Network Intrusion Detection Systems Using Artificial Intelligence/Machine Learning

Subjects: Computer Science, Artificial Intelligence

Contributor: Patrick Vanin , Thomas Newe , Lubna Luxmi Dhirani , Eoin O’Connell , Donna O’Shea , Brian Lee , Muzaffar Rao

The rapid growth of the Internet and communications has resulted in a huge increase in transmitted data. These data are coveted by attackers and they continuously create novel attacks to steal or corrupt these data. The growth of these attacks is an issue for the security of systems and represents one of the biggest challenges for intrusion detection. An intrusion detection system (IDS) is a tool that helps to detect intrusions by inspecting the network traffic. Although many researchers have studied and created new IDS solutions, IDS still needs improving in order to have good detection accuracy while reducing false alarm rates.

Intrusion Detection Systems (IDS) machine learning network security AI

1. Intrusion Detection Systems (IDS)

An IDS is a tool, either a hardware device or software, that raises an alarm when it detects malicious activity on the network. IDSs are the “watch-eye” of the network. It is an important component of the security architecture of modern networks. It allows early stages of attacks to be detected and thus, gives the opportunity to mitigate against them. In addition, they allow detection of a variety of attacks, e.g.: Denial of Service (DoS), Man in the Middle (MitM), etc. IDSs can monitor and log any, and all, network traffic as specified. Since IDSs can provide details on attacks as they occur, it helps security administrators to understand what has happened. Therefore, in the case of future similar attacks the security of the system can be configured to detect and prevent attacks of this type. There are two types of IDS, Network Intrusion Detection System (NIDS) and Host Intrusion Detection System (HIDS) [1]:

- Network Intrusion Detection Systems are IDSs placed in the network at strategic points. NIDS analyses the overall traffic of the network to detect if there are malicious activities in the network. NIDS helps to detect attacks from your own hosts and is a key component of the security of most organization networks.

- Host Intrusion Detection Systems are IDSs that are in all client computers (hosts) of the network. Contrary to NIDS, HIDS analyses the traffic of a single host as well as its activities, if it detects abnormal behaviour, it will raise an alarm.

There are three different detection types for IDS: Misuse Detection, Anomaly Detection, and Hybrid detection [1]:

- Misuse detection, also known as signature detection, searches for known patterns of intrusion in the network or in the host. Each attack has a specific signature, for instance, it can be the payload of the packet, the source IP address, or a specific header. The IDS can raise an alarm if it detects an attack that has one of the signatures listed in the list of known signatures of the IDS. The advantage of this approach is its high accuracy to detect known attacks. However, its weakness is that it is inefficient against unknown or zero-day (never seen before attack) patterns.

- Anomaly detection defines a normal state of the network or the host, called a baseline, and any deviation from this baseline is reported as a potential attack. For instance, anomaly-based IDS can create a baseline based on the common network traffic such as the services provided by each host, the services used by each host and the volume of activity during the day. Thus, if an attacker accesses an internal resource at midnight, and if in the baseline there should be almost no activity at midnight, then the IDS will raise an alarm. The advantage of anomaly detection is its flexibility to find unknown intrusion attacks. However, in most cases it is difficult to precisely define what the baseline of a network is, thus, the false detection rate of these techniques can be high.

- Hybrid detection combines both of the aforementioned detections. Generally, they have a lower false detection rate than anomaly techniques and can discover new attacks.

Nowadays, IDS by themselves are not enough for companies that want to protect themselves from attacks. IDS are more and more being replaced by Intrusion Prevention Systems (IPS). An IPS is similar to an IDS but with active components to stop attacks before they are successful. Usually, an IPS consists of a firewall with IDS rules. Contrary to IDS, IPSs are placed inline, this means that an IPS will continuously scan the traffic as the traffic passes through it. Thus, an IPS needs to be fast and have high computing capacities to avoid causing latency issues in a network which can affect network performance for its users.

One of the main disadvantages of many IPS is false positive attack detection. With an IDS, a false positive can be an inconvenience but for an IPS it can cause DoS, as legitimate traffic will be blocked. In addition, since IPSs, and especially Network Intrusion Prevention Systems (NIPSs), form single point of failure in the network, they need to be highly stable and robust against attacks.

