Machine Learning for Network Intrusion Detection

Student Name: Christos Christou Mavropoulos

Supervisor Name: Dr. Noura Al Moubayed

Submitted as part of the degree of BSc Computer Science to the

Board of Examiners in the Department of Computer Sciences, Durham University

April 2020

Abstract —

Context/Background The increased use of online storage and processing alongside the internet of things has allowed individuals and corporations to store an accumulative amount of sensitive information on networks. There has also been an increase in the capabilities of nefarious individuals and organisations to attack or expose such networks in hope of either gaining access to these networks or to the sensitive data stored in them. Thus, the research into detecting these kinds of attacks is now more prevalent than ever and the development of Network Intrusion Detection Systems.

Aims The aim of this project is while working with a recently created dataset that approximates both normal and abnormal network traffic, to test novel and existing methods to detect and classify such attacks. Both the methods and the dataset will be analysed in the process and compared with similar research on this sector.

Method Already established classifiers will be used such as Support Vector Machine and k-Nearest Neighbours as well as encoding schemes such as Variational Autoencoders. The data will be pre-processed to improve performance of said algorithms and to improve the detection rate on the different attack categories.

Results It was discovered that many of the features in the original dataset can be abused by some classification algorithms as they can act as a label for an attack category. By accounting for this and using autoencoder losses as features, the classifier results have demonstrated to have better results than using just the original data for the more problematic attack categories.

Conclusions The solution indicates the importance of understanding and analysing the data before entering them into algorithms as well as demonstrating the potential of using the reconstruction loss of autoencoders as features. When compared to related work, the proposed classification structure can achieve improved results than a standard Support Vector Machine or Extreme Gradient Boosting classifier.

Keywords — Network Intrusion Detection System (NIDS), UNSW-NB15 dataset, feature extraction, variational autoencoder

I INTRODUCTION

With the exponential advancement in computer systems and their capabilities people and organisations store more and more sensitive information online. This is an incentive for nefarious individuals to attempt and gain access to such information. A report on data breaches of 2018 and 2019 found that for the first half of 2019, the number of successful attacks increased by 54% and the total number of records exposed increased by 284%. Most of these attacks targeted users' personal details, mainly emails and passwords and originated from sources outside the targeted organisation (Risk Based Security 2019). In addition, attacks in modern networks can expose more information that in previous years as the amount stored in databases has increased. Moreover, systems with more capabilities now use networks from which they can be accessed and controlled. For this reason, there is a need for detecting abnormalities or attacks in the dynamically changing networks to secure personal details and control of the computer systems. A method by which this can be achieved is a Network Intrusion Detection System that is mainly based upon Anomaly and Misuse Detection (Moustafa & Slay 2015*a*). Anomalies are patterns in data that do not conform to a defined notion of normal behaviour (Chandola et al. 2009). For the purposes of this paper, normal behaviour is defined as a connection of a user to the system without the intention of attacking the system in any way.

The main difference of a NIDS as opposed to other systems such as Host-Based Intrusion Detection Systems (HIDS) is that the former operates on live network traffic data and is able to detect and prevent attacks in real time. The latter, receives the log-files of a connection in the system and analyses the events that occurred. This technique allows for more analytical methods to be applied that take into account for example differences in geographical locations of connections and their importance based on domain information (Anandapriya & Lakshmanan 2015). However, these systems do not operate in real time and often analyse this information after a connection has been made. HIDS are also known to have a high false alarm rates (FAR).

The earliest effort to generate realistic data for testing such systems was made by DARPA (Defence Advanced Research Project Agency) in 1998 when they created the KDD98 dataset. Afterwards, they also improved the dataset by creating KDDCUP99 which is based on KDD98 and has been extensively used in research. These are datasets that approximate live network traffic and thus are used in developing NIDS. Since 1999 there has been a significant lack of testing and training data of normal and abnormal behaviour in internet traffic that can be used for intrusion detection systems. This has led to the reuse of the aforementioned datasets that although important, their accuracy and capability to consider real-life conditions have been broadly criticised (Creech & Hu 2014). Additionally, as these sets are now more than 20 years old, it has been demonstrated that they do not inclusively reflect modern network traffic and low footprint attacks (Moustafa & Slay 2015*b*).

In 2015, a new dataset was generated, named UNSW-NB15 which contains a hybrid of real modern normal behaviour and the contemporary synthesised attack activities of the network traffic. Thus, it simulates both normal and abnormal behaviour split among one normal and 9 different abnormal categories (Moustafa & Slay 2015*b*). These are: Backdoor, Analysis, Exploits, Worms, DoS (Denial of Service), Worms, Reconnaissance, Fuzzers, Shellcode and another class Normal that holds all the data that fall under normal user behaviour 1. The process of detecting data points that do not conform to the normal behaviour is referred to as anomaly detection (Deshpande 2019). The dataset consists of approximately 87.35% Normal records which reveals that the dataset is heavily imbalanced, a property that will be important when considering which algorithms to use for this task as well as the interpretation of the results. Furthermore, each of the 2.5 million data points collected contain 49 different features (dimensions) such as source Internet Protocol(IP)/port, destination IP/port, timestamps, protocols used and others. There are two target features which are a binary label that distinguishes normal from abnormal data and an attack category that classifies the data under the 10 distinct categories, some of which hold only a few hundred data points. The exact classes as well as their cardinality are presented in Table 1

High dimensional data are prone to issues often categorised as curse of dimensionality. This term refers to various phenomena that emerge when operating on high-dimensional data. Thus, it is common practice to reduce the number of dimensions while retaining as much of the original information as possible. This procedure is known as dimensionality reduction and can be achieved through a variety of methods (Zimek et al. 2012).

A Objectives

The research question proposed is *How can normal and abnormal network behaviour be correctly classified?*. To tackle this task, the existing dataset will be used to train a system which will attempt to classify network connections between normal and abnormal. This dataset will need to be analysed first to reduce the bias and errors of the classification. Thus, the classification system should be trained and tested on the dataset after any pre-processing.

As mentioned above, the different classes are heavily imbalanced which makes using some supervised learning methods not favourable for anomaly detection. This is because the system might assume normal behaviour for all evaluations and get high a accuracy number as a result of the imbalance (Nguyen et al. 2008). Unsupervised algorithms were considered and more detailed metrics than accuracy were used to counteract this problem. Another step that can be taken to avoid that is to balance the different classes. However, this process might add noise and bias to the data and thus to the results or destroy information valuable to a classifier.

