Continual Learning for Intrusion Detection Systems

A report submitted in partial fulfillment of the requirements of the requirements for the Project Part-II

Dual Degree Programme - B. Tech. and M.Tech

in

Computer Science & Engineering

by

Sai Prasath S

17CS02002

Under the supervision of

Dr. Padmalochan Bera

Department of Computer Science and Engineering
School of Electrical Sciences
Indian Institute of Technology Bhubaneswar
Orissa, India - 752050

DECLARATION BY THE SCHOLAR

I certify that:

- The work contained in this thesis is original and has been done by me under the guidance of my supervisor.
- The work has not been submitted to any other Institute for any degree or diploma.
- I have followed the guidelines provided by the Institute in preparing the thesis.
- I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
- Whenever I have used materials (data, theory and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the reference section.
- In case if something is used and it belongs to you, let author know about it. He will either attribute you or take it out from here.

CERTIFICATE

This is to certify that the thesis entitled **"Continual Learning for Intrusion Detection Systems "**, submitted by **Sai Prasath S** to Indian Institute of Technology Bhubaneswar, is a record of bonafide research work under my supervision and I consider it worthy of consideration for the degree of Dual Degree Programme - B. Tech. and M.Tech of the Institute.

Place: IIT Bhubaneswar
Date: May 13, 2021

Supervisor
Department of Computer Sc. and Engg.
School of Electrical Sciences
Indian Institute of Technology Bhubaneswar

ACKNOWLEDGEMENT

I would like to express my deep and heartfelt gratitude for my respected guide and mentor, Dr. Padmalochan Bera , for his evergreen blessings, valuable advice, support, good wishes and encouragement all alone the journey of this work. Without his undisputed support, this work would not have been true in the light of daylight. I am also grateful to Mr. Kamalakanta Sethi and Mr.Dinesh Mohanty for assisting me in the implementation works and for their valuable comments.

Place: Indian Institute of Technology Bhubaneswar
Date: May 13, 2021

Sai Prasath S

Abstract

Network attacks are a growing security risk in our modern societies, especially with the growing number of smart devices. Along with our daily use of utility devices such as televisions and mobile phones, growing interests in the Internet of Things (IoT), cloud computing and safety-critical systems that are essential for the society, such as electric and water grids, are often under severe threat and are vulnerable against such attacks. Thus protecting these critical infrastructures considering the fast-growing consumer network becomes of paramount importance. Manually detecting large scale intrusions is very challenging and usually time-consuming; although statistical analysis techniques help detect the attacks to an extent, finding complex patterns and hidden relationships across various parameters might not always be possible. To this extent, artificial intelligence (AI) has emerged as the go-to tool to tackle this issue.

However, deep neural network suffers from catastrophic forgetting where they completely and abruptly forget previously learned information upon learning new information. This study presents a continual learning-based deep neural network for an intrusion detection system developed to overcome catastrophic forgetting. We present two main strategies: the rehearsal-based Dark Replay buffer and regularisation-based learning without forgetting(LwF) algorithms. The ability of the model to learn new attacks is verified using multiple metrics such as accuracy and false-positive rates across different attack types.

We test and verify our models in the context of the famous NSL-KDD dataset, which suffers from a dataset drift problem between the train and the test set. We also present how the dataset drift problem can be detected, analyzed and quantified using various statistical and machine learning-based techniques. We compare the performance of our models with joint-training based learning and fine-tuned models to evaluate their relative performance. We conclude that application of continual learning for NIDS is crucial and certain rehersal based continual learning models perform better than other techniques but require sufficient memory for guaranteeing performance.

Keywords: Continual learning, NSL-KDD, catastrophic forgetting, dataset drifting

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Networked computing has become indispensable to people's life. From daily communications to commercial transactions and small businesses to large enterprises, all activities can be or will soon be done through networked services. Any vulnerabilities in the networked devices and computing platforms can expose the whole network to various attacks and bring about disastrous consequences. Hence, effective network intrusion detection (NID) solutions are ultimately essential to modern society.

Initial NID designs are signature-based, where each type of attacks should be manually studied beforehand, and the detection is performed based on the attack's signatures. This kind of approaches are, however, not suitable to the fast- growing network and cannot cope with attacks of increasing volume, complexity and volatility. For the security of such a large scale ever-expanding network, we need an intrusion detection system that is not only able to quickly and correctly identify known attack types but also is adaptive and intelligent enough to learn unknown and evolved attacks, which leads to our AI-based solutions. The machine learning-based NID enables the detection system to discover network attacks without much need for human intervention [1].

So far, investigations on AI-based solutions are mainly based on two schemes: anomaly detection and misuse detection. The anomaly detection identifies an attack based on its anomalies deviated from the profile of regular traffic. Nevertheless, this scheme may have a high false-positive rate, mainly if the regular traffic is not well profiled, and the profile used in the detection is not fully representative [2]. Furthermore, obtaining a fully representative regular traffic profile for a dynamically evolving and expanding network is unlikely. The misuse detection, on the other hand, focuses on the abnormal behaviour directly. The scheme can learn attacks based on a labelled training dataset, where both normal and attacked traffic data are marked. Given sufficient labelled data, a misuse-detection design can effectively generate an abstract hypothesis of the boundary between normal and malicious traffic. The hypothesis can then be used to detect attacks for

future unknown traffic. Therefore the misuse detection is more feasible and practical than anomaly-based detection [2], [3] and has been adopted in real-world systems, such as Google, Qihoo 360 and Aliyun.

Most of the misuse based NID models are trained once offline and are deployed online for continuous monitoring. However, constantly varying attack patterns and day-zero attacks affect the system's performance, which is usually not desirable, especially in safety-critical infrastructures. For instance let us consider that we have trained a neural network to detect DDOS attacks and Probe attacks. When the neural network encounters new attacks such as R2L or U2R, it fails to detect them with high accuracy because it has not encountered them previously and thus has no idea what are the features/patterns of those attacks. To solve this problem usually models are fine-tuned to learn new patterns but this leads to them forgetting previously learnt patterns. In this case the model learns the R2L and the U2R patterns but forgets the DDOS and the Probe attack patterns. This is termed as catastrophic forgetting more details about this are explained in the later sections of this paper. In our research work we aim to learn the new attack types(U2R and R2L) without forgetting the previously learnt patterns(DDOS and Probe).

The University of Maryland estimated a new cyberattack somewhere on the internet every 39 seconds, and around 64% of all companies worldwide have at least experienced one cyber-attack to date. It also estimated that the average cost of a data breach in 2020 would exceed $150 million. Any lack of security may potentially destroy the company financially, and its public relations might face catastrophic damages. Therefore this is a common problem faced by many institutions world-wide.

