Received 2 October 2022, accepted 7 November 2022, date of publication 17 November 2022, date of current version 23 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3222715

RESEARCH ARTICLE

Analysis of Continual Learning Models for Intrusion Detection System

SAI PRASATH[1], KAMALAKANTA SETHI[2], DINESH MOHANTY[1], PADMALOCHAN BERA[1], (Member, IEEE), AND SUBHRANSU RANJAN SAMANTARAY[1], (Senior Member, IEEE)
[1]IIT Bhubaneswar, Kansapada 752050, India
[2]Indian Institute of Information Technology Sricity, Sri City 517646, India
Corresponding author: Kamalakanta Sethi (ks23@iitbbs.ac.in)

This work was supported by the Research Scheme on Power Grant, Central Power Research Institute, Bangalore, India.

ABSTRACT Deep Learning based Intrusion Detection Systems (IDSs) have received significant attention from the research community for their capability to handle modern-day security systems in large-scale networks. Despite their considerable improvement in performance over machine learning-based techniques and conventional statistical models, deep neural networks (DNN) suffer from catastrophic forgetting: the model forgets previously learned information when trained on newer data points. This vulnerability is specifically exaggerated in large scale systems due to the frequent changes in network architecture and behaviours, which leads to changes in data distribution and the introduction of zero-day attacks; this phenomenon is termed as covariate shift. Due to these constant changes in the data distribution, the DNN models will not be able to consistently perform at high accuracy and low false positive rate (FPR) rates without regular updates. However, before we update the DNN models, it is essential to understand the magnitude and nature of the drift in the data distribution. In this paper, to analyze the drift in data distribution, we propose an eight-stage statistics and machine learning guided implementation framework that objectively studies and quantifies the changes. Further, to handle the changes in data distribution, most IDS solutions collect the network packets and store them to retrain the DNN models periodically, but when the network's size and complexity increase, those tasks become expensive. To efficiently solve this problem, we explore the potential of continual learning models to incrementally learn new data patterns while also retaining their previous knowledge. We perform an experimental and analytical study of advanced intrusion detection systems using three major continual learning approaches: learning without forgetting, experience replay, and dark experience replay on the NSL-KDD and the CICIDS 2017 dataset. Through extensive experimentation, we show that our continual learning models achieve improved accuracy and lower FPR rates when compared to the state-of-the-art works while also being able to incrementally learn newer data patterns. Finally, we highlight the drawbacks of traditional statistical and non-gradient based machine learning approaches in handling the covariate shift problem.

INDEX TERMS Intrusion detection systems, catastrophic forgetting, covariate shift, continual learning.

I. INTRODUCTION

Millions of people use networked services for their day-to-day activities, from buying groceries online to banking transactions; networks have become a part of our daily life. They are only projected to grow given the expanding reach of technology and the need to connect with others. Monitoring large scale complex systems is essential to protect the privacy and ensure security for the users, but this task is becoming increasingly complicated due to the size and the intricate nature of these networks. Many network service providers and cyber security experts have been developing robust and reliable solutions to handle this issue. However, the rise of artificial intelligence as a potential tool to address the

The associate editor coordinating the review of this manuscript and approving it for publication was Donato Impedovo.

problem has led to significant improvements in IDS solutions. We have already progressed from ineffective rule-based solutions [34] (where the IDS extracts certain features from the network packets and monitors them individually to determine their authenticity) to more advanced machine learning and deep learning techniques [43] which understands complex feature interactions and provide robust detection capabilities.

Despite the appreciable success of AI-based solutions, the path to reliable IDS solutions is still far from reach. While most research works focus on the performance of AI models in an offline setting(where the data distribution remains the same), they ignore evaluating the performance of the models in an online setting(where data distribution changes continuously), which is more reflective of real-world systems. For example, a recent survey by The University of Maryland [25] estimated that a new cyber attack occurs somewhere on the internet every 39 seconds, and around 64% of all companies worldwide have experienced web-based attacks. It is also estimated that the average cost of a data breach in 2020 would exceed $150 million [30]. Hence, the assumption of changing and drifting data distribution is practically valid and requires an efficient and cost-effective solution. Therefore, any models trying to protect these systems must be able to quickly learn and dynamically adapt to the changes for performing well in an online setting. In our research work, instead of analyzing the offline performance of the AI models, we study their online performance.

Unfortunately, most AI models today do not have this capability as they follow the train and deploy technique [41]. Firstly, the models are trained on data previously collected from the system, and secondly, once the training is complete, the models are implemented to detect anomalies/intrusions in the system. In most rapidly evolving systems, the data encountered after implementing the model might not be from the same distribution on which the model was initially trained due to changes in system architecture and behaviours; therefore, over time, the model's performance deteriorates. This phenomenon is termed covariate shift which is explained in detail in sections III-B. Thus, in this research work, we analyze how deep learning models can be continuously updated to learn the new data distribution during the deployment stage without forgetting their previous knowledge so that they can perform well in the online setting.

However, before developing models capable of continuously learning to adapt to new data distributions, it is necessary to understand the nature and the magnitude of those changes so that we can accordingly develop the right continual learning technique for handling the change. To achieve this goal, we propose a comprehensive eight-step framework that combines various statistical and ML techniques along with our feature importance based technique that can help us detect, quantify and understand the nature of the change.

Once a drift in data has been detected, it is necessary to update the AI model. Most organizations today update the model in an offline setting [29]. In this technique, they collect and store all the data packets they encounter during the deployment stage along with the original data on which the model was trained. Once a drift is detected, they remove the deployed model and retrain it again on all available data. After the model is trained, they are deployed once again, and this process continues over and over. Although this technique maintains high accuracy levels, it can become really expensive to store all the data and train the model from scratch periodically as the system complexity increases. Therefore, to efficiently handle this problem, we must be able to achieve high accuracy levels while updating the model while also minimizing the storage and computational requirements of the update process. To solve this problem, we study continual learning algorithms that are usually used in the computer vision domain and extend them to large scale systems security. We empirically compare and analyze the performance of three major categories of continual learning algorithms on the NSL KDD and CICIDS 2017 datasets. We also explain why conventional statistical techniques and non-gradient based ML models fail in addressing this problem efficiently.

In summary, the major contributions of our current research work are as follows:

1) **Proposed framework for covariate shift analysis:** We present a comprehensive eight-step technique to understand the nature and the magnitude of the covariate shift by combining multiple statistical and machine learning-based strategies (univariate and multivariate).
2) **Introduction of feature importance based analysis:** As a part of our framework, we introduce novel feature importance based technique that analyses the impact of the individual feature drift by comparing it with their relative importance in the attack classification problem.
3) **Emulating real-world distribution:** To test and verify the performance of our models on rapidly changing attack distributions that emulate real-world characteristics, we generate batches of data with alternating attack types that arrive sequentially using the NSL KDD and CICIDS 2017 dataset.
4) **Continual Learning for IDS:** We compare multiple continual learning algorithms to address the issue of changing data distribution and also demonstrate the superiority of memory rehearsal based techniques. We also reason why conventional strategies and specific continual learning algorithms fail to address this problem while others succeed.

The rest of the paper is organized as follows: Related works in IDS, covariate shift and continual learning are discussed in Section II, followed by background on covariate shift, catastrophic forgetting, continual learning, neural networks and dataset analysis is detailed in Section III. Section IV elaborates on the eight-step covariate shift detection framework, and Section V explains the continual learning experiments, comparisons and results. Finally, Section VI summarizes and concludes the research work.

II. RELATED WORK

This section presents various research that has analyzed ML-based IDS solutions and rule-based statistical models. We also analyze strategies designed to counter the covariate shift problems along with the computer vision-based continual learning algorithms that have inspired this research work.

A. EXISTING IDS MODELS

Most rule-based classifiers fail to detect complex attack patterns and intricate dependencies due to their relatively simple detection process; therefore, implementing such classifiers on large scale security systems is not efficient. For instance, Rastegari et al. [34] tested multiple ruleset-based models, such as JRip, J48, MPLCS etc., on the NSL-KDD dataset and only achieved accuracies around 80%.

Whereas ML models have gained popularity because of their ability to analyze vast amounts of data and detect attacks efficiently. To this extent, Decision Trees(DT), Support Vector Machines(SVM), Clustering algorithms, Random Forests(RF), Artificial Neural Networks(ANN) etc., [2], [4] have been extensively examined as potential solutions for IDS. All these models achieve very high accuracies and low false-positive rates, which are within the acceptable threshold, thus qualifying them as potential options for real-world implementations. Other strategies such as ensemble learning [51], dataset analysis based recommendations [31], feature extraction [20] and feature selection [2] (like stacked feature selection) [38] have also been successfully utilized to improve the performance of the models.

Recently, despite the success of ML models, Deep Learning based solutions have been gaining traction primarily due to their ability to efficiently comprehend vast amounts of data and reliably understand the underlying representations. Deep Neural Networks(DNN) [43], Convolutional Neural Networks(CNN), and Recurrent Neural Networks (RNN - GRU and LSTM) [3], [46] have all been extensively analyzed and have been shown to perform better than ML models. Combining these architectures with denoising autoencoders [43], residual blocks [47] and attention mechanisms [43] has also been shown to increase the model's performance.

Siddique et al. [41] have proposed a comprehensive approach to develop an efficient IDS with emphasis on tackling big data problems in large-scale networks. They used the ABB technique to select optimal feature subsets and performed classification tasks using efficient machine learning techniques. They used the ISCX-UNB dataset (which is a relatively newer dataset) for their evaluation problem. They were able to achieve high accuracy with a low FPR Rate. Neha Yadav et al. [29] proposed Autoencoder based novel IDS for 5G IoT systems. They used the benchmark UNSW-NB15 dataset for training and testing purposes. They used several Machine Learning and Deep Learning methods for their implementation and presented a detailed comparison.Their proposed solution gave higher accuracy than the state of the art solutions.

In their paper [42], Tama and Lim have presented a systematic study of various ensemble-based IDS. A total of 124 publications were analyzed and categorized based on their year of publication, ensemble techniques, IDS techniques, etc. They also mention a new classifier ensemble approach called stack of ensemble (SoE), which combines ensemble learners like the random forest, gradient boosting machine, and extreme gradient boosting machine in a homogeneous manner by leveraging a parallel architecture. They also provided an empirical investigation of the Stack of Ensembles classifier ensemble approach. Their investigation suggested that ensemble techniques generally brought significant improvements over individual classifiers.

Mahzad Mahdavisharif et al. [24] have proposed a BDL-IDS algorithm based on the LSTM architecture, which they claim to capture complex and long term dependencies between the network packets of large scale computer networks. They perform the experiments on the NSL-KDD dataset using the BigDL directly on top of the Spark framework. They claim that their BDL-IDS algorithm outperforms conventional IDS in metrics like detection rate (20%), accuracy (15%), false alarm rate (60%), and training time (70%).

Although various IDS models achieve very high accuracy levels and low false-positive rates, none of the research works focus on the problem of covariate shift and continual learning in large scale security systems. All these research works measure the ability of models to achieve high accuracy and low fpr in their deployment stage. However, unlike these research works, we study the ability of models to continuously learn and adapt to the dynamic nature of today's systems, where the data distribution changes constantly. In this research work, we analyze how to efficiently detect the drifts in the data distribution and how to handle the drifts to ensure the high accuracy levels and low fpr rates of the model while also minimizing the storage and computational requirements involved in the process.

B. COVARIATE SHIFT DETECTION

Usually, many research works utilize statistical techniques to detect covariate shifts in the dataset. Raza et al. [35] implemented Exponentially Weighted Moving Average(EWMA) and Kolmogorov–Smirnov statistical hypothesis test(KS test) in a two-step detection process on an EEG dataset. Nair et al. [48] used reweighting strategies to assign more importance to points similar to the test set. The weights of the data points were generated by taking the ratio of the test probability to the training probability predicted by various models such as KNN, DT, NB and LDA. Rabanser et al. [33] tested different dimensionality reduction techniques coupled with multiple statistical tests to detect covariate shifts in various datasets. Both univariate and multivariate shifts were analyzed. Feutry et al. [9] experimented with various mean metrics such as quadratic mean, geometric mean and harmonic mean to analyze the change in output probabilities to detect the shift.

For relatively smaller data sets with fewer features, these techniques work well; however, in large scale security systems, the dimensionality of the data is very high. Hence, to solve this problem efficiently, we combine multiple ML-based covariate shift detection techniques with the above-mentioned statistical measures to propose a robust covariate shift framework that detects, quantifies, and analyses the nature of the drift in the dataset.