Table 1: Categories of the UNSW-NB15 dataset and cardinality

Class	Description	No. of Records
Normal	Natural transaction data.	2,218,761
Generic	A technique to clash the block-cipher configuration by using hash functions.	215,481
Exploits	Attacker uses prior knowledge of system vulnerability within an OS or software and leverages that knowledge to expose the system.	44,525
Fuzzers	Feeds randomly constructed data to the system and looks for an indication that a failure in response to that input has occurred.	24,246
DoS	Denial of Service attack aimed at making a network resource unavailable to its intended users by temporarily or indefinitely disrupting services.	16,353
Reconnaissance	Contains all records that can simulate attacks that gather information to expose system vulnerabilities.	13,987
Analysis	It contains different attacks of port scan, spam and html files penetrations.	2,677
Backdoors	A technique in which a system security mechanism is bypassed stealthily to access a computer or its data.	2,329
Shellcode	A small piece of code used as the payload in the exploitation of software vulnerability. Common example is Trojan virus.	1,511
Worms	A worm virus is a malicious, self-replicating program that can spread throughout a network relying on security failures.	174

Additionally, some of the 49 features in the data set are categorical and need to be encoded in a numeric format. This is because most algorithms can only operate on numerical data but also because some of the categories can act as labels. Categorical features can also add unwanted and wrong information to the data while being processed by a neural network if they are not encoded correctly. Also, some techniques can operate faster or produce more desirable results with quantitative data and, categorical variables have too many levels and can significantly impact an algorithm's performance. Lastly, dealing with integers or floats instead of strings will improve performance of the subsequent algorithms by allowing scaling and normalisation processes.

As input to a NIDS will be a set of data records that imitate information gathered from network traffic. Each of these records consists of attributes (features) that can be either categorical or numerical and be of different data types such as binary, float, strings and others. Most commonly there is a binary label that indicates if the record is of normal (0) or abnormal (1) behaviour in such datasets. In many sets, there is also other more specific classifications for the different attack categories that might exist (DoS, Generic, Exploits etc.).

One of the most used and quoted dataset in this field has been the KDDCUP99 and its improved version: NSLKDD which were created in 1999. Both of these originate from the 1998 DARPA dataset that was one of the earliest attempts to create a dataset for an IDS. Even though they have been widely used even in recent research, these datasets have been heavily criticised for their representation of reality and the methods used to generate them. Moreover, the KDD set's main problem is the large number of redundancies included. There have been many attempts to alleviate the redundancy but it is not an easy task and often introduces bias to the data (Tobi & Duncan 2018). In addition, these datasets were created before 2000 and thus, do not include examples of modern low foot print attack environments and information that may be available today. Therefore, there is still a need for NIDS datasets as any classification algorithm is heavily dependant on the data it is trained on.

The dataset selected and used in this project was the UNSW-NB15 that was generated by Moustafa N. and Slay J. in 2015 (Moustafa & Slay 2015*b*). Along with the complete raw pcap files there are also 2 smaller sets generated from the original containing less redundant data. These are the training and testing partitions of the dataset. The data contains 2 label features that specify normal and abnormal behaviour as well as the 9 different attack categories present. In total there are 49 attributes for each connection, over the 42 existing in the KDDCUP99. However, even with the attempted improvements, this dataset still has its own problems that need to be tackled. For this project, only the newer dataset will be analysed and used.

II RELATED WORK

A *UNSW-NB15 Dataset*

In most of the papers testing novel and older methods for network intrusion with UNSW-NB15, the training and testing data used is only a partition of the original dataset generated. The normal class is significantly down-sampled and the split is not said to how this was performed although redundancy of data points has been removed.

Furthermore, many papers state that the only categorical features that exist in the data are that of the normal and abnormal label as well as the attack category. However, some features exist in the data that might perform as a label because of their distribution among normal and abnormal features, thus causing results of algorithms such as Support Vector Machines (SVM) biased. These categorical features are states, protocols, services, IPs, ports and timestamps. Specifically, out of the 135 different protocols that a data point can have, only 3(2.2%) of them contain both normal and abnormal data. From the remaining protocols, 126 of them contain only abnormal data points and 115 of them hold exactly 137 data points. This noise adds bias to the data as a new normal record that uses one of these 126 protocols might be classified as an attack because of this and vice versa. Thus, these implications must be minimised to get accurate classification when feeding new data to the classifier. Other papers either remove these problematic features or don't mention how they are encoded.

As training a NIDS and other machine learning algorithms is heavily data dependant, many papers are focused on assessing or improving the existing cyber security data sets. One paper used the NSL-KDD dataset and applied Synthetic Minority Oversampling Technique (SMOTE) to synthetically generate more minority class data points. The classifiers trained on this dataset exhibit significant performance improvements over the NSL-KDD on the same minority classes. (Divekar et al. 2018). This is also applied to the UNSW-NB15 dataset generating NB15-SMOTE, however only the much smaller and processed training and testing partitions of the original set were used. Random undersampling was also used to reduce the cardinality of the majority classes. The best results for this set were provided by Random Forest classifier with a weighted F1-Score of 78% over the 77% for the original version of the dataset. Even though this is a small increase, the F1-Score for Analysis in the original data was 0 as opposed to 13 in the oversampled set and the weighted average does not depict this change. K-Means was the second worst classifier when run on the UNSW-NB15 and NB15-SMOTE sets. SVM achieved the best results in the KDD dataset and its variations while Neural Networks and Random Forest followed closely behind. Finally, they conclude that SMOTE and its variations can be used on the UNSW-NB15 set to improve classification of the minority classes while under-sampling the majority class leads to records being ignored and recommend not using random under-sampling for this application.

In many papers, in attempts to make training more efficient and improve a classifier's performance, the number of total features used is reduced. One way to achieve that is by feature selection where either manually or automatically, a subset of the features is selected and used, which provides the greatest classification accuracy. The goal of this subset is to either achieve much better performance without a significant drop in accuracy or even accomplish a better performance than the full set of features. A paper analysed the features of the UNSW-NB15 dataset to find the most significant ones (Janarthanan & Zargari 2017). They also used the partitions of the data instead of the full dataset while they also compared their finding with the KDD99 dataset. It was found that through machine learning techniques the most significant features was $service$, $sbytes$, $sttl$, $smean$ and $ct\_dst\_sport\_ltm$. They were able to achieve better classification results with the Random Forest classification method, than the subset of features initially proposed by the creators of the dataset. These features were $sttl, ct\_dst_sport\_ltm, spkts, dload, sloss, dloss, ct\_s$ and ct_srv_dst and were adopted by an Association Rule Mining (ARM) based algorithm, which the creators of the dataset designed (Moustafa & Slay 2015a). However, as seen in Table 2, some of these features reference the source and destination IPs of the last 100 rows of data. These IPs only exist in the full dataset and not in the testing and training dataset. This could be a reason why the latter feature selection process did not find these features as important.

B Classification

One paper applied multi-class SVC to the training partition of the data and got high accuracy results on the original data (Jing & Chen 2019). Specifically, they had accuracy of 99.9% for the Worms attack category that only holds 174 samples in both the training and testing dataset. However, the detection rate (recall) is at only 9% instead. By using the original data without taking into account the features, these results might be skewed. Furthermore, in this particular paper, a non-linear scaling method was used in the data pre-processing step as opposed to linear normalisation that is most commonly found in machine learning applications. The reasoning for this decision is the differences found in the various features that the UNSW-NB15 data set has. However, in this project, these features were the subject of other feature extraction and encoding methods thus the effect with the standard scaling was applied. Moreover, weighted average is used for their metrics however this means that large classes that perform better in classification due to the many example data points will skew the results.