Despite the recent success in building AI-based models with high accuracy rates and low false-positive rates, the task of building a continual learning model for intrusion detection systems mostly remain unexplored. In our research work, we use the famous NSL-KDD dataset[4] to analyze and compare our results with other states of the art models. The following are the major contributions of our research work:

1. Study the KDDTrain and the KDDTest dataset for shifting patterns
2. Quantify the shift in the dataset using novelty detection techniques and discriminative distance
3. Analyze the problem of catastrophic forgetting in AI-models
4. Build AI-based continual learning systems to solve this problem

The remainder of this paper is organized as follows: Section II provides a brief background of machine learning for network intrusion detection. The basics of the techniques used in the research work, such as continual learning and catastrophic forgetting, are detailed in Section III. The detailed analysis of the data set used along with the ML

models is presented in Section IV. The evaluation and comparison of continual learning-based models with other state-of-the-art designs are given in Section V. We conclude the research work in Section VI along with the proposed future works in Section VII.

1.2 Related Works

The use of AI-based models for IDS has gained huge popularity recently with the development of powerful and sophisticated models along with the access to computational power. This section will describes some important results examined during our research.

Many traditional machine learning (ML) methods such as SVM [4,5], KNN [6], ANN [7], decision tree [8,9], and random forest (RF) [10] have been used for developing intrusion detection system, and they achieved reasonable success with smaller datasets. Support Vector Machines(SVM) models finds a decision boundary to separate various attack types based on the features of the dataset, where as Artificial Neural Networks(ANN) performs very similar operation by optimizing the loss function but after extracting more complex features from the dataset. K-Nearest Neuighbours(KNN) models find k-nearby points in the dataset using statistical distance(euclidean, manhattan etc.) to find the attack type of new data points. RF and Decision Tree models use entropy based algorithms to create a tree and segregate various datapoints by imposing conditions on features.

The ensemble machine learning algorithm is an ML technique that generally combines multiple weak learners to build a strong learner [11]. These models use majority voting based strategies to arrive at a single conclusion and generally perform better than individual classifiers due to pooled knowledge. Recently, deep learning (DL)-based approach has been used for developing efficient and reliable IDS. Researchers have explored various possible architectures such as attention based learning to remember only the important features, auto-encoder based learning for feature extraction and LSTM/GRU based deep learning models for recalling previous knowledge.

Javaid et al. [12] developed a network intrusion detection system (NIDS) based on self-taught learning (STL), which combines a sparse autoencoder with softmax regression. Shone et al. [13] proposed a DL-based NIDS. Their classification model is constructed using a stacked non-symmetric deep autoencoder (NDAE) for efficient feature extraction. Intelligent deep learning based models have been developed by Peilun et al. [14] by using a combination of CNN and RNNs to capture time as well as spatial dependencies.

Although most studies achieve higher accuracy and lower false positive rates, they fail to analyse the dataset used for any shift in patterns. When the dataset distribution changes from the train test to the test set, the ML based systems developed fails. To the best of our knowledge this is the first work in the field of continual learning for intrusion detection systems.

Chapter 2

Background

2.1 Covariate Shift

In most real-world systems we assume that the dataset distribution remain unchanged that is data collected at any instance is independent and identically distributed(iid) but that is not always the case.

Dataset shift is present in most practical applications for reasons ranging from the bias introduced by experimental design, to the mere irreproducibility of the testing conditions at training time. For example, in an image classification task, training data might have been recorded under controlled laboratory conditions, whereas the test data may show different lighting conditions. Some researchers consider the problem of spam email filtering: successful "spammers" will try to build spam in a form that differs from the spam the automatic filter has been built on. Without adapting to these new and varying attacks, the developed AI system is useless in a short period of time.

Dataset shift[15,16] is a challenging situation where the joint distribution of inputs and outputs differs between the training and test stages. Covariate shift is a simpler particular case of dataset shift where only the input distribution changes (covariate denotes input), while the conditional distribution of the outputs given the inputs remains unchanged.

$$P_{train}(X) \neq P_{test}(X)$$
$$P_{train}(X \mid Y) = P_{test}(X \mid Y)$$

In the above figure 2.1, the blue points represents the train set data points. The green line is the model learnt from the available data but due to covariate shift where the test points(black crosses) drifted away from the points in the train set the true function represented in red couldn't be learnt. Thus when the model encounters the test set data points it's performance is affected.

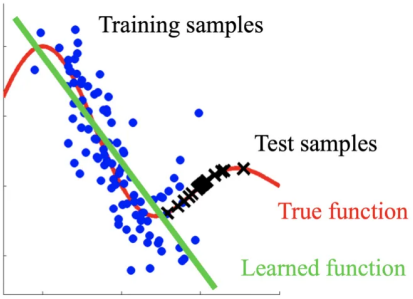


Figure 2.1: Covariate Shift between Train and Test sets

2.2 Catastrophic Forgetting

Catastrophic forgetting or catastrophic interference [17] was first recognized by McCloskey and Cohen [1989]. They found that, when training on new tasks or categories, a neural network tends to forget the information learned in the previous trained tasks. This usually means a new task will likely override the weights that have been learned in the past, and thus degrade the model performance for the past tasks. Without fixing this problem, a single neural network will not be able to adapt itself to all the scenarios, because it forgets the existing information/knowledge when it learns new things. This was also referred to as the stability-plasticity dilemma in Abraham and Robins [2005]. On the one hand, if a model is too stable, it will not be able to consume new information from the future training data. On the other hand, a model with sufficient plasticity suffers from large weight changes and forgets previously learned representations. We should note that catastrophic forgetting happens to traditional multi-layer perceptrons as well as to DNNs. Shadow single-layered models, such as self-organizing feature maps, have been shown to have catastrophic interference too.

A concrete example of catastrophic forgetting is transfer learning using a deep neural network. In a typical transfer learning setting, where the source domain has plenty of labeled data and the target domain has little labeled data, fine-tuning is widely used in DNNs to adapt the model for the source domain to the target domain. Before fine-tuning, the source domain labeled data is used to pre-train the neural network. Then the output layers of this neural network are retrained given the target domain data. Backpropagation-based finetuning is applied to adapt the source model to the target domain. However, such an approach suffers from catastrophic forgetting because the adaptation to the target domain usually disrupts the weights learned for the source domain, resulting inferior inference in the source domain.

2.3 Continual Learning

Given a potentially unlimited stream of data, a Continual Learning algorithm should learn from a sequence of partial experiences where all data is not available at once. Continual learning algorithms may have to deal with imbalanced or scarce data problems, catastrophic forgetting, or data distribution shifts. We consider continual learning a synonym of Incremental Learning, Lifelong Learning and Never Ending Learning. For the sake of simplicity, for the remainder of the chapter we refer to all Continuous, Incremental and Lifelong learning synonyms as Continual Learning (CL).

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

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

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 $C^t = \{c_1^t, c_2^t, c_3^t, ..., c_{n^t}^t\}$ and D^t is the corresponding training dataset. The total number of classes in the task is represented as $N^t = \sum_{i=1}^{n} \mid C^i \mid$. The training dataset D^t is a set of datapoints $\{(x_1^t, y_1^t), (x_2^t, y_2^t), ...(x_{m^t}^t, y_{m^t}^t)\}$ where x are the input features and $y \in \{0, 1\}^{N^t}$ is a one hot ground truth label vector corresponding to x[25]. 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 [23].

