Rasd: Semantic Shift Detection and Adaptation for Network Intrusion Detection

Fahad Alotaibi(✉) and Sergio Maffeis

Department of Computing, Imperial College London, London, UK
{f.alotaibi21,sergio.maffeis}@imperial.ac.uk

Abstract. Network Intrusion Detection Systems (NIDSs) based on Deep Neural Network have demonstrated impressive performance in multi-class, closed-world settings, where training and test data follow the same distribution. However, when deployed in real networks, these systems have a limited ability to detect novel attacks which do not belong to already known classes. In this work, we aim to tackle *semantic shift*, that is the emergence of unknown classes, by proposing a two-phase approach to *detect* new classes and *integrate* them into the classification model, while minimising the need for human intervention. While contrastive learning is a promising techniques to tackle semantic shift, it has high computational cost and it is sensitive to imbalanced data. We propose a novel contrastive learning approach based on synthetic centroids which has low computational cost and is robust to class imbalance, making it suitable for application to NIDS. To integrate the shifted samples in the existing model, we also design a novel adaptation method that combines manual labeling and pseudo-labeling to reduce labeling costs. We evaluate our system, *Rasd*, on two NIDS datasets, finding it excels in both detection and adaptation. For example Rasd improves on the nearest detection baseline F1-score by 6.83% for IDS 2017 and 19.21% for IDS 2018.

Keywords: Distribution Shift · Shift Detection and Adaptation · Network Security · Intrusion Detection

1 Introduction

Computer networks are vital for modern organizations, facilitating efficient information and resource exchange. However, they are often targets of cyberattacks, with 39% of UK organizations experiencing cybersecurity incidents [5]. Protecting these networks is essential to safeguard sensitive data and restrict unauthorized access. Network Intrusion Detection Systems (NIDS) are commonly used for security, but their traditional signature-based approach, involving threat collection, analysis, and periodic off-line NIDS updating, is becoming increasingly challenging due to the growing complexity of network communication and attack

Published by Springer Nature Switzerland AG 2024
N. Pitropakis et al. (Eds.): SEC 2024, IFIP AICT 710, pp. 16–30, 2024.
https://doi.org/10.1007/978-3-031-65175-5_2

patterns. To address these limitations, Machine Learning (ML) techniques are being employed to automate NIDS. However, designing an effective ML-based NIDS still demands significant manual effort in modeling attacks and defining relevant features. Deep Neural Networks (DNNs) offer a solution by learning from raw data, and recent research shows promising results in developing DNN-based NIDS (e.g. [9]).

DNN-based NIDS typically operate on the closed-world assumption, meaning that training and testing data come from the same distribution [1,9]. This assumption is violated by network security data. Network traffic is diverse and unpredictable, influenced by a mix of benign and malicious activities which frequently lead to distribution changes. Such variability degrades the performance of learning-based systems in real-world scenarios, as these systems are not fundamentally designed to adapt to constant changes [1]. These changes are known as *distribution shift*, and can be divided into *covariate shift* (changes in the distribution of features within a class) and *semantic shift* (emergence of new classes). Distribution shift has been proven to impact learning-based security applications severely [1,4,7,11,15]. In security settings, both types of shift are prevalent.

Covariate shift typically represents changes in behaviour from known users or attackers. It is believed to be a consequence of not defining a robust feature space, and results in slow model performance degradation [11]. While semantic shift may have a limited adverse impact on binary classification, it severely impacts multi-classification as new classes will be predicted as one of the training classes. In NIDS multi-classification settings, this can result in new attacks being incorrectly identified as known attacks or, worse, as benign. This challenge leads us to focus on the identification of new classes, and their integration into a multi-class NIDS classifier. Suppose we have a classifier $\mathcal{M}$ which has been trained on classes $\mathcal{C} = \{C_1, \ldots, C_n\}$. During inference, $\mathcal{M}$ encounters a mixture of both familiar classes from $\mathcal{C}$ and a new class C_{n+1}. The key problems to solve are how to effectively differentiate samples of the new class C_{n+1} from those of the existing classes in $\mathcal{C}$, and how to subsequently update $\mathcal{M}$ to incorporate C_{n+1}.

Our solution is the *Rasd* framework, which integrates a shift detector alongside an existing NIDS classifier. Rasd identifies key semantic shifts for human labeling, trains a pseudo-labeler with these labeled samples, and uses the pseudo-labeler to label the remaining shift samples. Both labeled and pseudo-labeled samples are then used to retrain the original classifier, enabling it to adapt to emerging threats.

Our main contributions are:

- We designed a novel shift detection model, that accurately detects semantically shifted network flows with high recall and diversity, based on a novel cost-effective contrastive learning approach relying on synthetic centroids.
- We defined a new semantic shift adaptation model that selects the most informative shifted samples for human labeling using the Farthest-First Traversal algorithm and leverages these labeled samples to pseudo-label the non-selected samples.