Another attempt to develop a NIDS system based on the UNSW-NB15 dataset was to build such a model using Extreme Gradient Boosting (XGBoost) (Husain et al. 2019). Initially, all nominal type features such as the source and destination IP were removed along with the time features resulting in 39 features. In this project, these features were encoded instead to maximise the information that can be gained from the data. Then they split the dataset for training and testing, with a further split of the training partition into training and validation subsets. They also highlighted the features that have a high importance based on the information gain analysis drawn from the XGBoost. Mainly, the $sbytes$, $smean$ and $sload$ features were found to have high importance among most of the abnormal classes. Using this data, their final model uses only 23 features instead of the 39 and the original 49 with only a minor drop in the results. While the accuracy metric of the classifier is reasonably high for the testing data, 75%, there are no data points from the Analysis attack category. Moreover, the precision is less than 0.1 for the classes of Analysis, Backdoor and Fuzzers and less than 0.5 for the classes of Worms and Shellcode.

Table 2: List of most significant features found

Name	Description
service	Application layer protocol such as http, ftp and smtp.
sbytes	Source to destination bytes.
sttl	Source to destination time to live.Time to live is a mechanism that limits the lifespan or lifetime of data in a computer or network.
smean	Mean of the packet size transmitted by the srcip(source IP).
ct_dst_sport_ltm	No of rows of the same dstip(destination IP) and the sport(source port) in 100 rows.
spkts	Source to destination packet count.
dload	Destination bits per second.
dloss	Destination packets retransmitted or dropped.
ct_src_ltm	No. of rows of the srcip in 100 rows
ct_srv_dst	No. of rows of the same service and dstip in 100 rows.

Another attempt at creating an effective NIDS was using a feed-forward neural network. This was based on full dataset instead of the prepared training/testing sets and they found that there is a difference in features in the prepared and full dataset. Using 10 layers and 10 neurons in each totalling 100 neurons, they were able to ac hive very high accuracy of 99.5% (Zhiqiang et al. 2019). They also achieved a low False Acceptance Rate of 0.45 which is a metric that specifies how many abnormal records were classified as normal. However, it is not mentioned if this is a binary or multi-class classification. This is problematic as we cannot accurately compare results. Moreover, their results were not compared with popular methods such as SVM, KMeans or Random Forest.

III SOLUTION

A Data Pre-Processing

Apart from the specific methods outlined below, the given data were also scaled with the use of a simple min-max scale which transforms features by scaling each to a given range. Specifically, the range used was 0 and 1. This is helpful for some algorithms to produce better results faster but for some algorithms such as Principal Component Analysis (PCA) and the VAE it is a necessary step. This was done to all numeric data and to all categorical data after they were converted to a numeric format.

A.1 Categorical Data Conversion

A naive method of encoding categorical data is to use a Label Encoder that encodes each class of a category as a number. This method however adds relationships to the data that did not previously exist. For example, if 3 colours need to be encoded: red, green and blue, the label encoder might assign them as 1,2 and 3 respectively. This means that red is closer to green than it is to blue and that blue is three times red. To avoid adding this bias to our data, other encoding methods must be used. A popular method to encode categorical data without adding bias is to use One-Hot Encoding instead of just a label encoder (Gori 2017). OneHot creates a new column/feature for each category that holds a binary label that reveals if that data point belongs to that category or not, as outlined in Table 3. Even though this method adds the least amount of bias, the cardinality of the categorical feature directly impacts the number of new features created. Thus, features with high cardinality, such as more than 15, will need another encoding method as to not impact the performance of the other algorithms. Such a method is Target Encoding (Rodríguez et al. 2018) and its extension, LeaveOneOut Encoding. With this type of encoding, the cardinality of the features stays the same as for each category of a categorical feature with the mean of the target variable. LeaveOneOut is the variation that does not take into account the current data point when calculating the mean for each data point with the same category. From the 4 example, for the first row, the FTP targets are (-,1,1,0) the mean of which is 0.67. Similarly, for the second row the targets are (0,-,1,0) thus the mean is 0.33. Then, some random noise is added for the final result.

Table 3: OneHot Encoding Example

index	Protocol	HTTP	SMTP	FTP
1	HTTP	1	0	0
2	SMTP	0	1	0
3	FTP	0	0	1

The categorical features encoded with One-Hot were the services and the states. LeaveOneOut was used for the IPs, the protocols and the timestamps. The source and destination IPs do not exist as a feature in the training and testing

Table 4: LeaveOneOut Encoding Example

Protocol	Target	Mean Target	Random	LeaveOneOut Protocol
FTP	0	0.67	1.05	0.70
FTP	1	0.33	1.97	0.32
FTP	1	0.33	1.98	0.33
FTP	0	0.67	1.02	0.68
HTTP	0	0	0	0

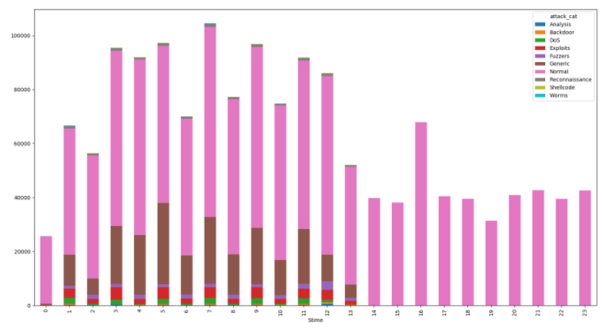

Figure 1: Hour of day attack count visualisation

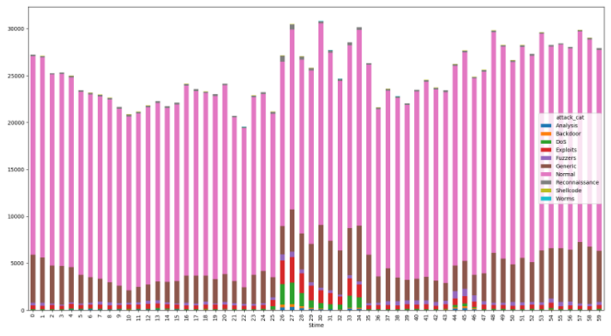

Figure 2: Minute of hour attack count visualisation

partitions that were extracted from the originally created set.

A method to encode IPs is to use character encoding as a step to use OneHot encoding for IPs. With this method, the IPs were padded with zeros to reach 12 digits if not already. Then, each digit of each location was treated as a category and thus was encoded with OneHot. So, instead of having each different IP be its own category and thus having a high cardinality, the cardinality of the new features is capped at 120 as there are 12 positions, each with 10 possible digits (Pasumarthy 2018). However, this still proved to be a lot of new dimensions added to the data and another method was used. Generally, each IP (IPv4) has 4 distinct parts and 2 sections seperated by dots. The 2 sections are for the network and the host/computer. The attack distribution of the first, second and third different parts were identical however the fourth differed. Therefore, the IPs were split into these parts but only the first and fourth part were encoded while the other 2 were discarded. This was done so redundant information does not affect the algorithm. The encoding of the 2 parts was done with LeaveOneOut as well as the cardinality of the different IP parts was quite high.