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

Chapter 3

Methodology

3.1 Dataset Description

The KDD-CUP99 dataset is a widely used dataset for testing systems developed to detect computer network traffic anomalies. This dataset contains many records in different attack types. However, in the studies performed on this dataset, it has been determined that there are some cases that adversely affect the performance of the systems tested in the dataset. To solve this problem, it has been suggested to use a new dataset called NSL-KDD to test the proposed systems, eliminating some records in the KDD CUP99 dataset. In the NSL-KDD dataset, the unnecessary samples from the dataset to be used for training are cleared and the size of the dataset is set to a reasonable value for the anomaly detection.

The following table summarizes the attack distribution in the train and test set. The train set consist of 23 different types of attacks but the test set consists of 38, some of which are day 0 attacks meaning the ML model wasn't trained on these attacks. All the data points are divided into 5 major categories namely Normal, DDOS, Probe, U2R and, R2L for multi-class classification task.

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

Further analysis of the dataset reveals that both the train and the test set share 21 different attack types which constitutes 99.29% and 83.36% of the datapoints in their respective datasets. Two attacks are exclusive to the train 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 above described values are summarized in the following table:

There are only around 1,50,000 data points which are comparatively less to train sophisticated neural networks. Many research works points out this shortcoming of the

Attack Type	Train Dataset	Test Dataset
Shared	99.29%	83.36%
Exclusive to Train	0.71%	0.0%
Exclusive to Test	0.0%	16.64%

dataset which restricts then from traning huge neural networks which might overfit when trained on smaller dataset. Also such small datasets might not be able to completely test and verify the various requirements of a robust NIDS. The NSL-KDD dataset itself is outdated and doesn't 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'll explore later makes it suitable for our analysis to test and compare various continual learning based models.

3.2 Dataset Preprocessing

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

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. 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.

3.3 Models Tested

In our research all the experiments are performed on the random forest(RF) classifier and a deep neural network(DNN) classifier. This subsection explains the models considered in detail:

3.3.1 Random Forests

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees

at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees.

$$Entropy = \sum_{i=1}^{n} p_i log(p_i)$$

$$GiniIndex = 1 - \sum_{i=1}^{n} (p_i)^2$$

Random forests are an extended version of decision trees that usually over fit on the training data and fail to generalize. Random forests are made up of many such decision trees trained on subset of the data and the features considered. Each tree in the random forest model is trained on a subset of the given dataset by sampling with replacement. This is called bagging. Also the features based on which the branches are generated are random chosen from a subset of features. This has shown to reduce the bias in the model and perform better across various data sets. The strength of the random forest model relies on creating decision tress that are not correlated with one another such that the majority decision is the best possible outcome.

They use the entropy/gini index metric defined above to create the necessary splits for learning the patterns in the dataset. The metric values are computed for a subset of features as discussed above and the feature that reduces the uncertainty in the split the most is selected. In our model we consider 50 decision trees for the RF classifier along with entropy as the splitting criteria.

3.3.2 Deep Neural Network

The deep neural network is an advanced model of the classical feed-forward network (FNN). A DNN consists of an input layer to provide the dataset features, a set of hidden layers used to extract complex features from the input features that can help explain the relationship better, and an output layer that generates the expected output for a given set of inputs. Adjacent layers in the neural network are connected through weights that help extract high-level features from the input data. The exact computations through which this is achieved is elaborated in the below equation. In conclusion, the whole neural network serves as a complex non-linear function that finds the relationship between the input features and the output by incrementally modifying the initial weights assigned to the nodes in the network.

As the number of layers in the DNN increases it causes the vanishing and exploding gradient issue. To handle the vanishing and exploding gradient issue, the ReLU non-linear activation was introduced. ReLU helps to protect weights from vanishing by the

gradient error. Compared to other nonlinear functions, ReLU is more robust to the first-order derivative function since it does not zero for high positive and negative values of the domain. The proposed DDN architecture is as follows: an input layer, 5 hidden layers and an output layer with varying output units depending on the algorithm being implemented. Softmax activation function is applied on the output layer so that the output probabilities are normalized and they sum to 1.The model uses a gradient descent based optimization to approximate the function for the given task by minimizing the below loss equation:

$$Loss = \sum_{i=1}^{n} [e_i log(p_i) + (1 - e_i) log(1 - p_i)] \tag{3.1}$$

Chapter 4

Results

4.1 Evaluating Performance

In this subsection we evaluate the performance of the Random Forest model and the DNN model on the NSL-KDD Dataset for a binary classification problem. The models are trained on the KDDTrain dataset and are tested on the KDDTest dataset. The below table summarizes the results of the analysis:

Model	**Train Dataset**	**Test Dataset**
Random Forest	99.67%	77.22%
DNN	96.42%	74.29%

Both the Random Forest model and the DNN model performed well on the train dataset but their performance in terms of accuracy dropped by 22% on the test set. The result of the analysis is in agreement with the previous analysis of the dataset where we found that 16.63% of points in the test dataset are unique to the testset attack types which affects the performance of our models.

Multiple research works develop complex models and test it on the NSL-KDD dataset but the performance on the test data is not satisfactory. The problem is not with the models but rather due to the shift in the dataset where new classes of attacks are being added and the old classes of data has shifted. Thus trying to build models to solve this issue will never improve the performance rather concentrating on how to develop algorithms to continuous learning new knowledge is necessary, this process is termed as continual learning which is explored in detail in this research work.

4.2 Quantifying the shift

4.2.1 Histogram Overlap Computations

This section analyses whether any individual features create the shift in the dataset. For identifying the particular features, we bin the values of the corresponding feature in the train set and the test set separately into 100 bins. The values in the bin are then normalized for a fair comparison across multiple features. The overlap between the histogram of the train and the test set is computed and used as a metric to measure the similarity of the features. The higher the value(closer to 1), the higher is the similarity between the features and the lower the value(closer to 0), the lower the similarity between the features.

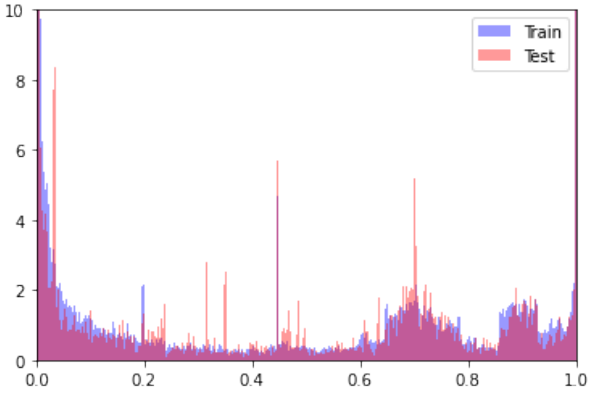


Figure 4.1: Histogram Overlap for feature 28

Features	Overlap (%)
dst_bytes	0.947
duration	0.953
service_pop_3	0.963
hot	0.978
dst_host_srv_count	0.979
dst_host_count	0.979
num_failed_logins	0.981
is_guest_login	0.981
service_ftp	0.984
srv_count	0.985

Table 4.1: Top 10 Least Feature Histogram Overlap

Figure 2 visualizes the histogram overlap of the feature dst_host_count which is then used for our calculations. The above table summarizes the top 10 drifted features obtained using the histogram overlap computations. Usually a set of features which 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 important for the classification purpose. However in our dataset the maximum drifted feature has an overlap of 94.7% meaning that the drift of the features are very limited. Therefore we conclude that there are no significant drifts of individual features, and hence we refrain from dropping any features.