C. CONTINUAL LEARNING ALGORITHMS

Once the magnitude and the nature of the drift are detected, we utilize continual learning algorithms to learn newly encountered classes and changing attacks distributions incrementally. Lomonaco et al. [22] used a rehearsal free Copy Weight and Reinitialize(CWR) technique to learn new classes by increasing the output nodes of the neural network and used weight-freezing to train the nodes in the network selectively. Li and Hoiem [21] introduced a modified loss function (Learning Without Forgetting) to learn the new classes without forgetting the older ones by adding a regularization term with distilled outputs. Buzzega et al. [5] utilized a reservoir buffer to store the encountered data points and trained the model periodically on the buffer data with a regularized loss function termed dark experience replay. They achieved accuracy rates as high as 85% for the class-incremental learning task on the CIFAR-10 dataset over other architecture and regularization techniques whose accuracy was lesser than 60%. Many other algorithms in this domain, such as [19], [37], and [36], fall into one of these three broad categories, namely, architecture-based, regularization-based or rehearsal-based.

Many works in the domain of continual learning has been specific to computer vision and has not been extended to more complex domains such as large scale systems security. In our research, we empirically study the performance of the three major categories of continual learning algorithms by extending them to the field of systems security and testing them on the NSL KDD and CICIDS 2017 dataset, and also compare the storage and computational requirements of these algorithms.

Recent works by Sethi et al. [39] handle the problem of new attacks and drifting data distribution by utilizing a Deep-Q-Learning based approach to retrain the neural network by using constant feedback from the network administrator. Unfortunately, such strategies suffer from catastrophic forgetting as the models forget the older classes. Also, Wiewel and Yang [45] introduced a variational autoencoder(VAE) coupled with a rehearsal based continual learning model to handle the problem of growing datasets in the anomaly detection problem. They also address the need for continual learning-based models for IDS but fail to analyze the issues of covariate shift in the newer datasets. While some research works have tried to address the problem of continual learning, they have either failed to analyze the problem of covariate shifts in the dataset or have failed to create a robust

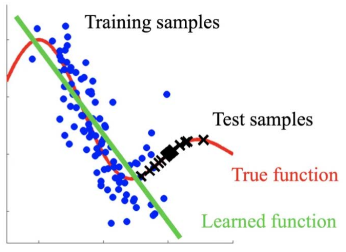


FIGURE 1. Covariate shift between train and test sets.

TABLE 1. Abbreviation and expansions.

Abbreviation	Expansion
RF	Random Forests
DNN	Deep Neural Networks
CL	Continual Learning
ER	Experience Replay
LwF	Learning Without Forgetting
DER	Dark Experience Replay
IDS	Intrusion Detection System

solution for catastrophic forgetting. In our research work, we efficiently overcome these problems by first detecting the shift in the dataset using the comprehensive eight-step covariate shift detection framework, followed by utilizing the best continual learning algorithm for handling the drift.

The abbreviations used in this research work are listed in Table 1.

III. BACKGROUND

In this section, we introduce and discuss three major concepts used in this research work, namely: covariate shift, catastrophic forgetting and continual learning.

A. COVARIATE SHIFT

Usually, in most real-world applications, we assume that the data encountered is independent and identically distributed; i.e. the data encountered post-training is derived from the same distribution on which the model was trained. Unfortunately, such an assumption always does not hold. New attack patterns are created every day in networked systems, which do not always overlap with the older patterns. For example, a classifier trained to identify spam mails using specific keywords extracted from a mail might fail once the spammers stop using those words and start using other similar words. The classifier needs to update itself to keep up its performance by learning the newer words used by spammers. Therefore without the ability to adapt to the changing conditions, the classifier becomes obsolete in a short amount of time. This property of the data to change from the initial training distribution is termed dataset shift.

Dataset shift [13] is a challenging situation where the joint distribution of inputs and outputs differs between the training and test stages. Whereas covariate shift is a specific case of dataset shift where only the input distribution changes $P_{train}(X) \neq P_{test}(X)$, while the conditional distribution of the outputs given the inputs remains unchanged $P_{train}(Y \mid X) = P_{test}(Y \mid X)$; meaning the mapping of the input features X to the output variable y remains unchanged, but the distribution of X varies between the train and the test set. Although the underlying decision function remains the same, part of that relationship is data-sparse, omitted, or misrepresented; the test set and training set do not reflect the same distribution.

Figure 1 [15] is a pictorial representation of the covariate shift problem. The blue points represent the training sample from which the classifier learns the underlying function(green). However, the test sample(black) has drifted from the initial train set distribution due to the changing nature of attacks, which affects the model's performance. Therefore the learnt function fails to perform efficiently on the test samples.

B. CATASTROPHIC FORGETTING

Catastrophic forgetting or catastrophic interference [11] was first recognized by McCloskey and Cohen [1989]. They found that when deep learning models are trained on newer tasks, the neural nets forget what was learnt previously on older tasks. This phenomenon usually occurs because the neural net weights are overwritten/fine-tuned by the optimization function applied when the model is trained on the newer tasks, thus degrading the model's performance on the older tasks. Therefore without solving this complexity, neural networks will never be able to learn new tasks and retain knowledge from the older ones. This was also referred to as the stability-plasticity dilemma in [1] by Abraham and Robins. If a model is too stable, it will not consume new information from the future training data. On the other hand, a model with sufficient plasticity suffers from significant weight changes and forgets previously learned representations. Hence it is essential to balance the plasticity and the stability of the models for optimum performance.

A glaring example of this issue is model transferability. Many organizations directly import well trained convolutional neural network(CNN) models for image classification such as ImageNet and fine-tune the final few layers to suit their requirements. Although the model's performance on the new tasks will be excellent, the performance on the older tasks where the model used to excel once would have dropped considerably. This trade-off where the organization compromises the stability of the model to achieve high performance is an inherent problem in neural networks. Hence any real-time long term IDS solutions should address the problem of catastrophic forgetting.

C. CONTINUAL LEARNING

In many real-world situations, all the data is not available at once; instead, we obtain them in batches once every minute, hour or day, depending on the type and the organization's requirement. Continual learning(CL) is a particular class of algorithms capable of learning and retaining information from these partial experiences, which are tantamount to the knowledge earned when trained on the whole data at once. CL saves a lot of computation and storage costs and improves the efficiency of the process by reusing previously learnt knowledge.

They should be able to handle various problems such as imbalanced or scarce data problems, catastrophic forgetting, or data distribution shifts. Continual learning can be considered as a synonym for Incremental Learning, Lifelong Learning and Never-Ending Learning.

In this research work, we model the problem as a set of tasks T where

$$T = \{(C^1, D^1), (C^2, D^2), (C^3, D^3), \dots, (C^n, D^n)\}$$

here each task t is a set (C^t, D^t) obtained at time $= t$, and C^t represents the set of classes encountered at that given time $C^t = \{c_1^t, c_2^t, c_3^t, \dots, c_{n^t}^t\}$ and D^t is the corresponding training dataset. The total number of classes in the task is represented as $N^t = \mid \bigcup_{i=1}^{n} C^i \mid$. The training dataset D^t is a set of datapoints $\{(x_1^t, y_1^t), (x_2^t, y_2^t), \dots (x_{m^t}^t, y_{m^t}^t)\}$ where x are the input features and $y \in \{0, 1\}^{N^t}$ is a one hot ground truth label vector corresponding to x [26]. Note that at any given time t, we only have access to the corresponding dataset at time t and all previously encountered data is not available for training. The theoretical context of applying continual learning in security based systems is well analysed in [7].

The type of data encountered at each interval can be divided into two main types, namely new classes and new instances. As time progresses and we receive the latest data, new classes that were not previously known can be encountered. The ability to learn new classes without forgetting the features of the old classes is termed incremental class learning. Along with new classes, data points belonging to old classes can also be shifted due to the non-stationary nature of the environments common among cyberattack domains. This corresponds to learning under dataset drift. Both these tasks are considered a part of the CL algorithms.

D. DEEP NEURAL NETWORK

Deep Neural Network(DNN) has become the state of the art model for multiple tasks such as image classification, time-series analysis and security solutions. DNNs can be described as layers(of nodes) that extract complex and non-linear features from the input to perform the assigned task. Each layer can be represented as l_i, and the input to layer l_i is the output of layer l_{i-1} meaning they are structured as a nested function. Therefore the final output of the model $y = l_n(l_{n-1}(\dots l_1(x)))$ is a series of nested layers. Each layer has its parameters that process the input and produce the outputs; there are two major classes of parameters, i.e. weights and biases. Consider the output of l_i as h_i, which is fed as the input to layer l_{i+1} where the input is processed as $h_{i+1} = g(w_{i+1}^T h_i + b_{i+1})$. The "w" in the equation represents the layer's weights, and the "b" represents the biases. Function g creates the non-linearity in the model without which the learnt function will be a weighted sum of the inputs. The same set of computations is repeated until the final layer to calculate the model's prediction y. At the final layer, the error is computed using

TABLE 2. NSLKDD dataset distribution.

Dataset	Normal	DOS	Probe	U2R	R2L
Training	67,343	45,927	11,656	52	995
Test	9,711	7,458	2,241	67	2,887

loss functions such as mean squared error for regression, binary cross-entropy for binary classification, and categorical cross-entropy for multi-class classification by comparing the prediction with the expected output.

To optimize the model's performance, the losses computed at the final layer have to be minimized by tuning the model's parameters. In this context, the weights and biases of each layer are the parameters; therefore, the task is to find the ideal values of these parameters such that the loss is minimized. This task is achieved by computing the gradients of the parameters and utilizing a gradient-based optimization function such as stochastic descent or ADAM to backpropagate through the model. At each layer, the parameters are fine-tuned by updating them using their respective gradients. Note that as the model uses gradient-based optimization, all the layers, parameters and the function of the model must be differentiable. This ability to train the model using gradient-based approaches is efficiently utilized in our research work.

E. DATASET DESCRIPTION

In this section, the dataset is analyzed, and certain inherent biases/inconsistencies are discussed. We also discuss pre-processing the dataset and why we refrain from feature selection/reduction strategies.

1) NSL KDD DATASET

a: DATASET ANALYSIS

The MIT Lincoln Labs generated the original KDD-CUP99 dataset in 1999, where the US Air Force Lan was simulated and multiple attacks were performed. The TCP packets were collected as binary dumps across seven weeks, and multiple network features were extracted from the raw dataset. The researchers ensured that the distribution of the training and the test attacks were not from the same distribution to emulate real-life scenarios. The NSL-KDD dataset [28] is a refined version of the KDD-CUP99 dataset where particular imbalances and inconsistencies in the dataset were eliminated by removing recurrent data points. The dataset was also reduced to a reasonable size for more accessible analysis and testing.

Table 2 summarizes the attack distribution in the train and test set. The train set consists of 23 different types of attacks, but the test set consists of 38, some of which are day 0 attacks meaning the ML model was not trained on these attacks. All the data points are divided into five major categories: Normal, DDOS, Probe, U2R and R2L for the multi-class classification task.

Further analysis of the dataset reveals that both the train and the test set share 21 different attack types, constituting 99.29% and 83.36% of the data points in their respective datasets. Two attacks are exclusive to the training dataset, which accounts for the remaining 0.71% of the training dataset and seventeen attacks are exclusive to the test dataset accounting for the remaining 16.64% of the test dataset. The introduction of new attacks which were not initially present leads to the change in the distribution causing the covariate shift between the training and the testing dataset.

There are only around 1,50,000 data points which are comparatively less to train sophisticated neural networks. Many research works point out this shortcoming of the dataset, which restricts them from training massive neural networks because they might overfit when trained on a smaller dataset. Also, such small datasets might not thoroughly test and verify the various requirements of a robust NIDS. The NSL-KDD dataset itself is outdated and does not reflect the current state of the network attacks. Nevertheless, the compact nature of the dataset and the dataset shift between the train and the test set, which we will explore later, makes it suitable for our analysis to test and compare various continual learning-based models.

b: DATASET PREPROCESSING

The raw dataset consists of 41 features values for each data point describing the various network parameters. Three('protocol-type', 'service', 'flag') out of the forty-one values are categorical, and all else are continuous. We convert all the categorical values to continuous by applying the one-hot encoder technique, creating a dataset with 122 features. All the features are then scaled between 0 and 1 using min-max normalization to help the models learn the pattern easily and prevent any feature bias.