The timestamps were first converted to display the hour of a day of a connection and the minute per hour and then each was encoded with LeaveOneOut as well. This was because the distribution of the attack categories was different for the hour of day and minute of hour so more information could be gathered if both these representations were used. As it is encoded with LeaveOneOut the number of features does not change. Moreover, the start time of the connection, the end time and the duration of a connection exist in the data. Thus the end time of the connection was removed.

B Balancing

As mentioned before, the different attack categories are heavily imbalanced. This might cause supervised algorithms to assume normal behaviour as that is the majority class and still have a high accuracy number. Thus, balancing the dataset is key to achieving better detection rates even for low count classes. Mainly, there are 2 methods used to balance a dataset. One is by either down-sampling the majority class and the other is to over-sample the minority classes. Oversampling might multiply noisy features in the data and downsampling might lead to loss of information by removing data points. Most commonly oversampling is used but the noisy features might lead to the classifiers overfitting which means learning the noise of the features. This is hard to detect because while the classifier will perform well in the training and validation set, only the results of the testing set will be affected. The opposite of this is underfitting, where a model is not able to learn the training data or generalise to new data. This is easier to detect as the performance on the training data will be poor. Oversampling can be achieved by randomly duplicating existing data points or by synthetically generating data points from the existing data.

The most popular technique is the Synthetic Minority Over-sampling Technique (SMOTE). Essentially, a minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbours. Depending upon the amount of over-sampling required, neighbours from the k nearest neighbours are randomly chosen (Chawla et al. 2002). This method has achieved improved results in most applications and has been proven useful in cyber security datasets (Divekar et al. 2018). An extension of SMOTE is the Adaptive Synthetic (ADASYN) sampling approach. It uses a weighted distribution for the different minority class examples according to their level of difficulty in learning. Therefore, the harder that a sample is to learn, the greater impact it will have on the new synthetically generated example. ADASYN was able to outperform SMOTE and Decision Trees when run in multiple datasets (Haibo He et al. 2008).

When oversampling, especially on a large scale, the generated data points might fail to relate to the original data points and thus not help when training a classifier to detect these points. This means that while the classifier might perform extremely well on the oversampled data, the original data might still be miss-classified. Thus, this technique should be applied in moderation.

C Dimensionality Reduction

After the prepossessing of the original data, the count of the different dimensions was 67, including the 2 target features. This, although not as high a number as in image or video applications, can still have a significant effect on the performance of the techniques used subsequently for classification. Thus, it is good practice to attempt to reduce the cardinality of the dimensions. Mainly, this process can be achieve linearly and non-linearly. An example of a linear method is Principal Component Analysis (PCA) while t-Distributed Stochastic Neighbour Embedding (t-SNE) is an example of a non-linear method. As the type of relation between the features is not known, both linear and non-linear methods had to be tested. These techniques can help visualise the dataset as they can reduce the dimensions enough to be plotted in a 2D graph.

PCA is one of the most popular techniques for dimensionality reduction. It is a linear mapping of data to a lower dimension space where variance is maximised. The features in this new space are independent and are referred to as components. This method is deterministic, computationally efficient and provides substantial results with Gaussian distributed data, however, it cannot interpret complex polynomial relationships between features. There can also be some information loss when selecting the number of principal components to be used (Seghouane et al. 2019).

On the other hand, t-SNE is a widely used non-linear method that uses a normal distribution in the higher dimension and a t-distribution in the lower one to measure similarities of the data points and the corresponding embedding of them. It aims to keep the relative similarity of the data points as close to the original space as possible. Thus, it performs well with non-linear relationships among features and can also be tuned to achieve better performance in different data sets. However, these hyper-parameters can have a significant negative impact on the results of this methods as it can be prone to overfitting if the parameters are not set up appropriately. Even then, running the model twice will provide different results as the process is non-deterministic.

Lastly, there is also a danger of losing large scale information during the embedding which is the global structure of the data (Chan et al. 2018).

Even with reduced dimensions, many attack categories overlap as illustrated by applying these techniques to visualise the original high dimensional data. This is an indication of the complexity of the problem.

Another way to perform dimensionality reduction is with the assistance of autoencoders (AE). An autoencoder is a type of unsupervised neural network where the input is the same as the ouput. They are split into two distinct parts that are the encoder and decoder. The encoder takes as input the original data and compresses them into a lower-dimensional code. After this transformation, this latent space is used as input for the decoder as it attempts to reconstruct the original data from this latent representation. During its iterative training process both the encoder and decoder improve to generate realistic results. An important part of this training process is the use of a loss function that judges how well the system was able to approximate the example input.

Mainly, the fundamental problem with standard autoencoders is that the latent space is not continuous and does not allow easy interpolation. This is why more complex models are being used, one of them being Variational AutoEncoder (VAE). In VAEs 3, a probabilistic distribution is built to estimate the distribution of the feature data and these probabilistic parameters can be calculated by the network (Doersch 2016). These models give the ability to get more accurate output and generate data that closely approximate the original data. This feature can also be used as a method to oversample the data by using the generated output of the autoencoder.

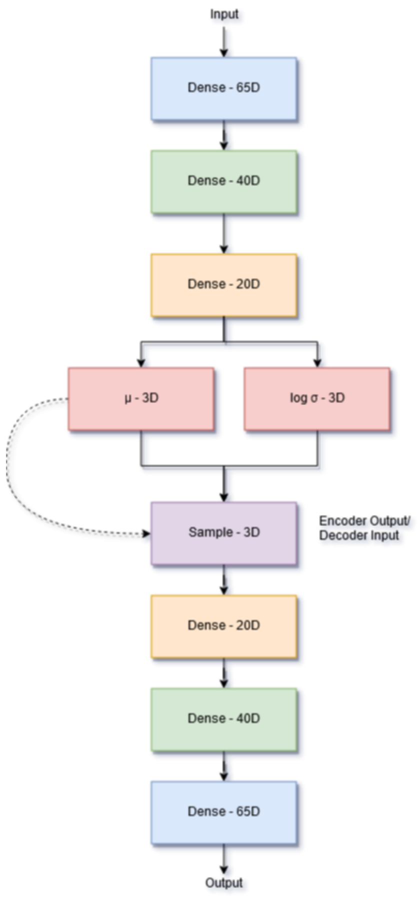

Figure 3: Variational Autoencoder Structure

D Classification

After the data has been processed different classification methods were used to determine which one was the best as well as other machine learning algorithms to improve the accuracy of the final results. The final model was the one that produced better results in detection rate and f1-scores which are going to be explained in the Results section. This model is built on the premise that some attack families are easier to detect and distinguish than others that possibly overlap and have many similarities. Moreover, because of the imbalance of the data, the categories with high cardinality are going to have more data to train a classifier which should provide higher accuracy results. Thus, the final model has 4 distinct classifiers for all 10 categories.

D.1 Binary Classification

The main distinction in classification techniques originates from the use of the labels. If the labels are used in the training process then the classification technique is supervised. If the labels do not exist or are not used then the method is unsupervised.

