A Survey on Machine Learning-based Intrusion Detection with Encrypted Data

Chris Regy Vallikunnathu (Student ID: 40232485)

Abstract

Over the last decade, more individuals and businesses have begun exploiting the internet and other communication technologies. This surge in demand was accompanied by rapid advances in this domain as well. Such huge surge in demand and advances in technology has drawn our attention and focus to the question of digital security. A key aspect of digital security is network security. Networks are among the most targeted points of attack by malicious actors. An Intrusion Detection System (IDS) is one among the many tools that have been developed in an attempt to ensure security of networks. An Intrusion Detection System constantly monitors the network in order to detect any malicious or suspicious activity and report such activity. Although such systems have proven their worth over the decades, the alarming increase in network traffic and data over the last few years have given rise to security risk and threats that cannot be effectively dealt with by intrusion detection systems. In an attempt to deal with such situations, modern-day researches have attempting to employ machine learning and deep learning algorithms in order to create and efficient and effective network intrusion detection system. We attempt to survey several research papers and articles published on the topic of machine learning based intrusion detection systems and summarize the key machine learning and deep learning approaches that have been proposed.

Keywords – Network Security; Intrusion Detection System; Machine Learning; Deep Learning; Detection Rate; Accuracy; False Alarm Rate.

I. INTRODUCTION

RECENT developments in the field of internet and other communication technologies have revealed a compelling need to develop technologies that are able to reliably protect and safeguard networks against modern-day attacks. This realization has led to network security becoming one among the most crucial domains of research and development. According to IBM's 'Cost of Data Breach' 2022 report, it has been estimated that over 83% of all organizations have been victims to network attacks or data breaches, and the average cost of each data breach is close to USD 4.35 million. Many different tools and technologies

have been developed over the decades in order to protect individuals and businesses against such attacks. Some of the more common tools that are used to secure networks include Firewalls, Network Access Control (NAC), Virtual Private Networks (VPN) and Network Intrusion Detection Systems (NIDS). A Network Intrusion Detection System is a listen-only security device that functions by continuously monitoring the network to detect any suspicious activity that the system predicts might lead to a possible security violation. Such devices are employed in an attempt to detect any malicious activity before it is able to infect the entire network. Once such an event is detected, the system notifies or alerts the relevant responder. An Intrusion Detection System is different from an Intrusion Prevention System (IPS), which is another popular network security tool, which also monitors network traffic but aims to prevent detected threats as opposed IDS which is primarily focused on detecting threats. An IDS is places out of the network band in use and makes use of a Test Access Point (TAP) or a Switched Port Analyzer (SPAN).

The idea of an intrusion detection system was conceived by James P. Anderson in 1980 [1]. This research was sponsored by the United States Department of Defense following the realization of the need to protect critical electronic networks that were used on a daily basis. In the decades since, a number of intrusion detection systems have been developed and designed in order to enhance network security. Although intrusion detection systems are extremely useful in detecting and thwarting a many attacks, traditional intrusion detection systems are largely unable to keep up with the volume of data that is being sent over modern networks and the wide variety of attack types that are exploited by malicious actors. These systems have proven inadequate when it comes to detecting attacks that target zero-day vulnerabilities. Unacceptably

high False Alarm Rates (FAR) is another problem that haunted these systems. In order to tackle this situation, a large number of researches are underway to develop an efficient intrusion detection system that is able to detect modern network attacks. Many of these researches focus on the use of machine learning and deep learning algorithms to achieve the desired objectives.

Both machine learning and deep learning concepts fall under the umbrella of Artificial Intelligence. Despite the close similarities, it is important to understand what differentiates the two concepts. Machine Learning represents a set of techniques and algorithms that enable machines to examine large amounts of data, to learn from this data and to apply what was learned to make informed predictions or decisions.

Deep Learning is a subset of Machine Learning, which makes use of many layers of processing algorithms to create an artificial neural network that can learn and make informed predictions or decisions. Deep learning techniques typically require less human intervention than machine learning, but often demands heavy computational resources. Machine learning and deep learning techniques are proving to be extremely useful in the development of modern network security solutions. They are able to efficiently learn network traffic characteristics and use this knowledge to detect abnormal behavior.

Through this report, we present an overview of the major developments that have been made in the field of network intrusion detection systems through the use of machine learning and deep learning algorithms. We examine what are the different proposals that are being put forward and understand their workings. We also compare and contrast the performance of the proposed models and attempt to identify existing gaps in this research area.

II. Definition & Classification

Any unauthorized attempt to compromise a system's confidentiality, availability or integrity qualifies as an intrusion. An intrusion detection system is a security mechanism or device that constantly monitors all traffic that flows through the host and the network to detect any suspicious behavior that might result in a possible security violation. In the event that any such activity is detected, the IDS is generally tasked with alerting a network administrator who will be able to initiate defensive or remedial actions. In the most general passive deployment use-case, a network intrusion detection system is connected to a network switch. The network switch in use is port mirroring enabled, allowing it to transfer a copy of network packets from one port to a monitoring connection on another port, which in this case is the NIDS. This allows the NIDS to monitor all traffic to detect any intrusions.