2) CICIDS 2017 DATASET

a: DATASET ANALYSIS

Unlike the NSL KDD dataset, the CICIDS 2017 dataset is a more recent and relevant dataset with the most up-to-date attacks present in the dataset. They closely simulate real-world data packets (PCAP) and also extract more information using network traffic analysis tools that are included in the dataset. To create a realistic dataset, the dataset captures the abstract behaviour of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols using the B-Profile system [40].

The data is captured across five days where on each day, new attacks are being introduced. The day and the type of attack introduced are listed in Table 3. In our research work, we divide these five days into two classes to compare the ability of various models to continually learn new attack distributions. The first three days, Monday, Tuesday and Wednesday, are grouped as Class 1, whereas the remaining days, that is, Thursday and Friday, are grouped as Class 2. We consider Class 1 as the train set and Class 2 as the test set.

Unlike the NSL KDD dataset, there is no overlap in the attack traffic between Class 1 and 2. All attacks present in

TABLE 3. Distribution of attacks.

Day and Time	Traffic Type	Count
Monday	Benign	529,918
Tuesday	Benign	432,074
	SSH-Patator	5,897
	FTP-Patator	7,938
Wednesday	Benign	440,031
	DoS Hulk	231,073
	DoS GoldenEye	10,293
	DoS Slowloris	5,2499
	Heartbleed	11
Thursday	Benign	456,752
	Web Attack Brute Force	1,507
	Web Attack SQL Injection	21
	Web Attack XSS	652
	Infiltration	36
Friday	Benign	414,322
	Bot	1,966
	Portscan	158,930
	DDoS	128,027

Class 2 are exclusive to the class and have not been encountered in Class 1. The attacks present in Class 2 contribute around 25.06% of the whole Class 2 data. Therefore, the task of learning these attacks is more difficult in the CICIDS 2017 dataset when compared to the NSL KDD dataset. This introduction of new attacks not present in Class 1 leads to a covariate shift in the dataset and therefore requires continual learning models to learn the distribution of these new attack types. A distinguishing feature of the CICIDS 2017 dataset is that it is more realistic and also has a reliable benchmarking that follows the 11 step approach proposed in [40].

b: DATASET PREPROCESSING

The dataset contains 78 features values for each data point describing both network traffic parameters and PCAP data. Ten features, namely (‘Bwd PSH Flags’, ‘Fwd URG Flags’, ‘Bwd URG Flags’, ‘CWE Flag Count’, ‘Fwd Avg Bytes/Bulk’, ‘Fwd Avg Packets/Bulk’, ‘Fwd Avg Bulk Rate’, ‘Bwd Avg Bytes/Bulk’, ‘Bwd Avg Packets/Bulk’, ‘Bwd Avg Bulk Rate’) have a constant value across all the data points; therefore, we remove these features because they do not contribute to our model’s learning process. The dataset also contains eight binary features and no categorical variables. All other features which are continuous are scaled between 0-1 using the min-max normalization, which helps the models learn better by removing any bias between the features. The final pre-processed dataset contains 68 features.

Note that we do not perform any feature selection/reduction algorithms due to changing nature of the attack patterns in the long term, which might affect the relative importance of the features across various attacks. Thus we expect our models to learn the importance of features by themselves and adapt to the changing circumstances when required. The importance of features describing an attack might also vary across time, considering the non-stationary nature of the attack domains. Therefore, as the features’ importance changes through time and across attacks, we abstain from implementing any selection/reduction algorithms.

TABLE 4. RF and DNN performance on the dataset.

Dataset	Model	Train	Test
NSL KDD	RF	99.67%	77.22%
	DNN	99.72%	78.29%
CICIDS 2017	RF	99.66%	79.83%
	DNN	99.81%	79.14%

IV. PROPOSED COVARIATE SHIFT ANALYSIS FRAMEWORK

This section presents a detailed description of our proposed eight-stage framework for analyzing any shift in the dataset. We objectively tried to visualize, quantify and justify the nature of the shift, which will help us in analyzing CL algorithms. We describe the proposed techniques in the following subsections.

A. EVALUATING PERFORMANCE

In the first technique, we directly verify whether there is any shift by analyzing the accuracy of machine learning and deep learning models. If the drop in accuracy across the training and the test set is considerable, then there is reason to consider that the dataset distribution might have shifted. We evaluate the performance of the Random Forest(RF) model and the DNN model on the NSL-KDD and CICIDS 2017 dataset for a binary classification problem. Table 4 summarizes the results of the analysis:

Both the Random Forest model and the DNN model performed well on the NSL KDD training dataset, but their performance in accuracy dropped by 22% on the test set. The analysis results agree with the previous analysis of the dataset, where we found that 16.63% of points in the test dataset are unique to the test set attack types, which affects the performance of our models. Similarly, in the case of the CICIDS 2017 dataset, the models perform exceedingly well on the train set but their performance drops by around 20% on the test set. The introduction of new attack types in the test set, which contributes around 25% of the set, clearly explains this drop. We ensured that the models did not overfit by restricting the depth and size of the RF and the DNN. No significant change was achieved in the model’s performance after an extensive hyperparameter search.

Multiple research works develop complex and sophisticated models to address this issue, but the performance remains relatively unchanged despite their best efforts. The problem, in this case, is with the data itself and not the model; hence altering the model would not create any significant changes in the performance. Instead, concentrating on developing algorithms to continuous learning the changing data distribution is necessary; this process is termed continual learning, which is explored in detail in this research work.

B. DATASET SHUFFLING

To ensure that the drop inaccuracy from the train to the test set is not due to the inherent lack of patterns or features but only occurs due to the data drifting, we implement a shuffling

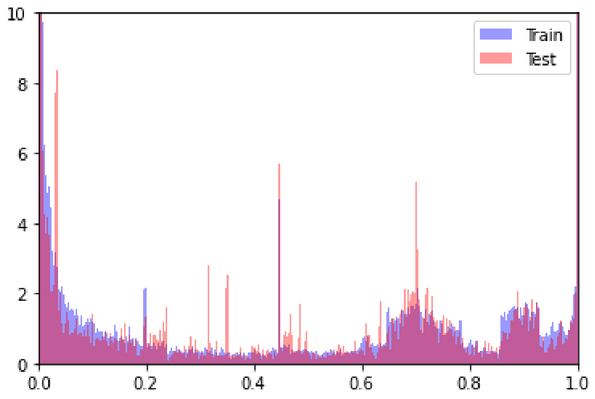


FIGURE 2. Histogram overlap for feature dst_host_count.

TABLE 5. Top 10 least feature histogram overlap.

Dataset	Feature	Overlap
NSL KDD	dst bytes	0.947
	duration	0.953
	service pop 3	0.963
	hot	0.978
	dst host srv count	0.979
CICIDS 2017	Average Packet Size	0.843
	Avg Bwd Segment Size	0.854
	Bwd Packet Length Mean	0.887
	Destination Port	0.898
	Bwd Packet Length Min	0.909

based analysis. The train and the test set data points are mixed and shuffled, ensuring that when this combined dataset is split into train-test for analysis, the attacks points are uniformly distributed, meaning there are no new attack types in the test set. This effectively removes the dataset shift between the newly created train and test set. A binary attack classification is performed on the combined dataset. The random forest classifier achieves a very high 10-fold cross-validated accuracy of 99.49% and an fpr value of 0.39% on the NSL KDD dataset. Similarly, on the CICIDS 2017 dataset, the random forest classifier achieved an accuracy of 99.56% and an fpr of 0.33%.

This helps us conclude that ML models can learn the attack patterns in the dataset and perform well if all the attacks were available during the training stage itself. This concludes that there is no inherent lack of patterns or features, but the poor performance is due to the dataset shift that changes the distribution of the existing attacks and introduces new attacks in the test dataset.

C. HISTOGRAM OVERLAP COMPUTATIONS

In this technique, we analyze whether any particular features are responsible for the drift. To identify the particular features, we bin the corresponding feature values in the train and test sets separately into 100 bins. The values in the bin are then normalized for a fair comparison across multiple features. The value of the histogram overlap is computed to compare the similarity between the distribution of features in the train and test set. The higher the overlap(closer to 1), the more the similarity and the lesser the overlap(closer to 0), the more diverse the distribution is. Any features that have an overlap of less than 70% can be considered as shifted.

Figure 2 visualizes the histogram overlap of the feature dst_host_count, which is then used for our calculations. Table 5 summarizes the top 5 drifted features of the NSL KDD and the CICIDS 2017 dataset obtained using the histogram overlap computations. Usually, a set of features that drift a lot are dropped from the dataset to improve the performance of our models as it reduces the bias across the datasets, provided that the feature itself is not essential for the classification task. However, in the NSL KDD dataset, the maximum drifted feature has an overlap of 94.7%, meaning that the drift of the features is minimal. Similarly, in the case of the CICIDS 2017 dataset, the maximum drifted feature has an overlap of 84.3%, which is higher than the 70% threshold considered for this technique. Therefore, we conclude that there are no significant drifts of individual features, and hence we refrain from dropping any features.

D. KOLMOGOROV SMIRNOV TEST

Kolmogorov–Smirnov Test or KS Test is a non-parametric statistical testing technique used to determine whether a given continuous sample distribution is similar to a reference distribution (such as normal distribution, binomial distribution) or another sample distribution. It computes the empirical distribution function F_n for all the n data points for both the considered samples and from that computes the supremum distance between the computed empirical distribution functions.

$$F_n = \frac{1}{n}\sum_{i=1}^{n} I_{[-\inf,x]}(X_i) \tag{1}$$

$$I_{[-\inf,x]}(X_i) = \begin{cases} 1, & X_i \leq x \\ 0, & X_i > x \end{cases} \tag{2}$$

$$D_{n,m} = \sup_x \mid F_{1,n}(x) - F_{2,m}(x) \mid \tag{3}$$

$$D_\alpha = c(\alpha)\sqrt{\frac{l_1 + l_2}{l_1.l_2}} \tag{4}$$

Equation 1 is the definition of the empirical distribution function, where the indicator function I is computed using equation 2. Once the sample distributions' empirical distribution is computed, the supremum distance between the distributions is calculated using equation 3. The KS statistics value computed by the function is compared with the critical value obtained from equation 4. The l_n values in equation 4 are the corresponding lengths of the datasets. The distributions are considered to be different if:

- p_value of the distribution $<$ 0.05 (Standard)
- KS Statistics $>$ Critical value

The above-described computations are applied to every feature individually, and the following results were obtained: a total of 47 features has a p-value less than 0.05, meaning that they might have different distributions, but upon further analysis when compared with the critical value of 0.00983 [critical_value(0.05) = 1.36] none of the KS statistics was above this limit. Similarly, for the CICIDS 2017 dataset, 66 features have a p-value lesser than 0.5, but none of those features has a significant KS statistic of 0.00126. Hence, we can conclude that all the features are derived from the same distribution for a p-value of 0.05, i.e., no individual feature shift. Similarly, other statistical tests such as chi-squared analysis and Mann-Whitney test can also be used to verify the shift in the distribution.

E. 2D VISUALIZATION

One of the best ways to understand and study the dataset drift is to plot the values and visualize the changes. However, in most cases, visualizing is not always possible due to the size of the dataset. In the NSL KDD dataset, we have 122 features after processing, and 68 features in the CICIDS 2017 dataset, thus, comparing the data points will be difficult. Hence we reduce our dataset to 2 dimensions using the Principle Component Analysis(PCA) algorithm, which helps us visualize the data points using a scatter plot. These two top features account for 71.35% of the total variance in the dataset in the NSL KDD dataset and 60.46% in the CICIDS 2017 dataset.

In figure 3, the blue and the red data points represent the shared attacks of the train and test set, respectively. Whereas the yellow data points represent the day-0 attacks introduced in the test set(exclusive test attack types). From this plot, we can observe that the distribution of the shared attacks points has drifted. Also, the day-0 attacks have a different distribution than what was initially learnt. However, figure 3b, which visualizes the normal(no attack) points, has a perfect overlap between the train and the test set. We conclude from these plots that although the normal(no attack) points remain unchanged, new attack types have been introduced(yellow), and the shared attacks have drifted from their previous distribution. Therefore due to the change in distribution, relying on the previous decision boundaries will lead to poor performances; hence the decisions boundaries need to be learnt again to accommodate these changes.

Similarly, for the CICIDS 2017 dataset, from figure 3d we can see that the normal points overlap between the train and the test set. However, we can also observe that new distributions have been introduced as there are regions where the red points do not overlap with the blue ones. In the case of the attack distribution, we conclude from figure 3c that all the attacks present in the test set are day-0 attacks as there are only yellow points visible(exclusive test attacks) which are covering the red points. Although in some regions, the yellow data points overlap with the blue points, there are many other regions where they do not. Therefore, in this case, the model needs to learn both the changes in the distribution of the normal data points and the new attacks introduced in the test set.