The first three classifiers are Support Vector Classifiers (c-SVC or c-SVM). This technique also referred to Support Vector Machines (SVM) is a supervised learning model with associated learning algorithms. It was originally developed for binary classification problems and can also be used for regression tasks (Cortes & Vapnik 1995). The non-linear version of the algorithm, maps input vectors to very high-dimension feature space and is used for data that is not linearly spreadable. As is illustrated in the results section from PCA and t-SNE visualisations, our data most likely have non-linear relationships. Then, in this high-dimensional space hyperplanes are generated that separate the data. These hyperplanes are boundaries that are in the form of 1 dimension less than the feature space. If the feature space is 2 then the hyperplane will be a line and so on. There are many possible hyperplanes that could be chosen, the objective of the algorithm is to find the one with the maximum margin between that and data points of different classes. This is the objective as future unseen data will have a higher chance of being correctly classified if these margins are wide. Support vectors are the records that are closest to these margins and thus have the most influence on the algorithm. c-SVM is a version where the regularisation parameter c is controlled. This parameter scales from 0 to infinity and controls the margins to balance the training and testing errors. Its purpose is to improve the classifier's performance on the unseen data.

To compare SVM with other classification methods, Multi-Layer Perceptron(MLP) and k-Means (KMeans) were used. MLP, unlike SVM relies on a feed forward artificial neural network to perform the classification. It is a non-linear supervised technique and has similar structure to an autoencoder where there is an input layer, a hidden layer and an output layer. It uses the gradient of the loss function and compares the input with the output and the difference between them. As a neural network it requires more data to be trained than an SVM which might be problematic for classes like Worms that only have 174 records. It managed to achieve similar results to the SVM as seen in the Results section and thus was substituted by it because of the cardinality of some attack categories. KMeans is a clustering method that partitions the data into clusters that will be the classes. It is an unsupervised method that is somewhat related to the kNN algorithm.

D.2 Multi-class Classification

There are two approaches to creating a binary classifier into a multi-class one. These are by either comparing each class against all others, referred to one-against-all (1AA) or by comparing one-against-one(1A1). For an N-class classification problem, the first one involves the division of the dataset into N two-class cases. The latter, which is the version used in this project, constructs a machine for each pair of classes resulting in $\frac{N(N-1)}{2}$ machines. This is a more intensive process however, by splitting the classifiers and grouping the classes into one, it creates less machines and is more efficient (Anthony et al. 2007).

The first classifier, is a multi-class Support Vector Classifier (SVC). The purpose of this SVC was to decide if records were part of the Normal or the Generic class, or neither. However, SVC and its variations, can only classify a point based on the classes in that model so it needed to be trained on all of the other classes as well.Thus, the data were split in Normal, Generic and Other, where other holds records from the other 8 classes. For the subsequent classifiers, the Normal and Generic categories were removed from the training and testing data. This is because as the classification of these 2 categories would already have taken place. The following classifiers would only be used if this one classified a record as ”Other”.

After the first classification there exist 2 binary classifiers for the more problematic classes. These classes are Exploits and Reconnaissance. Even in the initial tests, increasing the cardinality of the training records of one would decrease the accuracy of the other class. A reason why this might be happening is the overlap that these classes might have. So after the second classifier is a binary SVM to decide if a point belongs in the Exploits class or some of the other. The normal and generic classes are not regarded at this point as it is assumed that the classification would have been made from the first classifier. After the Exploits SVM, there is another binary SVM to decide if a give point is part of the Reconnaissance class or not. These two are separate as the classifiers have exhibited improved results with the two classes split in this format rather than existing in the same classifier or being in the same classifier with the other attack categories.

The final classifier works based on the VAE. Specifically, each of the remaining 6 attack categories have their own VAE that is trained on data from their own attack category. In order to achieve that, some classes needed to be oversampled as the cardinality of the samples available was not enough to train an autoencoder. However, the VAE used has a slightly different structure than the one outlined in the original paper. By using either the mean or standard deviation dense layers separately as the latent space, the VAE was able to achieve overall lower reconstruction losses than using the sample of both. In Figure 3, this change is depicted by the dotted line that moves from the μ to the sample that is the input of the decoder. Moreover, after a certain number of epochs, the reconstruction loss would not improve if the sample was used however this problem did not exist in the other 2 cases, when either μ or σ layers were used by themselves.

These classes were Worms, Fuzzers, Shellcode, DoS, Backdoor and Analysis. This oversampling happened with ADASYN. After the 6 different VAEs had been trained, the rest of the data was passed through and the reconstruction loss of each of the VAE on the individual point was stored. So, each point now had 6 features which were the losses of the 6 different VAEs. These losses were then used to train a k-Nearest Neighbour (k-NN) classifier with the k being 6, as the number of distinct attack classes that are considered in this point of the classification process. kNN is another supervised

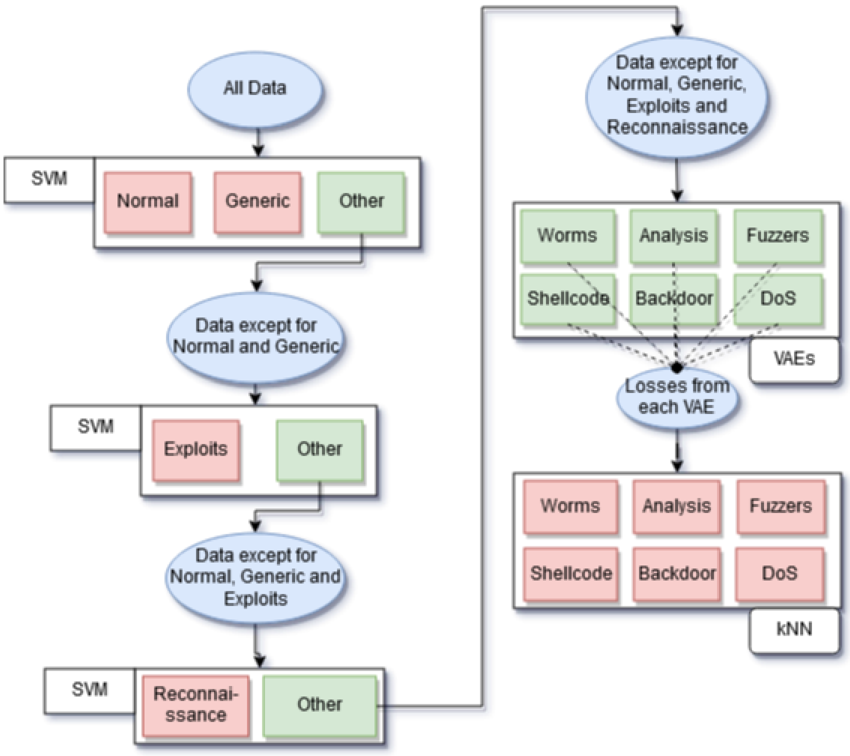

Figure 4: Structure of the multiple classifier model. Red boxes signify an endpoint to the prediction.

one-against-one classifier.
It groups examples together based on a distance metric based on the assumption that similar records will be closer together. It then generates k clusters and for each new example, its distance is also compared and assigned the most likely candidate label.

Applying the k-NN on the losses instead of the original classes produced superior results. In theory, the lowest reconstruction loss should belong to the VAE of the same class as that data point however, after some testing this proved to not be true. This is another indication of the complexity of the dataset and its sub classes.