4.2.2 Kolmogorov–Smirnov Test

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

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

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

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

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

The equation 1 is the definition of the empirical distribution function where the indicator function I is computed using the equation 2. Once the empirical distribution of both the sample distributions 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

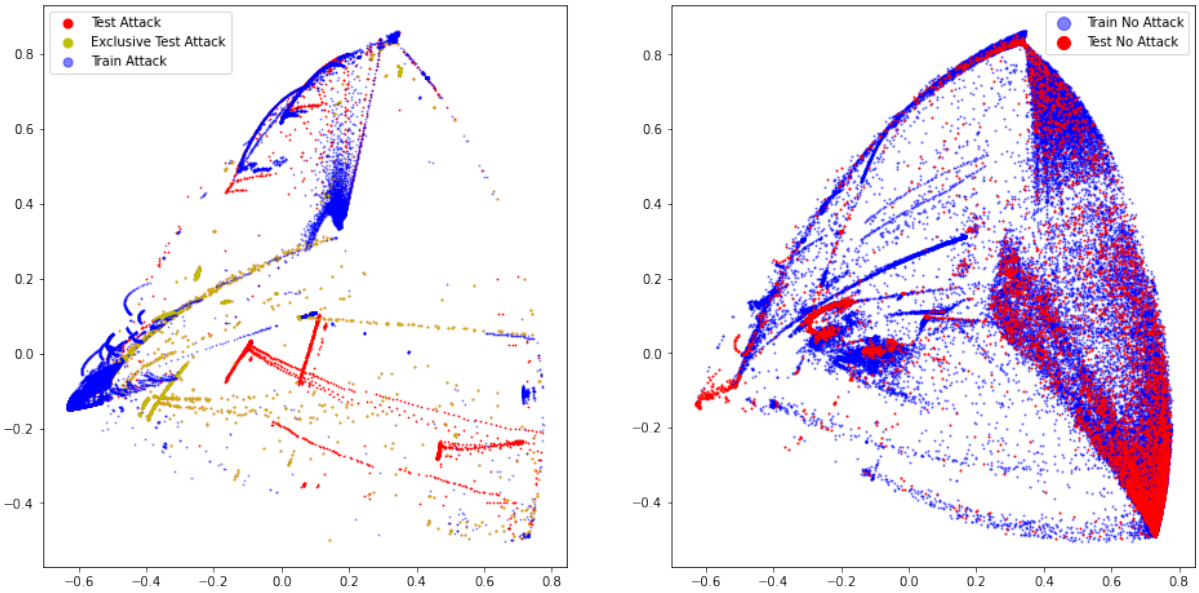


Figure 4.2: Distribution of the attack datapoints

that they might have different distributions but upon further analysis when compared with the critial value of 0.00983(c(0.05)=1.36) none of the KS statistics was above this limit. Hence we can conclude that all the features are derived from the same distribution for a p-value of 0.05.

4.2.3 2D visualization

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

In figure 4.2 the blue points represents the train attack data points. The dataset shift in the test attack points is visible where the red dots which share the same attack types as blue have drifted due to the non-stationary nature of the attacks also new cyber attacks have also been introduces which don't fir any previously learnt pattern, these are represented in the yellow points. However in figure x(2) which visualizes the normal(no attack) points, there is a very good overlap between the train and the test set. Hence the definition of what a benign datapoint is hasn't changes but as new attacks are introduced and there is a shift in the old attacks there is a need for relearning the decision boundary.

4.2.4 Novelty Detection

It is the identification of new or unknown data or signals that a machine learning system is not aware of during training. Novelty detection methods 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 available in the sklearn library is used to analyse and visualize the data distribution.It measures the local deviation of density of a given sample with respect to its neighbors. It is local in that the anomaly score depends on how isolated the object is with respect to the surrounding neighborhood. More precisely, locality is given by k-nearest neighbors, whose distance is used to estimate the local density. By comparing the local density of a sample to the local densities of its neighbors, one can identify samples that have a substantially lower density than their neighbors. These are considered outliers.

The local outlier detection algorithm creates a boundaries to differentiate between inliers and outliers. All the datapoints within the boundary are considered as inliers and the points outside are considered as outliers meaning they don't belong to the data distribution that these models were previously fit on.

Along with the Local Ourlier Factor algorithm, we also use a One-Class SVM that can be used to detect outliers in data by fitting a SVM model on a dataset containing a single class. We use a constant ν for representing the impurity in the training dataset(These fraction of ν points are considered as outliers in the training process). The impure points are labelled as -1 where as all the other points are labelled as 1. Using this labelled dataset we train the SVM Classifier which is applied on the testset to detertime the shift in the dataset.

Algorithm	**Train-Inliners**	**Test-Outliers**
Local Outlier Factor	98.14%	19.90%
One Class SVM	98.00%	17.19%

The above table summarizes the results of the novelty detection algorithm: For both the algorithms we use a ν value of around 2% for the training stage, as expected around 17-20% of datapoints in the test set are outliers which corresponds to the 16.64% data points that are exclusive to the testset.

4.2.5 Discriminative Distance

All the above analysis performed in the dataset involved individual features or 2 PCA components but not the whole dataset, to utilize the whole dataset we use the discriminative distance metric to quantify the shift.

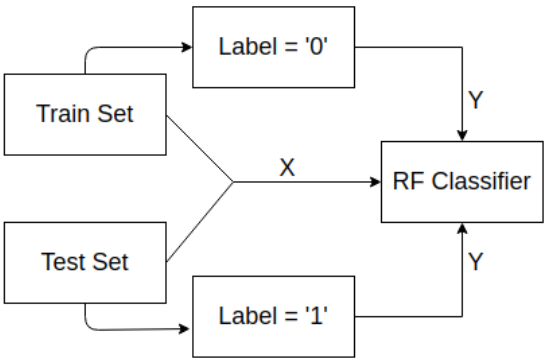


Figure 4.3: Discriminative Distance

Discriminative distance technique tries to differentiate between the training and the testing data using ML-based classifiers. The intuition behind the technique is that if the classifier is not able to distinguish between the datapoints then they share the same distribution, whereas if the classifier can distinguish them with high accuracy rates then 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's much more powerful than the previous ones.

The RF classifier was able to successfully classify the train and the test set with a 10-fold cross validation accuracy of 91.28%. This clearly shows the change in distribution between the train and the test set.

4.2.6 Feature Importance

In this subsection we calculate the importance of various features in the binary attack classification problem using the Extra Trees Classifier and compute how many of those features drift during the dataset shift from the train to the test set. We analyse the importance of the top-10 drifting features in the attack classification problem individually in the train and the test set. The below table summarizes the top-10 drifting features, which are computed using their relative importance foe the train-test classification problem described in the previous subsection.

From Table 4.2 it is clear that none of the features drift a lot, the maximum drift is 6.10% which is corresponds to our results from the statistical analysis. Also these top-10 drifted features contribute towards 74.78% importance and 61.20% importance in the train and test set attack classification respectively. Hence dropping these features to reduce the bias in the dataset will affect the attack classification problem as they are significantly important for that task.