TABLE 6. Novelty detection.

Dataset	Algorithm	Train-Inliners	Test-Outliers
NSL KDD	Local Outlier Factor	98.14%	19.90%
	One Class SVM	98.00%	17.19%
CICIDS 2017	Local Outlier Factor	97.79%	22.82%
	One Class SVM	98.03%	21.25%

F. NOVELTY DETECTION

It is the identification of new or unknown data or signals that a machine learning technique is not aware of during training. Novelty detection techniques try to identify outliers that differ from the distribution of ordinary data, more specifically, the training dataset whose distribution the model learns.

The Local Outlier Factor function [32] available in the sklearn library is used to analyze and visualize the data distribution. It measures the local deviation of the density of a data point with respect to its neighbours. The anomaly score of a data point depends on how isolated the object is to the surrounding neighbourhood. More precisely, the locality is given by k-nearest neighbours, whose distance is used to estimate the local density. By comparing the local density of a sample to the local densities of its neighbours, one can identify samples that have a substantially lower density than their neighbours. These are considered outliers.

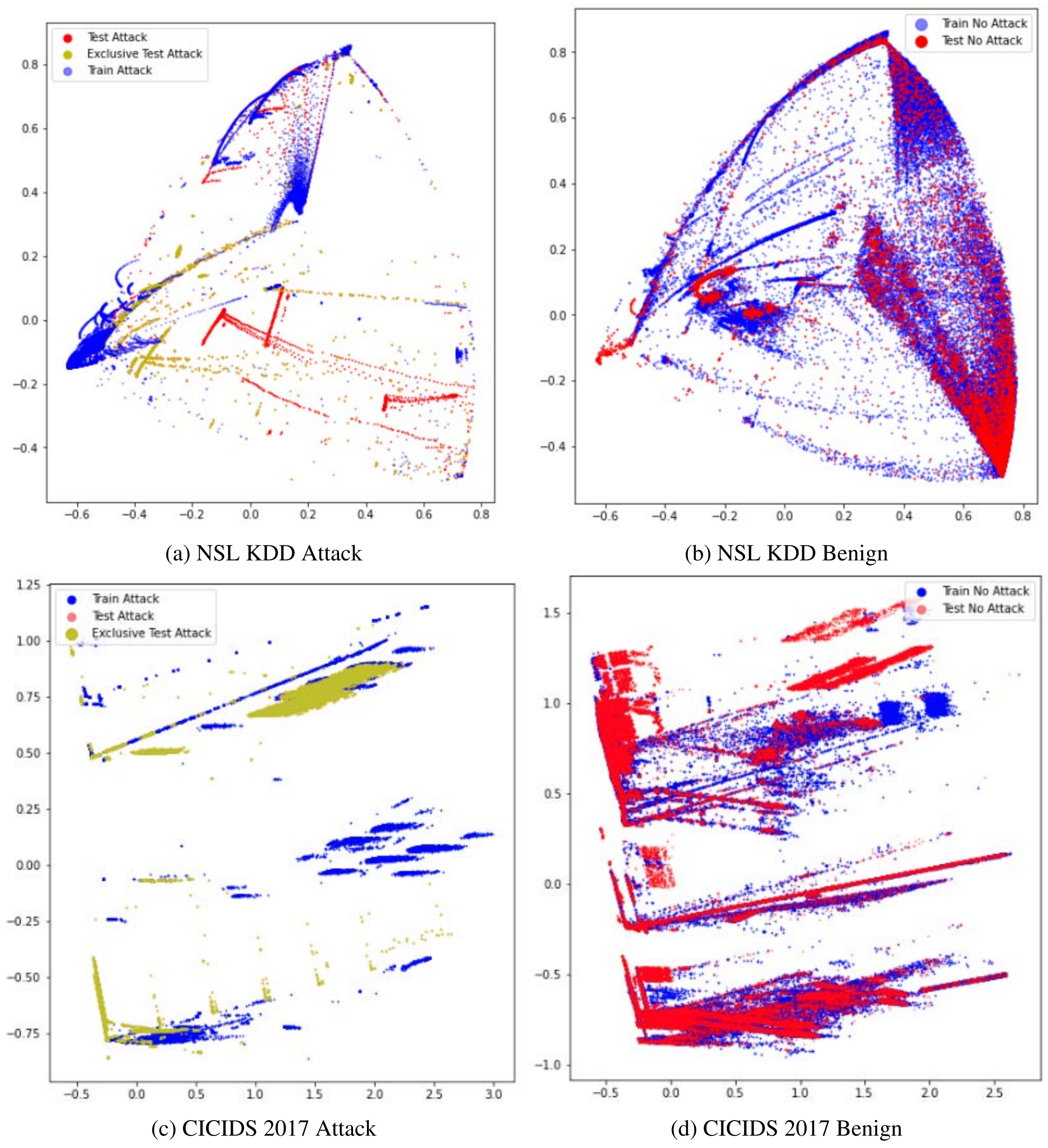
The local outlier detection algorithm creates boundaries to differentiate between inliers and outliers. All the data points within the boundary are considered inliers, and the points outside are considered outliers, meaning they do not belong to the data distribution that these models previously fit on.

Along with the Local Outlier Factor algorithm, we also use a One-Class SVM that can be used to detect outliers in data by fitting an SVM model on a dataset containing a single class. We use a constant ν to represent the training dataset's impurity (These fractions of ν points are considered outliers in the training process). The impure points are labelled as -1, whereas all the other points are labelled as 1. Using this labelled dataset, we train the SVM Classifier, which is applied to the test set to determine the shift in the dataset.

Table 6 summarizes the results of the novelty detection algorithm: For both the NSLKDD dataset and the CICIDS 2017 dataset, we set the ν value to 2% for the training stage. As expected, the test outliers in the case of the NSL KDD dataset are around 17-19%, which is in accordance with the 16.64% of data points that are exclusive to the test set. In the case of the CICIDS 2017 dataset, the outliers are around 21-23%, as 25% of data in the test set corresponds to the new attack types introduced as day-0 attacks. Therefore this technique helps us conclude that there are significant changes in the distribution between the train and the test set, which requires us to relearn the decision boundaries.

G. DISCRIMINATIVE DISTANCE

All the above analyses performed in the dataset involved individual features or 2 PCA components but not the whole



(a) NSL KDD Attack

(b) NSL KDD Benign

(c) CICIDS 2017 Attack

(d) CICIDS 2017 Benign

FIGURE 3. **Distribution of data points.**

dataset. To utilize the whole dataset, we use the discriminative distance metric to quantify the shift.

Discriminative distance technique tries to differentiate between the training and the testing data using ML classifiers. The intuition behind the technique is that if the classifier cannot distinguish between the data points, they share the same distribution, whereas if the classifier can distinguish them with high accuracy rates, they are from different distributions. The classifier can learn complex patterns that the previous analysis might not have considered. Although this technique is more time consuming, it is much more potent than the previous ones.

We label all the train set data as "1" and test set as "0". A random forest classifier is used to classify the data point as either 0 or 1, i.e. whether the data point is from the train or the test set. We refer to this as the train-test problem in the following sections. The logic behind this technique is that the accuracy of such a classification can only be high if the data points from the train and test set are different, meaning they come from a different distribution, but if they are from the same distribution, then the accuracy will be low. This technique considers the complex interaction between multiple features that were neglected in the previous analysis. It might so happen that no individual feature is responsible for this change, but rather a combination of subtle changes might result in the final drift; these unique cases can also be efficiently analyzed using this technique.

TABLE 7. Feature importance for top-5 drifted features.

Dataset	Feature	Train-Test	Train Binary	Test Binary
NSL KDD	dst_host_srv_count	6.10	6.77	5.07
	dst_host_count	5.77	18.22	10.10
	count	5.35	13.58	5.64
	srv_count	5.06	2.89	3.90
	dst_host_diff_srv_rate	4.93	0.61	0.67
Total Contribution			42.07	25.38
CICIDS 2017	Destination Port	7.12	6.79	3.29
	Init_Win_bytes_forward	6.99	3.54	2.38
	Flow IAT Max	4.94	5.32	1.17
	Flow Duration	4.72	2.37	1.39
	Flow IAT Mean	4.12	1.31	0.43
Total Contribution			19.33	8.66

The RF classifier successfully classified the train and the test set with a 10-fold cross-validation accuracy of 91.28% on the NSL KDD dataset and 87.17% on the CICIDS 2017 dataset; clearly showing the change in distribution between the train and the test set. An accuracy higher than 80% is a strong indication of the drift; thus, there is a need to address this problem using continual learning algorithms.

H. FEATURE IMPORTANCE

In this subsection, we propose a unique technique to analyze the impact of the individual features by comparing their importance in the train-test classification task with the binary attack classification problem. The importance of various features is computed using the Extra Tree Classifier algorithm for both the tasks individually. In the case of the train-test classification problem, the higher the importance, the more the drift, as the features that have drifted the most can help distinguish between the train and the test set data points. Whereas the features that have not drifted cannot help distinguish between the train and the test set data points, hence they have little importance to this task. In the case of the binary attack classification problem, the importance of the features reflects their direct contribution to efficiently identifying the attack data points. Therefore by comparing the importance of features in both these tasks, we can understand the relation between the drifted features and their relative importance in the binary attack classification problem. This analysis helps us identify features with large drifts that do not have significant importance in the binary attack classification problem. We can drop such features to reduce the bias in the dataset without affecting the model's performance. Features with high importance need to be retained to ensure the model's performance, whereas features with low importance but low drift can be either retained or ignored depending on the problem.

One of the standard techniques to address the shift in the data is to drop the drifted features; it is based on the fact that dropping those features reduces the bias between the datasets hence providing similar distributions. However, suppose the feature is essential for the binary classification problem; in that case, we will have to compromise the accuracy of the task, whereas if the feature is not essential, then dropping the features does not necessarily improve the performance of the binary classification task. Therefore this trade-off between the feature importance and drift has to be handled efficiently to ensure good performances.

From Table 7 it is straightforward that none of the features drifts(in terms of importance) a lot; the maximum drift is 6.10% for the NSL KDD dataset and 7.12% for the CICIDS 2017 dataset which corresponds to our results from the statistical analysis that there is no significant drift in the individual features. Also, these top-5 drifted features contribute towards 42.07% importance and 19.33% importance in the train set attack classification of the NSL KDD and CICIDS 2017 dataset, respectively. Hence dropping these features to reduce the bias in the dataset will affect the attack classification problem as they are significantly crucial for that task. Therefore, directly dropping these features is not viable due to their little drift and high importance values.

Therefore, the feature importance based analysis helps us determine the nature and the impact of the feature drifts on the binary attack classification problem that was previously unknown. With the help of the analysis, we conclude that the feature drifts are minimal, but the drifted features are important in the binary attack classification problem; hence dropping those features is not an option.

V. EXPERIMENTAL STUDY OF CONTINUAL LEARNING MODELS

In the previous section, we have analyzed various problems associated with implementing IDS solutions in large scale networks with rapid changes in attack patterns. We have also created a framework to detect and quantify such drifts in the dataset. This section studies various solutions and how a specific technique performs better than others. The work flow of the covariate shift detection framework and the continual learning algorithm is depicted in Fig.4

A. REWEIGHTING DATA POINTS

In general model training, all data points are given equal weightage, meaning no point is given more priority over the rest. We can allot weights to the data points in the training set such that the ones with higher weights are given more priority over the ones with the lower weights; this is reflected in the

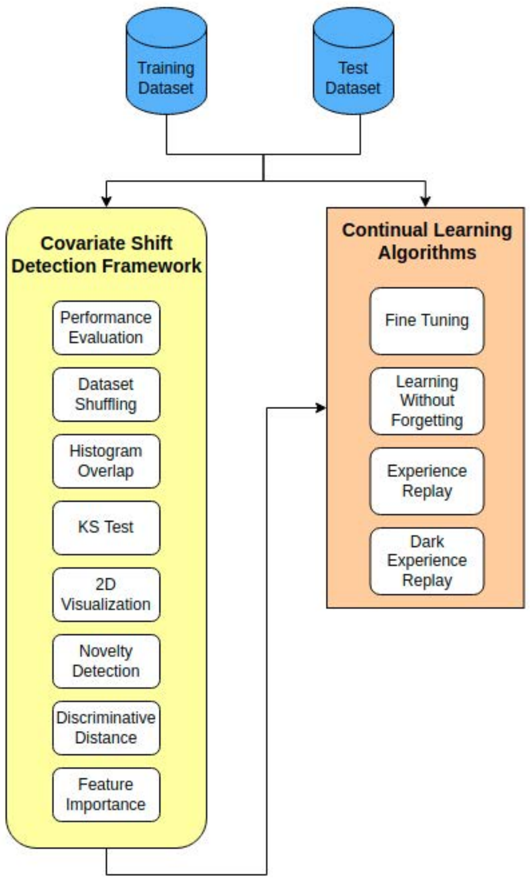


FIGURE 4. Work Flow of the Algorithm.

loss function that the model is optimizing.

$$total\_loss = \frac{1}{n}\sum_{i=1}^{n} w_i * loss\_function(y, f(x_i)) \tag{5}$$

$$\sum_{i=1}^{n} w_i = 1 \tag{6}$$

In equation 5 which is a generalised loss function the parameter w_i (constraint on equation 6) represents the weight allotted to the data point x_i. In general, the weights are set to $\frac{1}{n}$, but to assign more importance to certain points over the others, the weights corresponding to those points could be modified. In our case, we assign higher weights to points in the train set that are similar to a small set of points that we sampled from the test set. This technique provides a platform for the model to alter its predictions based on what might be expected in the future and not completely focus on the current data distribution.