– We implemented our models as the Rasd framework for shift detection and adaptation. We evaluated Rasd on two NIDS datasets, showing that it outperforms baselines and related approaches. For instance, on the IDS2018 dataset [13], Rasd improves the detection F1-score by at least 19.21% and the adaptation macro F1-score by at least 8.24%, with only 1% of samples requiring human labeling.

Section 2 provides an overview of the relevant background and related literature. Section 3 describes Rasd methodology. Section 4 presents the evaluation results of Rasd and the related baselines. Finally, Sect. 5 concludes the paper.

2 Background

Distribution shift refers to a change in the underlying probability distribution between the training and test sets. In the context of NIDS, distribution shift occurs when the network data in the training set does not accurately represent the data observed during deployment. It can occur as both *covariate* and *semantic* scenarios [8]. *Covariate shift* occurs when the distribution of features changes while the underlying concept remains the same. For example, when an attacker strategically alters their behavior to manipulate the feature space, the classifier is deceived into misinterpreting this malicious behavior as benign. *Semantic shift* represents a fundamental change in the underlying concept, such as the emergence of a new attack or a concept flip (e.g., malicious features turn to be benign and vice-versa).

Contrastive learning is a type of representation learning that focuses on learning robust representations by contrasting and comparing different examples [3,14]. It aims to choose an *anchor* sample and reduce its distance to similar samples, or *positives* (e.g. those with the same label as the anchor), while increasing its distance from dissimilar samples, or *negatives* (e.g. those with different labels). It is applicable in both supervised and unsupervised learning contexts. In unsupervised settings, such as in the SimCLR framework [3], it leverages data augmentation to learn effective data representations without labels. Conversely, in supervised learning, such as in the Lifted Structured Loss (LSL) framework [14], it utilizes labels to push same-label samples together and push them from different label samples.

2.1 Related Work

Semantic shift poses two major challenges: firstly, detecting the shifted samples, and secondly, adapting to these samples with minimal human effort.

To detect shifts, two main approaches have been proposed recently: confidence-based [1] and distance-based [15]. Confidence-based methods rely on the confidence scores of a DNN to pinpoint uncertain predictions as potential semantic shifts. Yet, this method faces limitations, as DNNs tend to be overconfident in unrelated inputs [10]. Distance-based methods, such as contrastive

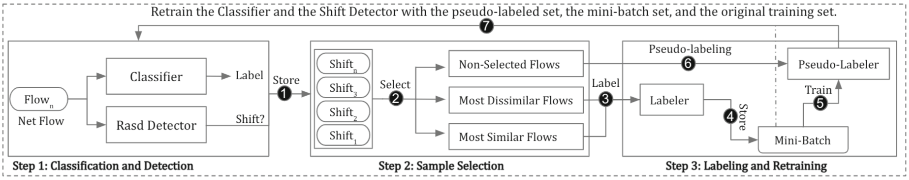


Fig. 1. Overview of the Rasd framework.

learning, develop a distance function for identifying semantic shifts. Although effective [4,15], these demand significant computational resources for extensive training on large batches and prefer balanced data to generate robust representations [3].

To adapt these shifts, pseudo-labeling [1] and prioritization [15] have been suggested. Pseudo-labeling, as used in [1], employs a binary model trained on known classes to categorize shifted samples as either malicious or benign. This technique relies on the nearest centroid for pseudo-label assignment but requires robust feature engineering and faces challenges in translating binary labels to a multi-classification context. Conversely, [15] prioritizes manual labeling for samples furthest from the centroids, constrained by a user-defined budget. This approach, however, might struggle with noisy shift, as distant samples in noisy scenarios often share labels.

3 The Rasd Framework

Rasd encapsulates a supervised network flow classifier and provides a framework for shift detection, pseudo-labeling, and retraining. The diagram in Fig. 1 illustrates the key steps, which are briefly described below.

Inside the *Rasd Detector*, an encoder is trained to map labeled samples together by minimizing their distance to the label centroid. The centroid can be pre-defined automatically or determined manually by data analysts. Following training, thresholds for each class centroid are set based on the k-th percentile. During deployment, the distance between the incoming sample and each class centroid is calculated and compared to the thresholds. If the sample distance to all the classes exceeds the per-class thresholds, the sample is identified as a semantic shift sample and stored in a pool.

The pool of semantic shift samples is then processed by the *Rasd Adaptation* method, which sequentially divides it into small batches and selects the most informative samples for manual labeling, while respecting a cost constraint. The manually-labeled samples are subsequently utilized to train the *Pseudo-Labeler* oracle, which will be used to pseudo-label the non-selected samples. By a combination of manual labeling and pseudo-labeling, we thus obtain labels for all of the semantic shift samples, which can be used to retrain the original classifier and the detector.