Features	Train-Test	Train Binary	Test Binary
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
src_bytes	4.42	8.42	7.73
dst_host_same_srv_rate	4.39	0.58	1.22
dst_bytes	4.23	22.26	23.37
same_srv_rate	4.10	0.63	1.22
protocol_type_tcp	3.91	0.81	2.28
Total Contribution		74.78	61.20

Table 4.2: Feature Importance for top-10 drifted Features

4.2.7 Shuffling the Dataset

To ensure that the drop in accuracy from the train to the test set is not due to the inherent lack of patterns or features but only occurs due to the drifting of the data, we implement a shuffling based analysis. To verify this claim, the train and the test set data points are mixed and shuffled, which ensures 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 is able to achieve a very high 10-fold cross validated accuracy of 99.49% and an fpr value of 0.39%.

This helps us conclude that ML models are able to learn the attack patterns in the dataset and perform well if all the attacks were available during the training stage itself.

4.3 Catastrophic Forgetting while learning

Catastrophic forgetting is a problem in which the AI-models forget old data distributions when learning new ones. However as the distribution changes continuously learning new patterns without forgetting the older ones is of paramount importance and it forms the basis of continual learning.

In this subsection we analyse the effect of catastrophic forgetting in neural networks. We create batches of new data from the actual data such that two different attack patterns repeat consecutively. Each batch consist of 10,000 points out of which 5000 points are normal where as the other 5000 points are either DOS(Denial Of Service) or Probing attack. R2L and U2R are ignored for this analysis due to the lack of enough data points in those attack types. We consider the DOS attack to be Type 1 and the other as Type

2. The dataset is created such that the attack types alternate between type 1 and 2 after five consecutive batches.

By testing our models on the newly created batches, the capacity of knowledge retained by the models can be determined. In order mimic a real world scenario we assume that the batches arrive in an sequential manner and the model only has access to the batch available at that time meaning no previous or future batches can be accessed by the model. As batches become available one after another the neural network models are fine tuned to fit to the latest batch without enforcing any constraints.

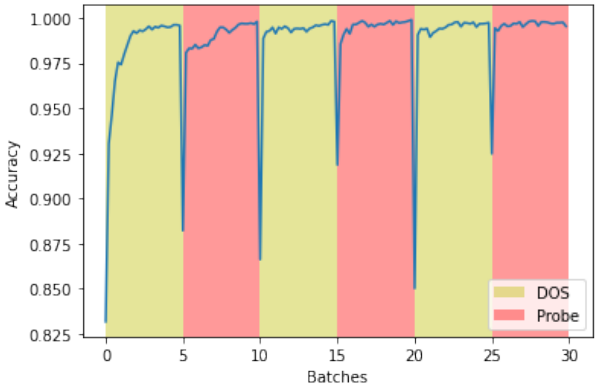


Figure 4.4: Catastrophic Forgetting in ML Models

From figure 4.4 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 the DOS attacks when it encounters the Probe attacks and vice versa. This is reflective of the networks problems of forgetting old learnt patterns when new data distributions are encountered. For instance in this analysis the initial performance of the model on the DOS data is around 99.9% at the end of batch 5, but when the DOS attacks emerge once again at the end of batch 10, the accuracy drop to 86.61% which is a 13.29% drop. This is a serious concern as the model is not able to retain any previous knowledge once it encounters new information. This problem is termed as catastrophic forgetting.

4.4 Solving Covariate Shift

4.4.1 Reweighting Similar Datapoints

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 datapoints 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 loss function that is being optimized by the model.

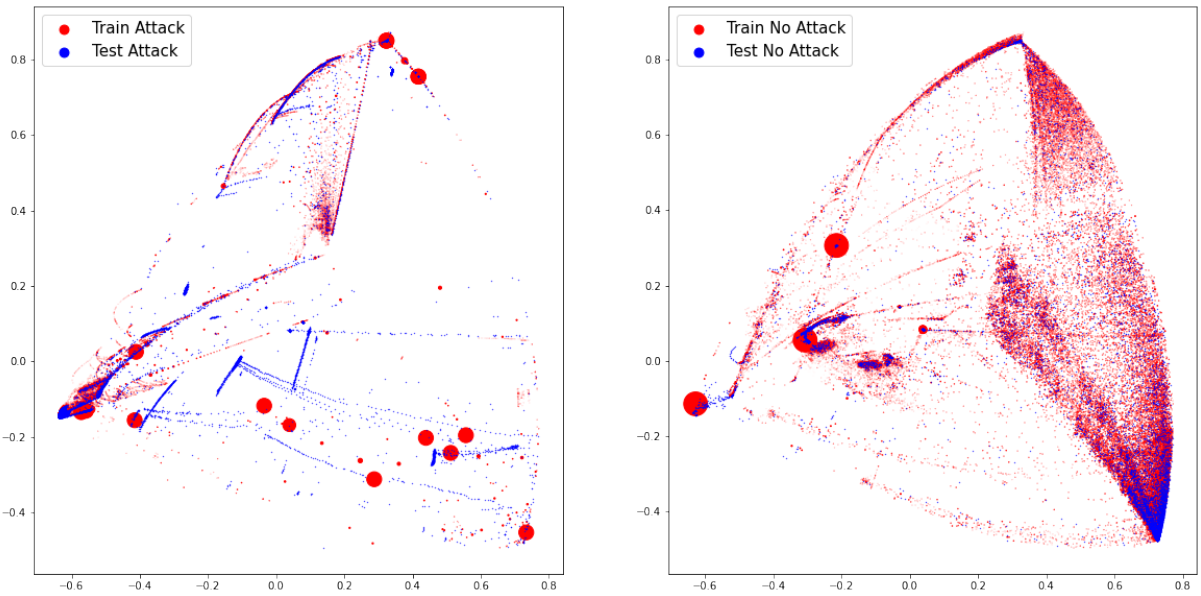


Figure 4.5: Reweighting trainset data

$$total\_loss = \frac{1}{n}\sum_{i=1}^{n} w_i * loss\_function(y, f(x_i)) \tag{4.1}$$

$$\sum_{i=1}^{n} w_i = 1 \tag{4.2}$$

In equation 4.1 which is a generalised loss function the parameter w_i 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. Hence the model will learn data patterns that overlap with the test set and not only those that are exclusive to the train set.

To implement the technique we use a random forest classifier to generate the weights of the datapoints in the train set. We randomly sample 2000 points from the test dataset which will represent 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 datapoints are labelled as '1'. After combining and shuffling the labelled dataset, we use a 10-fold cross validation technique is used generating the weights of the train set data points by using the output of the random forest classifier. In the 10-fold validation technique the dataset is segmented into 10 parts out of which 9 are used for training and models predicts the output for the remaining segment. This is performed 10 time until the output values of all the train set datapoints are obtained.

$$r_i = \left(\frac{1}{f(x_i)} - 1\right) \tag{4.3}$$

$$w_i = c_1 + c_2 * \frac{r_i}{\sum_{i=1}^{n} r_i} \tag{4.4}$$

If the output of the classifier 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 doesn't belong to the test set distribution. The weights are computed using equations 3 and 4 where r_i is the prediction probability of the random forest classifier and c_1 and c_2 are constants.