An IDS can be classified on the basis of the deployment method or on the basis of the detection method. On the basis of the deployment method, IDS can be classified into host-based IDS and network-based IDS. A host-based IDS monitoring only the traffic a single host in a lookout for any suspicious activity. However, this type of IDS necessitates a deployment on every single host that needs to be scanned for possible intrusions. This approach might not always be suitable in a real-time scenario, primarily due to huge processing overloads.

Unlike host-based IDS, a network-based IDS is deployed on the network and will monitor all traffic in the network. This approach is more useful than the previous one as it is able to protect the entire network and all hosts connected to the network from intrusions. Most researches that propose the use of machine learning or deep learning models focus on network-based IDS.

On the basis of the detection method used, intrusion detection systems can be classified into signature-based IDS and anomaly-based IDS. Signature-based intrusion detection systems work by defining a signature for a particular type or pattern of attacks/intrusions and storing this pattern in a signature database. The idea here is that an intrusion can now be detected by comparing the pattern to the signature that is already stored in the database. This mechanism offers a high level detection accuracy and efficiency in the case of previously defined attack types as it can be easily and reliably matched with the signature stored in the database. However, signature-based IDS is completely ineffective in the case of a zero-day attack that has no corresponding signature stored in the database to be compared and matched with. Another significant disadvantage of this approach is the need to store a large sized signature database. Many modern devices that heavily rely on networks and hence need to be protected from intrusion, including IoT devices and smart wearable come severely restricted storage space and computational power. Maintaining a large signature database and matching possible attack patterns with stored signatures might be largely infeasible given the restriction of available resources.

An anomaly-based IDS, on the other hand, works by measuring and learning from network traffic and usage patterns in order to define standard/expected behavior. Any significant deviation from this expected behavior profile is then considered an anomaly and hence flagged as suspicious. The primary advantage here that an anomaly-based IDS, unlike the signature-based IDS, is able to detect an new types of attacks which do not have any known patterns. This approach gains from being highly flexible as

a result of being able to define different expected behavior profiles for different networks and services. This approach, however, from high false alarm rates as it is highly complex and difficult to define the fine line between what is expected behavior and what is not.

III. General ML-Based NIDS Methodology

A. *General Methodology*

Close to all implementations that make use of machine learning algorithms to develop intrusion detection systems go through three major steps, (i) Data pre-processing phase (ii) Training phase, and (iii) Testing phase.

The data pre-processing stage ensures that the data being inputted to the algorithms are clean and well organized. It simply transforms the large amounts of raw data available into a form that is suitable for the algorithm. This is an integral step of all machine learning algorithms as their ability to learn and generate meaningful outcomes maybe directly impacted by the input data quality. Once the data pre-processing phase is completed, the data is then divided into two different groups. It is important that the division here remains random. The first group consists of roughly 80% of the original pre-processed dataset and is referred to as the training dataset. The remaining 20% data comprises the testing dataset.

Data from the training dataset will be inputted into the machine learning algorithm in order for it to note patterns and learn. A number of factors including the size of the data being fed into the algorithm and the complexity of the model involved will determine the time that to train the algorithm. As explained earlier, deep learning algorithms typically tend to be more sophisticated than machine learning algorithms and this can mean generally higher training times in case of DL algorithms. Upon completion of the training phase, the testing dataset will be used to gauge the accuracy of the predictions or decisions made by the algorithm. In the specific case of a network intrusion detection system, this decision is whether a particular pattern on network behavior is an intrusion or not.

B. *Commonly Used Datasets*

Most of the proposed models were tested using one of the following popularly used datasets: (i) KDD Cup 1999, (ii) NSL-KDD, and (iii) CICIDS2017 dataset.

KDD Cup'99 is one of the most widely used datasets for testing intrusion or anomaly detection capabilities. This dataset contains close to 4,900,000 connection vetors. Each of these connection vectors possesses 41 features and labeled as an attack or normal. The various attack types present in the dataset include (i) Denial of Service (DOS) attacks, (ii) Remote to Local (R2L) attacks, (iii) Probing attacks, and (iv) User to Root (U2R) attacks. A factor that makes this dataset realistic and useful is the presence of specific attack types in the test dataset that are not present in the training dataset. This feature enables developers to test the performance of their proposed models in relation to new attack types that have not been encountered by the model before. However, this dataset is quite old and might not able to accurately reflect the nature of modern-day network attacks and intrusions. It has also been criticized by McHugh due to the characteristics of synthetic data [2].

The NSL-KDD dataset is derived from the KDD Cup'99 dataset and was suggested with the objective of tackling some of the problems associated with the KDD Cup'99 dataset. Even though many of the criticisms faced by KDD Cup'99 is still applicable to the NSL-KDD dataset, it remains an effective benchmark to test intrusion detections. An important advantage of the NSL-KDD dataset is the inverse proportionality between the instances of chosen records from each difficulty group and the percentage of records in the original dataset. This results in a wider variation of the classification rates of different machine learning methods. This enables a more accurate evaluation of the various proposed models.

The CICIDS2017 dataset is the latest and most up-to-date among the three popularly used datasets for intrusion detection models. This fact makes the dataset extremely relevant modern-day researchers in this domain. The CICIDS2017 dataset consists of labeled network flows, including full packet payloads in pcap format, the corresponding profiles and the labeled flows (GeneratedLabelledFlows.zip) and CSV files for machine and deep learning purpose (MachineLearningCSV.zip) are publicly available for researchers [3].