E Testing

To evaluate the results of the above classifications a range of different methods was used. As mentioned before, even if a classifier achieves high accuracy it might still not be as good as perceived so precision and recall need to be taken into account. For each testing, the binary or multi-class confusion matrix was generated and more advanced metrics were used such as macro and weighted accuracy average and f1-score that takes into account both precision and recall. Additionally, graphs such as the IP and time attack distribution as well as the graphic representation of the dataset were qualitatively assessed in order to make decision on which techniques might yield the greatest results.

IV RESULTS

The training and testing of this solution as well as the collection of results took place on the Google Colab platform using a Tesla K80 GPU accelerated runtime.

A Metrics

To evaluate the classifiers, multiple metrics were be used. These are Accuracy, Precision, Recall and F1-Score. It is important for multiple to be used as individual metrics can sometimes be deceiving especially in cases of imbalanced datasets. These were calculated per class through the confusion matrix. The Macro Average of these was also calculated which is an average that does not account for the number of samples in each class. This is because using a weighted version, the majority classes would skew the results. Moreover, the support column in the per class classifiers indicates the number of test samples in that particular class.

- Accuracy (AC) indicates the total percentage of correct classifications according to the true values, both True Positive (TP) and True Negative (TN).

$$Accuracy(AC) = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

- Precision (P) refers to the ability of a classifier to identify only relevant data points. It is also knows as the Positive Predictive Value.

$$Precision(P) = \frac{TP}{TP + FP} \tag{2}$$

- Recall (R) signifies the ability of a classifier to find all the relevant cases within the dataset. This metric is widely used in anomaly detection and is often referred to as Detection Rate (DR) or Sensitivity (S) however, the equation is identical.

$$Recall(R) = \frac{TP}{TP + FN} \tag{3}$$

- F1-Score (F1) conveys the balance between Precision and Recall as it is the harmonic mean of these metrics and thus is more sensitive to low values. It is often referred to as the F-Score or the F-Measure. It is a simple metric to focus on as maximizing both Precision and Recall is not possible.

$$F1 - Score(F1) = \frac{2 * P * R}{P + R} \tag{4}$$

B Dimensionality Reduction

For this step PCA, t-SNE and VAE were used for visualising the data by reducing the number of features to 2 in order to be projected on a 2D graph. PCA and t-SNE were just used for visualisation while VAE was also used in different parts of the testing and the final solution. From the attached figures, by looking at the distribution of the latent space and the clusters fromes, there was some indication that the latent space can be used to get better accuracy results and detection rate for the data. This is because there is more seperation among the classes in the encoded data rather than the original one.

C Binary Classification

During the early stages of this project, different classification methods were tested on the binary classification that separated normal from all abnormal data. The algorithms tested were SVM, and unsupervised classification methods Multi-Layer Perceptron (MLP) and k-Means Clustering(KMeans). The latter is the unsupervised version of the k-NN algorithm. These were tested both on the original data and on data encoded by the VAE. The training and testing split on the data for them was 70:30.

Table 5: Binary Classification results on original data

Classifier	Accuracy	Precision	Recall	F1-Score
SVM	99%	0.98	1.00	0.99
MLP	99%	0.99	1.00	0.99
KMeans	68%	0.65	0.80	0.72

Table 6: Binary classification results on VAE latent space

Classifier	Accuracy	Precision	Recall	F1-Score
SVM	90%	0.98	0.87	0.92
MLP	62%	0.85	0.62	0.71
KMeans	84%	0.90	0.85	0.87

So, all classifiers perform better on the UNSW-E67 data instead of the latent space, even though SVM achieved high results even in the encoded data. While KMeans is much worse,

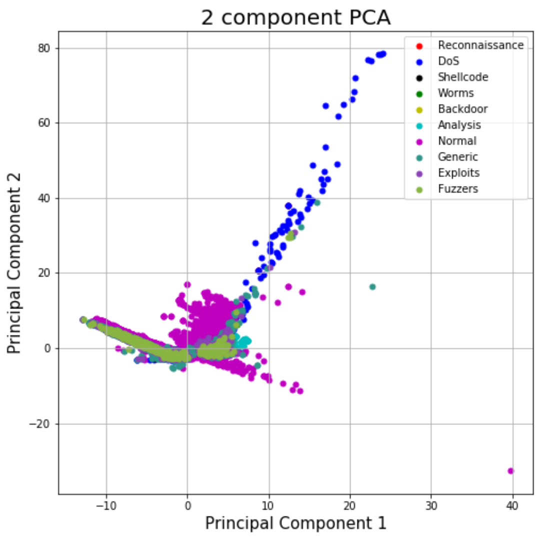

Figure 5: PCA on original data

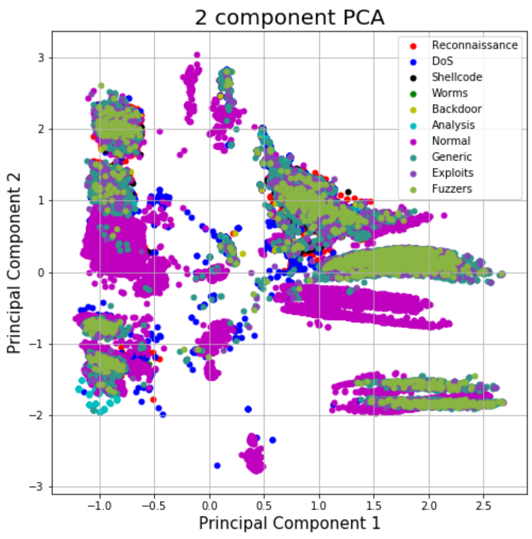

Figure 6: PCA on latent space

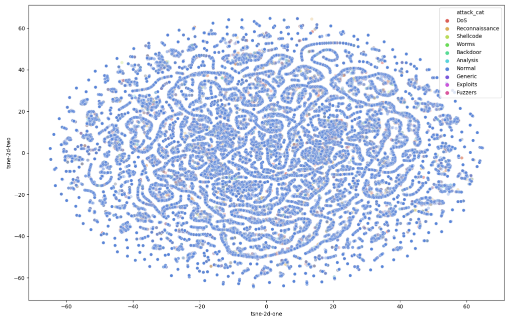

Figure 7: t-SNE on original data

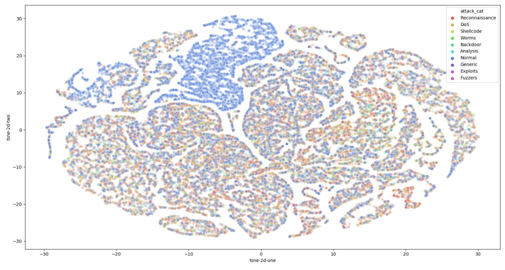

Figure 8: t-SNE on latent space

MLP and SVM are similar in their performance. However, this is just on the binary classification. When training the per-class classifier, MLP failed to achieve as good results as SVM mainly because as a machine learning technique, it requires more data to be trained. Moreover, isolation forests were tested for binary classification. Isolation forests or iForest build a normal profile and are able to detect outliers. In this case, these outliers would be the data that are part of the abnormal class. Unfortunately, the algorithm was not able to detect any abnormal types when trained on the normal behaviour so it was not used in the final model.