3.1 Distance Learning for Shift Detection

Learning Robust Latent Representations. At the core of the *Rasd Detector* is our novel, cost-effective centroid-based loss. This loss function is inspired by the principles of contrastive losses [3,4,14,15], but it stands out due to its reduced computational cost (Table 1) and reduced sensitivity to data imbalance (Table 3). Our insight is that instead of contrasting class samples with other samples from other classes, we define a centroid for each class and minimize the distance of samples to their label centroid, as illustrated in Fig. 2.

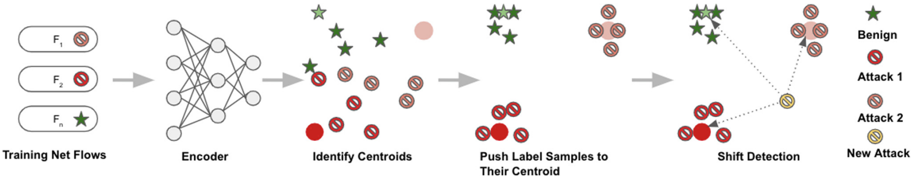


Fig. 2. The high-level idea of Rasd loss function for shift detection.

Specifically, our approach employs an Encoder to transform network flows into low-dimensional latent space representations, thereby addressing the *curse of dimensionality* [2]. The Encoder is then optimized to ensure that representations with identical labels are proximate, while those with differing labels are distant. To achieve this, we introduce a loss function that incorporates predefined synthetic centroids. Our loss function takes three inputs: Z, Y, and C. Z represents the set of all latent representations of network flows in a batch, defined as $Z = \{z_1, z_2, \ldots, z_n\}$. Y denotes the corresponding set of labels for these network flows, expressed as $Y = \{y_1, y_2, \ldots, y_n\}$. C denotes the set of all pre-defined centroids. Let k be the number of unique labels in the batch, then $C = \{c_1, c_2, \ldots, c_k\}$.

For each unique label y_j within the batch, the average squared distance of the elements in Z having that label to their corresponding centroid c_{y_j} is calculated as follows:

$$\text{Label\_Loss}_{y_j}(Z, Y, C) = \frac{\sum_{i=1}^{n} \left(\delta(y_i, y_j) \times ||z_i - c_{y_j}||^2\right)}{\sum_{i=1}^{n} \delta(y_i, y_j)}$$

where $\delta(y_i, y_j)$ is the Kronecker delta function, which is 1 when $y_i = y_j$ and 0 otherwise, and c_{y_j} is the centroid associated with label y_j. Finally, our Rasd loss is defined as:

$$\text{Rasd}(Z, Y, C) = \frac{1}{k} \sum_{j=1}^{k} \text{Label\_Loss}_{y_j}(Z, Y, C)$$

Centroids Search. We need to define the synthetic centroids to train the Encoder using our loss function. To achieve this, we implement a Genetic Algorithm (GA) to maximize the separation between centroids. The GA takes two main inputs: a fitness function for optimization and a set of initial centroids as the starting point. Our fitness function is defined as $-\min(\text{dist}(C_i, C_j))$, where *dist* refers to the Euclidean distance, and C represents either the initial centroids or the evolving centroids generated during the GA process. To generate the initial centroids, we use the untrained Encoder to project the training data into a latent space, then we use KMeans++ to set up the initial centroids. This step is crucial to avoid numerical instability that might arise from randomly assigned centroids, as it provides a more stable and representative starting point for GA optimization. Additionally, we find that a relatively modest number of GA generations, specifically 50, achieves well-separated centroids with minimal computational complexity (see Appendix A.2 and Sect. 4.2).

In summary, our learning approach starts with the Encoder –before training– and KMeans++ for the initial centroid definition, followed by GA optimization to identify the most distant centroids. After identifying the centroids, we can proceed to train the Encoder using the Rasd loss function.

Detecting Semantic Shift Samples. Post-training with the Rasd loss, the Encoder functions as a semantic shift detector. Specifically, before deployment, we map training samples into latent space using the Encoder. For each class, we calculate its centroid in Euclidean space and establish a class-specific threshold at the k-th percentile of the distances between the class sample's latent representations and the centroid. During inference, we map each incoming sample into latent space and measure its distance to each class centroid. A sample is identified as a shift if its distance exceeds all the thresholds.

3.2 Semantic Shift Adaptation

The identified samples of semantic shift can be then used in updating both the classifier and the shift detector. However, given the typically large volume of these detected samples, manually labeling each one is impractical. To address this, Rasd employs a two-step approach to minimize the need for human labeling: first, through sample selection, and second, by applying pseudo-labeling.