Up until now, we have described the broad and general methodology of implementation typically followed by all machine learning-based NIDS proposals. We have also briefly looked at the various datasets that are in popular use by a majority of surveyed researches. In the coming section, we explore the details of implementation and workings, specific to the proposed approaches mentioned in the research papers and articles that we surveyed.

IV. Machine Learning Algorithms

A. *Decision Tree*

A decision tree is an ML algorithm which, as its name suggests, uses a tree-like hierarchical structure of decisions and their consequences, including outcomes of chance events and the cost of the utilized resources. The

two types of decision trees are (i) Regression Trees and (ii) Classification Trees. Regression trees are generally used for continuous quantitative target variables whereas classification trees are typically used for discrete categorical target variables.

A decision tree consists of three types of nodes: chance nodes, decisions nodes and end nodes. In simple terms, a chance node signifies multiple uncertain outcomes, a decision node signifies a decision that has to be made and an endpoint node indicates the final outcome. It is possible to reduce possible risk and increase the chances of obtaining a desirable result. Decision trees are widely used for the purpose of classification. It divides features and tests the value of each feature. This process of feature division continues up until a point where each branch has just one classification. Some of the common decision tree algorithms in popular use include C4.5, ID3 and CART [4]. Decision trees have also led to the development of multiple powerful algorithms including Random Forest and XG Boost [4].

Through the IEEE published research article titled, "A Reliable Network Intrusion Detection Approach Using Decision Tree with Enhanced Data Quality", ***Guezzaz et al*** [5] proposed an interesting new approach to network intrusion detection based on the decision tree algorithm. The algorithm is used to develop and train a binary classifier model. The two main contributions claimed by the authors include feature selection using entropy decision technique and a decision tree algorithm based classifier model.

The proposed approach consists of three major phases. The first phase is the data quality process during which the data being used of training and testing phases goes through sanitizations and pre-processing in order to avoid any incompatibility issues during runtime. During this phase, feature selection is applied in order to ensure good training data by minimizing the number of features. Entropy decision technique is made use of to perform feature selection to ensure the selection of most relevant features in the data the removal of irrelevant ones. The second phase involves the development of the binary classifier model. The sanitized and pre-processed data is the input to this phase. The decision tree classifier model goes through training and testing phases. The third phase is the deployment of the developed model. The decisions made by this model were tested against the NSL-KDD dataset and CICIDS2017 dataset. The authors report a good performance in terms of high detection rates and low false alarm rates when tested using these datasets.

B. K-MEANS CLUSTERING

K-means clustering is a simple and popular unsupervised centroid-based iterative ML algorithm which works by grouping similar data together into a cluster in order to identify and expose the underlying patterns. In K-means clustering, 'k' refers to the number of required centroids. Each centroid represents the center of a cluster, which is nothing but a collection of data which has been grouped together based on their similarities. The K-means algorithm begins by using a group of randomly chosen centroid as the starting points for every cluster and then performing calculations iteratively to optimize the centroid positions. The objective here is to minimize the sum of the distances between the data points and their corresponding centroids.

Yao et al [6] proposes a multi-level semi-supervised machine learning framework for intrusion detection named MSML. Firstly, a data generator process is used to generate the training and testing data to be used by the MSML framework. The training data contains both labeled and unlabeled samples. During this phase, the data also goes through normalization and sanitization. The MSML framework itself consists of four key modules: (i) pure cluster extraction, (ii) pattern discovery, (iii) fine-grained classification, and (iv) model updating.

The objective of the pure cluster extraction module is to locate pure clusters using a semi-supervised k-means algorithm. A pure cluster can be defined, in general terms, as a cluster where a significant majority of the samples fall into the same category. In the pattern discovery module, cluster based methods are applied to identify unknown patterns. Patterns can be classified as known or unknown. A known pattern is one that is contained in the training dataset whereas an unknown pattern is one that is not present in the training dataset. Patterns can also be classified as intrusion patterns or normal patterns. After processing by the pattern discovery module, some patterns do not fall into either category and is classified as 'new'. Fine-grained classification is needed in such cases to classify these patterns. The unknown patterns go through k-means clustering. A few samples a randomly selected from each cluster for manual inspection. Fine-grained classification can be considered complete if all the selected samples have the same ground-truth category. Finally, the model updating module offers a retraining mechanism.

The proposed model is able to effectively tackle class imbalance and the non-identical distribution problem. The authors conclude that the MSML framework can effectively differentiate between known and unknown pattern samples while achieving 99.3% accuracy for known pattern samples. It is able to detect attacks with low number of instances in the dataset.

C. ARTIFICIAL NEURAL NETWORK

Artificial Neural Networks, or simply neural networks, are supervised machine learning algorithms that attempt to mimic the neural functionality of the human brain. These neural networks are made of several node layers, where

each node works like an artificial neuron which sends passes on signals to the next neuron. The node layers include an input layer, one or more hidden layers, and an output layer. The nodes, which act as neurons, are connected to each other. One node sends data to next layer of the network only once it is activated. This is decided by the weight or the threshold corresponding to the particular node. If the node's output crosses the set threshold value, it is activated. This activation results in the transfer of data onto the next node. Some of the popularly used activation functions include linear regression, logistic regression, binary sigmoid, bipolar sigmoid and rectified linear activation function (ReLU). A major drawback of the artificial neural network is its complex nature and resulting high time consumption due to slow learning processes.