2. Machine Learning

Machine learning is closely linked to Artificial Intelligence (AI) technology. It trains an algorithm to find regular patterns in a dataset. This training results in a model that can be used to predict or automate things. For IDSs, machine learning can be used to detect either known attacks or unknown attacks if the model has been sufficiently trained.

2.1. Supervised Machine Learning

Supervised machine learning uses labelled data to generate a function that maps an input to an output. The function is constructed from labelled training data.

Supervised learning can be categorized into two types of models, classification and regression:

- Classification models are used to put data into specific categories. From a dataset it recognizes the category it belongs to based on its features. The model will be trained on labelled input and output data to understand what the features of the input data are so as to correctly classify it. Classification models are very useful to detect attacks. For instance, for traffic coming into network, a well-trained classification model can classify the traffic as normal traffic or as abnormal traffic. If the model performs well, the abnormal traffic can then be classified into subcategories of well-known attacks such as DoS, phishing, worms, port scan, etc. Common classification algorithms are decision tree, k-nearest neighbour, support vector machine, random forest, and neural network [2].

- Regression models are used to predict continuous outcomes. The model is trained to understand the relationship between independent variables and a dependent variable. Regression is used to find patterns and relationships in datasets that can then be applied to a new dataset. Regression models are mainly used for forecasting the evolution of market prices or predicting trends [3]. Common regression algorithms are linear regression, logistic regression, decision tree, random forest, and support vector machine.

One of the main advantages of supervised learning is to use previous experiences to produce outputs. In addition, previous results can be used to improve the algorithm by optimizing the performance criteria to reach a precise model.

Supervised learning is used to solve many computational problems. However, the model needs precise and good input during the training phase to produce good outputs. In addition, this training requires a lot of computation time. Finally, it can be tough to classify big data, thus, if the dataset is too big, unsupervised learning is often a better choice.

2.2. Unsupervised Machine Learning

Unsupervised machine learning is used with unlabeled data. As suggested by its name, unsupervised machine learning is not supervised by the user to improve the model. The model will improve by itself as it will discover patterns and information from the dataset it was given. Usually, the algorithm will group the different data into categories that have the same similarities or by differences. Unsupervised learning is useful for big data analysis. As shown in **Figure 1**, unsupervised learning can be categorized as three types of problems clustering, association, and dimensionality reduction:

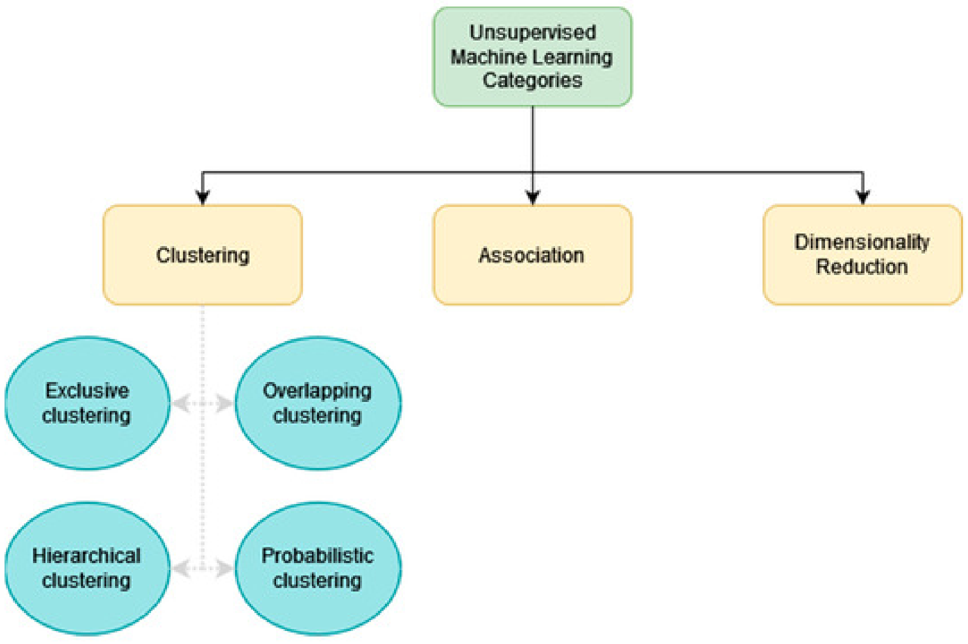

Figure 1. Unsupervised Machine Learning Categories.