To implement the technique, we use a random forest classifier to generate the weights of the data points in the train set. We randomly sample 2000 points from the test dataset, representing the test set distribution, and the whole train dataset is considered for this analysis. The newly created test set data points are labelled as '0', and the new train set data points are labelled as '1' (similar to the train-test classification problem discussed in section IV-G). After combining and shuffling the labelled dataset, we use a 10-fold cross-validation technique to generate the weights of the train set data points by using the output of the random forest classifier.

$$r_i = \left(\frac{1}{f(x_i)} - 1\right) \tag{7}$$

$$w_i = c_1 + c_2 * \frac{r_i}{\sum_{i=1}^{n} r_i} \tag{8}$$

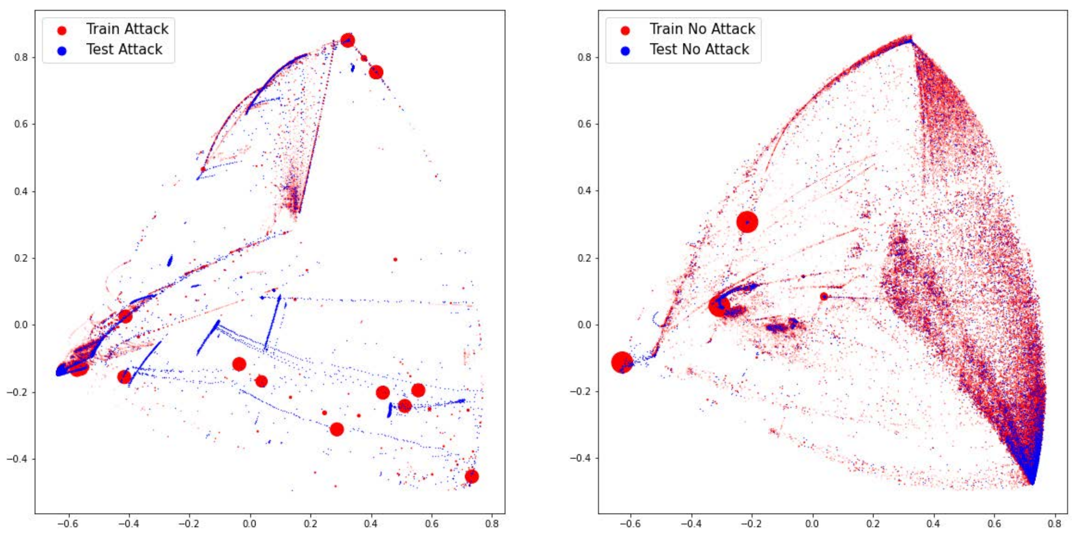
If the classifier's output $f(x_i)$ is closer to '0', then the point is similar to the test dataset distribution; hence should be given more weight. On the other hand, if the output is closer to '1', the point should be given lesser weight as it is not similar to the test set distribution. The weights are computed using equations 7 and 8, where r_i is the prediction probability of the random forest classifier and c_1 and c_2 are constants.

In figure 5a and 5b, the size of the points correspond to the weights allotted to them. Most of the significant points are located where there is considerable overlap with the test set distribution, and in locations where the overlap is less, the size of the points is small. We use the first two components of the PCA algorithm to plot the above figure. Although the test set accuracy improved by 2.53 % to 79.75% for the NSL KDD dataset and 3.19% to 83.02% for the CICIDS 2017 dataset, the train set accuracy dropped by 1.63% to 94.79% and 2.16% to 97.50% respectively. This trade-off is expected due to the different weights allotted to the train set's data points. The 2-3% improvement in performance is not enough as the accuracy is only around 80%.

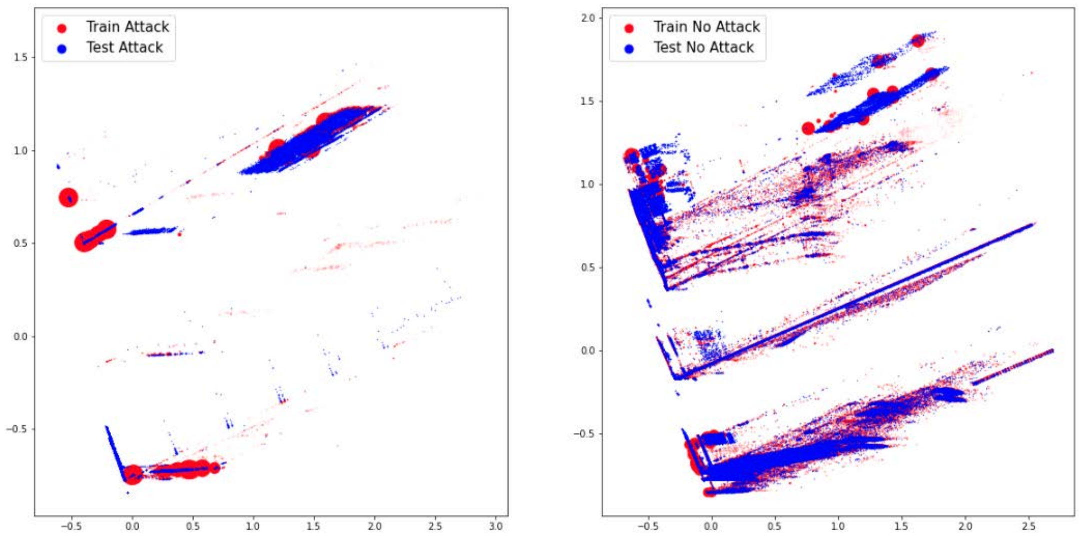
One of the primary reasons for the insufficient improvement in accuracy is the nature of the shift. Along with the existing attack points that have shifted significantly, new attacks have been introduced that follow entirely different distributions. Therefore just reweighting the data points is not an ideal solution due to the magnitude and the nature of the shift. Hence we need to explore algorithms that can handle such significant drifts and ensure good performance levels. Also, note that getting adequate test data to train our classifiers in real-time scenarios is not always practically feasible due to the unpredictable nature of the changing data distributions.

B. NON-GRADIENT-BASED MACHINE LEARNING MODELS

To solve covariate shift and the arrival of new classes, we need a model that can be tuned to remember previous patterns and learn new ones simultaneously. Unfortunately, most ML-based models are offline models meaning they are trained before deployment, and their parameters cannot be changed after the initial training phase. For example, in random forests, the branches and their associated conditions generated during the training phase cannot be altered if the data distribution changes; instead, we are forced to retrain the whole model once again. When the model is retrained, the previously available knowledge is not utilized, leading to a wastage of computational power and storage. Although in random forests, one potential solution is to add more trees



(a) NSL KDD



(b) CICIDS 2017

FIGURE 5. Reweighting data points.

to the existing classifier to retain previous knowledge; this increases the model's complexity and does not guarantee any significant improvement in the model's performance. Therefore the only solution is to store all the encountered data packets and retrain the model periodically.

Hence the use of non-gradient based models in the continual learning algorithms require massive computational power and storage capacity due to their need to be frequently retrained, especially in the case of large scale security systems. Whereas in gradient-based optimization models, the parameters can be fine-tuned to adapt to the changing attack patterns without any additional requirements. In conclusion, any non-gradient based models cannot continually learn without significant aid. Hence we explore gradient-based models that can be trained online and can sequentially learn new data patterns without significant additional requirements.

C. FINETUNING

In many real-world applications, deep learning models are often tuned to match the changing data distribution. This helps the models stay updated and generate accurate predictions based on the existing requirements. For instance, in fast fashion, the trend keeps changing rapidly; hence the recommendation engines should adapt themselves to these changes to serve their customers efficiently, even at the cost of forgetting previous trends. Unfortunately, in the cybersecurity domain, this is not the case; although new attacks emerge every day, the older attacks remain in use. Therefore, tuning the models to detect new patterns might lead to significant consequences due to the catastrophic forgetting nature of such models where the model fails to detect previously learned patterns. Hence there is a need to learn new attack patterns without forgetting the older ones.

Batch Creation: The NSL-KDD dataset contains four different attack types, out of which two (U2R and R2L) amount to less than 1% of the dataset. Hence we avoid them and focus on the remaining two attacks, namely **Attack Type 1 - DOS and Attack Type 2 - Probe**. Similar to the NSLKDD dataset, where DOS and Probe attacks are considered as the two different types of attacks, in the case of the CICIDS dataset, we consider the attacks in Class 1 and Class 2 as the two major attack distributions. Therefore, in Class 1 we have **Attack Type 1 - {SSH-Patator, FTP-Patator, DoS Hulk, Dos GoldenEye, DoS Slowloris, DoS Slowhttptest, Heartbleed}** and in Class 2 we have **Attack Type 2 - {Web Attack - Brute Force, Web Attack SQL Injection, Web Attack XSS, Infiltration, Bot, Portscan, DDoS}.**

We create a total of 30 batches, each of size 5000 for the NSLKDD dataset and 20000 for the CICIDS 2017 dataset. Each batch contains around 40-60% of benign data points and the remaining with attack data points. The nature of the attack alternates after every five batches meaning the first five batches are of attack type 1, the next five are of attack type 2, the next five once again attack type 1 and so on. In order to mimic a real-world scenario, we assume that the batches arrive sequentially and the model only has access to the batch available at that time, meaning the model can access no previous or future batches. As batches become available one after another, the neural network model is fine-tuned to fit the latest batch without enforcing any constraints, and the model is tested on the subsequent batch of data. We repeat this for all 30 batches.

In this section, we demonstrate the effects of catastrophic forgetting by fine-tuning the model without any constraints and additional resources by emulating a real-world scenario using batches of alternating attack patterns. From Figure 5a and figure 6a, we can observe that the accuracy rates in the neural network models keep fluctuating when it moves from one attack type to another. The network forgets the patterns and properties of attack type 1 when it encounters attack type 2 and vice versa. This is reflective of the network's problems of forgetting old learnt patterns when new data distributions are encountered. In this simulation, the model's accuracy in detecting DOS attacks falls from 99.9% to 86.6%(-13.3%) from batch 5 to batch 10 for the NSL KDD dataset. Similarly, in the case of the CICIDS 2017 dataset, the accuracy of the model falls from 98.7% to 84.2%(-14.5%) when we transition from class 1 to class 2 attacks. When the model is tuned on attack type 2, it forgets the previously learned attack type 1 patterns, thereby decreasing its ability to detect them. This is a severe concern as the model cannot retain previous knowledge once it is fine-tuned on new data. This phenomenon is termed catastrophic forgetting in neural networks.

The standard solution to this problem is joint training which stores the previously encountered data and retrains the models periodically to retain the model's performance. This approach is computationally expensive and requires high amounts of storage to ensure good performance levels. Given the rapidly changing nature of the attack patterns, the frequency of retraining will also be high; therefore, this is not a viable solution. To counter the inefficiencies of these standard algorithms, we explore replay and network structure based continual learning models to achieve similar performance levels without significant additional requirements.