Ali et al [7] proposes to resolve this problem through a model based on fast learning network and particle swarm optimization. Particle Swarm Optimization (PSO) is a parallel evolutionary computation technique based on the social behavior metaphor developed by Mishra and Sengupta[999]. The performance of a particle swarm optimization algorithm varies based on a tradeoff between exploration and exploitation. The ability to identify a desirable global optimum by evaluation the many regions in the problem space is referred to as exploration. On the other hand, exploitation is the ability to locate the optimum as quickly and efficiently as possible by concentrating the search to the close neighborhood of a highly probable or likely candidate.

The fast learning network used in the proposed model, developed by *Sahu et al* [8] is a parallel connection of a single hidden layer feedforward neural network and three-layer Feedforward Neural Network (FNN), hence forming a Double Parallel Forward Neural Network (DPFNN). A problem that arises during the PSO-based optimization of the fast neural network is having to choose values for the two weights and also selecting the number of required neurons in order to maximize accuracy. This problem is tackled by consider the maximum number of neurons while assigning a length for the particle.

The developed model has been tested using the KDD Cup'99 dataset. The FLN used in model has been evaluated through a comparison against other optimization algorithms and results suggest that the proposed PSO-based FLN exhibits better performance than the compared algorithms. The authors demonstrate that accuracy can be maximized by increasing the number of nodes in the hidden layer. However, it is important to note that the detection rate accuracy is lesser than desirable for lower attack classes.

D. RECURRENT NEURAL NETWORKS

A Recurrent Neural Network (RNN) is sub-category of Artificial Neural Networks that models sequential or time-series data. It is deep learning technique that has gained wide popularity in the domains of handwriting prediction, speech recognition, natural language processing, image captioning and others. *Figure 1* demonstrates the basic distinguishing principle of a recurrent neural network. A prediction of a traditional neural network on the current input is not dependent is anyway to the output of the previous input. The factor that distinguishes recurrent neural networks from traditional neural networks is the use of output from the previous input to help make a decision on the current input. Recurrent neural networks also share the same weight parameters between the different layers of the network. RNN also makes use of back propagation through time (BPTT) algorithm to establish gradients.

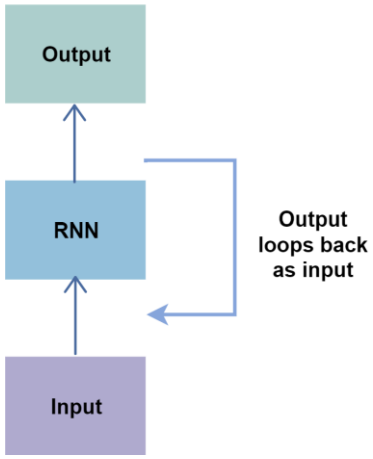

Figure 1: Simple representation of a Recurrent Neural Network

A key extension to recurrent neural networks is the bidirectional recurrent neural networks (BRNN) [9]. BRNN works by combining two opposite recurrent neural networks that makes use of the same input and output layers. The advantage of this technique is the trained data is now linked to both past and future data. The forward and backward layers in the BRNN are hidden layers which have are not connected to each other so as to prevent any loops.

Recurrent neural networks are valuable in the development of intrusion detection systems as they can be used to supervised classification and feature extraction. A notable disadvantage of a recurrent neural network is the limitation on sequence length. In case of longer than expected sequences, a recurrent neural network may face short-term memory issues.

Xu et al [10] alleviates this problem through the use of gated recurrent unit as the main memory unit instead of long short term memory (LTSM). Their intrusion detection system proposal consists of a recurrent neural network with gated recurrent units, multilayer perceptron and a softmax module. The authors argue that a gated recurrent unit is a simplification over LTSM that offers comparable performance results by reducing the number of gates used. This is made possible by combining the input gate and the forget gate into the update gate. This results in a simpler construction with a minimum number of parameters offering better performance and convergence. The multilayer perceptron consists of a unidirectional artificial neural network that has many layers. The three components that constitute the multilayer perceptron are the input layer, the output layer and many hidden layers. The sigmoid function is a good example for the activation function used by the multilayer perceptron as it is continuous and monotonically increasing.

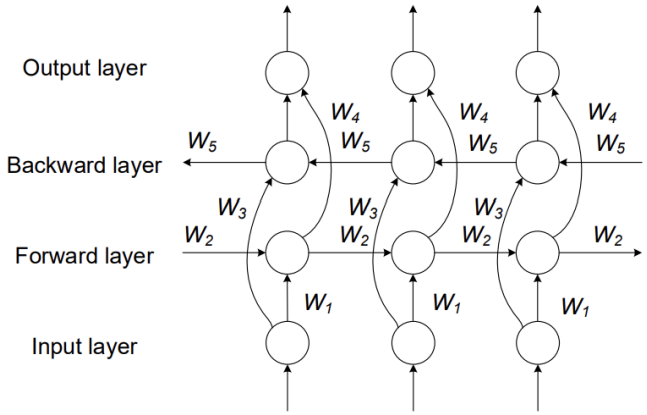

Figure 2: Bidirectional Recurrent Neural Network [10]

Figure 3 demonstrates the overall architecture of the proposed intrusion detection system. It consists of four modules, (i) the pre-processing module (ii) the gated recurrent unit, (iii) the multilayer perceptron, and (iv) the output module. The purpose of the first module is to process the data to make it more suitable and compatible for the neural network. The gated recurrent unit consists of many bidirectional layers used to identify and store relevant features. The multilayer perceptron performs mapping from the output of the gated recurrent unit which makes a non-linear classification decision. The softmax output module normalized the probability of classification and outputs it as a final result.