D Multi-Class Classification

Following are the results of the final classification structure that assigns a record to one of the 10 classes. The python sklearn library was used for the SVM and k-NN classifiers while Keras, which used Tensorflow 1.x, was used to implement the VAE. The SVM used was based on a non-linear version of the algorithm implemented on LibSVM which is a library for Support Vector Machines. The fit time scales quadratically with the number of samples and thus, was

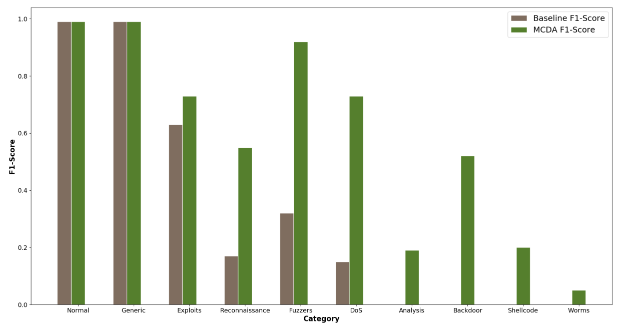

Figure 9: Barplot of per-category F1-Scores between baseline SVM and MCDA method

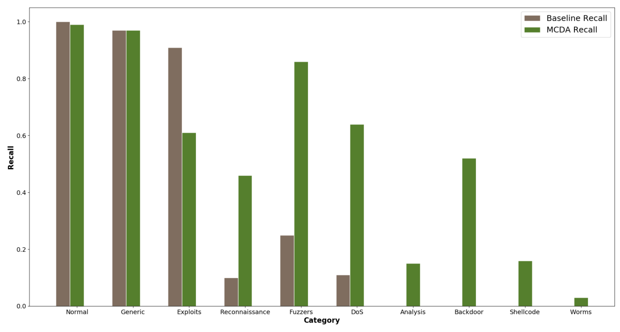

Figure 10: Barplot of per-category Recall between baseline SVM and MCDA method

impractical for the original data. However, splitting the dataset and iteratively, removing classes previously classified by another SVM made the task more manageable. A linear version of SVM was also tested as it is more efficient but it failed to produce the desired results. All the other settings remained the same as the default.

For the k-NN classifier, the sklearn version was aga. The only setting changed was the number of neighbours that was prespecified. This was set to the number of classes that needed to be fit which was 6. Setting this number to something different from the specified number of classes resulted in lower scores for all the metrics.

For the VAE model, both the encoder and decoder activation was set to ReLU as it provided the lowest losses among linear and sigmoid. Moreover, Adam was chosen as the model's optimiser with a 0.001 learning rate. This was the result of seeing the loss of optimisers after 20 epochs for all classes combined. Adam was the best at 0.11 while RMSprop, Adadelta, Adamax and Adagrad were close at 0.12. SGD performed the worst at 0.6. The batch size was 64 and the final model included 4 layers on both the encoder and decoder. The final results and losses were generated after training each VAE for 10 epochs. In **??** are the results of the first SVM classifier that splits the data in either Normal, Generic or one of the other categories. All the data points that are not Normal or Generic are renamed to Other. This will happen in the following classifiers as well with the respective attack categories. These results are similar to the binary SVM, however as expected adding the Generic class, the abnormal classification score is lower.

After the Normal-Generic-Other classification, the next classifier can discard the normal and generic data points and be trained on just the others. This will subsequently happen with all the classifiers. The next two classifiers are the binary SVMs for Exploits and Reconnaissance. In Table **??**, are the results of the Exploits classifier. While the Precision and Accuracy scores are high, the recall score for exploits is not. This is also the case for the binary Reconnaissance classifier, results of which are found in Table **??**. The recall metric is high for the Other classes in both of these classifiers.

After these, the data are passed to the VAE. To train and validate each of the 6 VAE just data of the specified class were used. Then all the data were passed to these VAEs and the reconstruction loss for each record from each VAE was calculated.

Lastly, these losses were used to train and test the kNN classifier on these 6 classes. In Table **??** the results of these losses can be seen. These are all in the oversampled data because the original data were not enough to train the VAEs. However, after training the VAEs with the oversampled data we tested just the original data on the classifiers. These results are found in

Table 7 where they are comprared with the oversampled and the original data from the SVM paper (Jing & Chen 2019).

V EVALUATION

In this section, based on the results obtained by our proposed solution as well as results from related work, we will evaluate strengths and weaknesses of this NIDS reflecting upong the original research question: *How can normal and abnormal network behaviour be correctly classified?.*

A Stengths and Limitations

As illustrated by the Results section, the proposed solution has achieved better detection rate and accuracy as compared to a standard SVM classifier. Especially for the case of the binary classification and deciding if the records are either Normal or Generic. The other SVM classifiers also performed well on Exploits and Reconnaissance however these were lower than the ones achieved in the nonlinear scaling SVM (Jing & Chen 2019). However, in that paper, while the accuracy of the classes was high, the detection rate of the Analysis class was 0% and less than 5% for the Backdoor and DoS attack categories. Additionally, the Worms category had a detection rate of 9.1%. While the macro average accuracy of 95.0% is higher of the achieved accuracy of our own 83.3%, the macro average detection rate was 84.8% over the 45.0% that the quoted paper found.

Furthermore, the paper used the partitions of the dataset instead of the original full available set. This could be a reason why both the accuracy and the DR of the Normal data is lower than what we achieved with the categorically encoded full data. This is because, the majority class was undersampled to create the training and testing partitions which can lead to loss of information. This loss could affect the ability of a classifier to correctly assign a record to a specific class.

A major **limitation** when comparing just the original encoded data and the ADASYN oversampled data, there was a clear performance drop. In Table 7 the detection rates of the different methods were compared. While there was a significant increase in some attack categories such as Analysis, DoS and Fuzzers, there was a performance loss in others. This points again to the overlapping of these categories and the difficulty to identify one class from another. A reason why the encoded data could not perform as well as the oversampled could be that the synthetically oversampled data failed to approximate reality. As a random oversampled had even lower results this was an issue that originates from the original cardinality of the classes such as Worms and Shellcode. Without much data to draw from, most of the oversampled data were synthetically generated from other synthetic data. This is an example of a model overfitting by learning the features of the synthetic data too well and not being able to accurately classify the original data. Moreover, as the non-linear SVM running time is quadratic and is inefficient for large datasets such as this. While it provided good results, the training time for the classifier was very long especially for the majority, Normal class. The linear version of the classifier while much more efficient did not provide good results. By splitting the categories to different classifiers, the process was sped up. A reason for this is the one-against-one structure of the multi-class SVM classifier. For a 10-class SVM classification, the classifier would need to build 45 binary machines. In our model, combined the 3-class classifier, the two binary classifier and the 6-class classifier only required 20 machines. Additionally, the data used for these machines were re-

Table 7: Comparison of methods detection rate on original, encoded and oversampled data

Class	Original SVM	MCDA on UNSW-E67	MCDA on ADASYN UNSW-E67
Normal	0.76	**0.99**	0.99
Analysis	0.00	**0.15**	0.73
Backdoor	0.30	**0.52**	0.60
DoS	0.50	**0.64**	0.54
Exploits	**0.93**	0.55	0.55
Fuzzers	0.47	**0.86**	0.65
Generic	0.96	**0.97**	0.97
Reconnaissance	**0.78**	0.50	0.50
Shellcode	**0.53**	0.16	0.93
Worms	**0.90**	0.03	0.97

duced as each subsequent classifier was only trained on the classes that were left to classify, not the ones that were involved in the previous classifiers.