- Clustering is a technique which groups unlabeled data according to their common or uncommon features [4]. At the end of this process, the unlabeled data will be sorted into different data groups. Clustering algorithms can be subcategorized as exclusive, overlapping, hierarchical and probabilistic [5]. With exclusive clustering, data that are grouped can only belong to one cluster only. Overlapping clustering allows data to belong to multiple clusters to have a richer model when data can belong to different categories. For instance, overlapping clustering is required for video analysis, since video can have multiple categories [6]. Hierarchical clustering is when the different isolated clusters are merged iteratively based on the similarity of the clusters until only one cluster is left. Different methods can be used to measure the similarities between two different clusters. The similarities between clusters are often measured as the distance between the clusters or their data. One of the most common metrics used is the Euclidean distance. The Manhattan distance is another metric often used [5]. With hierarchical clustering, it is possible to use the differences instead of the similarities to merge the clusters into a single cluster. It is known as Divisive clustering. With probabilistic clustering data points are put into clusters based on the probability that they belong to this cluster. Common clustering algorithms of the different clustering types are K-means, Fuzzy K-means, and Gaussian Mixture.

- Association is a method that finds relationships between the input data. Association is often used for marketing purposes to find the relationship between products and thus, to propose other products based on customer purchases. A common association algorithm is the Apriori algorithm [7].

- Dimensionality reduction is a method used to reduce the number of features in a dataset to make it easier to process. Nowadays, due to the growing size of the dataset that is used, it is common to use a dimensionality reduction method before applying a specific machine learning algorithm to a dataset. There are two main methods to reduce the dimensionality of a dataset. The first one is feature elimination. It consists of removing features that will not be useful for the prediction that researchers want to make. The second one is feature extraction. Feature extraction will create the same number of features that already exist in the dataset. These new features are a combination of the old features. These features are independent and ordered by importance. Thus, researchers can remove the least important new features. The advantage of this approach is that researchers reduce the dataset while keeping the important part of each original feature. One of the most used methods for feature extraction is Principal Component Analysis (PCA) [8].

One of the main advantages of unsupervised learning is that it does not require human intervention. In addition, unsupervised learning helps to discover links or differences in large datasets quicker than manual analysis. Thus, unsupervised learning can be useful to find an unknown anomaly in large traffic if used to improve IDSs.

However, due to the high amount of data required to train the model, it requires high computational power and a lot of time. In addition, unsupervised learning has a higher risk of inaccurate results compared to supervised learning because it learns by itself. Thus, human intervention is often needed at the end of the training to verify whether the output variables are correct or not. This verification is also time-consuming.

2.3. Semi-Supervised Machine Learning

Finally, semi-supervised learning is the combination of both types of machine learning; supervised and unsupervised. With semi-supervised learning, only a part of the dataset is labelled. All the previous methods and techniques can be applied to this dataset.

One of the main advantages of semi-supervised learning is that it allows using techniques and algorithms from both machine learning types. Thus, new machine learning algorithms can be created to reach a better accuracy. In addition, semi-supervised learning is less time consuming since it does not need to use an entire set of labelled data. However, semi-supervised learning also has the disadvantages of both above techniques.

3. Datasets

To train and test their models, the researchers used datasets. Following is a discussion of the most known and used datasets for Intrusion Detection System training and testing.

3.1. KDDcup99

The KDDcup99 dataset has been one of the most widely used datasets to assess IDS. It is based on the DARPA'98 dataset. The KDDcup99 contains approximately 4,900,000 samples. Each sample has 41 features and is labelled as Normal or Attack. The attack samples are classified into four categories: Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L), and Probe. There are three different datasets for KDDcup99, the first one is the whole dataset, the second corresponds to 10% of the whole dataset the third one is a test dataset which contains 311,029 samples. One of the main disadvantages of this dataset is that it is imbalanced, i.e., many samples are like each other for major classes such as DoS and Probe whereas for R2L and U2R there are few. Depending on which part of the dataset is used some classes might be completely absent [9].

