datasets: Kitsune [73], IDS17 [29, 89], and IDS18 [64, 89]. The Kitsune dataset contains nine attack-specific files, each captured independently alongside its benign traffic. When a model is trained on a single file and tested on the others, its performance significantly deteriorates. This occurs because each file is designed for a distinct attack scenario and captured in a unique environment, resulting in concept shifts within both attack and benign activities. The IDS17 dataset features a mix of both benign and malicious

traffic gathered from the same network over five days. The IDS18 dataset provides a mix of traffic captured over three weeks from a larger network. IDS17 and IDS18 exhibit covariate shifts in benign traffic, marked by gradual and subtle changes in traffic properties over time, and concept shifts, defined by the sudden appearance of new attack patterns as time evolves.

Data Split. We divided each dataset into training and testing sets, adopting time-aware settings to mitigate bias, as recommended in [6, 81]. The Kitsune dataset lacks timestamps, so we used the first file (ARP MitM) for training, and the remaining files for testing. For IDS17, we used the initial two days as the training set, and the following three days for testing. For IDS18, we used the first week for training and the following week for testing. The testing sets were then sequentially divided into batches, each comprising 50,000 samples, as in [4, 39]. The benign-to-malicious ratio changes across batches, hence we can say that each dataset also demonstrates a ratio shift.

Shift Scenarios. The testing sets illustrate two types of benign shift: covariate and ratio shifts, as seen in IDS17 and IDS18, and concept and ratio shifts, as seen in Kitsune. We also created additional simulated shift scenarios. From IDS18 (covariate shift), we removed the effect of ratio shift to focus solely on analyzing the impact of covariate shift. Specifically, we first calculate the benign-to-malicious sample ratio within the entire testing set. Subsequently, the testing set is partitioned into sequential batches, each comprising 50,000 samples, while maintaining the same ratio. For example, given an 80:20 benign-to-malicious ratio, each batch is formed by sequentially selecting 40,000 benign and 10,000 malicious samples. The first batch includes the initial benign and malicious samples as per the ratio, the second batch comprises the next sequence of benign and malicious samples, and this pattern continues with each batch until the testing set is fully segmented.

From Kitsune, we created a scenario maintaining a constant malicious-to-benign ratio (mKitsune), and another designed to replicate a recurring concept (rKitsune). For mKitsune, we adjusted the Kitsune testing set for a uniform ratio, distributing attacks across batches akin to our IDS18 strategy, allowing us to compare Kitsune and mKitsune results to assess the ratio shift impact on adaptation results. For rKitsune, we created a test set by mixing ARP MitM and Active Wiretap files, interspersing two Active Wiretap batches after every four ARP MitM batches, aiming to test the frameworks' ability to maintain old valid distributions (ARP MitM file) while adapting to new ones (Active Wiretap file).

Table 1 presents the statistical breakdown of the datasets post-processing, while Appendix A offers further details on the datasets and their processing steps.

Dataset	Train	Test	
	# Samples	# Samples	# Batches
IDS17	693K	1.4M	29
IDS18	2.5M	7.5M	151
Kitsune	751K	5.3M	107
mKitsune	751K	5.3M	107
rKitsune	751K	1.7M	35

Table 1: Statistical information of the datasets.

4.1.2 Baselines. We compare Mateen to four baselines. The *first* baseline is No-Update, which is an offline one-class DAE that is trained on benign data from the training set and does not conduct any updates to address shifts. The *second* baseline is OWAD [39], which stands out as the only framework specifically aimed at detecting and adapting to benign shifts in security contexts. It uses a fixed-size memory, periodically updated by replacing outdated training samples for new benign shift samples upon detecting shifts. The model is retrained with this updated memory. To prevent catastrophic forgetting, OWAD employs a customized version of UNLEARN [27], ensuring crucial parameters for previous valid distributions remain constant while updating less critical ones for new distributions.

The *third* baseline is INSOMNIA [4], which is a supervised framework designed specifically for adapting to distribution shifts in multi-class NIDS. INSOMNIA does not employ a shift detection mechanism; instead, it updates the model with every incoming batch of data. It utilizes US to select the least confident samples for human labeling. These selected samples are then incorporated into the training set for model retraining.

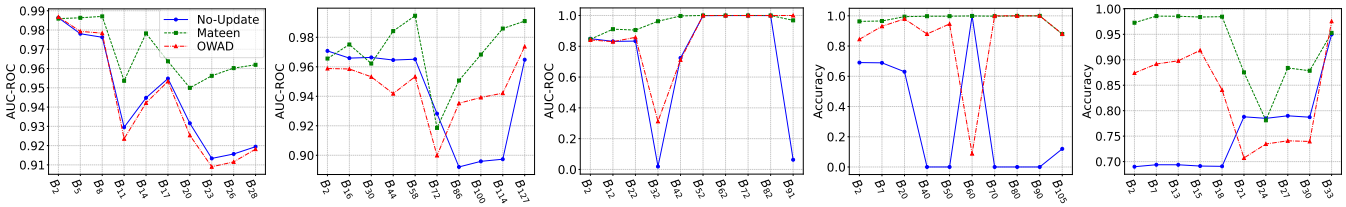
The *fourth* baseline is CADE, which is considered the state of the art in out-of-distribution detection and sample selection for multi-class security applications. CADE utilizes supervised contrastive loss [38] to minimize the distance between samples sharing the same label while increasing the distance between samples with different labels. After training, samples that are distant from all label clusters are flagged as shifts, with those exhibiting the highest distance chosen for labeling. Note that CADE is used only for comparison with our selection method, as it focuses on shift detection and sample selection.

Training and Updates. All models are given the same training set and batches of testing data but vary in their training and sample selection strategies for updates. Each adaptive framework utilizes a unique method to select an identical number of samples (δ), forming a subset that is then labeled by humans, and used for updates. OWAD and Mateen initialize with the No-Update model, which is exclusively trained on benign samples from the training set. Subsequently, they update their underlying DAE(s) to address benign shifts, using only the benign samples from the selected subset. INSOMNIA and CADE, as supervised frameworks, are trained on the complete training set, covering both malicious and benign samples. Their updates involve the full selected subset, integrating samples from both classes.

For a detailed description of the baselines, the hyperparameter search strategy, and the hyperparameters used, please refer to Appendix B.

4.1.3 Experimental Design. Our experimental design encompasses five key elements, each addressing a distinct aspect of our framework's performance and characteristics:

- **End-to-End Performance** (Section 4.3.1): We examine the end-to-end performance of Mateen alongside adaptive learning baselines under diverse distribution shift conditions and varying selection rates (δ).



a. IDS17 (AUC-ROC) b. IDS18 (AUC-ROC) c. mKitsune (AUC-ROC) d. Kitsune (Acc) e. rKitsune (Acc)

Figure 6: Adaptation performance over B_n.

- **Learning Under Latency** (Section 4.3.2): We examine the effects of update latency and restricted sample size on adaptation performance.
- **Learning Under Label Noise** (Section 4.3.3) We assess the impact of mislabeling the selected samples on adaptation performance.
- **Sub-Component Analysis** (Section 4.3.4): We perform an ablation study to assess the impact of our proposed adaptive learning strategy, complexity reduction module, and selection method.
- **Hyperparameter Impact** (Appendix B.4): We analyze the effects of varying hyperparameters, such as ρ, σ%, maximum ensemble size, and λ_0, on Mateen's performance.

4.1.4 Evaluation Metrics. We set the threshold for the DAE-based models at the 95th percentile and computed three key metrics at this threshold: F1-Score, the harmonic mean of precision and recall; Macro F1-Score, averaging the F1-Scores for each class; and Accuracy, the proportion of correct predictions. To mitigate potential bias from the threshold setting, we also included the Area under the ROC Curve (AUC-ROC) as a metric, assessing the model's ability to differentiate between classes across various thresholds. For the threshold-dependent metrics, we sequentially combined predictions from all batches into a single file. Next, we directly compared this consolidated predictions file with the actual labels to calculate the metrics. Concurrently, the AUC-ROC was calculated for each data batch, and the overall score was derived by averaging these individual batch scores.

It should be noted that in our process, after predicting a batch of data with a model, we stored these initial predictions for metrics calculation. If a shift was detected within this batch, we updated the model accordingly. However, we did not reapply the updated model to the same batch of data for a second round of predictions. This procedure ensures that our evaluation remains consistent with the principles of the test-then-train scheme [34].

4.2 Computing Platform

The experiments were conducted on a Linux-based system (Ubuntu 22.04.3 LTS) with an Intel Xeon Processor (Skylake), featuring 16 cores across two NUMA nodes. The system was equipped with dual NVIDIA Tesla T4 GPUs, each offering 16 GB of memory, and was supported by 128 GB of RAM. All frameworks have been implemented using PyTorch version 2.0.1 [80], utilizing CUDA version 12.2 and operating on Python 3.11.3.