Sample Selection. To ensure that the selected samples are both representative and diverse, covering most of the data distribution and encompassing all detected classes, we employ the Farthest-First Traversal (FFT) algorithm [12] from computational geometry. This algorithm starts with a randomly chosen sample and progressively selects the sample that is farthest from those already chosen. In this process, we utilize the latent representations generated by the Detector, employing Euclidean distance to identify the farthest samples.

However, FFT presents two primary challenges: significant computational demands when processing large data batches and a heightened sensitivity to outliers. To mitigate these challenges, we adopt a two-phase approach. First, we partition the detected samples into sequential batches to reduce the computational cost. Second, to reduce the influence of outliers, we calculate a cosine similarity score for each sample relative to others in the same batch. Based on these scores, we arrange the samples and divide them equally into two groups: one comprising the most similar (higher scores) and the other containing the most dissimilar samples (lower scores). FFT is then applied separately to each group within every batch.

Pseudo-labeling and Retraining. When the selection is done, we construct a labeled mini-batch that contains the selected samples ground-truth labels, which are given by a human annotator. We could retrain the classifier using both the original training data and the mini-batch data. However, since the mini-batch contains only a few samples from each new class, there is a risk that the classifier would develop a bias toward the more prevalent, older classes. To mitigate this, we cannot increase the number of samples selected for manual labeling, as it would be too expensive. Instead, we use pseudo-labeling: we train an oracle model on the labeled mini-batch and use this oracle to pseudo-label all detected unlabeled samples. We then retrain the classifier and the shift detector using the pseudo-labeled data, mini-batch data, and original training data. In this way, we leverage all detected samples to update the models, enhancing their generalizability to new classes.

4 Evaluation

In this Section, we describe the experimental setup and present the main results. We used the revised IDS2017 [6] and IDS2018 [13] datasets processed as stated in Appendix A.1. Our code, datasets, and models are publicly available[1].

4.1 Experimental Setup

Baselines. For our detection baselines, we chose the cutting-edge semantic shift detector, CADE [15], and the LSL [14] —a recent contrastive loss— as another baseline. The process of hyperparameter search for each model is detailed in Appendix A.2. For thresholding, we set all the methods to accept a 7% error, which maximizes the relative performance across methods and datasets (see Fig. 3). We conducted the experiments on a machine with an AMD EPYC Processor with 16 cores, 70 GB of RAM, and 2 NVIDIA A30 GPUs with 24 GB of memory each. To evaluate adaptation strategies, we selected two baselines for sampling: Uncertainty Sampling (US) as proposed in [1], and the CADE strategy, which selects the farthest samples (FS) to the classes centroid. Specifically, CADE computes the distance of the detected semantic shift samples to their nearest centroid and ranks them in descending order. We set sample selection rates ranging from 1–5%. Since the selected samples are few, we cannot employ a DNN classifier as an oracle; instead, we used a random forest classifier. To train the random forest, we relabeled the selected known samples as 0, regardless of their ground truth labels. For the main classifier retraining, we selected 70%

[1] https://github.com/ICL-ml4csec/Rasd.

Table 1. Semantic shift detection performance of Rasd and baselines.

Dataset	System	FPR	Recall	mRecall	Precision	F1	Diversity	Accuracy	Train Cost
IDS2017	CADE	**8.75**	67.82	60.81	68.87	68.34	100	86.03	1.47 h
	LSL	9.51	71.24	67.56	68.15	69.66	100	86.21	1.20 h
	Rasd	10.84	**85.43**	**89.47**	**69.24**	**76.49**	**100**	**88.32**	**40.62 mins**
IDS2018	CADE	18	60.25	64.55	52.65	56.19	100	76.57	52.75 mins
	LSL	**9.16**	60.99	57.93	68.86	64.69	100	83.39	41.08 mins
	Rasd	12.45	**99.35**	**77.59**	**72.6**	**83.9**	**100**	**90.49**	**24.70 mins**

of the pseudo-labeled samples for retraining and 30% for testing. We relabeled the samples predicted as known (i.e., 0) using the current version of the main classifier (i.e., before retraining).

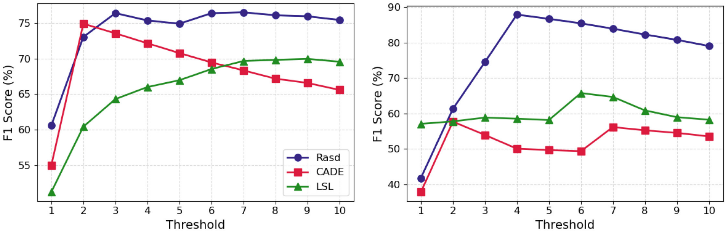
Evaluation Metrics. We consider the shift samples as positives and the known samples as negatives. We calculate both global and unweighted metrics. Specifically, for the detection, we calculated precision, recall, diversity (the ratio of detected unique shift classes to the total unique shift classes in the dataset), and accuracy as global metrics. Additionally, we calculated macro recall as an unweighted metric to identify the performance of the detectors on minority classes. For the adaptation, we focus on unweighted metrics due to the post-detection imbalance in our dataset, where some classes have few samples. This avoids bias in performance measures from class frequency-weighted or global metrics. Therefore, we calculate macro F1, macro precision, and balanced accuracy.