Testing using the KDD Cup'99 and NSL-KDD datasets demonstrates that the use of gated recurrent units as the main memory unit results in considerable performance improvements over LTSM.

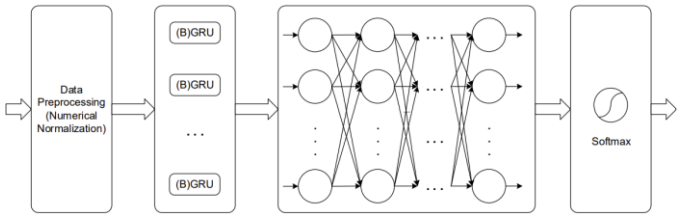

Figure 3: Architecture of model proposed by Xu et al [10]

E. AUTOENCODER

The AutoEncoder is an unsupervised artificial neural network is able to encode data and also learns how to construct back the same data from the encoded version. The AutoEncoder consists of three main layers, the input layer, the output layer and one or many hidden layers. It is important to note that the input layer and the output layer has the same number of neurons. *Figure 4* demonstrates the architecture of a simple AutoEncoder with one hidden layer.

The working of the AutoEncoder can be represented by two processes, namely encoding and decoding. The encoding process compresses the dimensions of the input data to generate an encoded version of the original data. This process is performed between the input layer and hidden layers. The objective of the decoding process is to reconstruct the original input data from the encoded or compressed version. The reconstructed version should as closely match the original input data as possible and degree to which it matches the original data is represented by the reconstruction loss.

Through the research article titled "A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection", ***Khan et al*** [11] proposes a two-stage deep learning based intrusion detection model that uses a stacked AutoEncoder with a softmax classifier. The two stages that forms the proposed model are the initial decision stage and the final decision stage which are constructed using a deep neural network. The use of the deep neural network is primarily because of the advantage it offers in terms of performance and speed during classification. The first stage of the proposed model not only classifies the network traffic as either normal or abnormal but also assigns a score representing the probability of attack which helps to better train the second or final decision stage for classification of normal and multi-class attacks.

The deep neural network used to build the initial and final decision stages consists of a deep stacked AutoEncoder with two hidden layers and a softmax module on the top. The training phase consists of two steps. In the pre-

training step, an unsupervised learning technique is used to train each layer of the deep stacked AutoEncoder individually which helps to reduce the errors that result from reconstructing the input features at each layer. The first layer outputs the compressed features which are then inputted into the second layer. Once both the layers have been pre-trained, the fine-tuning task is performed. During fine tuning, the two layers are stacked and a softmax classifier module is added on top. The parameters of the pre-trained deep stack AutoEncoder is then tuned with back-propagation algorithm based supervised learning mechanism to ensure the reduction of prediction errors through the use of labeled features. Once the training phase is completed, the model is able to perform accurate classification of new or zero-day network attacks. The authors claim that the proposed model is able to efficiently learn useful features from large amounts of unlabeled data and is also able to automatically classify them. The authors conclude that the model is able to outperform existing approaches when tested using the KDD Cup'99 and UNSW-NB15 datasets.

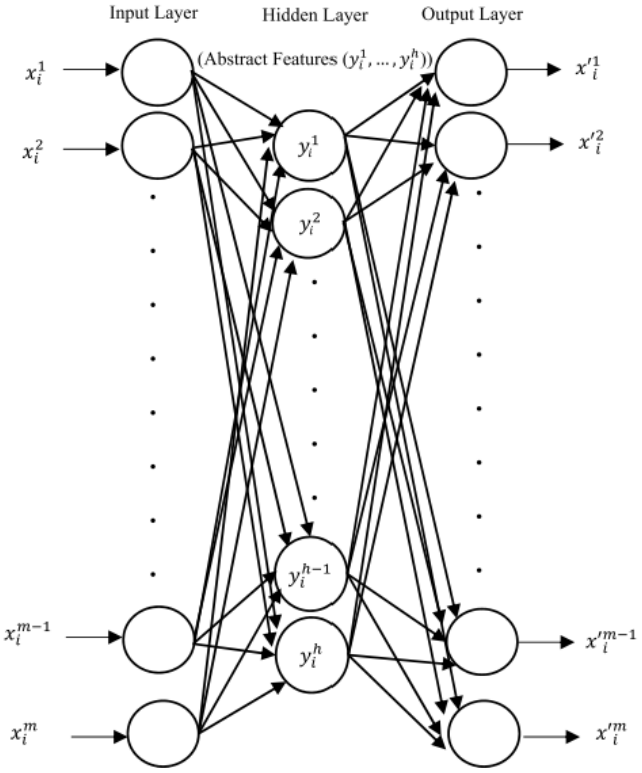

Figure 4: The basic architecture of the AE with a single hidden layer [11]

F. *CONVOLUTIONAL NEURAL NETWORK*

Convolutional neural network is a deep learning structure that has gained wide popularity in the computer vision field. Convolution neural networks possess the ability to identify and learn spatial hierarchies of features through backpropogation. This neural network is particularly suited for data stored in arrays. The several layers that together constitute a convolutional neural network are the input layer, the stack of convolutional and pooling layers, a fully connected layer and a softmax classifier in the classification layer. Convolutional neural networks are of high value to the development of intrusion detection systems as they can be used for classification and feature extraction.