3.2. Kyoto 2006

This dataset was created by deploying honeypots, darknet sensors, email servers, web crawlers, and other network security measures outside and inside Kyoto University to collect various types of traffic. Based on the 41 features from the KDDcup99 dataset, they extracted 14 statistical features. In addition, they also extracted 10 additional features to form the dataset, thus, each sample has 24 features. The most recent version of the Kyoto dataset includes traffic from 2006 to 2015 [10].

3.3. NSL-KDD

This dataset was created to fix the main issue of the KDDcup99 dataset. It was proposed in 2009 by Tavallaee et al. [9]. It keeps the four attack categories of the KDDcup99. The NSL-KDD proposes two files, a training set, and a testing set. The training set is made of 21 different attacks and has 126,620 instances. The testing set is made of 37 different attacks and has 22,850 instances [10].

3.4. UNSW-NB15

This dataset was created by the Australian Centre for Cyber Security. It was created to generate traffic which is a hybrid of normal activities and attack behaviours. This dataset has nine types of attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. The UNSW proposes two files, a training set, and a testing set. These files contain records from different types of traffic, attacks and normal, from the original dataset. The original dataset has a number of records of 2,540,044 while, the training set has 175,341 records, and the testing set has 82,332 records [11].

3.5. CICIDS2017

This dataset was created by the Canadian Institute for Cybersecurity (CIC) in 2017. This dataset was built using real-world traffic containing both normal and recent attack samples. The results were analyzed based on the time stamp, source, and destination IP, protocols, and attacks using CICFlowMeter. In addition, they implemented common attacks such as Brute Force FTP, Brute Force SSH, Denial of Service (DoS), HeartBleed, Web Attack, Infiltration, Botnet and Distributed Denial of Service (DDoS) [12].

4. Literature Review

The solution proposed by Lirim et al. [13] used a Convolutional Neural Network (CNN) with a multi-layer perceptron for its model. Multi-layer perceptron can be considered as a fully connected network where a neuron corresponds to one layer and is connected to all neurons in the next layers. For neural networks, a CNN is composed of an input layer, hidden layers, and an output layer. Contrary to a traditional neural network, in one of its hidden layers, CNN uses a mathematical operation called a convolution instead of using a multiplication matrix. In CNN, the input is a tensor made of different parameters such as the number of inputs, the input height, width, and channels. The convolutional layer convolves the input and forwards the result to the next layer. However, the tensor size can grow tremendously after multiple convolutions. To tackle this issue, Lirim et al. [13] use padding to reduce the tensor dimension. They trained their model by optimizing the hyperparameters until a decrease in performance is met. Their final model uses ten classes (nine for attacks and one for normal traffic) and is made of multiple dual convolutional layers followed by a pooling and a dropout layer to avoid oversize. However, their model presents a class imbalance between the upper and bottom class which needs the use of bootstrapping to solve the issue. They tested their model on the pre-partitioned UNSW-NB15 dataset and on a user-defined dataset which corresponds to 30% of the whole dataset. They obtained, respectively an accuracy of 94.4% and 95.6% for both datasets.

Lin et al. [14] proposed another IDS based on CNN. Their solution is composed of two parts. The first one is offline training using CNN, where in their model, they start with an input layer of 9 × 9 and reduce it through successive convolutional layers and a maximum pooling layer to reach an output layer of 1 × 1. The second part of their system is the online detection phase, where they use Suricata, an open-source IDS, to catch the traffic. Then, the packets are pre-processed, and the trained model is used on the network traffic to produce the outcome of the detection. To test their model, they used the CICIDS2017 dataset. They tested it on the feature dataset and the raw traffic dataset. They obtained, respectively an accuracy of 96.55% and 99.56%, showing that their model is better with raw traffic than with an extracted feature set.

Rohit et al. [15] proposed an ensemble approach to detect intrusion. They perform three tests to show how their approach proposed better results. They first performed normalization on the KDD Cup99 dataset, then, they use a correlation method to perform feature selection. The feature selection used information gain as a decision factor, and finally, they use an ensemble approach combining three algorithms: Naïve Bayes, PART, and Adaptive Boost. The result of each algorithm is then compared, and the average of the results or most voting results is used to decide the outcome. In addition, they use the bagging method to reduce the variance error. They obtained an accuracy of 99.9732% on the KDD Cup99 dataset using their solution.