Framework	F1-Score	mF1-Score	Accuracy	AUC-ROC
(a) IDS17 (Covariate & Prior)				
No-Update	88.09	80.53	83.46	94.56
INSOMNIA	78.45	49.99	64.55	49.76
OWAD	85.1	71.59	78.01	94.4
Mateen	**92.22**	**88.48**	**89.69**	**96.79**
(b) IDS18 (Covariate & Fixed Ratio)				
No-Update	94.74	78.68	90.78	93.81
INSOMNIA	96.92	84.56	94.46	78.32
OWAD	85.01	56.24	85.01	91.97
Mateen	**97.03**	**91.32**	**95.63**	**98.17**
(c) Kitsune (Concept Shift & Prior)				
No-Update	24.50	27.72	27.86	63.98
INSOMNIA	87.23	51.38	77.81	47.31
OWAD	91.28	67.21	84.89	70.97
Mateen	**97.1**	**91.6**	**95.2**	**72.15**
(d) mKitsune (Concept Shift & Fixed Ratio)				
No-Update	24.50	27.72	27.86	72.58
INSOMNIA	88.11	50.14	79.06	50.65
OWAD	93.59	81.82	89.44	79.98
Mateen	**95.86**	**87.41**	**93.09**	**94.76**
(e) rKitsune (Recurring Concept Shift)				
No-Update	86.02	77.18	80.61	88.43
INSOMNIA	86.84	67.79	79.06	65.24
OWAD	91.09	82.89	86.82	87.14
Mateen	**94.52**	**89.63**	**91.93**	**93.47**

Table 2: End-to-End adaptation performance.

4.3 Results and Discussion

4.3.1 End-to-End Performance. In this experiment, we assess the end-to-end performance of Mateen—specifically, how well it detects anomalies after adapting to benign shifts—under various scenarios of distribution shifts. We compared Mateen against three baselines: a non-updated DAE, OWAD, and INSOMNIA. The non-updated DAE (No-update) represents the initial model that was trained on a large dataset before being utilized by OWAD and Mateen. We report two results: the first details the overall performance aggregated across all batches (Table 2); the second result visualizes the adaptation performance over time, which is calculated from individual batches (Figure 6). For updates, each model was restricted to selecting only 500 samples from batch B, equivalent to a selection rate (δ) of 1%. It is important to note that Mateen and OWAD initiate

updates only upon detecting shifts, whereas INSOMNIA performs updates with each new batch of data.

Covariate Shift Adaptation. The results of end-to-end adaptation for the covariate shift datasets IDS17 and IDS18 are presented in Tables 2a and 2b, respectively. Notably, Mateen significantly surpasses all baseline models in both datasets across all the metrics. More importantly, Mateen shows a notable performance improvement compared to the No-Update; for instance, it increases the F1-Score by 4.13% on the IDS17 dataset, while OWAD reduces the performance by 2.99%. Further examinations, illustrated in Figures 6a and 6b, demonstrate that OWAD consistently underperforms compared to No-Update across most batches. In contrast, Mateen consistently enhances performance.

We can also notice that Mateen significantly improves the mF1-Score when compared to all the baselines. This metric underscores Mateen's ability to balance precision and recall across both benign and malicious classes. For example, on the IDS17 dataset, Mateen achieves an mF1-score of 88.48, outperforming the highest-performing baseline (No-Update) by 7.95%. More importantly, other baselines, in contrast to Mateen, demonstrate a bias toward the more common class (i.e., the benign class), marked by high F1 scores but notably lower mF1-scores, with a discrepancy exceeding 10%. This indicates that they classify a significant number of malicious samples as benign.

We further analyzed the technical design choices of each framework to understand their impact on adaptation outcomes. Both OWAD and INSOMNIA employ incremental updates, whereby selected samples from the batch are combined with historical samples prior to retraining the model on this enriched dataset. However, OWAD utilizes a memory-based strategy that necessitates the removal of some historical samples to accommodate new ones. We identified two main reasons why this approach may lead to reduced performance. First, historical samples may remain relevant for future batches; thus, removing them can result in the loss of valuable knowledge. Second, addressing covariate shift by integrating both all the historical and new data is more effective, as it enables the model to learn patterns that more accurately represent both past and current data.

On the other hand, INSOMNIA is hindered by its binary supervised framework. More specifically, in contrast to OWAD and Mateen, it adapts to shifts in both malicious and benign classes. Within this context, malicious classes demonstrate concept shifts across both datasets. Consequently, the performance is significantly affected by the manner in which attacks are represented in these datasets. To be specific, INSOMNIA performs exceptionally well on the IDS18 dataset, where the attacks exhibit semantic similarities—for instance, the training set includes FTP brute force attacks, whereas the testing set features web brute force attacks. Additionally, the attacks are distributed over multiple batches. In contrast, the IDS17 dataset presents challenges, as the attacks significantly differ semantically from those in the testing set (e.g., brute force attacks compared to DDoS), and certain attacks are confined to a single batch.

Concept Shift Adaptation. Table 2 presents the performance of Mateen and the baselines on the Kitsune (c) and mKitsune (d) datasets. Mateen demonstrates strong performance across both datasets, except the AUC-ROC score for the Kitsune dataset. The lower AUC-ROC score in Kitsune is due to its structure, which only includes two batches suitable for AUC-ROC calculation, as these are the only batches containing a mixture of benign and malicious instances.

We can also notice that OWAD and Mateen significantly enhance the evaluation metrics compared to the quickly outdated No-Update baseline, which suffers from concept shifts in benign traffic. Specifically, OWAD improves accuracy in most batches within the Kitsune dataset, as illustrated in Figure 6d. In this scenario, the memory-based approach proves beneficial, as it discards outdated concepts that no longer match the incoming data concepts. However, Mateen exhibits even stronger performance, surpassing OWAD in the majority of batches across both the mKitsune and Kitsune datasets, as evidenced in Figures 6c and 6d, respectively.

INSOMNIA, on the other hand, struggles with concept shifts, exhibiting a mF1-Score around 50%. This issue originates from its incremental learning approach, which incorporates only a small number of samples (500 per shifted batch) into the already large training set. As a result, the classifier is retrained predominantly with outdated, less relevant data and a minimal number of new distribution samples. This causes the model to focus on minimizing errors for the larger, outdated dataset, in alignment with empirical risk minimization principles.

Recurring Concept Adaptation. Table 2e and Figure 6e present the performance of Mateen and the baselines on the rKitsune dataset. We can notice that Mateen is outperforming the baselines, which suggests that Mateen can handle short concept shift data without sacrificing the performance on the consistent concept. For example, Mateen's AUC-ROC score is 93.47, surpassing OWAD's 87.14. More importantly, both OWAD and Mateen show improvements in evaluation metrics over the No-Update, although OWAD's AUC-ROC score decreases slightly by 1.29%. However, it is crucial to note that the rKitsune dataset offers limited data (only 2 out of 35 batches) for AUC-ROC calculation. On the other hand, INSOMNIA, while generally less effective, demonstrates improved performance compared to its results on the Kitsune dataset. This improvement is attributed to its incremental learning method, which effectively keeps the classifier up-to-date with consistent concepts. However, its performance in adapting to concept shifts remains compromised.

Selection Rate Impact. We here evaluate the effect of varying the δ value on the end-to-end adaptation results, utilizing the IDS17, mKitsune, and rKitsune datasets with δ values of 0.5%, 1%, 5%, and 10%. These percentages correspond to the frameworks being updated with 250, 500, 2500, or 5000 samples per batch upon detecting shifts. We assessed the frameworks based on Accuracy and AUC-ROC scores, as illustrated in Figure 7.

We can observe three main findings. First, Mateen consistently outperforms all baseline frameworks across the entire range of δ values. Second, it exhibits strong performance even when only 0.5% of the samples from each shifted batch require labeling. Most notably, Mateen's performance with a δ of 0.5% surpasses that of all baseline models across their full spectrum of δ values. This indicates that Mateen can achieve superior performance with fewer required labels. For example, in the mKitsune dataset, Mateen achieves an accuracy of 93.09% with only 250 labeled samples (equivalent to a δ of

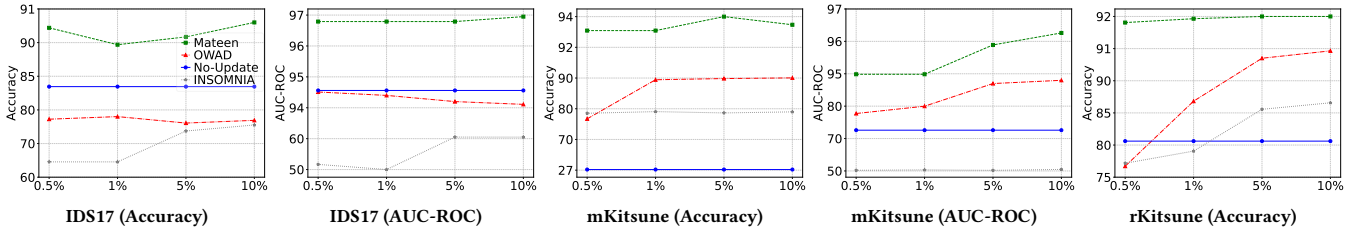


Figure 7: End-to-End adaptation performance with δ of 0.5%, 1%, 5%, and 10%.

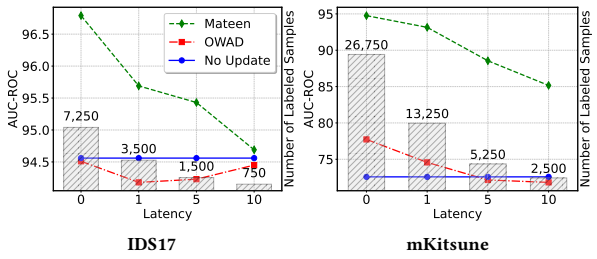


Figure 8: The impact of latency and limited sample selection on performance.

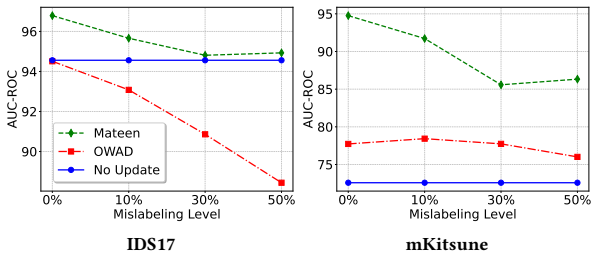


Figure 9: The impact of mislabeling on performance.