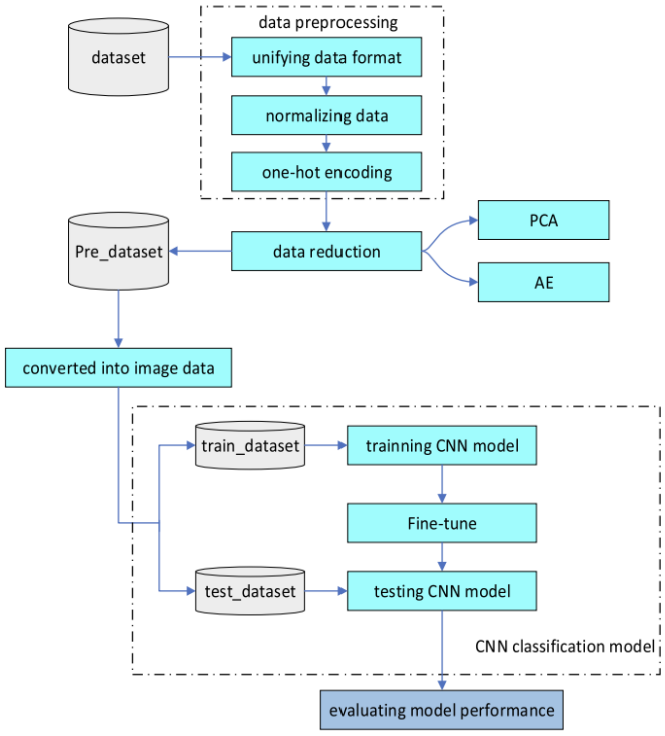

Figure 5: Architecture of model proposed by Xiao et al [12]

Xiao et al [12] proposed an intrusion detection model based on convolutional neural networks. The model can be summarized into three steps as seen from the figure 5. In the first step, the datasets are digitized and normalized to obtain standardized datasets. The input data form of the convolution neural network is two-dimensional, and in accordance with this data form, each dataset is converted into a two-dimensional dataset. In the second step, the authors obtain the optimal features by training the pre-processed dataset. The five attack states identified using the softmax classifier include (i) Denial-of-Service/DOS, (ii) Probe, (iii) Root to Local/R2L, (iv) User to Root/U2R, and (v) Normal. In step 3, the performance of the model is enhanced through model training and fine-tuning using backpropagation. The proposed model is able to bring down computational cost by reducing the time taken in the training and test phases. The authors conclude that the proposed model is able to perform better than traditional algorithms in terms of higher accuracy and lower false alarm rates.

V. RESULTS

A. *PERFORMANCE METRICS*

In order to be able to objectively compare the performance of the different proposed IDS models, we need the performance of many different approaches to be measured using the same metrics. The most commonly used metrics to measure the performance of IDS models based on machine learning and deep learning are based on the attributes used in the confusion matrix. A confusion matrix is used to evaluate the performance of a machine learning classification models through a comparison between the actual and predicted values of the target variable. The four attributes of a confusion matrix are explained below from an IDS perspective:

i. **True Positive** (TP): The number of attacks that have been correctly classified as 'Attack'.

ii. **False Positive** (FP): The number of normal traffic instances that have been incorrectly classified as 'Attack'.

iii. **False Negative** (FN): The number of attacks that have been incorrectly classified as 'Normal'.

iv. **True Negative** (TN): The number of normal traffic instances that have been correctly classified as 'Normal'.

Actual Classification	Predicted Classification	
	Attack	**Normal**
Attack	True Positive	False Negative
Normal	False Positive	True Negative

Table 1: Confusion Matrix

Some useful performance evaluation metrics based on the attributes of a confusion matrix popularly used to evaluate proposed IDS models include:

Precision: It can be defined as the ratio of the number instances correctly classified as attacks to the total number of instances classified as attacks.

$$\text{Precision} = TP / (TP + FP)$$

Recall Ratio/Detection Rate: It can be defined as the ratio of the number of instances correctly classified as attacks to the total number of actual attack instances.

$$\text{Recall} = \text{Detection Rate} = TP / (TP+FN)$$

False Alarm Rate (FAR): It can be defined as the ratio of the number of instances incorrectly classified as attacks to the total number of actual normal instances.

$$FAR = FP / (FP + TN)$$

True Negative Rate: It can be defined as the ratio of the number of instances correctly classified as normal to the number of actual normal instances.

$$\text{True Negative Rate} = TN / (TN + FP)$$

Accuracy (ACC): It can be defined as the ratio of the number of instances correctly classified as either attack or normal to the total number of instances.

$$ACC = (TP + TN) / (TP + TN + FP + FN)$$

B. *PERFORMANCE OF PROPOSED APPROACHES*

This section describes the reported performance of the various machine learning and deep learning based approaches to IDS that were discussed in earlier sections:

Decision Tree: *Guezzaz et al* tested their decision tree based IDS approach using the NSL-KDD and CICIDS2017 datasets. Using the NSL-KDD dataset, the accuracy of the proposed model is 99.42%. The authors also report a detection rate of 88.5% and a false alarm rate of 2.64%. When tested using the CICIDS2017 dataset, the proposed model achieves an accuracy of 98.8%, a detection rate of 97.3% and a false alarm rate of 3.10%. The reported evaluation metrics indicate good performance in terms of high accuracy and low false alarm rates, demonstrating the effectiveness and relevance of the proposed approach.