4.2 Rasd Detector Performance

We evaluate the *Rasd Detector* and compare it to the selected baselines. Our comparison, which is detailed below, shows that Rasd outperforms all baselines in detection metrics and computational efficiency.

Table 1 presents the detection results of the Rasd and the selected baselines. Rasd excels among detectors, offering lower computational demands and superior performance in most detection metrics. Specifically, when looking at the recall at the IDS2017 dataset, we can notice that Rasd outperformed the closest system by 14.19%. The difference is even higher when looking at the IDS2018 dataset, where the difference between Rasd and LSL is 38.36%. The other metrics are also favoring Rasd, however, with less differences when compared to the recall. For example, in the IDS2017 dataset, Rasd accuracy outperforms LSL by only 2.11%. However, when looking at the False Positive Rate (FPR), Rasd performs worse than the baselines.

Threshold rate significantly impacts performance; we explored rates from 1% to 10%, as shown in Fig. 3. Rasd typically outperforms other methods across most thresholds, although it can occasionally perform less, particularly at lower rates. For example, at a 1% threshold in the IDS2018 dataset, LSL surpasses Rasd, but



(a) IDS2017

(b) IDS2018

Fig. 3. F1 performance of Rasd detector and baselines over different thresholds.

Table 2. Performance comparison of Rasd and the baselines in detecting semantic shifts, with selecting the best model and a targeted false positive rate of 10%.

Dataset	System	Recall	mRecall	Precision	F1 score	Diversity	Accuracy
IDS2017	CADE	87.97	63.69	71.2	78.7	100	89.41
	LSL	77.17	76.93	68.13	72.37	100	86.9
	Rasd	**92.87**	**91.18**	**72.26**	**81.28**	**100**	**90.49**
IDS2018	CADE	55.91	61.31	64.29	59.81	100	81.25
	LSL	61	58.19	66.5	63.63	100	82.61
	Rasd	**99.22**	**74.55**	**76.4**	**86.32**	**100**	**92.16**

Rasd performs better in the IDS2017 dataset. Rasd's best performance, 87.87% at a 4% rate in IDS2018, notably exceeds the closest baseline, LSL, which reaches 65.78% at 6%.

The detection experiment outcomes might be influenced by the hyperparameter optimization approach (see Appendix A.2) and false positive rates. As Table 1 shows, Rasd has a higher FPR, suggesting improved performance in the other metrics. We thoroughly evaluate and select hyperparameters for each baseline model based on the best F1 score, then adjust the FPR to a fixed 10%. This setup, while not entirely realistic, aims to create a controlled environment to identify the most efficient model through thresholding and hyperparameter search, keeping the false positive rate constant.

Table 2 shows the detection results of Rasd as well as the considered baselines. In the IDS2017 dataset, we can notice that all of the baselines' metrics have improved when compared to those in Table 1. However, Rasd still has a higher detection performance. Conversely, the baselines in the IDS2018 are showing different conclusions. CADE performance has decreased, which is due to fixing the FPR at 10%, where previously CADE had an FPR of 18% (ref. Table 1). LSL performance, on the other hand, has increased as the FPR at rate 10% is higher when compared to the one in Table 1. Rasd has a slight decrease in the recall metrics, however, with a higher increase in the others, which implies that Rasd can distinguish semantic shift samples from the known samples accurately.

Table 3. Evaluation of latent representations quality via clustering metrics.

System	IDS2017			IDS2018		
	SS	NMI	Acc	SS	NMI	Acc
CADE	69.42	38	64.82	71.17	14.37	34.34
LSL	57.62	59.69	79.85	56.52	66.36	79.02
Rasd	**80.59**	**85.78**	**96.3**	**84.58**	**92.32**	**98.46**

Although we selected the best models for all the baselines, they still performed less than Rasd. We hypothesize that the contrastive functions may not have effectively differentiated between the classes. This hypothesis aligns with current research, which suggests that these functions are susceptible to data imbalances, potentially leading to the misclassification of new classes as existing ones [4]. To further investigate this hypothesis, we examine the latent representation quality of training samples using K-Means++ and clustering metrics. The underlying assumption is that if these representations are well-separated by their labels, then they should be clustered with precision. We evaluate using three metrics: Normalized Mutual Information (NMI), clustering accuracy, and the intrinsic Silhouette Score (SS). NMI measures the correspondence between two sets of clusters, while SS assesses the compactness of clusters and their separation using predicted labels, irrespective of ground-truth labels. In our analysis, SS contrasted with accuracy reveals cluster quality. High SS but low accuracy indicates distinct yet mixed-label clusters, and high NMI suggests a consistent mislabeling pattern.