Al-Yaseen et al. [16], proposed a new model for intrusion detection systems, using a hybrid multi-level model combining SVM (Support Vector Machine) and ELM (Extreme Learning Machine). In their model there are five levels, the first level distinguishes the traffic into DoS or Other. The second level distinguishes the previous unknown traffic into Probe or Other. The third distinguishes the previous unknown traffic into User to Root attack (U2R) or Other, and, the fourth level distinguishes the previous unknown traffic into Remote to Local attacks (R2L) or Other. Finally, the previous unknown traffic is distinguished between normal or unknown traffic in the fifth level. R2L and U2R are placed at the bottom level because they are similar to normal connections. At each level, a classifier is used. Their model is composed of 4 SVM classifiers at levels 1, 3, 4, and 5 and of 1 ELM classifier at level 2. They choose to use an ELM classifier to detect Probe because ELM has shown better results than SVM. After pre-processing the training set from the KDD dataset, they performed a modified K-means for feature extraction to have the 5 different categories that their solution can detect. Using their solution, they obtained an accuracy of 95.75%, which is slightly better than if they only used multi-level SVM (95.57%). In addition their hybrid model has a lower false alarm rate, 1.87%, compared to multi-level SVM at 2.17%.

Kanimozhi et al. [17], proposed a solution using oppositional tunicate fuzzy C-mean for detecting cloud intrusions. In their model, they first pre-processed the data and performed a normalization to have two datasets, one for training and one for testing. They performed a feature selection using logistic regression to keep the more relevant features, and they used the OPTSA and FCM clustering model. The dataset is split into C clusters using the fuzzy C-means algorithm. Once the data is clustered, they performed a cluster expansion and integration to reduce redundant clusters. They tested their solution on different datasets such as CICIDS2017 and obtained an accuracy of 80%.

Yiping et al. [18] created an intrusion detection system for wireless networks based on the random forest algorithm. They first created a signal detection model to catch the important features of signals, then, they created the model to detect malicious nonlinear scrambling intrusion signals. An improved random forest algorithm was used to extract the spectral features of the malicious signal, and then, optimal detection of malicious traffic in a wireless network was performed using a reinforcement learning method and static feature fusion. They obtained a mean accuracy of 96.93%.

Jabez et al. [19] created a system using an outlier detection approach to detect unknown attacks. The outlier detection approach is based on identifying data points that are isolated from clustered points. This approach uses the neighbourhood outlier factor to detect points that are not close to each other. They trialled their solution on the KDDcup99 datasets. In addition, the main advantage of their solution is its execution time which is significantly better compared to other solutions such as back propagation neural network which requires a lot of computing resources.

Kurniawan et al. [20] proposed an improved solution of Naïve Bayes for intrusion detection systems. The Naïve Bayes algorithm is based on the Bayes equation:

$$P(H|U) = \frac{P(U|H) * P(H)}{P(U)}$$

where:

- *U* is the data with an unknown class
- *H* is the hypothesis class of *U*
- *P*() is the Probability

The Naïve Bayes algorithm has an issue when one of the probabilities is 0 [21]. This results in its low accuracy when used. In their solution, Kurniawan et al. proposed two modifications to the Naïve Bayes algorithm. The first one is removing each variable that has a probability of 0. The second modification is to change the multiplication operation by an addition operation when the probability is 0. In their solution, they first realized a feature selection using the correlation-based features selection (CFS). Thus, the number of features was reduced from 41 to 10 features. They tested their two modifications on the NSL-KDD dataset, the second modification showed promising results with an accuracy of 89.33%.

Gu et al. [22] proposed a new solution to improve IDS using SVM. Their solution used Naïve Bayes algorithm to perform feature selection. Then, they trained the model with the transformed data from the feature selection. They tested their solution on the UNSW-NB15 and the CICIDS2017 datasets. Compared to using only the SVM classifier, the use of Naïve Bayes for feature extraction before using the SVM classifier shows better results. Indeed, they obtained an accuracy of 93.75% on the UNSW-NB15 dataset and an accuracy of 98.92% on the CICIDS2017 dataset. However, their solution only shows if there is an intrusion, it cannot be used to detect what kind of attack is in operation.