0.5%), while OWAD and INSOMNIA, despite requiring 5000 labeled samples, attain accuracies of just 90.01% and 78.98%, respectively.

4.3.2 Learning Under Latency. The previous experiments do not consider latency and assume that ground truth labels are immediately available and the model updates instantaneously. In reality, both manual labeling and model updates will introduce delays, and new samples continue to arrive before the update is completed. To more accurately simulate these real-world conditions, we introduce latency and a limited labeling budget into the process.

Assume updating a model takes as long as accumulating n batches of data. When a distribution shift is detected within a batch B_j, we continue using the current model M_k to classify the next n batches, and in the meantime we create the updated model M_{k+1}, which will replace M_k for B_{j+n}. For updates, we use a selection rate of 0.5%, which equates to selecting only 250 samples for manual labeling and updating. Once updated, the processed batches (n batches) are excluded from the shift detection and sample selection steps. We test this setup with n set at 1, 5, and 10. For the entire testing set of IDS17, this configuration requires analysts to label 3,500 samples for n=1, 1,500 for n=5, and 750 for n=10.

Figure 8 shows the results of this experiment using IDS17 and mKitsune datasets. Generally, as expected, Mateen's performance declines with increasing delays. However, it is noteworthy that even with significant delays (i.e., n=10) and a very low number of manually labeled samples (approximately 0.05% of the testing set), Mateen still outperforms the baseline models. Conversely, OWAD shows improved performance with delays in the IDS17 dataset, benefiting from retaining old, still-valid samples.

4.3.3 Learning Under Label Noise. Security analysts can mistakenly mislabel samples. To simulate this, we randomly changed the ground-truth labels for 10%, 30%, or 50% of the samples in each selected subset, while the rest remained correctly labeled. We then used these subsets, containing both accurate and inaccurate labels, to update the model and evaluate its performance under label noise conditions. Figure 9 presents the results for Mateen and OWAD on the IDS17 and mKitsune datasets. The performance of the non-updated model (No-Update) is also included, as it does not undergo updates and is thus unaffected by mislabeling. This shows whether noisy updates cause the frameworks to perform worse than their non-updated counterpart.

On the IDS17 dataset, Mateen's performance gradually declines as noise increases, yet it still outperforms both OWAD and the non-updated model. In contrast, OWAD shows a significant decrease in performance with increasing label noise. On the mKitsune dataset, despite the noise, both Mateen and OWAD are still better than the non-updated model. However, OWAD's performance is particularly affected by the highest noise level (50%). Overall, Mateen proves to be the most robust under label noise conditions.

4.3.4 Sub-Component Analysis. We undertook ablation studies to dissect and assess the individual contributions of Mateen's components. Our focus was particularly on evaluating the effects of the adaptive learning component and the efficacy of our proposed selection method.

Adaptive Learning Strategies. We evaluate the proposed adaptive learning strategy by contrasting Mateen with four setups: an incrementally updated DAE (*INC*), solely temporary models (*T*), Mateen without its transfer learning component (No Transfer), and Mateen without its merging component (No Merge). The No Merge setup

IDS17 (Covariate & Prior)				
Method	F1-Score	mF1-Score	Accuracy	AUC-ROC
INC	**92.73**	**89.03**	**90.48**	96.69
T	78.32	50.28	66.1	65.29
No Transfer	91.31	87.04	88.45	96.51
No Merge	92.43	88.75	89.96	96.61
Mateen	92.22	88.48	89.69	**96.79**
mKitsune (Concept Shift & Fixed Ratio)				
INC	24.03	27.69	27.88	73.9
T	79.86	57.6	69.29	78.3
No Transfer	79.95	59.23	69.77	82.95
No Merge	95.5	86.72	92.53	94.01
Mateen	**95.86**	**87.41**	**93.09**	**94.76**

Table 3: Comparison of the ensemble components.

IDS17 (Covariate & Prior)				
Method	F1-Score	mF1-Score	Accuracy	AUC-ROC
US	83.64	68.97	75.91	79.29
CADE	91.1	86.6	88.11	95.21
OWAD	91.56	87.27	88.71	96.5
Mateen	**92.22**	**88.48**	**89.69**	**96.79**
mKitsune (Concept Shift & Fixed Ratio)				
US	90.82	76.9	85.29	78.9
CADE	81.54	66.31	73.2	87.28
OWAD	95.29	**87.45**	92.35	93.85
Mateen	**95.86**	87.41	**93.09**	**94.76**

Table 4: Comparison of the selection methods.

specifically excludes Mateen's complexity reduction module. Comparative results for the IDS17 and mKitsune datasets are detailed in Table 3.

We can notice three important observations. Firstly, Mateen demonstrates consistently high performance across both datasets, though the efficacy of its integral components, *INC* and *T*, varies with the type of data shift. Specifically, *INC*'s performance on the covariate shift is marginally better than Mateen's overall performance; however, it is significantly worse than all baselines on the concept shift. This discrepancy arises because the *INC* model utilizes its large and growing dataset to learn patterns more effectively in the covariate shift scenario. Conversely, this extensive dataset hampers its adaptability in the concept shift scenario, as acquiring representative patterns of diverse concepts proves challenging. On the other hand, the temporary models, *T*, underperform relative to most baselines in both datasets. This underperformance is attributed to their limited training data, which compromises their ability to learn the task adequately. Specifically, each model is provided with only 500 labeled samples (a δ of 1%), an amount insufficient to accurately represent the task's distribution, thereby predisposing the models to overfit.

Secondly, the transfer learning component significantly influences the effectiveness of learning. The results clearly demonstrate that models lacking transfer learning capabilities underperform markedly on the concept shift dataset compared to the complete Mateen framework, underscoring the vital role of transfer learning in improving model generalizability. However, this benefit does not apply to the covariate shift dataset. This is primarily because our best-performing model selection strategy tends to favor the *INC* model over the *T* model for most batches, owing to the low performance of the latter. Thirdly, Mateen's performance closely matches that of the No-Merge approach, indicating that our complexity reduction module is key to both preserving an optimal ensemble size and retaining essential knowledge. This suggests the module's effectiveness extends beyond simple size management to ensuring that critical insights are not sacrificed for efficiency.

Selection Method Comparison. In this experiment, we evaluate three baseline methods – CADE, Uncertainty Sampling (US), and OWAD – in comparison to our sample selection method on the adaptive learning approach of Mateen. CADE and US both target the most uncertain samples for updates, differing in their strategies for assessing uncertainty. CADE, specifically, utilizes a contrastive learning approach during training and selects samples most distant from the centroids of the training classes for adaptation. US, tailored to our context, chooses samples based on the highest reconstruction scores. Conversely, OWAD operates at a distribution level, comparing reconstruction error scores between the training data and the incoming batch data distributions. It selects samples from the batch of data that show significant deviations from the training distribution.

Table 4 presents the outcomes of this comparison for the IDS17 and mKitsune datasets. The US strategy emerges as the least effective, primarily because it selects samples with high MSE scores, which are predominantly malicious. This selection hinders the adaptive learning approach's capacity to adapt to new distributions and adversely affects the identification of the best-performing model. However, CADE demonstrates better performance compared to US in the adaptive learning ensemble, despite being a form of uncertainty sampling. Our analysis indicates that CADE's contrastive loss results in both benign and malicious shifted samples being distanced from the training data, facilitating the selection of shifted samples from both classes. Nonetheless, the Mateen and OWAD selection methods exhibit significantly superior results, particularly in the mKitsune dataset, with Mateen slightly outperforming OWAD. For instance, in the mKitsune dataset, Mateen, OWAD, and CADE achieve AUC-ROC scores of 94.76, 93.85, and 87.28, respectively.

5 DISCUSSION AND FUTURE WORK

Hyper-parameter sensitivity. We conducted a sensitivity analysis of hyperparameters and have documented the default configuration of Mateen, along with the configurations of the considered baselines, in Appendix B. Through our extensive analysis, we discovered that Mateen consistently delivers robust performance across a range of hyperparameter combinations. For example, different hyperparameter combinations for IDS17 resulted in an average of 96.19 and a standard deviation of 0.64 concerning AUC-ROC scores. This observation suggests that Mateen can achieve strong results even when the optimal hyperparameters are not precisely identified. We also provide guidance on how to set these values and the factors that need to be considered in Appendix C.

Costs of Manual Labeling. Mateen adopts the most dependable updating strategy by using only a few labeled samples, a method well-regarded in security contexts [16, 39, 55, 95, 101]. However, this presents a significant challenge in highly dynamic networks where labeling even a small number of samples can become prohibitively expensive due to frequent shifts occurring in short spans of time. Recent research efforts have investigated learning without reliance on ground-truth labels, employing label estimators instead [4, 98]. Nonetheless, this method has been found to inadvertently lead to self-poisoning of models, as inaccurately labeled samples can degrade model performance [47]. This issue arises from the direct use of estimated labels for model updates without verifying their accuracy. Consequently, our future work aims to investigate learning from high-quality estimated labels, seeking methods to ensure the reliability of these labels before model integration.