In figure 4.5 the size of the points corresponds to the weights allotted to them. Most of the large points are located where there is considerable overlap with the test set distribution and locations where the overlap is less the size of the points are small. We use the first two components of the PCA algorithm to plot the above figure. Using this technique the accuracy of the model has increased from 76.40% to 78.93% in the test set which is a 2.53% increase in the accuracy but the train set accuracy has reduced from 98.36% to 96.73% which is a reduction of 1.63%. This tradeoff is expected due to the different weights that were allotted.

4.4.2 Non Gradient Based Machine Learning Models

To solve the problem of covariate shift and arrival of new classes, we need a model that can be tuned to remember previous patterns and learn new ones simultaneously. Most ML-based models are offline models meaning they are trained exactly once and don't change their parameters after the initial training phase. Due to this property they can't be retrained and fit on new batches of data, rather they have to be completely retrained which means that they'll definitely lose any previously learnt patterns.

In this analysis we retrain the models only when the accuracy falls below 90%. For example in case of Random Forests the model initially calculates the entropy for each features and determines the criteria for splitting. However when new batches arrive the entropy changes according to the data distribution of the arrived batch, therefore forcing the model to recompute the entropy once again to determine the feature split.

From figure 4.6 it can be clearly observed that when new batches arrive the RF models fails in detecting them due the difference in the data distribution on which it was trained. Even after learning the DOS attacks in batches 1 through 5 and relearning them in batches 10 through 15 the model under performs when similar data values are obtained in batch 25. This is expected due to the retraining nature of the algorithm.

This wouldn't be a problem if the model used any gradient based optimization techniques where the parameters can be tuned accordingly. In conclusion any non-gradient

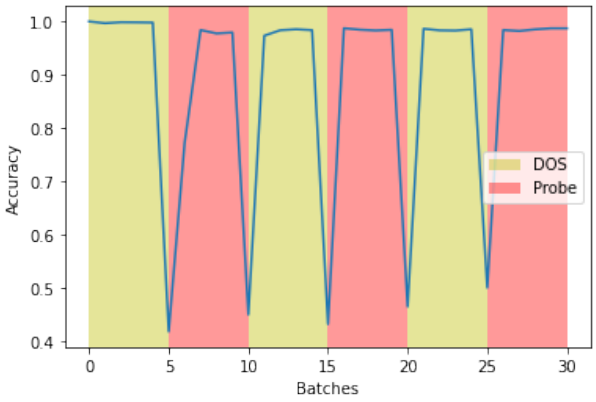


Figure 4.6: Catastrophic Forgetting in Random Forests

based models don't have the ability to continually learn. Hence we concentrate on gradient based models that can be trained online and have the ability to sequentially learn new data patterns.

4.4.3 Experience Replay

Rehearsal based Experience Replay using the Reservoir buffer technique tackles the problem of catastrophic forgetting by replaying a subset of the batch data that has been encountered uptil now. It uses a buffer of a predefined size to store the datapoints encountered, on a random basis and uses this buffer data to train the deep learning model. By selecting a subset of the data encountered till now we believe that the subset will be a combined representation of all the batches so far.

Algorithm 1: Reservoir Sampling

Input: Memory buffer M, number of samples encountered N, datapoints (x, y)
if $M < N$ **then**
  M[N] $\longleftarrow (x, y)$
else
  j = sample random integer (min = 0, max = N);
  if $j <| M |$ **then**
    M[N] $\longleftarrow (x, y)$
  end
end

Algorithm 1 summarizes the process of selecting a subset of datapoints to fill the buffer from the current batch of data. If the buffer has empty spaces, the batch data is directly loaded, where as if the buffer is already full we randomly select a datapoint that already exist in the buffer and decide whether or not to replace it with a point from the current batch.

Figure 4.7b summarizes the results of the model, compared to fine tuning based model the reservoir buffer performs better as the accuracy doesn't drop after every single attack transition. Also as more and more batches become available the drop in accuracy gradually reduces overtime leading to a model that is capable of learning both the DOS and the Probe attacks without any compromises.

4.4.4 Dark Experience Replay (DER)

Dark Experience Replay[21] is an updated version of the reservoir buffer technique where along with the buffer based training the loss function also includes the ability to retain knowledge about previous distributions. 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 of all the encountered tasks t_c thus far is minimized:

$$\underset{\theta}{argmin} \sum_{t=1}^{t_c} L_t \tag{4.5}$$

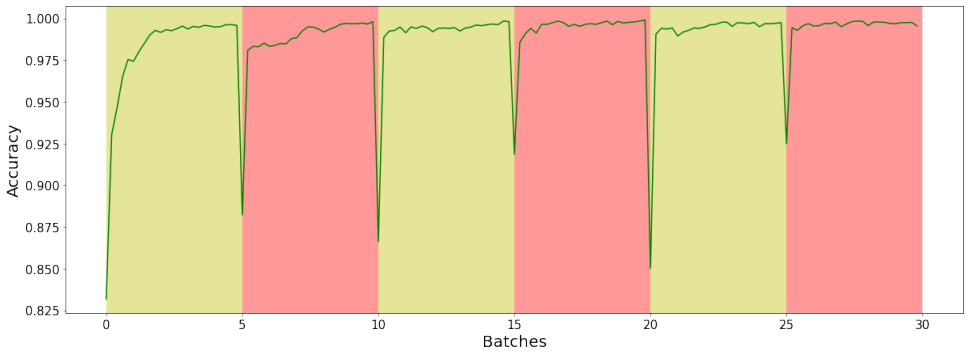
$$L_t = E_{(x,y)\sim D_t}[l(y, f_\theta(x))] \tag{4.6}$$

However as we don't store the all the datapoints from the previous batches due to the limited memory available, computing the loss for batches $1, 2, ..t_c - 1$ is not possible. Therefore the loss function 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 can't be stored after every intermediate batch optimization, we store the probability outputs for all the datapoints in each of the 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 for each datapoint x.

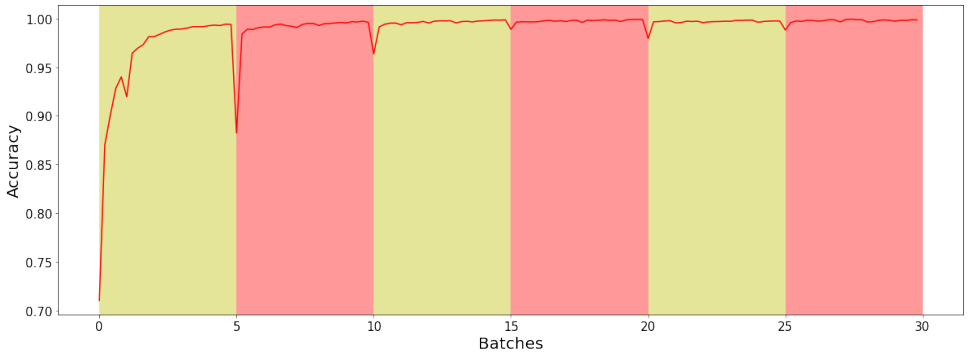
$$L_{t_c} + \alpha \sum_{t=1}^{t_c-1} E_{x\sim M}\left[|| \ z - h_\theta(x) \ ||_2^2\right] \tag{4.7}$$