Pan et al. [23] conceived a solution to detect intrusion in wireless networks. Their solution was based in the cloud to have the maximum efficiency in terms of computational power. They used sink nodes based in the fog to lessen the burden on the cloud computing section. In order to have a solution as light as possible, they used a combination of Polymorphic Mutation (PM) and Compact SCA (CSCA), as CSCA helps to reduce the computing load by reducing the density of the data by using probability. They added Polymorphic Mutation to reduce the loss of precision when using CSCA. They used PM-CSCA to optimize the parameters of KNN algorithms to have the best configuration. They tested their solution on the NSL-KDD and UNSW-NB15 datasets. They, respectively obtained an accuracy of 99.327% and 98.27%.

Xiao et al. [24], proposed a solution based on CNN. They first performed feature extraction using both Principal Component Analysis (PCA) and Auto-Encoder (AE). Auto-Encoder is a dimension reduction method using several hidden layers of neural networks to remove insignificant data. Then, they transformed the dimension of the data from one into a two-dimensional matrix and forwarded it to the CNN model to train it. The model is trained and improved using back propagation algorithms. They tested their model on the KDDcup99 and obtained an overall accuracy of 94%. They compared their model with DNN and RNN models and got slightly better results. However, their model has a low detection rate of U2R and R2L which are not represented enough in the dataset.

Zhang et al. [25], proposed a multi-layer model to detect attacks. Their solution combined two machine learning techniques: CNN and GcForest. The GcForest is a random forest technique which generates a cascade structure of decision trees. Their model is composed of two main parts. In the first part, they run a CNN algorithm to detect different kinds of attacks and normal traffic from the input data. Their CNN algorithm is an improved model of GoogLeNet called GoogLeNetNP. The second part consists of using a deep forest model to create more subclasses of the attacks. This second layer improves the precision of their solution by classifying the abnormal classes into N-1 subclasses. The second layer uses the cascade principle of gcForest but instead of the random forest, it uses XGBoost. XGBoost is like a random forest, however, the construction of the trees is done one after another until the objective function is optimized. They tested their solution on a combination of the UNSW-NB15 and CICIDS2017 datasets. They obtained an overall accuracy of 99.24% which is better compared to the algorithms used singularly.

Yu et al. [26], proposed an IDS model based on Few-Shot Learning (FSL). FSL is a deep learning method that can learn from a small amount of data. In their solution they used two embedding models, CNN and DNN, to perform feature extraction. Those models help to reduce the dimension of the input data without losing important information. They tested their model on the UNSW-NB15 and NSL-KDD datasets. Their solution obtained, respectively an accuracy of 92.34% and 92%.

Gao et al. [27], proposed an ensemble machine learning IDS. They used the Principal Component Analysis method for feature extraction. After different tests on the NSL-KDD datasets, their ensemble algorithm is combining Decision Tree, Random Forest, KNN, DNN and MultiTree. The results of the ensemble algorithm are made by a majority vote using weights for each algorithm to have better accuracy. They obtained an accuracy of 85.2% which is better than the accuracy if they were only using one algorithm. However, their model lacks efficiency when analyzing attacks that are not in large quantity.

Marir et al. [28], proposed a solution using a Deep Belief Network (DBN) and an ensemble method composed of multiple SVMs. A DBN is a succession of unsupervised networks such as Restricted Boltzmann Machines (RBM). An RBM is composed of an input and a hidden layer where the nodes are connected to the previous and next layers but are not connected within their layer. DBN uses an unsupervised pre-training based on the greedy layer-wise structure. Then, they use a supervised fine-tuning approach to learn the important features. In their solution, they use DBN for feature extraction. Then, the extracted features are forwarded to the multi-layer ensemble SVM. The output is generated by a voting algorithm. They tested their solution on KDDcup99, NSL-KDD, UNSW-NB15, and CICIDS2017 datasets. They, respectively obtained a precision of 94.76%, 97.27%, 90.47% and 90.40%. However, it was shown that when more layers are used their solution is more time consuming.