Latency and Mislabeling. Manual labeling introduces two key challenges: latency and error risk. Latency occurs because analysts need time to manually label samples, and during this period, the non-updated ensemble continues to make predictions, which may not be accurate due to shifts in data. Additionally, there is a risk of error, as analysts might mislabel some samples. These inaccuracies can lead to updates with incorrect data, resulting in degraded performance of the ensemble. Despite these challenges, our findings indicate that even with high latency and a significant level of mislabeling, Mateen still outperforms both the best baseline model and the non-updated version. In future work, we aim to further reduce the impact of latency and error risks. To achieve this, we plan to investigate three interconnected research areas: label estimation [4, 47, 98], learning from noisy data [14, 32, 96, 97], and advanced labeling strategies [37].

Adversarial Machine Learning. Mateen is purpose-built to effectively handle benign shifts and improve the DAE's performance. This design incorporates an updating mechanism that carefully selects samples for human labeling. However, this mechanism is a potential vector for poisoning the ensemble [52, 97, 100]. Analysts might inadvertently mislabel non-poisonous samples, as previously mentioned, or attackers could introduce poisonous samples, such as through label flipping and backdoor attacks [88, 92, 100], to deceive analysts and compromise the ensemble. Therefore, in our future research, we plan to investigate the effects of advanced poisoning attacks and explore defensive strategies, such as robust learning [65, 69, 83], to mitigate their impact. Additionally, Mateen may be vulnerable to evasive samples [41, 43, 54, 105], where adversaries attempt to disguise attacks as benign behaviors to evade detection. However, this susceptibility depends on the design of the underlying DAE and its associated feature space. In future work, we plan to investigate the interplay between feature space, DAE architecture, and Mateen when confronted with evasive inputs, considering the potential impact on anomaly detection accuracy and robustness.

6 RELATED WORK

Distribution Shift in Security Applications. Generally, two key approaches are employed to address distribution shift in security: learning from robust features [2, 13, 21, 23, 53, 94, 106] and continuous shift adaptation [4, 16, 39, 101]. The first approach focuses on defining robust features that minimize the impact of distribution shifts. However, this approach cannot be considered a universal solution, as it is complex to define the nature of future attacks. The second approach seeks to mitigate the impact of shifts by updating the learning framework with samples that represent the shift. Building upon this approach, our work introduces a framework specifically designed to adapt well to various types of benign shifts (e.g., covariate, concept, etc.), utilizing a minimal number of labeled representative samples. Additionally, our framework demonstrates superior performance with fewer labeled samples compared to the baselines and exhibits less sensitivity to shift types, unlike each baseline, which excels in specific shift scenarios.

Batch-based Ensemble Learning. Batch-based ensemble learning diverges from the common ensemble approach by avoiding dependence on a fixed set of models trained on the same dataset. It instead continuously trains a new sub-model for each incoming batch of data, incorporating it into a growing pool of models. Various batch learning strategies exist [11, 25, 28, 30, 44, 67, 68, 99, 103], distinguishable by factors such as batch formulation, with examples including fixed-batching [25, 28, 99, 103] and dynamic-batching [11, 12, 68, 95]. This learning strategy trains on every sample from incoming data batches using either unsupervised batch training [99, 103] or focusing on labeled benign samples [95]. Unsupervised training can lead to self-poisoning by indiscriminately incorporating all data, including potentially harmful anomalies. Conversely, focusing solely on benign samples requires accessible labels, presenting challenges in security applications such as NIDS, where comprehensive labeling is expensive. A straightforward solution is to select the most uncertain samples from each batch to train new sub-models. However, our results indicate that this approach falls short for batch-based learning in NIDS, as the small size of the chosen samples leads to inadequate generalization. To overcome this issue, we utilize transfer learning combined with a coreset-like selection method, enabling effective generalization of sub-models even when trained with a small number of samples.

7 CONCLUSION

In this paper, we present Mateen, a novel framework designed for detecting and adapting to benign shifts within single-class DAE-based NIDS. Mateen utilizes a hybrid ensemble learning strategy incorporating an incremental model that is capable of adjusting to minor shifts, such as covariate shifts, alongside temporary models designed to tackle major shifts, including concept shifts. This framework successfully mitigates the reliance of batch-based learning on extensive labeled data sets through the application of transfer learning and a coreset-like selection strategy. Additionally, a complexity reduction module is employed to optimize the size of the ensemble, ensuring that performance remains uncompromised. When evaluated across five network datasets, Mateen exhibits exceptional adaptability, effectively maintaining the relevance of the DAE while requiring minimal labeled data.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful critiques and the shepherd for his invaluable guidance during revisions. Alotaibi's research is funded by a scholarship from the Deanship of Scientific Research at Najran University, Kingdom of Saudi Arabia.

REFERENCES

[1] Andrea Agiollo, Enkeleda Bardhi, Mauro Conti, Riccardo Lazzeretti, Eleonora Losiouk, and Andrea Omicini. 2023. GNN4IFA: Interest flooding attack detection with graph neural networks. In *Proc. of EuroS&P*.

[2] Almuthanna Alageel and Sergio Maffeis. 2022. EARLYCROW: Detecting APT Malware Command and Control over HTTP (S) Using Contextual Summaries. In *Proc. of ISC*.

[3] Giuseppina Andresini, Annalisa Appice, Corrado Loglisci, Vincenzo Belvedere, Domenico Redavid, and Donato Malerba. 2021. A network intrusion detection system for concept drifting network traffic data. In *Proc. of DS*.

[4] Giuseppina Andresini, Feargus Pendlebury, Fabio Pierazzi, Corrado Loglisci, Annalisa Appice, and Lorenzo Cavallaro. 2021. INSOMNIA: Towards Concept-Drift Robustness in Network Intrusion Detection. In *Proc. of AISec@CCS*.

[5] Fabrizio Angiulli. 2007. Fast Nearest Neighbor Condensation for Large Data Sets Classification. *IEEE Transactions on Knowledge and Data Engineering* (2007).

[6] Giovanni Apruzzese, Pavel Laskov, and Johannes Schneider. 2023. SoK: Pragmatic Assessment of Machine Learning for Network Intrusion Detection. In *Proc. of EuroS&P*.

[7] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and Don'ts of Machine Learning in Computer Security. In *Proc. of USENIX Security*.

[8] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2023. Lessons Learned on Machine Learning for Computer Security. *IEEE Security & Privacy Magazine* (2023).

[9] Dor Bank, Noam Koenigstein, and Raja Giryes. 2023. Autoencoders. *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook* (2023), 353–374.

[10] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. 2022. Transcending TRANSCEND: Revisiting Malware Classification in the Presence of Concept Drift. In *Proc. of S&P*.

[11] Albert Bifet and Ricard Gavaldà. 2007. Learning from Time-Changing Data with Adaptive Windowing. In *Proc. of SDM*.

[12] Albert Bifet, Geoffrey Holmes, and Bernhard Pfahringer. 2010. Leveraging Bagging for Evolving Data Streams. In *Proc. of ECML PKDD*.

[13] Haipeng Cai. 2020. Assessing and Improving Malware Detection Sustainability through App Evolution Studies. *ACM Transactions on Software Engineering and Methodology* (2020).

[14] Jian Chen, Ruiyi Zhang, Tong Yu, Rohan Sharma, Zhiqiang Xu, Tong Sun, and Changyou Chen. 2024. Label-retrieval-augmented diffusion models for learning from noisy labels. *Proc. of NeurIPS* (2024).

[15] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proc. of ICML*.

[16] Yizheng Chen, Zhoujie Ding, and David A. Wagner. 2023. Continuous Learning for Android Malware Detection. In *Proc. of USENIX Security*.

[17] Zekai Chen, Dingshuo Chen, Xiao Zhang, Zixuan Yuan, and Xiuzhen Cheng. 2022. Learning Graph Structures With Transformer for Multivariate Time-Series Anomaly Detection in IoT. *IEEE Internet of Things Journal* (2022).

[18] Hyunsoo Cho, Jinseok Seol, and Sang-goo Lee. 2021. Masked Contrastive Learning for Anomaly Detection. In *Proc. of IJCAI*.

[19] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2020. Selection via Proxy: Efficient Data Selection for Deep Learning. In *Proc. of ICLR*.

[20] Steve Dias Da Cruz, Bertram Taetz, Thomas Stifter, and Didier Stricker. 2022. Autoencoder Attractors for Uncertainty Estimation. In *Proc. of ICPR*.

[21] Lei Cui, Jiancong Cui, Yuede Ji, Zhiyu Hao, Lun Li, and Zhenquan Ding. 2023. API2Vec: Learning Representations of API Sequences for Malware Detection. In *Proc. of ISSTA*.

[22] Tiago Fontes Dias, João Vitorino, Tiago Fonseca, Isabel Praça, Eva Maia, and Maria João Viamonte. 2023. Unravelling Network-Based Intrusion Detection: A Neutrosophic Rule Mining and Optimization Framework. In *Proc. of ESORICS*.

[23] Mirabelle Dib, Sadegh Torabi, Elias Bou-Harb, Nizar Bouguila, and Chadi Assi. 2022. EVOLIoT: A Self-Supervised Contrastive Learning Framework for Detecting and Characterizing Evolving IoT Malware Variants. In *Proc. of ASIA CCS*.

[24] Simone Disabato and Manuel Roveri. 2019. Learning Convolutional Neural Networks in presence of Concept Drift. In *Proc. of IJCNN*.

[25] Gregory Ditzler and Robi Polikar. 2013. Incremental Learning of Concept Drift from Streaming Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* (2013).

[26] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. 2016. Characterization of Encrypted and VPN Traffic using Time-related Features. In *Proc. of ICISSP*.

[27] Min Du, Zhi Chen, Chang Liu, Rajvardhan Oak, and Dawn Song. 2019. Lifelong Anomaly Detection Through Unlearning. In *Proc. of CCS*.