K-means Clustering: The MSML framework based on K-means clustering proposed by *Yao et al* demonstrates good performance in terms of accuracy, F1 score and unknown pattern recognition ability when tested using the KDD Cup'99 dataset. The framework obtains an accuracy of 99.3% for known pattern samples and an overall accuracy of 96.6%. A notable feature of this model is the ability to effectively detect attacks even with low instances in the dataset.

Artificial Neural Network: A major drawback faced by ANN is the high time consumed by the training process due its complex nature. The IDS model proposed by Ali et al based on fast learning network and particle swarm optimization is able to effectively tackle this problem. This model was tested using the KDD Cup'99 dataset. The proposed PSO-FLN model demonstrates superiority in performance over other FLN models that made use of alternative algorithms for optimization including Genetic algorithm (GA), Harmony Search Optimization (HSO) and

Ameliorated Teaching Learning based optimization. They also demonstrated that an increase in accuracy can be made possible through an increase in the number of neurons in the hidden layer. However, low detection rate accuracy was observed in case of lower attack classes like R2L. The authors observe that this tendency results from the low amount of training data when compared to other classes.

Recurrent Neural Network: In order to alleviate RNN's short-term memory issues while handling long sequences, *Xu et al* employed a Gated Recurrent Unit as the main memory. The model also uses a multilayer perceptron and softmax classifier. This model was tested using the KDD Cup'99 and NSL-KDD datasets. An overall detection rate of 99.42% was obtained when using the KDD Cup'99 dataset while 99.31% was obtained when using the NSL-KDD dataset. Low false alarm rates of 0.05% and 0.84% were obtained using the KDD Cup'99 and NSL-KDD datasets respectively. The results demonstrate that a GRU can be more suited for use in an intrusion detection system as it offers simplification and performance improvement over Long Short Term Memory. The only drawback here, similar to ANN, is the low detection rate for lower attack classes like R2L and U2R.

AutoEncoder: Khan et al's proposal of a two-stage model based on deep stacked AutoEncoder is able to efficiently learn and classify useful feature representations form huge amounts of unlabeled data. Performance was tested using the KDD Cup'99 and UNSW-NB15 datasets. Downsampling for KDD Cup'99 and upsampling using SMOTE for UNSW-NB15 datasets were performed to minimize the problems that might arise due to class imbalance of the datasets. This preprocessing results in a considerable improvement of detection rate efficiency for lower attack classes. Detection accuracy was observed to be 99.996% and 89.134% for KDD Cup'99 and UNSW-NB15 datasets. However, a dip of close to 10% is observed in detection accuracy when tested using newer datasets.

Convolutional Neural Network: Xiao et al's CNN-based IDS proposal performs feature extraction using Principal Component Analysis and AutoEncoder. Experimental analysis using the KDD Cup'99 dataset shows that the proposed model is able to efficiently detect intrusions by dimensionality reduction with an accuracy of 94%, detection rate of 93% and a false alarm rate of 0.5%. However, considerably lower detection rates were observed in case of U2L and R2L attacks at 20.61% and 18.96%. Low detection accuracy of these lower class attacks restricts the overall detection accuracy.

VI. Distribution of Workload & Individual Contribution

The objective of this project undertaken by me and my colleague is to perform a comprehensive analysis of the various machine learning algorithms and techniques that have been used in order to develop capable and efficient network intrusion detection systems. Although this research is in a nascent stage, significant improvements have already been made in terms of high accuracy and low false alarm rates. In the first stage of the project, myself, Chris V and my colleague, Harleen Kaur, attempted to understand traditional network security solutions and look at how machine learning might be able to address some of the challenges faced by traditional solutions. Then, we set out to review several research papers and articles in order to obtain a comprehensive understanding of ML and DL-based IDS implementations. We agreed that I would review proposals based on machine learning algorithms while my colleague reviewed deep learning-based models. Following are the research articles reviewed by me:

'A Reliable Network Intrusion Detection Approach Using Decision Tree with Enhanced Data Quality' by *Guezzez et al* published through the Hindawi Security and Communication Networks Journal in 2021. The authors propose an intrusion detection system based on decision tree. The model also features an enhancement of data quality through pre-processing and entropy decision feature selection.

'***MSML: A Novel Multi-level Semi-supervised Machine Learning Framework for Intrusion Detection System'*** by *Yao et al* published through the IEEE Internet of Things Journal (Volume: 6, Issue: 2) in 2019. The authors present a multilevel semi-supervised machine learning framework called MSML based on the K-means clustering algorithm.