Wei et al. [29], improved the performance of DBN for IDS by using an optimizing algorithm. To optimize their model, they used a combination of Particle Swarm Optimization (PSO), Artificial Fish Swam Algorithm (AFSA), and Genetic Algorithm (GA). The

PSO is first optimized using AFSA. Then, GA is used to find the global optimal solution of the initial particle search. The optimal solution is then used in the DBN model to improve its accuracy. They tested their solution on the NSL-KDD dataset and obtained an accuracy of 82.36%.

Vinayakumar et al. [30], proposed an IDS based on Deep Neural Network (DNN). Their DNN architecture is made of an input layer, five hidden layers and an output layer. Their solution is scalable, and it is possible to use between one and five hidden layers in the DNN models. They used the Apache Spark computing platform. Their solution can work in both cases, HIDS and NIDS. For NIDS, they tested their solution on KDDcup99, NSL-KDD, Kyoto, UNSW-NB15 and CICIDS2017 datasets. They, respectively obtained an overall accuracy of 93%, 79.42%, 87.78%, 76.48% and 94.5% when combining the accuracy for each number of DNN layers.

Shone et al. [31] proposed a solution combining Non-symmetric Deep Auto-Encoder (NDAE) and Random Forest. Usually, an auto-encoder uses the symmetric scheme from encoder-decoder, however, in their solution, they only used the encoding phase. It reduces the computational time without impacting too much on the accuracy of the IDS. To handle complex datasets, they choose to stack their NDAE. However, they discovered that using only NDAE was not enough to have an accurate classification. Therefore, they added Random Forest as their classifier after performing feature extraction using two NDAE with three hidden layers each. They tested their solution on the KDDcup99 and NSL-KDD datasets and compared it to a DBN solution. They obtained, respectively a total accuracy of 97.85% and 85.42%. However, their solution struggles to detect small classes such as R2L and U2R.

Yan et al. [32], showed the impact of feature extraction using a Stacked Sparse Auto-Encoder (SSAE) to improve IDS. A sparse auto-encoder is an autoencoder which uses a sparsity penalty, usually, the penalty is activated when hidden nodes are used. Thus, using a sparse auto-encoder reduces the number of hidden nodes used. Stacked sparse auto-encoder is the addition of further sparse auto-encoders. It allows for reducing the dimension of the input data without losing significant information. To optimize their SSAE they used the error back propagation method, and, to test their SSAE model they used the NSL-KDD dataset. They used different classifiers with and without their SSAE model to show how much the use of SSAE for feature extraction improves the accuracy. The best accuracy was obtained when the SSAE and SVM classifiers were combined. They reached an overall accuracy of 99.35%. One of the main advantages of using their solution is the large time reduction for training and testing, approximately a tenth of the time of other solutions is needed. However, the detection rate for R2L and U2R is lower compared to the other classes.

Khan et al. [33], proposed a two-stage deep learning model (TSDL) to improve IDS. In the first stage, they classify the traffic as normal or abnormal with a probability value. In the second stage they used this value as an additional feature to train the classifier, they used a DNN approach for both stages, where they used a Deep stacked auto-encoder (DSAE) for feature extraction and Soft-max as a classifier. Soft-max is often used in a neural network for multi-class classification problems. They tested their solution on the KDDcup99 and UNSW-NB15 datasets. They, respectively obtained an overall accuracy of 99.996% and 89.134%.

Andresini et al. [34], proposed a solution combining an unsupervised approach with two auto-encoders and a supervised stage to build the datasets. They trained the two auto-encoders separately using normal and attack traffic. Then, the auto-encoders reconstruct those samples and add them to the dataset that is used to train the model. The dataset goes through a one-dimension CNN. This is done to see the impact of one channel on the other to have a better distinction between the two classes: normal and attack. Finally, they used a Soft-max classifier to identify if the data was an attack or normal. They tested their model on KDDcup99, UNSW-NB15 and CICIDS2017 datasets. They, respectively obtained an overall accuracy of

92.49%, 93.40% and 97.90%. One of the drawbacks of their solution is that it does not provide details about the different types of attacks.

Ali et al. [35], proposed a model using Fast Learning Network (FLN) based on particle swarm optimization (PSO). They used PSO to improve the accuracy of FLN which can be inefficient due to the weights used in the neural network. They tested their solution on the KDDcup99 dataset against other FLN solutions. They obtained a better accuracy to detect the different classes than the other solutions. They achieved an overall accuracy of 89.23%. However, their overall accuracy is decreased by their low accuracy when identifying one of the small classes of attack (R2L).