[28] Ryan Elwell and Robi Polikar. 2011. Incremental Learning of Concept Drift in Nonstationary Environments. *IEEE Transactions on Neural Networks* (2011).

[29] Gints Engelen, Vera Rimmer, and Wouter Joosen. 2021. Troubleshooting an Intrusion Detection Dataset: the CICIDS2017 Case Study. In *Proc. of S&P Workshops*.

[30] Dewan Md. Farid, Li Zhang, M. Alamgir Hossain, Chowdhury Mofizur Rahman, Rebecca Strachan, Graham Sexton, and Keshav P. Dahal. 2013. An adaptive ensemble classifier for mining concept drifting data streams. *Expert Systems with Applications* (2013).

[31] Helmut Finner and Veronika Gontscharuk. 2018. Two-sample Kolmogorov–Smirnov-type tests revisited: old and new tests in terms of local levels. *The Annals of Statistics* (2018).

[32] Fahimeh Fooladgar, Minh Nguyen Nhat To, Parvin Mousavi, and Purang Abolmaesumi. 2024. Manifold DivideMix: A semi-supervised contrastive learning framework for severe label noise. In *Proc. of CVPR*.

[33] Chuanpu Fu, Qi Li, Ke Xu, and Jianping Wu. 2023. Point Cloud Analysis for ML-Based Malicious Traffic Detection: Reducing Majorities of False Positive Alarms. In *Proc. of CCS*.

[34] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. 2013. On evaluating stream learning algorithms. *Machine learning* (2013).

[35] João Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *Comput. Surveys* (2014).

[36] Dimitrios Giagkos, Orestis Kompougias, Antonis Litke, and Nikolaos Papadakis. 2023. ZeekFlow: Deep Learning-Based Network Intrusion Detection a Multimodal Approach. In *Proc. of ESORICS*.

[37] Jorge Luis Guerra, Carlos Catania, and Eduardo Veas. 2022. Datasets are not enough: Challenges in labeling network traffic. *Computers & Security* (2022).

[38] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *Proc. of CVPR*.

[39] 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. 2023. Anomaly Detection in the Open World: Normality Shift Detection, Explanation, and Adaptation. In *Proc. of NDSS*.

[40] Dongqi Han, Zhiliang Wang, Wenqi Chen, Ying Zhong, Su Wang, Han Zhang, Jiahai Yang, Xingang Shi, and Xia Yin. 2021. DeepAID: Interpreting and Improving Deep Learning-based Anomaly Detection in Security Applications. In *Proc. of CCS*.

[41] Dongqi Han, Zhiliang Wang, Ying Zhong, Wenqi Chen, Jiahai Yang, Shuqiang Lu, Xingang Shi, and Xia Yin. 2021. Evaluating and Improving Adversarial Robustness of Machine Learning-Based Network Intrusion Detectors. *IEEE Journal on Selected Areas in Communications* (2021).

[42] Zijun Hang, Yuliang Lu, Yongjie Wang, and Yi Xie. 2023. Flow-MAE: Leveraging Masked AutoEncoder for Accurate, Efficient and Robust Malicious Traffic Classification. In *Proc. of RAID*.

[43] Mohammad J. Hashemi, Greg Cusack, and Eric Keller. 2019. Towards Evaluation of NIDSs in Adversarial Setting. In *Proc. of Big-DAMA@CoNEXT*.

[44] Adriana S. Iwashita, Victor Hugo C. de Albuquerque, and João Paulo Papa. 2019. Learning concept drift with ensembles of optimum-path forest-based classifiers. *Future Generation Computer Systems* (2019).

[45] Roberto Jordaney, Kumar Sharad, Santanu Kumar Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. 2017. Transcend: Detecting Concept Drift in Malware Classification Models. In *Proc. of USENIX Security*.

[46] Md. Alamgir Kabir, Jacky W. Keung, Kwabena Ebo Bennin, and Miao Zhang. 2019. Assessing the Significant Impact of Concept Drift in Software Defect Prediction. In *Proc. of COMPSAC*.

[47] Zeliang Kan, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. 2021. Investigating Labelless Drift Adaptation for Malware Detection. In *Proc. of AISec@CCS*.

[48] ElMouatez Billah Karbab and Mourad Debbabi. 2021. Petadroid: Adaptive android malware detection using deep learning. In *Proc. of DIMVA*.

[49] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Proc. of NeurIPS* (2020).

[50] Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. 2021. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *Proc. of ICML*.

[51] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. 2021. Glister: Generalization based data subset selection for efficient and robust learning. In *Proc. of AAAI*.

[52] Marius Kloft and Pavel Laskov. 2010. Online Anomaly Detection under Adversarial Impact. In *Proc. of AISTATS*.

[53] Ke Kong, Zhichao Zhang, Ziyuan Yang, and Zhaoxin Zhang. 2022. FCSCNN: Feature centralized Siamese CNN-based android malware identification. *Computers & Security* (2022).

[54] Aditya Kuppa, Slawomir Grzonkowski, Muhammad Rizwan Asghar, and Nhien-An Le-Khac. 2019. Black Box Attacks on Deep Anomaly Detectors. In *Proc. of ARES*.

[55] Aditya Kuppa and Nhien-An Le-Khac. 2022. Learn to adapt: Robust drift detection in security domain. *Computers and Electrical Engineering* (2022).
[56] Maxime Lanvin, Pierre-François Gimenez, Yufei Han, Frédéric Majorczyk, Ludovic Mé, and Eric Totel. 2023. Towards understanding alerts raised by unsupervised network intrusion detection systems. In *Proc. of RAID*.
[57] Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. 2017. Characterization of Tor Traffic using Time based Features. In *Proc. of ICISSP*.
[58] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. 2013. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of experimental social psychology* (2013).
[59] Ruoyu Li, Qing Li, Yu Zhang, Dan Zhao, Yong Jiang, and Yong Yang. 2024. Interpreting Unsupervised Anomaly Detection in Security via Rule Extraction. *Proc. of NeurIPS* (2024).
[60] XuKui Li, Wei Chen, Qianru Zhang, and Lifa Wu. 2020. Building Auto-Encoder Intrusion Detection System based on random forest feature selection. *Computers & Security* (2020).
[61] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2020. On the Convergence of FedAvg on Non-IID Data. In *Proc. of ICLR*.
[62] Yuhua Li and Liam P. Maguire. 2011. Selecting Critical Patterns Based on Local Geometrical and Statistical Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2011).
[63] Anjin Liu, Jie Lu, Feng Liu, and Guangquan Zhang. 2018. Accumulating regional density dissimilarity for concept drift detection in data streams. *Pattern Recognition* (2018).
[64] Lisa Liu, Gints Engelen, Timothy M. Lynar, Daryl Essam, and Wouter Joosen. 2022. Error Prevalence in NIDS datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018. In *Proc. of CNS*.
[65] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. 2020. Early-learning regularization prevents memorization of noisy labels. *Proc. of NeurIPS* (2020).
[66] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. 2019. Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering* (2019).
[67] Yang Lu, Yiu-ming Cheung, and Yuan Yan Tang. 2017. Dynamic Weighted Majority for Incremental Learning of Imbalanced Data Streams with Concept Drift. In *Proc. of IJCAI*.
[68] Yang Lu, Yiu-Ming Cheung, and Yuan Yan Tang. 2020. Adaptive Chunk-Based Dynamic Weighted Majority for Imbalanced Data Streams With Concept Drift. *IEEE Transactions on Neural Networks and Learning Systems* (2020).
[69] Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah Erfani, and James Bailey. 2020. Normalized loss functions for deep learning with noisy labels. In *Proc. of ICML*.
[70] AA Makarov and GI Simonova. 2016. Some properties of two-sample kolmogorov–smirnov test in the case of contamination of one of the samples. *Journal of Mathematical Sciences* (2016).
[71] David M Mason and John H Schuenemeyer. 1983. A modified Kolmogorov-Smirnov test sensitive to tail alternatives. *The Annals of Statistics* (1983).
[72] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. of AISTATS*.
[73] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In *Proc. of NDSS*.
[74] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. 2020. Coresets for data-efficient training of machine learning models. In *Proc. of ICML*.
[75] Jose G. Moreno-Torres, Troy Raeder, Rocío Alaíz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. 2012. A unifying view on dataset shift in classification. *Pattern Recognition* (2012).
[76] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2020. What is being transferred in transfer learning? *Proc. of NeurIPS* (2020).
[77] Quoc Phong Nguyen, Kar Wai Lim, Dinil Mon Divakaran, Kian Hsiang Low, and Mun Choon Chan. 2019. GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection. In *Proc. of CNS*.
[78] Guansong Pang, Kai Ming Ting, and David W. Albrecht. 2015. LeSiNN: Detecting Anomalies by Identifying Least Similar Nearest Neighbours. In *Proc. of ICDMW*.
[79] Cheong Hee Park and Youngsoon Kang. 2016. An active learning method for data streams with concept drift. In *Proc. of BigData*.
[80] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Proc. of NeurIPS* (2019).
[81] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. 2019. TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time. In *Proc. of USENIX Security*.
[82] Adityanarayanan Radhakrishnan, Mikhail Belkin, and Caroline Uhler. 2020. Overparameterized neural networks implement associative memory. *Proc. Natl. Acad. Sci. USA* (2020).
[83] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *Proc. of ICML*.
[84] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. 2022. A Survey of Deep Active Learning. *Comput. Surveys* (2022).
[85] Phillip Rieger, Marco Chilese, Reham Mohamed, Markus Miettinen, Hossein Fereidooni, and Ahmad-Reza Sadeghi. 2023. ARGUS: Context-Based Detection of Stealthy IoT Infiltration Attacks. In *Proc. of USENIX Security*.
[86] Peter Schneider and Konstantin Böttinger. 2018. High-Performance Unsupervised Anomaly Detection for Cyber-Physical System Networks. In *Proc. of CPS-SPC@CCS*.
[87] Burr Settles. 2009. Active learning literature survey. (2009).
[88] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Proc. of NeurIPS* (2018).
[89] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proc. of ICISSP*.
[90] Robin Sommer and Vern Paxson. 2010. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *Proc. of S&P*.
[91] Ruming Tang, Zheng Yang, Zeyan Li, Weibin Meng, Haixin Wang, Qi Li, Yongqian Sun, Dan Pei, Tao Wei, Yanfei Xu, and Yan Liu. 2020. ZeroWall: Detecting Zero-Day Web Attacks through Encoder-Decoder Recurrent Neural Networks. In *Proc. of INFOCOM*.
[92] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. 2022. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *Comput. Surveys* 55, 8 (2022), 1–35.
[93] Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. 2002. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proc. of PNAS* (2002).
[94] Liang Tong, Bo Li, Chen Hajaj, Chaowei Xiao, Ning Zhang, and Yevgeniy Vorobeychik. 2019. Improving Robustness of ML Classifiers against Realizable Evasion Attacks Using Conserved Features. In *Proc. of USENIX Security*.
[95] Xian Wang. 2022. ENIDrift: A Fast and Adaptive Ensemble System for Network Intrusion Detection under Real-world Drift. In *Proc. of ACSAC*.
[96] Hongxin Wei, Huiping Zhuang, Renchunzi Xie, Lei Feng, Gang Niu, Bo An, and Yixuan Li. 2023. Mitigating memorization of noisy labels by clipping the model prediction. In *Proc. of ICML*.
[97] Xian Wu, Wenbo Guo, Jia Yan, Baris Coskun, and Xinyu Xing. 2023. From Grim Reality to Practical Solution: Malware Classification in Real-World Noise. In *Proc. of S&P*.
[98] Ke Xu, Yingjiu Li, Robert H. Deng, Kai Chen, and Jiayun Xu. 2019. DroidEvolver: Self-Evolving Android Malware Detection System. In *Proc. of EuroS&P*.
[99] Lijuan Xu, Xiao Ding, Haipeng Peng, Dawei Zhao, and Xin Li. 2023. ADTCD: An Adaptive Anomaly Detection Approach Towards Concept-Drift in IoT. *IEEE Internet of Things Journal* (2023).
[100] Limin Yang, Zhi Chen, Jacopo Cortellazzi, Feargus Pendlebury, Kevin Tu, Fabio Pierazzi, Lorenzo Cavallaro, and Gang Wang. 2023. Jigsaw Puzzle: Selective Backdoor Attack to Subvert Malware Classifiers. In *Proc. of S&P*.
[101] Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Ali Ahmadzadeh, Xinyu Xing, and Gang Wang. 2021. CADE: Detecting and Explaining Concept Drift Samples for Security Applications. In *Proc. of USENIX Security*.
[102] Yu Yang, Hao Kang, and Baharan Mirzasoleiman. 2023. Towards Sustainable Learning: Coresets for Data-efficient Deep Learning. In *Proc. of ICML*.
[103] Susik Yoon, Youngjun Lee, Jae-Gil Lee, and Byung Suk Lee. 2022. Adaptive Model Pooling for Online Deep Anomaly Detection from a Complex Evolving Data Stream. In *Proc. of KDD*.
[104] Shujian Yu and Zubin Abraham. 2017. Concept Drift Detection with Hierarchical Hypothesis Testing. In *Proc. of SDM*.
[105] Chaoyun Zhang, Xavier Costa-Pérez, and Paul Patras. 2020. Tiki-Taka: Attacking and Defending Deep Learning-based Intrusion Detection Systems. In *Proc. of CCSW*.
[106] Xiaohan Zhang, Yuan Zhang, Ming Zhong, Daizong Ding, Yinzhi Cao, Yukun Zhang, Mi Zhang, and Min Yang. 2020. Enhancing State-of-the-art Classifiers with API Semantics to Detect Evolved Android Malware. In *Proc. of CCS*.