Table 3 displays the clustering performance. Rasd consistently excels across both datasets, exemplifying its excellence in forming distinct clusters and pinpointing labels accurately. Take the IDS2018 dataset, for instance: Rasd achieves an SS of 84.58% coupled with a 98.46% accuracy rate. Furthermore, its notable NMI score of 92.32% reinforces Rasd's cluster assignment consistency. CADE's performance is inconsistent across datasets. In IDS2017, its SS, accuracy, and NMI are 69.42%, 64.82%, and 38%, respectively, indicating that 35.18% of its latent representations are randomly scattered, affecting clustering. In IDS2018, CADE shows reduced accuracy and NMI, suggesting poor differentiation of latent representations. In contrast, LSL consistently performs well in both datasets, with accuracies above 79% and closely aligned SS and NMI scores. For example, in IDS2018, LSL achieves an SS of 56.52% and an NMI of 66.36%, indicating effective label discernment but with some potential for misassignments due to closely packed labels.

4.3 Rasd Adaptation Strategy Performance

We analyze the performance of the *Rasd Adaptation* strategy against the baselines. Our results show that the proposed method demonstrates its effectiveness in adapting to semantic shift samples.

Table 4. A comparison of the baseline selection strategies, US and FS, with Rasd.

Batch Size	System	IDS2017				IDS2018			
		Diversity	mF1	mPrecision	bAccuracy	Diversity	mF1	mPrecision	bAccuracy
1000	US	71.42	73.46	78.41	70.55	33.33	50.74	56.07	53.61
	FS	85.71	63.23	72.71	64.68	66.66	49.81	53.87	52.3
	Rasd	**100**	**75.88**	**82.21**	**76.58**	**83.33**	**60.13**	**59.25**	**62.62**
2000	US	85.71	79.44	82.05	78.85	33.33	52.07	56.54	54.12
	FS	**100**	68.21	79.6	69.07	66.66	49.81	53.38	52.3
	Rasd	85.71	**79.72**	**85.69**	**81.25**	**83.33**	**62.34**	**65.09**	**64.5**
3000	US	71.42	63.13	64.89	62.18	33.33	51.36	56.3	53.18
	FS	100	74.1	86.5	75.3	66.66	49.65	53.21	52.3
	Rasd	**100**	**83.68**	**88.45**	**83.11**	**100**	**67.85**	**75.05**	**70.09**

Table 5. Comparing Rasd and baselines using low selection rates of 1% to 4% and a batch size of 3,000.

Selection Rate	Strategy	IDS2017				IDS2018			
		Diversity	mF1	mPrecision	bAccuracy	Diversity	mF1	mPrecision	bAccuracy
1%	US	57.14	60.76	61.75	62.13	33.33	47.43	50.04	53.62
	FS	71.42	64.91	72.09	66.86	50	48.54	52.25	51.64
	Rasd	**85.71**	**76.9**	**81.57**	**76.73**	**66.66**	**56.78**	**61.26**	**61.45**
2%	US	71.42	66.57	68.86	66.75	33.33	48.95	51.75	55.04
	FS	85.71	62.53	70.41	64.53	66.66	51.95	56.55	52.31
	Rasd	**85.71**	**78.72**	**80.16**	**79.98**	**66.66**	**59.09**	**57.76**	**61.57**
3%	US	71.42	68.12	68.33	69.41	33.33	47.6	51.5	54.11
	FS	100	71.75	**87.79**	70.69	66.66	50.1	53.46	52.53
	Rasd	**100**	**80.13**	87.02	**79.8**	**83.33**	**61.05**	**60.81**	**64.07**
4%	US	71.42	68.59	69.81	68.31	33.33	52.67	57.43	54.54
	FS	100	71.07	**92.75**	70.05	66.66	53.2	58.63	53.66
	Rasd	**100**	**82.89**	88.86	**82.74**	**83.33**	**65.48**	**66.39**	**66.5**

Table 4 displays the results of the Rasd strategy compared to selected baselines across various batch sizes, with a fixed selection rate of 5%. The results underscore significant disparities between the datasets. Across all strategies, the performance on the IDS2017 dataset consistently surpasses that on the IDS2018 dataset. This divergence can be attributed to Rasd's low micro recall ($\approx$ 78%) for this dataset, as elaborated in Table 1. This means we have only few samples detected from minor semantic shift classes.