D. CONTINUAL LEARNING MODELS

There are three main continual learning strategies namely:

- Architecture Based
- Regularization Based
- Rehearsal Based

Architecture-Based models manipulate the design of the neural network by adding more layers/nodes to the existing structure to learn new data distributions and mitigate forgetting. By increasing the model's parameters, it can learn new attack classes and other dataset shifts. In comparison, regularization based models achieve continual learning by altering the loss functions and using distillations for selective weight consolidation to help the model retain past memories. Alternatively, buffer based algorithms utilize a replay buffer to store a subset of data and fine-tune the model on this buffer data periodically to remember all the encountered attack classes. Many other techniques also exist that are a hybrid version of the above techniques. This section analyses which classes of algorithms can and cannot be applicable for our intrusion detection task and the reasons for such distinctions. In this research work, we explore regularization based and rehearsal based strategies.

1) LEARNING WITHOUT FORGETTING (LwF)

Regularization-based techniques such as LwF [21] handle the new attack types by increasing the nodes in the final layer, thereby creating an updated neural network. This is achieved by adding additional nodes to the existing neural network, thereby increasing the model's parameters thus increasing their ability to adapt to new distributions. They rely on a combined training approach where the weights of the newly added nodes can be learnt without disrupting the knowledge gained through previous iterations stored in the existing nodes.

In this model, each known class(i.e. attack types) is allotted a node in the final layer, and all the classes share all the layers except the last layer. The shared layer parameters are θ_s, and θ_o is the class-specific parameter of the known classes. LwF adds θ_n more parameters to the final layer to include the newly encountered classes. The model aims to find the optimal values of θ_n, which will help perform well on the newer attacks but also constrain any changes in the θ_s and θ_o values so that catastrophic forgetting can be avoided. Unlike standard models, which depend on high storage and computational capacity, regularization-based models do not require additional resources.

To ensure that the model performs well on both the old and new attacks, the joint training approach uses different loss functions for training the old and new attack nodes in the final layer. The training algorithm follows a cross-entropy loss for the new nodes, which helps the model fit nicely on the new attack data points. Whereas for the older nodes, Knowledge Distillation Loss designed by Hinton [16] is being used, allowing the current model to approximate the prediction distribution of the previous model, ensuring that the past knowledge is not lost. The total loss of the model is a weighted sum of the old and the new attack losses; the model's weights ensure that no one task is given more priority over the other.

Notably, compared to the joint training approach, the regularization technique requires no additional storage requirements and avoids retraining the model periodically. Using the previous model's logits and comparing it with the current model using the distillation loss has voided any storage requirements. Furthermore, instead of completely retraining the model, we combinedly train the old and new attack nodes collectively, thus efficiently utilizing the existing knowledge.

Algorithm 1: Learning Without Forgetting

```
1  **Input:**
2  Shared parameters $\theta_s$, old task parameters $\theta_o$,
3  New training data $(X_n, Y_n)$, Trained DNN model
4  $Y_o \longleftarrow DNN(X_n, \theta_s, \theta_o)$
5  $\theta_n \longleftarrow randomly\_initialize(\mid \theta_n \mid)$
6  **Train:**
7  $\hat{Y}_o = DNN(X_n, \hat{\theta}_s, \hat{\theta}_o)$
8  $\hat{Y}_n = DNN(X_n, \hat{\theta}_s, \hat{\theta}_n)$
9  $\theta_s^*, \theta_o^*, \theta_n^* \longleftarrow$
10 $\underset{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n}{argmin} \left( \lambda_o L_{old}(Y_o, \hat{Y}_o) + L_{new}(Y_n, \hat{Y}_n) + R(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \right)$
```

In algorithm 1, the L_{new} is the categorical cross-entropy loss, whereas the L_{old} is the modified cross-entropy with the Knowledge Distillation, which increases the weight assigned to lower probabilities and performs well in approximating the output of two networks.

2) EXPERIENCE REPLAY

Rehearsal based strategies such as experience replay rely on a reservoir buffer of a predefined size to store a subset of the data points encountered thus far. Introduced as an off-policy learning algorithm to store various experiences in the reinforcement learning(RL) domain, the stored experiences are then used to fine-tune the RL models. This ensures that the model utilizes the older experiences for future iterations and helps stabilize the model's performance. Similarly, this technique can be extended to continual learning domains where there is a need to efficiently store and retain knowledge about the attack pattern encountered.

The model relies on creating a buffer of data points representing the distribution of all the attack patterns encountered. The buffer itself is a subset of the data points selected using the reservoir sampling algorithm that selects points at random. Algorithm 2 summarizes the sample selection process; up until the buffer has empty spaces, the encountered data points are directly added to the buffer; however, once the buffer is full, we generate a random number to decide whether or not that particular data point will replace an existing point in the buffer.

Algorithm 2: Reservoir Sampling

```
1  **Input:** Memory buffer M, number of samples
   encountered N, datapoints (x, y)
2  **if** $M < N$ **then**
3      M[N] $\longleftarrow (x, y)$;
4  **else**
5      j = sample random integer (min = 0, max = N);
6      **if** $j <\mid M \mid$ **then**
7          M[N] $\longleftarrow (x, y)$
8      **end**
9  **end**
```

As soon as a new batch of data is encountered, the neural network is fine-tuned using the dataset generated by combining the current batch with a subset of data points sampled from the buffer and the binary cross-entropy loss optimizing the model. Once the model is tuned, the buffer is updated by using the reservoir sampling algorithm.

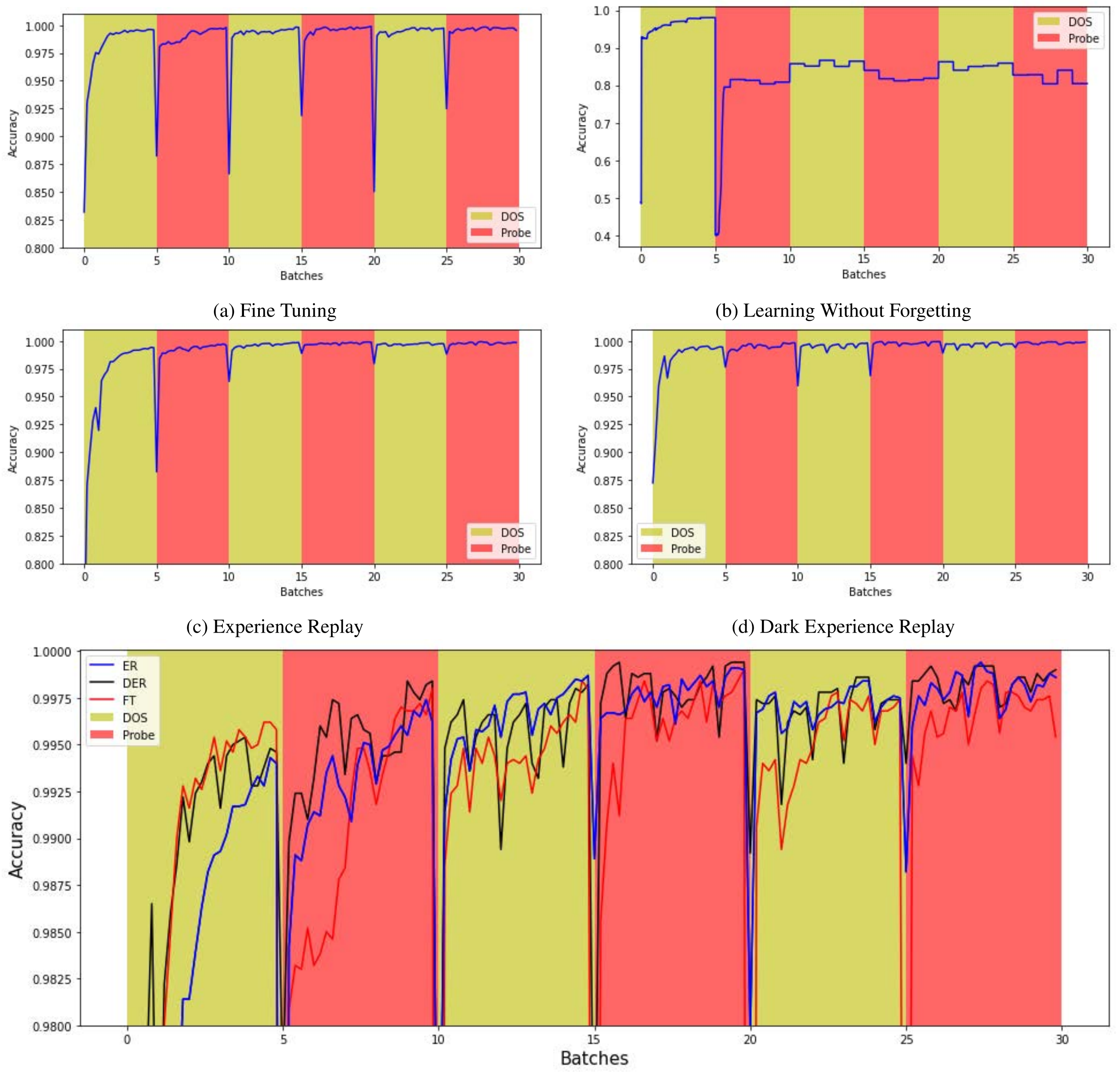
3) DARK EXPERIENCE REPLAY (DER)

Dark Experience Replay [5] is an updated version of the experience replay technique where each sample's output probability distribution (softmax) is stored along with its labels. This helps the model to approximate the original predictions of the models on past samples as described in Algorithm 3. The loss function is also modified to optimize both the current batch's performance and retain the knowledge of past tasks.

Algorithm 3: Dark Experience Replay

```
1  **Input:** dataset D, parameter $\theta$, trade-off value $\alpha$ and
   learning rate $\lambda$
2  M $\longleftarrow$ {}
3  **for** *(x,y) in D* **do**
4      $(x', z') \longleftarrow sample(M)$
5      $z \longleftarrow h_\theta(x)$
6      $reg \longleftarrow \alpha \mid\mid z' - h_\theta(x') \mid\mid_2^2$
7      $\theta \longleftarrow \theta + \lambda.\nabla_\theta[l(y, f_\theta(x)) + reg]$
8      $M \longleftarrow reservoir(M, (x, z))$
9  **end**
```

The approximation function of the neural network is f, the corresponding output logits are represented by $h_\theta(x)$ and the distribution of the output probabilities over various classes is $f_\theta(x) = softmax(h_\theta(x))$. The goal of the continual learning model is to optimize the parameters θ of the neural network model such that the combined loss function in equation 9 where L_t is defined in equation 10 of all the encountered tasks



(a) Fine Tuning

(b) Learning Without Forgetting

(c) Experience Replay

(d) Dark Experience Replay

(e) Combined Results

FIGURE 6. Comparison of continual learning models on NSL KDD.

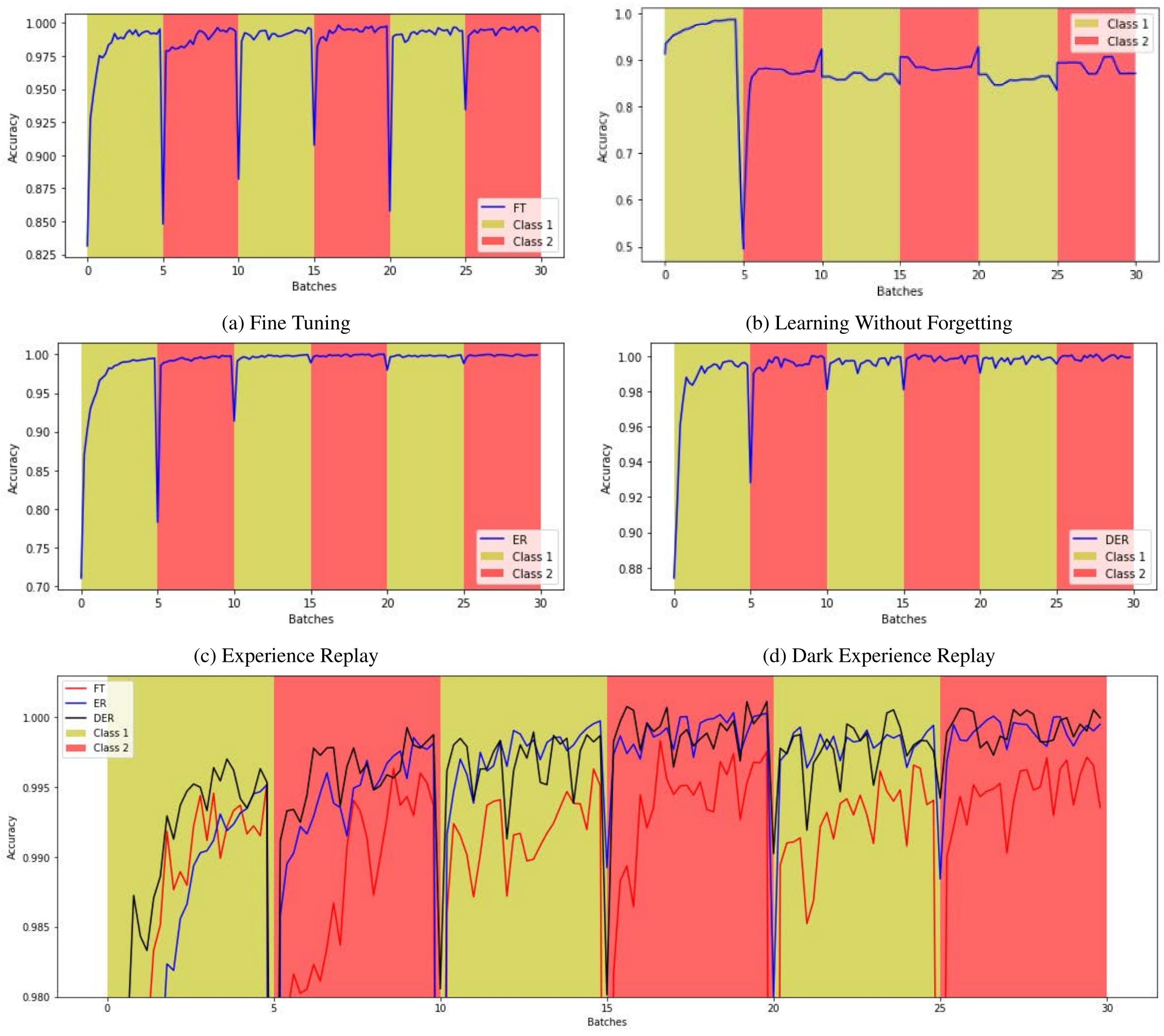
t_c thus far is minimized:

$$\underset{\theta}{argmin} \sum_{t=1}^{t_c} L_t \tag{9}$$

$$L_t = E_{(x,y)\sim D_t}[l(y, f_\theta(x))] \tag{10}$$

However, as we do not store all the data points from the previous batches due to the limited memory available, computing the loss for batches $1, 2, \ldots t_c - 1$ is not possible. Therefore the loss function in equation 9 is modified such that we search for the best parameters that fit our current batch t_c along with a regularization term that mimics the outputs of the previous batches. As the parameter values cannot be stored after every intermediate batch optimization, we store the probability outputs for all the data points in each batch directly in the memory buffer as soon as the batch is available. Therefore instead of only the class y, we store the complete probability distribution z(softmax) for each data point x.

$$L_{t_c} + \alpha \sum_{t=1}^{t_c-1} E_{x\sim M}\left[\| \, z - h_\theta(x) \, \|_2^2 \right] \tag{11}$$



(a) Fine Tuning
(b) Learning Without Forgetting
(c) Experience Replay
(d) Dark Experience Replay
(e) Combined Results

FIGURE 7. Comparison of continual learning models on CICIDS 2017.

The continual learning algorithm aims to minimize the loss function in equation 11 by tuning the model's parameters. To fine-tune the model, we utilize data points from both the current batch and the buffer. We sample a subset of the points from the buffer from which we compute the regularization loss, and the binary cross-entropy loss is computed for the current batch. The gradient of the parameters is computed with respect to the sum of these losses, using which they are fine-tuned in an iterative process.

E. COMPARISON OF CONTINUAL LEARNING ALGORITHMS

We use the batches created in section V-C for analyzing the performance of all the models.

1) LwF

The results of the LwF (regularization based) technique are displayed in Figure 5b and 6b. From these figures, we can understand that the catastrophic forgetting problem is eliminated as the accuracy rates remain almost constant even across attack transitions. Therefore the model does not forget one attack when trained on the other. However, the model's overall accuracy is only 84.92% and 88.35% for the NSL KDD and the CICIDS 2017 dataset, respectively, which is considerably lesser than our requirements for values above 95%. Although in batches 1-5, the model can reach accuracies as high as 98%, it drops as soon as we transition from Class 1 to Class 2 attacks.

The drop in accuracy post-transition can be attributed to the lack of discriminative training. Even though we train both the

TABLE 8. Performance comparison of algorithms.

Dataset	Algorithm	Overall Accuracy
NSL KDD	Finetuning	98.85%
	Experience Replay	98.93%
	Dark Experience Replay	99.36%
	Learning Without Forgetting	84.92%
CICIDS 2017	Finetuning	98.61%
	Experience Replay	98.95%
	Dark Experience Replay	99.46%
	Learning Without Forgetting	88.35%

old and the new nodes simultaneously, the losses computed for both the classes are independent of each other, meaning the predictions of the old and new nodes are never compared with one another. Therefore regularization based strategies work well when applied to classes significantly different from the existing ones as they can be easily differentiated. However, the accuracy drops when applied to classes with similar properties (such as DOS and Probe). For instance, regularization based strategies can distinguish a car and an apple more accurately than a car and a truck due to the dissimilarity between the classes. Therefore, when comparing two attack types from the same system, the model cannot easily distinguish between them because of their shared properties. Similar results were obtained by [44], where numbers were learnt on an incremental basis from the MNIST dataset.

Although catastrophic forgetting was eliminated, regularization based models are not a practical solution due to the lack of discriminative training in classes with similar distributions, which leads to poor performances. Therefore we explore efficient buffer based solutions that allow discriminative training between multiple similar classes, thus helping them differentiate efficiently.

2) EXPERIENCE REPLAY

In the fine-tuning technique, the model is trained on the current batch of data which contains either Class 1 or Class 2 attacks and not both. Therefore, at any given time, the model learns only one class of attacks. This leads to fluctuating accuracy rates during attack transitions. However, these issues are eliminated in the replay based models due to the presence of both Class 1 and Class 2 attacks in the buffer. Also, the presence of both the attacks classes in the buffer ensures that the models are trained in a discriminative setting, therefore, allowing the models to achieve high accuracy levels, unlike the regularization based techniques.

The accuracy of the model highly depends on the size of the buffer being utilized. Larger buffer sizes ensure that more information can be stored, consequently improving the model's accuracy but demands more resources. Whereas smaller buffer sizes require far fewer additional resources but the model's performance levels will decrease. In our experiments, we fix the buffer size to 10000 samples. Note that specific algorithms such as EXSTREAM [14] exist that help us select the data points intelligently by creating clusters for each class and determining the value of each point by measuring the distance from its label class. Depending on the value of the data point, it is either selected or rejected. Note that we achieve high-performance levels in our tasks despite ignoring such algorithms due to their high computational requirements.

Figures 5c and 6c summarize the results of the model; compared to the fine-tuning based model, the reservoir buffer achieves an accuracy of 98.93% for the NSL KDD dataset and 98.95% for the CICIDS 2017 dataset, and more importantly the accuracy does not drop significantly during attack transitions. Also, as more batches of data become available, the drop in accuracy gradually reduces, thus stabilizing the model's performance.

3) DARK EXPERIENCE REPLAY

The results of the analysis are presented in Figures 5d and 6d. Figures 5c, 5d, 6c and 6d conclude that, unlike the fine-tuning technique, there is no catastrophic forgetting due to the presence of both the attack classes in the buffer, and unlike regularization based techniques, the replay based techniques can achieve high accuracy levels due to discriminative training. Especially in the case of DER, the model's accuracy stabilizes after batch 15, therefore ensuring consistent results. From Figures 5e and 6e, we can observe that the accuracy of the DER model stays above the rest across multiple batches, closely followed by the ER model and then the fine-tuning based approach. Table 7 summarizes the overall performance of the algorithms on the batches of data from which we can observe that the overall accuracy of the DER model is 0.43% and 0.49% higher than the ER model for the NSL KDD and CICIDS 2017 dataset, respectively. This increase in accuracy is due to the nature of training, where the model is tuned to approximate its previous probability distribution rather than being forced to train on hard labels. The regularization loss ensures that the model creates smooth boundaries that, in turn, help them to generalize better. We can also observe that the replay strategies perform much better (around +15%) than the regularization based strategies. Also, note that although the fine-tuning based technique has better accuracy than the LwF technique, the problem of catastrophic forgetting is never addressed in that approach, whereas the LwF technique eliminates catastrophic forgetting at the cost of reduced performance.

Hence from the above analysis, it is clear that buffer based replay techniques perform better than other continual learning models but requires additional storage capacities. Considering that only buffer based models allow discriminative training, which is essential for differentiating between similar attack types in the continual learning setting, we conclude that the requirement of additional resources is an acceptable trade-off.

4) EXISTING WORKS

In tables 9 and 10, we compare the performance of various state of the art IDS models with our DER technique for both the NSL KDD and CICIDS 2017 dataset respectively. While

TABLE 9. Performance comparison with other IDS (NSL-KDD Dataset).

Algorithm and paper	Overall Accuracy	False Alarm Rate	Detection Rate
3CB-LSTM[24]	99.1%	0.5%	99.2%
AdaBoost[27]	98.9%	1.0%	99.61%
XGBoost[50]	83.34%	-	-
AutoEncoder, Extreme Learning Machine, DNN [23]	92.49%	14.72%	97.95%
Stacked Feature Selection with XG Boost [38]	99.87%	-	-
Random Tree, Naive Bayes Tree [18]	89.24%	-	-
XGBoost [20]	95.55%	2%	-
Dark Experience Replay	99.36%	0.35%	99.34%

TABLE 10. Performance comparison with other IDS(CIC-IDS dataset).

Algorithm and paper	Overall Accuracy	Precision	Recall
Deep Encoder[6]	99.2%	95.00%	98.90%
XGBoost [8]	96.00%	99.0%	79.00%
Logistic Regression+Decision Tree+Gradient Boosting [10]	98.8%	98.8%	97.1%
Deep Feed Forward Neural Network [12]	98.4%	97.79%	98.27%
Light GBM [49]	98.37%	98.14 %	98.37%
AdaBoost [17]	99.69%	99.7%	99.69%
Dark Experience Replay	99.46%	99.74%	99.20%

various models such as boosting techniques [27], [50] [17], LSTMs [24] and DNNs [12], [23] listed in table 9 and 10 perform well in the anomaly detection task and achieve accuracy over 95%, they do not have the capabilities to continuous learn new attack patterns. When the attack distribution changes and new patterns are introduced, none of these existing works will be able to efficiently update the models to adapt to the dynamic changes in the systems. Therefore, over time the performance of these models drops significantly. However, unlike these techniques, our continual learning models have the ability to learn new attack patterns and changes in data distribution while also maintaining high accuracy levels with low false-positive rates. Further, compared to existing works, our continual learning models are more robust and adaptable to changing system properties and handling day-0 attacks. The dark experience replay(DER) technique achieves an accuracy of 99.36% and 99.46% for the NSL KDD and the CICIDS 2017 dataset, respectively, while also handling constant changes in the data distribution without suffering from the catastrophic forgetting problem.

VI. CONCLUSION

In this research work, we introduced the problem of covariate shift and catastrophic forgetting in the context of intrusion detection systems for addressing the online learning problem. We propose an extensive eight-stage framework to help understand the nature and the magnitude of the shift in the dataset. As a part of the framework, we designed a novel feature importance based technique that determines the impact of individual feature drift on the performance of our IDS. We analyze the NSL KDD and the CICIDS 2017 dataset using the framework for understanding the nature and magnitude of drifts in the dataset, thus exemplifying the need for continual learning algorithms. After measuring the drift in the dataset, we efficiently address the problem by extending image-domain continual learning algorithms to cyber security in general and intrusion detection systems in specific and testing them on the NSL KDD and CICIDS 2017 data sets. We developed and compared the performance of three major continual learning models: learning without forgetting, experience replay and dark experience replay by analyzing their performances on alternating batches of attacks that emulate real-world distributions. We also demonstrate that the continual learning-based approaches can achieve an accuracy as high as 99% with false-positive rates below 0.5% with minimal additional requirements in storage capacity. From our analysis, we conclude that replay based continual learning models outperform traditional statistical techniques and state of the art boosting and DNN models in handling the covariate shift problem. To the best of our knowledge, this is the first application of continual learning strategies in the domain of large scale systems security to address the problems of covariate shift without suffering from catastrophic forgetting.

While doing the experimentation of the proposed study, we have used NSL-KDD and CICIDS 2017 dataset. We observed that there is reduction in degradation during change in attack dynamics. However we feel that there is a need to use formal real-world context-sensitive datasets related to various large-scale cyber–physical systems like cloud and smart grid networks. Therefore in future we plan to conduct extensive experimentation in real world large scale cyber-physical systems to further validate our study.