A DATASETS PROCESSING

In the following, we provide additional details about the datasets utilized and the methods employed in their processing.

IDS17. We used an improved version of the IDS17 dataset, initially developed by the Canadian Institute for Cybersecurity [29]. The original dataset [89], which included packet captures and network

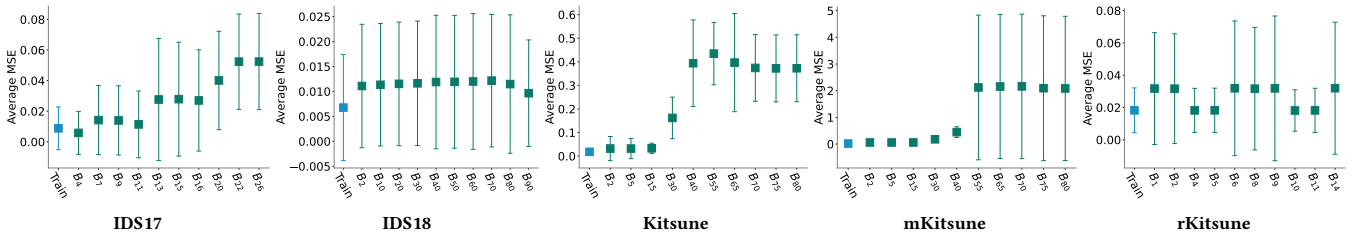


Figure 10: Reconstruction error (MSE) for benign samples using an offline DAE: This graph illustrates the performance of a model trained exclusively on the training dataset, with no further updates after the initial training. It presents the average and standard deviations of the reconstruction error, highlighting the training set in blue and testing set batches in green.

flows extracted with the CICFlowMeter tool [26, 57], had inaccuracies in feature extraction and labeling. To overcome these issues, Engelen et al. [29] released a revised version, which we adopted to enhance the accuracy and reliability of our findings.

For the purposes of our experiments, the dataset was divided into two segments: the training segment and the testing segment. The training segment, covering the first two days of data, contained 693,702 network flows. This portion was primarily benign traffic (99%), but it also included instances of two network attacks: FTP-Patator and SSH-Patator. The testing segment covered the last three days of data, comprising 1,406,274 network flows, with 64.55% being benign traffic alongside a variety of network attacks (13 different types). The testing segment was divided into batches of 50,000 network flows, resulting in a total of 29 batches of network flows. Crucially, to ensure integrity and objectivity in our analysis, we omitted certain features from the dataset that were used as a basis for original labeling. These included 'id', 'Flow ID', 'Src IP', 'Src Port', 'Dst IP', and 'Timestamp'. This step was taken to avoid any bias, such as the potential for models to use these features as shortcuts in classification and anomaly detection.

IDS18. IDS18, an extended version of the IDS17 dataset, incorporates data from over 450 machines across five departments, including 50 machines designated for attack simulations, collected over a three-week period. Similar to the original IDS17, this larger dataset initially suffered from flaws in the labeling and feature extraction process. However, these issues were addressed in a revised version, as detailed in [64], which we used for our experiments. Our experimental design was tailored to accommodate the substantial size of the dataset. We partitioned the data from the initial two weeks into training and testing segments. The training segment, including data from February 14th to 16th, consisted of over 10 million flows before processing. The testing segment covered data from February 20th to 23rd. Due to the large size of both datasets, we implemented a strategic sampling approach, randomly selecting 30% of flows from each label in every file. This selection was carried out in a time-based sequential order to avoid temporal bias [6, 81].

Post-processing, the training set was reduced to 2,548,780 flows, among which 1.47% were classified as attacks across six distinct classes. Likewise, the testing set was reduced to 7,501,307 flows, with 12.70% identified as attacks. These attacks span 10 classes, all of which differ from those in the training set. We segmented the test set into batches, each containing 50,000 flows, resulting in a total of 151 batches. To ensure unbiased analysis, we removed features that were used for labeling, specifically 'id', 'Flow ID', 'Src IP', 'Src Port', 'Dst IP', and 'Timestamp'. Furthermore, we adjusted the testing batches to consist of a fixed benign-to-malicious ratio to eliminate the impact of ratio shifts.

Kitsune. The Kitsune dataset [73], tailored for IoT network analysis, comprises 115 statistical features derived from network packets. It is organized into separate files by attack type, each containing a mix of benign and attack-specific records. In total, the combined files yield over 21 million records, presenting a significant computational challenge. To manage this, we adopted the same approach as with the IDS18 dataset, randomly sampling 30% of records from each label within each file. Due to the absence of timestamps in Kitsune, we arbitrarily selected the ARP Man-in-the-Middle file for training, which, after post-processing, contained 751,280 records, with 45% classified as malicious traffic. For testing, we chose other files, resulting in 5,324,759 records, of which 20.86% were identified as attacks. Segmenting the test set into batches of 50,000 each yielded a total of 107 batches.

Additionally, we created two new testing sets: one with a fixed malicious-to-benign ratio (mKitsune), and another designed to simulate a recurring concept (rKitsune). For mKitsune, we adjusted the already generated testing set to maintain this ratio, distributing attacks evenly across batches in line with the methodology employed in our processing of IDS18. For rKitsune, we generated a new test set by sampling 50% from both the original ARP Man-in-the-Middle and Active Wiretap datasets. We then interspersed two batches of Active Wiretap after every four batches of ARP Man-in-the-Middle. This arrangement produced a recurring concept test set with 1,739,344 records, which, when divided, yielded 35 batches.