However, Rasd consistently outperforms all baselines across batch sizes and datasets. Its best performance is observed at a batch size of 3,000. Yet, at a batch size of 2000 for the IDS2017 dataset, Rasd exhibits slightly less diversity compared to FS, but still surpasses FS in other classification metrics. The performance of FS tends to improve with larger batch sizes, except for balanced accuracy on IDS2018. In contrast, US achieves its optimal performance on the IDS2017 with the smallest batch size. A consistent observation with US is its diminished diversity performance, which has been highlighted previously in [1].

The selection rate signifies a trade-off: while higher rates increase labeling costs, they also suggest potential system enhancements. Therefore, we assess the effects of reduced selection rates on the strategies. We concentrate on the batch size of 3000 while adjusting the selection rates between 1% and 4%.

Table 5 presents the performance of the strategies on both datasets. For IDS2017, all strategies improved at 1% and 3% selection rates, with minor differences between 1–2% and 3–4%. Consider Rasd, which reflects advancements ranging from 0.56% to 3.25% across most metrics. However, its precision did register a decline of 1.41%. The disparity between 1% and 4% is more pronounced, leading to an uptick in most classification metrics across all strategies. For instance, balanced accuracy witnessed increases of 6.01, 3.2, and 6.18 percent for Rasd, FS, and US, respectively. When contrasting the 4% and 5% rates (see Table 4), outcomes are contingent on the strategy. Rasd exhibits a slight increase, FS experiences an uptick in F1 and balanced accuracy (ranging from 3–5%), but a drop in precision. Conversely, the performance of US deteriorates. The results from the IDS2018 draw distinct conclusions. While FS and US exhibit marginal enhancements with an increased selection rate, Rasd experiences a notable boost at a 4% selection rate compared to 3%. Moreover, when contrasting the 4% and 5% selection rates, the latter evidences a relative increase in classification metrics.

5 Conclusion and Future Directions

We presented Rasd, a novel framework comprising two complementary components that enable the detection and adaptation of semantically shifted network flows. Rasd identifies semantic shift samples in real-time and selects the most informative samples from the detected set for manual labeling, while assigning pseudo-labels to the remaining non-selected samples. Our evaluation shows that, on the IDS2018 dataset, the Rasd detector outperforms the baselines by margins of 7.1%, 38.36%, and 19.21% in accuracy, recall, and F1, respectively. The Rasd adaptation strategy guarantees high diversity and superior performance compared to baseline approaches. Specifically, a classifier updated with both labeled and pseudo-labeled samples attains over 80% performance across all considered metrics when using a 5% selection rate and a batch size of 3,000 on the IDS2017 dataset.

Although we achieved robust results, our approach encountered two primary challenges. Firstly, some of the pseudo-labels used to update the models were incorrectly labeled, introducing noise to the updates. Secondly, the selection rates depended on the detected shift sample size, leading to thousands of samples being selected for manual labeling in our setting. For example, Rasd identified 368,947 samples as shift samples within the IDS2017 test set, which at a 1% selection rate, required approximately 3,689 samples for manual labeling. To enhance efficiency in future efforts, we aim to limit the number of samples selected for human labeling to just a few dozen, while the remaining non-selected samples will be labeled using an oracle. However, this strategy could increase the occurrence of mislabeled pseudo-labels. We therefore plan to investigate methods to minimize the impact of these errors when updating the models.

A Appendix

A.1 Datasets Processing

We evaluated the performance using the revised IDS2017 [6] and IDS2018 [13] datasets. The datasets were divided into two sets: Set 1 of IDS2017 contains data from the first three days, while Set 2 includes data from the subsequent two days; Set 1 of IDS2018 consists of data from the first week, and Set 2 comprises data from the second week. Consequently, each set contains different attacks, representing the semantic shift classes.

For IDS2018, due to its large size (over 16 million flows), we randomly sampled 33% from each label. We also adjusted the high imbalance in Set 2 of IDS2018 by reducing the benign traffic to 71.42%. Table 6 displays the statistics and distribution of the data after division. Additionally, we randomly transferred 20% of the data from Set 1 to Set 2, using these modified sets as our training and testing sets, respectively. This setup allows us to assess the performance of our framework against both known and new attack types.

Table 6. Datasets distribution and statistics.