We use the above loss function to optimize the parameter values by sampling from the replay buffer. We use gradient descent based optimization to find the optimal values. The exact algorithm implemented is as follows:

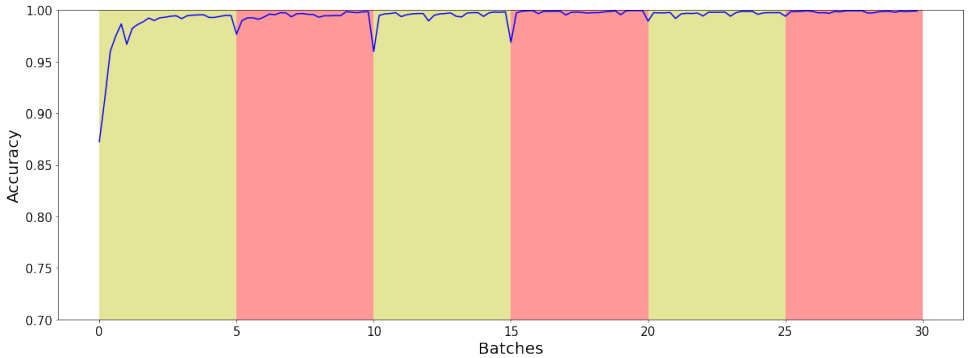
The results of the analysis are presented in figure 4.7c and it can be clearly observed that as more batches of data becomes available the the drop in accuracy is almost zero. Also from figure 4.7d the catastrophic forgetting is almost eliminated in both the expe-



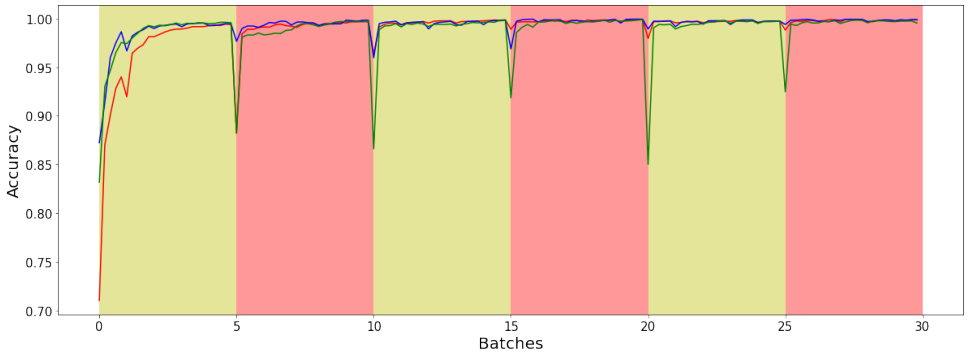
(a) Finetuning



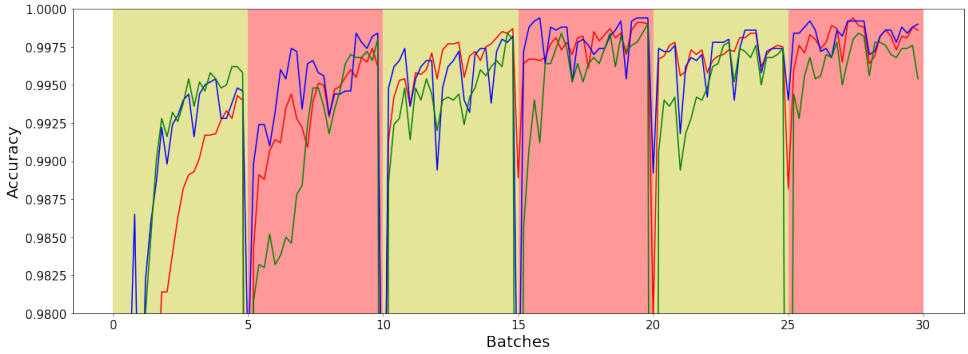
(b) Experience Replay



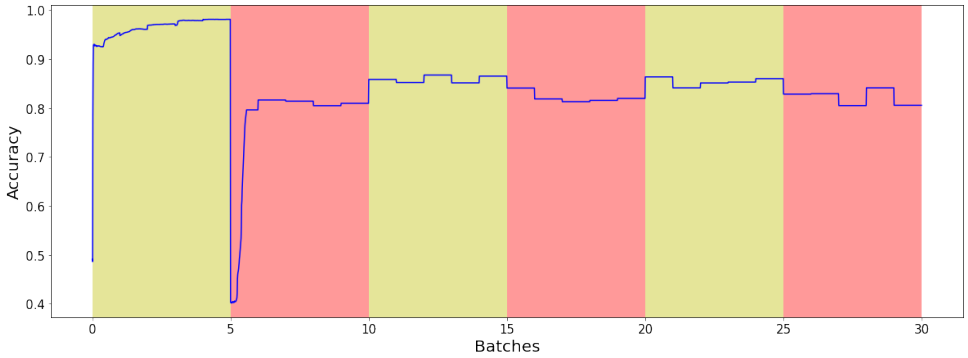
(c) Dark Experience Replay



(d) Combined Results



(e) Combined Zoomed



(f) Learning Without Forgetting

Figure 4.7: Results of Multiple Techniques

rience replay technique and the dark experience replay technique when compared to the fine tuning technique. Especially from figure 4.7e it clear that the DER has superior performance over the other two algorithms, this is expected as DER is the only technique that uses the actual probability distribution to retain previous knowledge.

4.4.5 Learning Without Forgetting (LwF)

The accuracy achieved by the replay based techniques highly depends on the size of the buffer being used. The higher the size of the buffer, more data can be stored leading to high accuracy and if the size of the buffer is relatively low it can affect the performance of the models significantly. Especially for very long term monitoring buffer based techniques might need their buffer size to increase over time to accommodate the new attack types and data drifts.

Regularization based techniques such as LwF[24] are a potential alternative to the buffer based systems, they introduce new parameters in the network to gain knowledge on the new attack types and enforce constraints on the old parameters so that the previous knowledge is not lost. In this model each known class(i.e. attack types) is allotted a node in the final layer and all the layers except the last layer is shared by all the classes. The shared layer parameters is θ_s, and θ_o is the class specific parameter of the known classes.