Visualization of benign shift. Figure 10 presents a comparison of MSE scores between the training set and the testing set's benign sample batches across all datasets. Clearly, the testing set exhibits higher reconstruction errors across all datasets when compared to the training set. This increase in error indicates a potential for the model to mistakenly identify these samples as malicious, especially since the training set MSE is often calibrated at a certain threshold to separate malicious from benign samples. The magnitude of this increase, however, varies among datasets.

For instance, the IDS17 dataset shows a modest and steady increase in MSE scores, with the training set's average MSE around 0.01 and a standard deviation slightly over 0.02. In contrast, the B7

dataset's average MSE nears 0.02, and B20's average approaches 0.04. This variance suggests that while the model may perform adequately with B7 samples, it is likely to encounter considerable difficulties with B20. The IDS18 dataset, on the other hand, presents an even more minimal increase in MSE, a factor that aligns with the strong performance metrics observed in the experiments detailed in Section 4.3.1. The Kitsune and mKitsune datasets manifest a significant shift, leading to a heightened risk of benign samples being incorrectly classified as malicious. This substantial discrepancy underscores the potential obsolescence of the model, as evidenced by the findings presented in Section 4.3.1. rKitsune, however, presents a different pattern due to the recurring concept shifts, where certain batch intervals exhibit significantly higher MSE averages than others.

B HYPERPARAMETERS SEARCH & ANALYSIS

In the following sections, we provide detailed information on the baselines (Section B.1), the architecture used for each model (Section B.2), our approach to conducting a hyperparameter search for each baseline (Section B.3), and conclude with an analysis of hyperparameter sensitivity for Mateen (Section B.4).

B.1 Baselines Details

In this subsection, we provide detailed descriptions of each baseline utilized. Emphasis is placed on their high-level functions, specifically how they detect, select, and adapt to shift samples.

OWAD. OWAD is an adaptive framework designed specifically for detecting, selecting, and adapting to benign shifts in one-class AD security applications. It employs a distribution-based shift detection method, comparing the data distribution of new upcoming batches against a control set derived from the training data. Upon detecting a shift, OWAD uses a custom optimization function to identify the top δ most significant samples for labeling and adaptation. This function is designed to align the distribution of the control set with that of the incoming batch, achieved by substituting portions of the control set's samples with samples from the incoming batch. After labeling the top δ most critical samples, OWAD updates the underlying DAE through a customized version of the UNLEARN framework [27]. In this framework, UNLEARN precisely regulates updates to the neural network weights, striking a balance between preventing catastrophic forgetting of the old distribution and adapting to the new one.

INSOMNIA. INSOMNIA is an adaptive, incremental learning framework specifically designed to improve the performance of multi-class NIDS amidst changing data distributions. Initially tailored for softmax classifiers, its flexibility allows for extension to any binary classification model that can generate probability scores. INSOMNIA offers two variants: the first requires human intervention, wherein the top δ most uncertain samples in each batch are manually labeled, added to the training set, and followed by retraining of the classifier. The second variant streamlines the process by using a Nearest Centroid Neighbor classifier [93] to estimate labels for new batches, thereby enabling automatic retraining. It is important to highlight that in both variants, shifts are not detected; instead, updates occur with each incoming batch of data. Comparative evaluations conducted by the authors of INSOMNIA underscore the benefits of manual labeling in significantly improving classifier accuracy. Accordingly, we have chosen the manually labeled version of INSOMNIA as the standard for our experiments.

CADE. CADE is a framework designed for detecting, selecting, and explaining shifts within multi-class security applications. It specifically targets concept shifts by identifying new classes, choosing relevant samples for adaptation, and providing explanations for why these samples are marked as shifts. At the heart of the CADE framework is supervised contrastive learning, particularly through the use of DrLIM [38]. This approach trains DrLIM in conjunction with a DAE to optimize the latent space by minimizing distances between samples of the same class while maximizing the distances between different classes. Once trained, CADE employs the Median Absolute Deviation (MAD) [58] method to detect shifted samples. This involves measuring the distance of each sample from the centroids of each training class and applying a class-specific MAD criterion to determine whether a sample significantly diverges from existing classes. A sample is flagged as a shift if it deviates significantly from all the training classes. The samples exhibiting the most significant distance from the classes' centroids are selected, in accordance with the δ selection rate, for labeling and subsequent adaptation. It should be noted that CADE itself does not introduce a novel adaptation function; rather, it employs the standard incremental learning approach to demonstrate the significance of its detection and selection mechanisms.

B.2 Architecture and Training

In both OWAD and Mateen, we employed an identical DAE architecture. Specifically, the encoder followed a progressive reduction in dimensions: from the original dimensions to 75%, then to 50%, followed by 25%, and finally 10%. Conversely, the decoder mirrored this structure in reverse. Each layer was coupled with a Rectified Linear Unit (ReLU) activation function, except for the encoder input and the decoder output layers. It is important to note that this architectural configuration is consistent with the one originally proposed by OWAD, which was inherited from Kitsune [73]. We then trained the DAE utilizing the MSE loss function. The DAE was trained with a batch size of 1024 for 100 epochs, employing the Adam optimizer with a learning rate of 0.00001. Regarding CADE, we adopted the training hyperparameters specified by the authors, with the exception being the architectural design, which remains consistent with that of OWAD and Mateen. These hyperparameters were retained due to their proven effectiveness in a contrastive learning setting, where, for instance, training with longer batches (e.g., 250) is advantageous [15, 16, 49].

Regarding the softmax classifier (INSOMNIA), we followed the hyperparameter search methodology outlined by the authors [4]. This search aimed to determine the best combinations of batch size, learning rate, dropout rate, and the number of neurons per hidden layer, with a limitation of 3 layers. The classifier is trained for 150 epochs using the Adam optimizer, with early stopping conducted after 10 epochs on the validation set.

When it came to updating the models, we configured the number of epochs to be 10% of the original training epochs.

B.3 Hyper-parameter Space and Configuration

The baseline frameworks, OWAD and CADE, require hyperparameter fine-tuning to optimize performance. Consequently, we undertook the following hyperparameter search:

OWAD. We explored the hyperparameter space of the OWAD selection method, with a primary focus on weight hyperparameters. These weights, namely 'acc_wgt,' 'ohd_wgt,' and 'det_wgt,' correspond to ensuring accurate representation of the shifted distribution, reducing the need for manual labeling in test samples, and emphasizing deterministic selection. We conducted empirical evaluations across various combinations, including the original values (5, 10, and 1, respectively), represented as (acc_wgt, ohd_wgt, det_wgt), as well as other configurations such as (3, 3, 3), (2, 3, 5), (3, 2, 5), (5, 2, 3), (5, 3, 2), (3, 5, 2), and (2, 5, 3). The most effective configurations, for a δ of 1%, were identified, such as (3, 2, 5) for IDS17, (3, 5, 2) for IDS18, (5, 10, 1) for Kitsune, (3, 5, 2) for mKitsune, and (2, 5, 3) for rKitsune. We also conducted the same empirical searches for all the considered δ values and experimental scenarios.

CADE. CADE's primary principle is that well-separated label clusters improve the accuracy of detecting shift samples. To achieve this, CADE combines MSE loss and contrastive loss [38], thereby creating distinct representations in latent space. This approach introduces two crucial hyperparameters: m and λ. The parameter m controls the separation distance between representations of different classes, while λ adjusts the contribution of contrastive loss to the overall objective. To optimize CADE, we conducted a hyperparameter search with a focus on the separation of class clusters and the compactness of individual clusters. For this purpose, we employed the Dunn Index, a metric measuring the smallest inter-cluster distance against the largest intra-cluster distance. Our search explored various combinations of m values (1, 5, 10, 15, 20) and λ values (1, 0.1, 0.01, 0.001) to identify the optimal settings that maximize the Dunn Index. The most effective hyperparameters were found to be $m = 1.0$ and $\lambda = 1.0$ for IDS 17, and $m = 10.0$ and $\lambda = 0.01$ for mKitsune.

B.4 Hyper-parameter Sensitivity

We conducted an empirical hyperparameter search for Mateen, starting by setting the ensemble size to 3 and fixing λ_0 at 0.1. Recognizing that representativeness more accurately mirrors the distribution than uniqueness, we uniformly set λ_1 to 1.0. This was followed by a grid search covering σ% of values 30%, 50%, and 90%, and ρ of values 500, 1000, and 1500. Utilizing the optimal values obtained, we further explored the best setting for λ_0 (considering values 0.5 and 1.0), and then determined the most effective ensemble size (options being 5 and 7).

In the following sections, we present the results of our hyperparameter search across the datasets at a δ of 1%, and include a comparison of sensitivity between Mateen and OWAD.

IDS17. Table 5 presents the impact of different σ% and ρ combinations. The most effective combination was identified as a σ% of 30% with a ρ of value 1000. Further analysis was conducted on λ_0 with values of 0.5 and 1.0, yielding results of 96.43 and 94.26, respectively. However, the initially set value of 0.1 for λ_0 proved to be superior. Additionally, we explored varying ensemble sizes, specifically 5 and 7, and found that an ensemble size of 3 yielded the best results. In detail, the original ensemble size of 3 achieved an AUC-ROC of 96.79, while sizes of 5 and 7 resulted in AUC-ROCs of 96.69 and 96.49, respectively. The final optimal parameters were determined as: λ_0 at 0.1, a σ% value of 30%, an ensemble size of 3, and a ρ of value 1000.