REFERENCES

[1] W. C. Abraham and A. Robins, "Memory retention—The synaptic stability versus plasticity dilemma," *Trends Neurosciences*, vol. 28, no. 2, pp. 73–78, Feb. 2005.

[2] I. Abrar, Z. Ayub, F. Masoodi, and A. M. Bamhdi, "A machine learning approach for intrusion detection system on NSL-KDD dataset," in *Proc. Int. Conf. Smart Electron. Commun. (ICOSEC)*, Sep. 2020, pp. 919–924.

[3] M. Ahsan and K. Nygard, "Convolutional neural networks with LSTM for intrusion detection," *EPiC Ser. Comput.*, vol. 69, pp. 69–79, 2020, doi: 10.13140/RG.2.2.24796.82567.

[4] A. O. Alzahrani and M. J. F. Alenazi, "Designing a network intrusion detection system based on machine learning for software defined networks," *Future Internet*, vol. 13, no. 5, p. 111, Apr. 2021, doi: 10.3390/fi13050111.

[5] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: A strong, simple baseline," 2020, *arXiv:2004.07211*.

[6] M. Catillo, M. Rak, and U. Villano, "2L-ZED-IDS: A two-level anomaly detector for multiple attack classes," in *Proc. AINA Workshops*, 2020, pp. 687–696.

[7] M. Choraś, R. Kozik, R. Renk, and W. Hołubowicz, "The concept of applying lifelong learning paradigm to cybersecurity," in *Proc. Int. Conf. Intell. Comput.* Cham, Switzerland: Springer, 2017, pp. 663–671.

[8] L. D'hooge, T. Wauters, B. Volckaert, and F. D. Turck, "Inter-dataset generalization strength of supervised machine learning methods for intrusion detection," *J. Inf. Secur. Appl.*, vol. 54, Oct. 2020, Art. no. 102564, doi: 10.1016/j.jisa.2020.102564.

[9] C. Feutry, P. Piantanida, F. Alberge, and P. Duhamel, "A simple statistical method to detect covariate shift," in *Proc. 27th Éme Colloque Francophone de Traitement du Signal et Des Images (Gretsi)*, 2019, pp. 1–4.

[10] Q. R. S. Fitni and K. Ramli, "Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems," in *Proc. IEEE Int. Conf. Ind. 4.0, Artif. Intell., Commun. Technol. (IAICT)*, Jul. 2020, pp. 118–124.

[11] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends Cogn. Sci.*, vol. 3, no. 4, pp. 128–135, Apr. 1999.

[12] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison," *J. Netw. Comput. Appl.*, vol. 169, Nov. 2020, Art. no. 102767.

[13] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *Proc. ESANN*, 2016.

[14] T. L. Hayes, N. D. Cahill, and C. Kanan, "Memory efcient experience replay for streaming learning," 2018, *arXiv:1809.05922*, doi: 10.1109/ICRA.2019.8793982.

[15] F. Herrera. (2011). *Dataset Shift in Classification: Approaches and Problems*. Accessed: Dec. 21, 2022. [Online]. Available: http://iwann.ugr.es/2011/pdf/InvitedTalk-FHerrera-IWANN11.pdf

[16] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[17] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset," *IEEE Access*, vol. 8, pp. 32150–32162, 2020.

[18] J. Kevric, S. Jukic, and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," *Neural Comput. Appl.*, vol. 28, no. S1, pp. 1051–1058, Dec. 2017.

[19] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, and A. Grabska-Barwinska, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[20] Y. N. Kunang, S. Nurmaini, D. Stiawan, A. Zarkasi, and J. Firdaus, "Automatic features extraction using autoencoder in intrusion detection system," in *Proc. Int. Conf. Electr. Eng. Comput. Sci. (ICECOS)*, Oct. 2018, pp. 219–224, doi: 10.1109/ICECOS.2018.8605181.

[21] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.

[22] V. Lomonaco, D. Maltoni, and L. Pellegrini, "Rehearsal-free continual learning over small non-I.I.D. batches," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 989–998.

[23] S. A. Ludwig, "Intrusion detection of multiple attack classes using a deep neural net ensemble," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2017, pp. 1–7, doi: 10.1109/SSCI.2017.8280825.

[24] M. Mahdavisharif, S. Jamali, and R. Fotohi, "Big data-aware intrusion detection system in communication networks: A deep learning approach," *J. Grid Comput.*, vol. 19, no. 4, pp. 1–28, Dec. 2021.

[25] University of Maryland. (2021). *Study: Hackers Attack Every 39 Seconds*. Accessed: Feb. 18, 2021. [Online]. Available: https://eng.umd.edu/news/story/study-hackers-attack-every-39-seconds

[26] A. Soutif–Cormerais, M. Masana, J. Van de Weijer, and B. Twardowski, "On the importance of cross-task features for class-incremental learning," 2021, *arXiv:2106.11930*.

[27] M. Mazini, B. Shirazi, and I. Mahdavi, "Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 31, no. 4, pp. 541–553, Oct. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157817304287

[28] G. Mohi-ud-din, "NSL-KDD," *IEEE Dataport*, Dec. 2018, doi: 10.21227/425a-3e55.

[29] N. Yadav, S. Pande, A. Khamparia, and D. Gupta, "Intrusion detection system on IoT with 5G network using deep learning," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–13, Mar. 2022.

[30] University of North Georgia. (2021). *Cybersecurity: A Global Priority Career Opportunity*. Accessed: Feb. 3, 2021. https://ung.edu/continuing-education/news-and-media/cybersecurity.php

[31] S. Pande, A. Kamparia, and D. Gupta, "Recommendations for DDOS attack-based intrusion detection system through data analysis," in *Proc. 2nd Doctoral Symp. Comput. Intell.*, D. Gupta, A. Khanna, V. Kansal, G. Fortino, and A. E. Hassanien, Eds. Singapore: Springer, 2022, pp. 899–909.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[33] S. Rabanser, S. Günnemann, and Z. C. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst., (NeurIPS)*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds. Vancouver, BC, Canada, 2019, pp. 1394–1406. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/846c260d715e5b854ffad5f7%0a516c88-Abstract.html

[34] S. Rastegari, P. Hingston, and C.-P. Lam, "Evolving statistical rulesets for network intrusion detection," *Appl. Soft Comput.*, vol. 33, pp. 348–359, Aug. 2015, doi: 10.1016/j.asoc.2015.04.041.

[35] H. Raza, G. Prasad, and Y. Li, "EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments," *Pattern Recognit.*, vol. 48, no. 3, pp. 659–669, Mar. 2015, doi: 10.1016/j.patcog.2014.07.028.

[36] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2001–2010.

[37] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016, *arXiv:1606.04671*.

[38] S. Pande, A. Khamparia, and D. Gupta, "Feature selection and comparison of classification algorithms for wireless sensor networks," *J. Ambient Intell. Hum. Comput.*, Aug. 2021, doi: 10.1007/s12652-021-03411-6.

[39] K. Sethi, E. S. Rupesh, R. Kumar, P. Bera, and Y. V. Madhav, "A context-aware robust intrusion detection system: A reinforcement learning-based approach," *Int. J. Inf. Secur.*, vol. 19, no. 6, pp. 657–678, Dec. 2020.

[40] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, vol. 1, Jan. 2018, pp. 108–116.

[41] K. Siddique, Z. Akhtar, M. A. Khan, Y.-H. Jung, and Y. K. and, "Developing an intrusion detection framework for high-speed big data networks: A comprehensive approach," *KSII Trans. Internet Inf. Syst.*, vol. 12, no. 8, pp. 4021–4037, Aug. 2018, doi: 10.3837/tiis.2018.08.026.

[42] B. A. Tama and S. Lim, "Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation," *Comput. Sci. Rev.*, vol. 39, Feb. 2021, Art. no. 100357, doi: 10.1016/j.cosrev.2020.100357.

[43] C. Tang, N. Luktarhan, and Y. Zhao, "SAAE-DNN: Deep learning method on intrusion detection," *Symmetry*, vol. 12, no. 10, p. 1695, Oct. 2020, doi: 10.3390/sym12101695.

[44] G. M. van de Ven and A. S. Tolias, "Generative replay with feedback connections as a general strategy for continual learning," 2018, *arXiv:1809.10635*.

[45] F. Wiewel and B. Yang, "Continual learning for anomaly detection with variational autoencoder," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 3837–3841.

[46] P. Wu and H. Guo, "LuNet: A deep neural network for network intrusion detection," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2019, pp. 617–624, doi: 10.1109/SSCI44817.2019.9003126.

[47] Y. Xiao and X. Xiao, "An intrusion detection system based on a simplified residual network," *Information*, vol. 10, no. 11, p. 356, Nov. 2019, doi: 10.3390/info10110356.

[48] N. G. Nair, P. Satpathy, and J. Christopher, "Covariate shift: A review and analysis on classifiers," in *Proc. Global Conf. Advancement Technol. (GCAT)*, Oct. 2019, pp. 1–6, doi: 10.1109/GCAT47503.2019.8978471.

[49] Y. Hua, "An efficient traffic classification scheme using embedded feature selection and LightGBM," in *Proc. Inf. Commun. Technol. Conf. (ICTC)*, May 2020, pp. 125–130.

[50] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 3854–3861.

[51] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," *Comput. Netw.*, vol. 174, Jun. 2020, Art. no. 107247. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128619314203

SAI PRASATH is currently pursuing the undergraduate degree with the School of Electrical Sciences, IIT Bhubaneswar, India. He has expertise in machine learning and cyber security.

KAMALAKANTA SETHI received the B.Tech. degree in information science from the National Institute of Science and Technology, Berhampur, India, in 2009, the M.Tech. degree from the National Institute of Technology Warangal, India, in 2014, and the Ph.D. degree from the School of Electrical Sciences, IIT Bhubaneswar, India, in 2021. He is working as an Assistant Professor with the Indian Institute of Information Technology, Sri City, India. He has expertise in cryptography and network security.

DINESH MOHANTY received the B.Tech. degree in computer science and engineering from the IIT Bhubaneswar, in May 2021. His research interests include wireless network security, cloud security, intrusion detection systems, and reinforcement learning.

PADMALOCHAN BERA (Member, IEEE) received the Ph.D. degree from the IIT Kharagpur, India, in 2011. He is currently working as an Associate Professor with the School of Electrical Sciences, IIT Bhubaneswar, India. He has more than ten years of applied research experience in his research areas. His work essentially focus on formal modeling and analysis of network configurations and security protocols for verification of different safety and security constraints towards detection and prevention of threats in networks. He also works on designing processes and protocols for software defined networks and applied cryptography. He has a number of research funding from Government agencies (DRDO, SERB, Meity) and Industries (BEL, CPRI, Cisco, Intel) on security assessment of defense networks, cryptography and software defined networks. He has published more than 40 research papers in reputed journals and conferences. His major research interests include network and cyber-physical systems security, cloud computing, formal verification and optimization, and cryptography. He is a member of ACM. He has research collaborations with various international academic institutes, such as Purdue University, University of North Carolina, University of Missouri, Oxford University, and Florida International University. He served on the editorial board and organizing committee in various journal and conferences.

SUBHRANSU RANJAN SAMANTARAY (Senior Member, IEEE) received the B.Tech. degree from UCE Burla, the Ph.D. degree from NIT Rourkela, and the postdoctoral degree from McGill University, Canada.

Currently, he is a Professor, a OPTCL Chair Professor, and the Head of the School of School of Electrical Sciences, IIT Bhubaneswar, India. His major research interests include PMU and wide area measurement, intelligent protection for transmission systems, including FACTs, micro-grid protection including distributed generation, micro-grid planning, wide-area based dynamic security assessment in large power networks, and smart-grid technologies.

He is a member of IEEE Power Systems Stability Sub-Committee. He is a fellow of Indian National Academy of Engineering (INAE) and a fellow of Institution of Engineering and Technology (IET), U.K. He has received the Prestigious SERB STAR Award-2021, the Director's Commendation for Outstanding Research-2021, the IEEE PES Chapter Outstanding Engineer Award-2020, the NASI-SCOPUS Young Scientists Awards-2015, the IEEE PES Technical Committee Prize Paper Award-2012, the Samanta Chandra Sekhar Award, and the Odisha Bigyana Acadmey-2010. He was a Editor of IEEE TRANSACTIONS ON SMART GRID and IEEE TRANSACTIONS ON POWER DELIVERY, an Associate Editor of IET, Generation, Transmission & Distribution, and a Guest Editor of IEEE SENSOR JOURNAL. He is an Associate Editor of IEEE SYSTEMS JOURNAL.