B Future Work

As the VAE losses for the final 6 categories were based on the oversampled data, the poor performance of the k-NN classifier on the original data could be because these reconstruction losses are more descriptive of the oversampled data. On the other hand, unless the data is oversampled, the VAE cannot be trained properly as there are not enough records in each class for the training to take place. As mentioned before, a Network Intrusion Detection System (NIDS) is heavily dependant on the dataset used to train it. Having a larger number of records for the attack categories would allow the training of the classifiers and of VAEs without the need to synthetically generate new data points. Thus, the generation of an adequate number of attack records needs to be a priority when creating new cyber security datasets in the future. While UNSW-NB15 has been a step in the right direction, it still suffers from problems that most cyber security datasets have such as low cardinality of attack categories.

VI CONCLUSION

In this project, we have demonstrated that even though the generated UNSW-NB15 tackles a lot of the problems of the past, heavily used KDD datasets, it has its own problems. Mainly, the substantial class imbalance as well as the arbitrary partition of the dataset into a training and testing subsets with less features. Moreover, a lot of the features, even if they are in numeric format they are still categorical and must be encoded, a step that other researchers often leave out. We used OneHot and LeaveOneOut encoding techniques according to the cardinality of these categories to add the least amount of bias to the data.

Compared to the standard SVM (Jing & Chen 2019) on the original data, our categorically encoded data were able to achieve higher scores in both accuracy and detection rate for binary classification. For the multi-class classification, while the oversampled dataset achieved high results in classification, the original data did not match that performance but did exhibit improvement in some classes compared to the original.

For these tests the entirety of the dataset was used instead of the training and testing partitions. This was because these partitions had less features and their data was undersampled leading to poor classification performance in the Normal class. Our final model included 4 distinct classifiers, three of which were c-SVM while the last one was a kNN classifier. Before the last classifier, the data was converted to VAE losses after each record was reconstructed using 6 different autoencoders. Then, the reconstruction losses from the different VAEs were used to train and subsequently test the kNN classifier. We demonstrate that the losses of an autoencoder could be a valuable feature to be used in order to train a classifier. This could be especially helpful when the original feature space is too high to use classifiers such as SVM. Future work can experiment with different neural network architectures to improve the performance and the accuracy of the final results.

References

Anandapriya, M. & Lakshmanan, B. (2015), Anomaly based host intrusion detection system using semantic based system call patterns, *in* '(ISCO)', pp. 1–4.

Anthony, G., Gregg, H. & Tshilidzi, M. (2007), 'Image classification using svms: one-against-one vs one-against-all', *arXiv preprint arXiv:0711.2914* .

Chan, D. M., Rao, R., Huang, F. & Canny, J. F. (2018), T-sne-cuda: Gpu-accelerated t-sne and its applications to modern data, *in* '(SBAC-PAD)', pp. 330–338.

Chandola, V., Banerjee, A. & Kumar, V. (2009), 'Anomaly detection: A survey', *(CSUR)* **41**(3), 1–58.

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002), 'Smote: synthetic minority over-sampling technique', *Journal of artificial intelligence research* **16**, 321–357.

Cortes, C. & Vapnik, V. (1995), 'Support-vector networks', *Machine Learning* **20**(3), 273–297.

Creech, G. & Hu, J. (2014), 'A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns', *IEEE Transactions on Computers* **63**(4), 807–819.

Deshpande, B. K. V. (2019), 'Chapter 13-anomaly detection'.

Divekar, A., Parekh, M., Savla, V., Mishra, R. & Shirole, M. (2018), Benchmarking datasets for anomaly-based network intrusion detection: Kdd cup 99 alternatives, *in* '(ICCCS)', pp. 1–8.

Doersch, C. (2016), 'Tutorial on variational autoencoders', *arXiv preprint arXiv:1606.05908* .

Gori, M. (2017), Chapter 5 - deep architectures, *in* 'Machine Learning: A constraint-based approach', Morgan Kaufmann, pp. 236–338.

Haibo He, Yang Bai, Garcia, E. A. & Shutao Li (2008), Adasyn: Adaptive synthetic sampling approach for imbalanced learning, *in* '(IEEE World Congress on Computational Intelligence)', pp. 1322–1328.

Husain, A., Salem, A., Jim, C. & Dimitoglou, G. (2019), Development of an efficient network intrusion detection model using extreme gradient boosting (xgboost) on the unsw-nb15 dataset, *in* '(ISSPIT)', pp. 1–7.

Janarthanan, T. & Zargari, S. (2017), Feature selection in unsw-nb15 and kddcup'99 datasets, *in* '(ISIE)', pp. 1881–1886.

Jing, D. & Chen, H. (2019), Svm based network intrusion detection for the unsw-nb15 dataset, *in* '(ASICON)', pp. 1–4.

Moustafa, N. & Slay, J. (2015*a*), The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems, *in* '(BADGERS)', pp. 25–31.

Moustafa, N. & Slay, J. (2015*b*), Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), *in* '(MilCIS)', pp. 1–6.

Nguyen, G. H., Bouzerdoum, A. & Phung, S. L. (2008), A supervised learning approach for imbalanced data sets, *in* '2008 19th International Conference on Pattern Recognition', pp. 1–4.

Pasumarthy, N. (2018), 'Encoding fixed length high cardinality non-numeric columns for a ml algorithm'.

Risk Based Security (2019), *2019 MidYear QuickViewData Breach Report*.

Rodríguez, P., Bautista, M. A., Gonzàlez, J. & Escalera, S. (2018), 'Beyond one-hot encoding: Lower dimensional target embedding', *Image and Vision Computing* **75**, 21 – 31.

Seghouane, A., Shokouhi, N. & Koch, I. (2019), 'Sparse principal component analysis with preserved sparsity pattern', *IEEE Transactions on Image Processing* **28**(7), 3274–3285.

Tobi, A. M. A. & Duncan, I. (2018), 'Kdd 1999 generation faults: a review and analysis', *Journal of Cyber Security Technology* **2**(3-4), 164–200.

Zhiqiang, L., Mohi-Ud-Din, G., Bing, L., Jianchao, L., Ye, Z. & Zhijun, L. (2019), Modeling network intrusion detection system using feed-forward neural network using unsw-nb15 dataset, *in* '(SEGE)', pp. 299–303.

Zimek, A., Schubert, E. & Kriegel, H.-P. (2012), 'A survey on unsupervised outlier detection in high-dimensional numerical data', *Statistical Analysis and Data Mining: The ASA Data Science Journal* **5**(5), 363–387.