Algorithm 2: Dark Experience Replay

Input: dataset D, parameter θ, trade-off value α and learning rate λ

M $\longleftarrow$ {}

for *(x,y) in D* **do**

$(x', z') \longleftarrow sample(M)$

$z \longleftarrow h_\theta(x)$

$reg \longleftarrow \alpha \mid\mid z' - h_\theta(x') \mid\mid_2^2$

$\theta \longleftarrow \theta + \lambda.\nabla_\theta[l(y, f_\theta(x)) + reg]$

$M \longleftarrow reservoir(M, (x, z))$

end

Algorithm 3: Learning Without Forgetting

Input: Shared parameters θ_s, old task parameters θ_o, New training data (X_n, Y_n), DNN model trained on old data

$Y_o \longleftarrow DNN(X_n, \theta_s, \theta_o)$

$\theta_n \longleftarrow randomly\_initialize(\mid \theta_n \mid)$

Train:

$\hat{Y_o} = \text{DNN}(\text{X}_n, \hat{\theta_s}, \hat{\theta_o})$

$\hat{Y_n} = \text{DNN}(\text{X}_n, \hat{\theta_s}, \hat{\theta_n})$

$$\theta_s^*, \theta_o^*, \theta_n^* \longleftarrow \underset{\hat{\theta_s}, \hat{\theta_o}, \hat{\theta_n}}{argmin} \left(\lambda_o L_{old}(Y_o, \hat{Y_o}) + L_{new}(Y_n, \hat{Y_n}) + R(\hat{\theta_s}, \hat{\theta_o}, \hat{\theta_n}) \right)$$

LwF adds θ_n more parameters to the final layer to include the newly encountered classes. The aim of the model is to find the optimal values of θ_n which will help perform well on the newer attacks but also to constrain any changes in the θ_s and θ_o values so that catastrophic forgetting can be avoided.

In algorithm 3, the L_{new} is the categorical cross entropy loss where as the L_{old} is the modified cross entropy with the Knowledge Distillation loss which increases the weight assigned to lower probabilities and performs well in approximating the output of two networks. From figure 4.7f the model reduces the catastrophic forgetting problem but the overall accuracy rates falls down to around 80-85% from the above 98% values that was previous found in the buffer based techniques. The model is able to learn both the attack types combined but not without affecting each others accuracy. This is a drawback of the regularization based technique. Similar results were obtained by [22] where different numbers from the MNIST dataset were learnt on a class incremental basis.

Algorithm	**Overall Accuracy**
Finetuning	98.85%
Experience Replay	98.93%
Dark Experience Replay	99.36%
Learning Without Forgetting	84.92%

Table 4.3: Performance of various Algorithms

Hence from the above analysis it is clear that buffer based replay techniques perform better than regularization based model but demand memory capacity that can increase with the growing collection of data.

Chapter 5

Conclusion

The research in the field of continual learning for IDS is limited as continual learning itself is an developing field. Continual learning provides extensive possibilities for IDS to be more efficient, intelligent and less expensive all at the same time. Given the rapid change in environments, attack patterns and evolving technology there is a requirement for proactive anomaly based IDS in IoT, Cloud and other large scale scalable infrastructures. In this research work we extensively explored the various properties and data distributions in the NSL-KDD dataset and analyzed multiple continual based learning algorithms to address the problem of dataset drifting as well as continual learning. We also evaluate the merits of the techniques in addressing the problem in the context of cyber security. We have shown that buffer based systems are more superior to other techniques especially for class incremental learning but also face certain drawbacks.

Chapter 6

Future Work

The following is the proposed work plan:

1. The continual learning techniques considered in this research work can be improved and new techniques can be considered. Such as using GAN to generate datapoints which reduces the size of the dataset required to be stored for experience relay. This technique is called "Generative replay"

2. Explainable AI models such as SHAP can help us exactly figure out where certain models succeeds and certain models fail. This can help us evaluate the performance of various techniques in an in-depth manner.

3. We are also looking into building privacy preserving machine learning models using homomorphic encryption that can guarantee performance without compromising user privacy.

4. The concepts and ideas can be extended to cloud/ smart grid based environments without any loss in performance. Thus deploying and evaluating the models in simulations of real-life scenarios will be an interesting task.

Chapter 7

References

[1] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," IEEE Network, vol. 8, no. 3, pp. 26–41, 1994.
[2] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in 2010 IEEE Symposium on Security and Privacy, pp. 305–316, IEEE, 2010.
[3] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," Computers Security, vol. 28, no. 1-2, pp. 18–28, 2009.
[4] https://www.unb.ca/cic/datasets/nsl.html
Kuang, F., Xu, W., Zhang, S.: A novel hybrid KPCA and SVM with GA model for intrusion detection. Appl. Soft Comput. 18, 178–184 (2014)
[5] Reddy, R.R., Ramadevi, Y., Sunitha, K.V.N.: Effective discrimi- nant function for intrusion detection using SVM. In: Proceedings of International Conference on Advance in Computing, Commu- nication and Information (ICACCI), pp. 1148–1153 (2016)
[6] Li, W., Yi, P., Wu, Y., Pan, L., Li, J.: A new intrusion detection system based on KNN classification algorithm in wireless sensor network. J. Electron. Comput. Eng. 2014, 240217 (2014)
[7] Bivens, A., Palagiri, C., Smith, R., Szymanski, B., Embrechts, M.: Network-based intrusion detection using neural networks. Intell. Eng. Syst. Artif. Neural Netw. 12(1), 579–584 (2002)
[8] Quinlan, R.: Induction of decision trees. Mach. Learn. 1(1), 81–106 (1986)
[9] Ross Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, Burlington (1993)
[10] Farnaaz, N., Jabbar, M.A.: Random forest modelling for network intrusion detection system. Procedia Comput. Sci. 89, 213–217 (2016)
[11] Polikar, R.: Ensemble based systems in decision making. IEEE Circuits Syst. Mag. 6(3),
[12] Javaid, A., Niyaz, Q., Sun, W., Alam, M.: A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on

Bio-inspired Information and Communications Technologies (BICT), pp. 21–26 (2015)
[13] Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q.: A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Top- ics Comput. Intell. 2(1), 41–50 (2018). https://doi.org/10.1109/ TETCI.2017.277279210.1109/TETCI.2017.2772792
[14] Peilun Wu, Hui Guo, Nour Moustafa, "Pelican: A Deep Residual Network for Network Intrusion Detection", Dependable Systems and Networks Workshops (DSN-W) 2020 50th Annual IEEE/IFIP International Conference on, pp. 55-62, 2020.
[15] https://cs.nyu.edu/ roweis/papers/invar-chapter.pdf
[16] Alexander Gepperth and Barbara Hammer. Incremental learning algorithms and applications. In European Symposium on Artificial Neural Networks (ESANN), Bruges, Belgium, 2016.
[17] Robert M. French. Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences, 3(4):128–135, 1999.
[18] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research
[19] Hunter, J. D., Matplotlib: A 2D graphics environment, Computing in Science & Engineering
[20] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
[21] Buzzega, Pietro Boschini, Matteo Porrello, Angelo Abati, Davide Calderara, Simone. (2020). Dark Experience for General Continual Learning: a Strong, Simple Baseline.
[22] Ven, Gido M. van de and A. Tolias. "Generative replay with feedback connections as a general strategy for continual learning." ArXiv abs/1809.10635 (2018)
[23] Choraś, Michał Kozik, Rafał Renk, Rafal Hołubowicz, Witold. (2017). The Concept of Applying Lifelong Learning Paradigm to Cybersecurity. 663-671. 10.1007/978-3-319-63315-2_58.
[24] Li, Zhizhong, and Derek Hoiem. "Learning without forgetting." IEEE transactions on pattern analysis and machine intelligence 40.12 (2017): 2935-2947.
[25] Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., van de Weijer, J. (2020). Class-incremental learning: survey and performance evaluation. arXiv preprint arXiv:2010.15277.