ρ	σ%	AUC-ROC
500	30%, 50%, 90%	96.42, 96.39, 95.87
1000	**30%**, 50%, 90%	**96.79**, 95.85, 95.65
1500	30%, 50%, 90%	96.23, 96.63, 95.66

Table 5: The impact of σ% and ρ values on IDS17.

ρ	σ%	AUC-ROC
500	**30%**, 50%, 90%	**98.17**, 97.41, 97.49
1000	30%, 50%, 90%	97.83, 97.64, 97.22
1500	30%, 50%, 90%	97.32, 98.11, 96.92,

Table 6: The impact of σ% and ρ values on IDS18.

ρ	σ%	F1-Score
500	30%, 50%, 90%	96.72, 96.72, 96.61
1000	30%, 50%, 90%	96.79, 96.80, 96.56
1500	30%, **50%**, 90%	97.04, **97.10**, 96.90

Table 7: The impact of σ% and ρ values on Kitsune.

IDS18. Table 6 illustrates the effects of various combinations of σ% and ρ. The optimal combination was found to be 30% for σ and 500 for ρ. Subsequent evaluations of λ_0 at 0.5 and 1.0 produced scores of 97.44 and 97.17, respectively. Therefore, the initial λ_0 setting of 0.1 was deemed most effective. Investigations into different ensemble sizes, notably 5 and 7, revealed that a smaller ensemble of 3 was most efficacious, with an original ensemble size of 3 reaching an AUC-ROC score of 98.17, compared to the 97.42 and 97.76 achieved by ensemble sizes of 5 and 7, respectively. Ultimately, the best-performing parameters were identified as λ_0 set at 0.1, σ at 30%, an ensemble size of 3, and ρ valued at 500.

Kitsune. Table 7 shows the F1 scores from the hyperparameter optimization performed on the Kitsune dataset. In this analysis, the AUC-ROC score was not considered a primary metric due to the limited number of batches containing a mix of labels. Nonetheless, for reference, the average and standard deviation of the AUC-ROC scores are provided in Table 10. The F1 results indicate only minor variations, with the lowest score recorded at 96.56 and the highest at 97.10, achieved with a ρ of value 1500 and a σ% value of 50%. Further assessment was made on the influence of λ_0 and the ensemble size. λ_0 values of 0.5 and 1.0 yielded scores of 96.55 and 97.05 respectively, while ensemble sizes of 5 and 7 led to scores of 96.43 and 96.57 respectively. The initial combination, featuring λ_0 at 0.1 and an ensemble size of 3, continued to show the strongest performance. Consequently, the optimal configuration for this dataset was determined to be a ρ value of 1500, a σ% value of 50%, λ_0 set at 0.1, and an ensemble size of 3.

ρ	σ%	AUC-ROC
500	30%, 50%, 90%	94.17, 93.90, 93.79
1000	30%, 50%, 90%	93.90, 93.85, 94.39
1500	**30%**, 50%, 90%	**94.42**, 93.78, 94.00

Table 8: The impact of σ% and ρ values on mKitsune.

ρ	σ%	F1
500	30%, 50%, 90%	93.95, 92.87, 94.11
1000	30%, 50%, **90%**	93.40, 93.81, **94.13**
1500	30%, 50%, 90%	93.32, 92.90, 93.20

Table 9: The impact of σ% and ρ values on rKitsune.

mKitsune. Table 8 showcases the AUC-ROC outcomes from the hyperparameter optimization conducted on the mKitsune dataset. The highest AUC-ROC value observed was 94.42, achieved with a ρ of value 1500 and a σ% value of 30%. Subsequent analysis focused on various combinations of λ_0 and ensemble size. A λ_0 value of 0.5 yielded a superior AUC-ROC score of 94.76, surpassing both the initial setting of 94.42 and the score obtained with λ_0 at 1.0 (94.02). In terms of ensemble size, the initial value of 3 proved to be the most effective, outperforming sizes of 5 and 7, which resulted in scores of 94.54 and 94.66, respectively. Thus, the final optimal settings for this dataset were determined to be a ρ of value 1500, a σ% value of 30%, λ_0 at 0.5, and an ensemble size of 3.

rKitsune. Table 9 presents the F1 scores obtained from the rKitsune dataset, which encounters a similar challenge to the Kitsune dataset: a limited number of batches is suitable for AUC-ROC computation, primarily due to the prevalence of batches dominated by a single label. Despite this, the average and standard deviation of AUC-ROC for this dataset are available in Table 10 for reference.

Regarding the hyperparameter combinations tested, the pairing of a ρ of value 1000 with a σ% value of 90% emerged as the most effective, achieving an F1 score of 94.13. Subsequent evaluation of λ_0 values revealed that both 0.5 and 1.0 offered slight performance improvements. Specifically, a λ_0 value of 0.5 resulted in an F1 score of 94.38, while a value of 1.0 achieved a score of 94.20. Further analysis of the ensemble size indicated that a size of 7, with an F1 score of 94.52, outperformed the initial value of 3 (F1 score of 94.38). In contrast, an ensemble size of 5 yielded an F1 score of 93.75. Therefore, the final optimal configuration for this dataset is identified as a ρ of value 1000, a σ% value of 90%, λ_0 set at 0.5, and an ensemble size of 7.

Dataset	OWAD		Mateen	
	AUC-ROC	F1-Score	AUC-ROC	F1-Score
IDS17	93.75±0.24	83.69±0.54	96.19±0.64	91.93±0.42
IDS18	90.80±0.97	91.97±0.26	97.61±0.39	97.41±0.04
Kitsune	66.09±3.01	50.29-±25.33	70.39±2.22	96.80±0.22
mKitsune	77.37±2.42	71.21±20.35	94.22±0.34	95.83±0.38
rKitsune	87.10±0.46	89.52±1.14	92.55±0.99	93.80±0.52

Table 10: Comparative evaluation of hyperparameter sensitivity: Mateen vs. OWAD.

Sensitivity Comparison. Table 10 presents the comparative results of the AUC-ROC and F1-Score from the hyperparameter sensitivity analysis of Mateen versus OWAD. The results reveal that Mateen consistently outperforms OWAD across all datasets, as indicated by the higher average scores and lower standard deviations. Notably, Mateen demonstrates minimal variability in its performance, with the exception of the AUC-ROC on the Kitsune dataset, which can be attributed to the limited number of batches containing a mix of labels. We can also observe that OWAD's performance diminishes on datasets with concept shifts, contrasting its relatively stable performance on datasets with covariate shifts. Overall, Mateen exhibits robustness to variations in its hyperparameters, often maintaining strong performance even with suboptimal settings.

C GUIDELINES FOR HYPERPARAMETER SELECTION

Mateen includes multiple hyperparameters that can be adjusted for different scenarios. Below, we provide guidance on how to set these values.

Performance threshold (t): This describes the acceptable performance level. Specifically, if the ensemble's performance on a selected subset exceeds a threshold t, no update is needed. This threshold can be set low (e.g., 90%), resulting in fewer updates, or high (e.g., 99%), leading to more frequent updates. The choice of this value balances desired performance against the computational cost of updates. If computational cost is not a concern, setting a higher threshold is recommended.

Maximum Ensemble Size: This establishes a maximum size for the ensemble. Once this limit is reached, the complexity module activates to merge and remove temporary models. Setting a lower limit, such as 3, means the module activates more often, optimizing the ensemble for recent shifts and potentially enhancing performance. However, this requires additional processing. Conversely, a higher limit, such as 10, results in less frequent activations but requires more storage to maintain more models.

Selection Rate (δ): This parameter sets the percentage of samples from a shifted batch that are selected for manual labeling. A higher setting (e.g., 10%) selects more samples, which can improve ensemble performance but increases the risk of mislabeling, latency, and labor costs. A lower setting (e.g., 0.5%) results in fewer labeled samples, which might lower performance but also reduces labor costs, likelihood of errors, and latency. Our findings indicate that the performance difference between high and low settings is slight, so selecting the lower value is recommended.

Mini-Batch Size (ρ): This represents the size of the mini-batches created from each shifted batch of data. Specifically, we divide each shifted batch into mini-batches of size ρ and then select a few samples from each mini-batch using distance-based measures. A smaller mini-batch size, such as 500, lowers computational costs, whereas a larger size, such as 1500, increases them. The impact on performance varies with the dataset. For instance, the smallest batch size achieves good performance at lower costs in the IDS17 and mKitsune datasets, but larger sizes can slightly improve performance if resources permit.

Retention Rate (σ%): This specifies the percentage of samples kept from a mini-batch to create a condensed version. We form

this condensed mini-batch by retaining σ% of the samples and discarding the remaining 100% − σ%, which are similar and thus redundant. The choice of σ is based on the level of redundancy observed in the network. For the datasets examined in this paper, setting σ at 30% generally results in good performance. However, it is important to note that this setting results in the removal of a substantial portion of the samples, specifically 70%.

Uniqueness vs. Informativeness (λ_0 & λ_1): These parameters assign weights to Uniqueness ($\hat{U}$) and Informativeness ($\hat{I}$) during the sample selection process for manual labeling. $\hat{I}$ indicates how many samples are similar to a particular sample, while $\hat{U}$ measures how distinct a sample is from the rest. Setting $\hat{U}$ to a high value and $\hat{I}$ to a low value prioritizes the selection of outliers. Conversely, setting $\hat{I}$ high and $\hat{U}$ low focuses the selection on more common, similar samples. Typically, $\hat{I}$ is set high (e.g., 1) to ensure the model learns from the majority of data, and $\hat{U}$ is set low (e.g., 0.1 or 0.5) to select only a few outliers.