Dataset	Set	Days	Distribution	Flows No.
IDS2017	1	Mon, Tue, and Wed	Benign (84.98%), DoS Hulk (13.31%), DoS GoldenEye (0.63%), FTP-Patator (0.33%), DoS Slowloris (0.33%), SSH-Patator (0.25%), DoS Slowhttptest (0.14%), and Heartbleed (0.0009%)	1,190,531
	2	Thu and Fri	Benign (71.96%), Port scan (17.48%), DDoS (10.44%), Bot (0.08%), Brute force (0.01%), Infiltration (0.003%), XSS (0.002%), and SQL Injection (0.001%).	910,283
IDS2018	1	02–14, 02–15, 02–16	Benign (83.09%), DoS Hulk (10.02%), FTP-Brute Force (0.0067%), SSH-Brute Force (4.43%), DoS SlowHTTPTest (0.0074%), DoS GoldenEye (1.95%), and DoS Slowloris (0.47%)	698,771
	2	02–20, 02–21, 02–22, 02-23	Benign (71.42%), DDoS LOIC-HTTP (19.79%), DDoS HOIC (8.68%), Bot (0.059%), Infiltration (0.020%), DDoS LOIC-UDP (0.0079%), and Brute Force - Web (0.0027%)	960,137

A.2 Hyperparameter Search

For the classifier, we followed the hyperparameter search strategy proposed in [1]. For the detectors, we used an Encoder comprising three hidden layers sized at 75%, 50%, and 25% of the input dimension, respectively. The output layer is set at 10% of the input dimension. The Decoder in CADE mirrors the Encoder structure. We used the Adam optimizer at a 0.0001 learning rate for 250 epochs with 1024 batch size, following current contrastive learning studies which suggest larger batch sizes and longer training improve performance [3,4,15].

In our context, hyperparameter optimization is challenging due to the unreliability loss scores. For example, CADE adjusts the contrastive loss weight with λ and sets dissimilar pair distances with a margin m, increasing the values of both parameters might lead to elevated loss scores. Our hyperparameters search strategy evaluates contrastive loss effectiveness in shift detection based on label sample proximity and clear separation between different label clusters [4,15]. We use the Dunn Index to identify optimal hyperparameters, measuring the ratio of the smallest inter-cluster to the largest intra-cluster distance.

For CADE's hyperparameter optimization, we conducted a grid search, testing λ and m values as suggested by CADE's authors. Our search included m values from $\{1, 5, 10, 15, 20\}$ and λ values from $\{1, 0.1, 0.01, 0.001\}$, resulting in 20 combinations. The best parameters for the IDS2017 dataset were $m = 20$ and $\lambda = 1.0$, while for IDS2018, were $m = 1.0$ and $\lambda = 0.1$. While tuning hyperparameters for LSL, we explored the following range for m: $\{1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0\}$. The best margin for the IDS2017 dataset was 1.5, while for IDS2018, was 2.25.

Although the Rasd loss lacks a margin value, our centroid optimization uses a genetic algorithm, focusing on two key hyperparameters: population size and generations. We fixed the population size at 100 and tested generations across $\{50, 100, 150, 200, 250, 300, 350, 450, 500, 550\}$, finding 50 generations optimal for both datasets.

References

1. Andresini, G., Pendlebury, F., Pierazzi, F., Loglisci, C., Appice, A., Cavallaro, L.: INSOMNIA: towards concept-drift robustness in network intrusion detection. In: 14th ACM Workshop on Artificial Intelligence and Security, AISec@CCS (2021)
2. Boukerche, A., Zheng, L., Alfandi, O.: Outlier detection: methods, models, and classification. ACM Comput. Surv. **53**(3), 55:1–55:37 (2021)
3. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.E.: A simple framework for contrastive learning of visual representations. In: Proceedings of the 37th International Conference on Machine Learning, ICML (2020)
4. Chen, Y., Ding, Z., Wagner, D.A.: Continuous learning for android malware detection. In: 32nd USENIX Security Symposium, USENIX Security (2023)
5. Ell, M., Gallucci, R.: Cyber security breaches survey 2022 (2022). https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2022/
6. Engelen, G., Rimmer, V., Joosen, W.: Troubleshooting an intrusion detection dataset: the CICIDS2017 case study. In: IEEE Security and Privacy Workshops, SP Workshops (2021)
7. Han, D., et al.: Anomaly detection in the open world: normality shift detection, explanation, and adaptation. In: 30th Annual Network and Distributed System Security Symposium, NDSS (2023)
8. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: a review. IEEE Trans. Knowl. Data Eng. **31**(12), 2346–2363 (2019)
9. Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A.: Kitsune: an ensemble of autoencoders for online network intrusion detection. In: 25th Annual Network and Distributed System Security Symposium, NDSS (2018)
10. Nguyen, A.M., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2015)
11. Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L.: TESSERACT: eliminating experimental bias in malware classification across space and time. In: 28th USENIX Security Symposium, USENIX Security (2019)
12. Rosenkrantz, D.J., Stearns, R.E., Lewis, II, P.M.: An analysis of several heuristics for the traveling salesman problem. SIAM J. Comput. **6**(3), 563–581 (1977)

13. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: 4th International Conference on Information Systems Security and Privacy, ICISSP (2018)
14. Song, H.O., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2016)
15. Yang, L., et al.: CADE: detecting and explaining concept drift samples for security applications. In: 30th USENIX Security Symposium, USENIX Security (2021)