'A New Intrusion Detection System Based on Fast Learning Network and Particle Swarm Optimization' by *Ali et al* published through IEEE Access (Volume: 6) in 2018. The proposed model is based on Artificial Neural Network and PSO-FLN algorithm.

After the review of these research articles, we came together to prepare a presentation and this final report based on our analysis. I confirm that the workload was fairly distributed between me and my colleague. I also confirm that the same grade can be assigned to the both of us.

VII. Conclusion

This report aims to summarize the various machine learning and deep learning algorithms that have proved to be extremely useful in the development of effective network intrusion detection systems. We first described the traditional network security solutions and their drawbacks. We explored the motivation behind the introduction of ML and DL algorithms into this domain and the general methodology adopted by all ML-based IDS approaches. We then elaborated on the specific details of the most popular ML and DL algorithms in use, their implementation techniques, strengths, weaknesses and reported performances.

Overall comparison of the different algorithms used indicates the superiority of DL-based intrusion detection systems in terms of maximization of detection accuracy and minimization of false alarm rates. AutoEncoders, like the one used by *Khan et al,* are among the most popular deep learning algorithms in use towards the development of intrusion detection systems. Although the performance of DL-based models is superior to that of ML-based algorithms, they are significantly more complex. As a result, DL-based algorithms demand considerably higher computing resources in terms processing power and storage capabilities. However, this presents a significant challenge as modern-day systems like IoT devices and smart wearables are lightweight and do not possess the necessary computing capabilities. In order to minimize the computational cost of the proposed DL-based IDS solutions, more researches must be undertaken to develop a more intelligent feature selection algorithm which can efficiently select the most relevant features in order to facilitate a faster learning process.

Another challenge that we observed is the extensive use of KDD Cup'99 and NSL-KDD datasets to test the proposed approaches. These datasets are quite old and fail to accurately represent the wide variety of modern network attacks. As a result, models might not be as effective during real-world attack scenarios. Hence, the use of more up-to-date datasets like CSE-CIC-IDS2018 might help to obtain more relevant results as to how the models will perform in real-world situations.

Despite these challenges, it is important to acknowledge that the various proposed approaches have been able to demonstrate superior performance of traditional network intrusion detection systems in terms of accuracy. Research in this domain is still at an early stage and there is long way to go before we can achieve peak performance. The future scope of research in this area focuses on the development of intrusions detection systems that are not only capable and efficient but also lightweight and able reliably detect network intrusions in real-world scenarios.

VIII. References

[1] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance," Technical Report, James P. Anderson Company, Fort Washington, 1980.

[2] McHugh, John. (2000), "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory" in ACM Transactions on Information and System Security 3(4), pp. 262-294, 2000. doi: 10.1145/382912.382923.

[3] Sharafaldin, Iman & Habibi Lashkari, Arash & Ghorbani, Ali. (2018), "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization" in the 4th International Conference on Information Systems Security and Privacy, pp. 108-116, 2018. doi: 10.5220/0006639801080116.

[4] Ahmad, Zeeshan & Shahid Khan, Adnan & Shiang, Cheah & Ahmad, Farhan, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches" in Transactions on Emerging Telecommunications Technologies 32(3). doi: 10.1002/ett.4150.

[5] Guezzaz, Azidine & Benkirane, Said & Azrour, Mourade & Khurram, Shahzada, "A Reliable Network Intrusion Detection Approach Using Decision Tree with Enhanced Data Quality" in Hindawi Security and Communication Networks, vol. 2021. doi: 10.1155/2021/1230593

[6] H. Yao, D. Fu, P. Zhang, M. Li and Y. Liu, "MSML: A Novel Multilevel Semi-Supervised Machine Learning Framework for Intrusion Detection System," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1949-1959, April 2019, doi: 10.1109/JIOT.2018.2873125.

[7] M. H. Ali, B. A. D. Al Mohammed, A. Ismail and M. F. Zolkipli, "A New Intrusion Detection System Based on Fast Learning Network and Particle Swarm Optimization," in IEEE Access, vol. 6, pp. 20255-20261, 2018, doi: 10.1109/ACCESS.2018.2820092.

[8] S. K. Sahu, S. Sarangi and S. K. Jena, "A detail analysis on intrusion detection datasets," 2014 IEEE International Advance Computing Conference (IACC), Gurgaon, India, 2014, pp. 1348-1353, doi: 10.1109/IAdCC.2014.6779523.

[9] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," in IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681, Nov. 1997, doi: 10.1109/78.650093.

[10] C. Xu, J. Shen, X. Du and F. Zhang, "An Intrusion Detection System Using a Deep Neural Network With Gated Recurrent Units," in IEEE Access, vol. 6, pp. 48697-48707, 2018, doi: 10.1109/ACCESS.2018.2867564.

[11] F. A. Khan, A. Gumaei, A. Derhab and A. Hussain, "A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection," in IEEE Access, vol. 7, pp. 30373-30385, 2019, doi: 10.1109/ACCESS.2019.2899721.

[12] Y. Xiao, C. Xing, T. Zhang and Z. Zhao, "An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks," in IEEE Access, vol. 7, pp. 42210-42219, 2019, doi: 10.1109/ACCESS.2019.2904620.