Dong et al. [36], proposed a hybrid solution combining clustering with SVM. In their solution, they first used K-means clustering to process the data and divided it into different subsets. Then, they used SVM on each of those subsets. They tested their solution on the NSL-KDD datasets and they obtained an overall accuracy of 99.45%. In addition, compared to other methods their solution improved the detection rate. Their solution also requires less time processing compared to SVM algorithms using different parameters. However, the authors provided no information concerning the accuracy of each attack classification.

Wisanwanichthan et al. [37], proposed a Double-Layered Hybrid Approach (DLHA). In their solution, they first create two groups in the NSL-KDD dataset. The first one contains all classes and the second one contains only the U2R, R2L and normal classes. They created these two groups to have better accuracy of the U2R and R2L classes which are often the weakness of most of the IDS solutions that researchers have seen. Then, they performed feature extraction in both groups. They first used Intersectional Correlated Feature Selection (ICFS). In ICFS, the Pearson Correlation Coefficient (PCC) is used to select important features between two random variables. PCC can determine how much two variables vary from each other. Once ICFS is done, they performed Principal Component Analysis (PCA) to reduce the dimension of the data. Finally, to have a ratio of 1:1 between attacks and normal data in the second group they randomly choose the same amount of data as R2L and U2R combined. Then, they used those two groups to train their model which is composed of a first layer using Naïve Bayes classifier and a second layer using SVM. The first layer is used only to detect DoS and Probe. If the outcome is not one of those two classes, then the data goes through the second layer to detect if it is a R2L, U2R or Normal data. They tested their solution on the NSL-KDD dataset. They obtained an overall accuracy of 93.11% and detection of 96.67% for R2L and 100% for U2R classes. Their solution outperformed other solutions when identifying the small classes, however, contrary to other efficient solutions, their accuracy for the large classes was not as good.

Elhefnawy et al. [38] proposed a Hybrid Nested Genetic-Fuzzy Algorithm (HNGFA) to detect attacks. They first performed feature selection using Naïve Bayes. Major features and Minor features are split into two groups. Their model is composed of two genetic-fuzzy algorithms. The first one is the Outer Genetic-Fuzzy Algorithm (OGFA) and the second one is the Inner Genetic-Fuzzy Algorithm (IGFA). Each of these algorithms used two nested genetic algorithms. The outer one is used for the fuzzy sets and the inner one is used for the fuzzy rules. The OGFA is used for classifying data with major features, whereas the IGFA is used for classifying data with minor features. The two genetic-fuzzy algorithms interact with each other to develop new solutions to have better accuracy. The goal is to make the interaction between the best results of the OGFA with weak results from the IGFA to have the best model possible. They tested their solution on the KDDcup99 and UNSW-NB15 datasets and they obtained an overall accuracy of 98.19% and 80.54%, respectively. In addition, their solution got a good accuracy for detecting small classes such as R2L and U2R. However, due to the complexity of their model, the training time is high.

References

1. Checkpoint. What Is an Intrusion Detection System? Available online: https://www.checkpoint.com/cyber-hub/network-security/what-is-an-intrusion-detection-system-ids/ (accessed on 19 May 2022).
2. IBM Cloud Education. Supervised Learning. Available online: https://www.ibm.com/cloud/learn/supervised-learning (accessed on 19 May 2022).
3. Seldon. Machine Learning Regression Explained. Available online: https://www.seldon.io/machine-learning-regression-explained (accessed on 19 May 2022).
4. Terence, S. All Machine Learning Models Explained in 6 Minutes. Available online: https://www.ibm.com/cloud/learn/unsupervised-learning (accessed on 19 May 2022).
5. IBM Cloud Education. Unsupervised Learning. Available online: https://www.ibm.com/cloud/learn/unsupervised-learning (accessed on 19 May 2022).
6. Ben n'cir, C.-E.; Cleuziou, G.; Nadia, E. Overview of Overlapping Partitional Clustering Methods. In Partitional Clustering Algorithms; Springer: Cham, Switzerland, 2015; pp. 245–275.
7. Joos, K. The Apriori Algorithm. Available online: https://towardsdatascience.com/the-apriori-algorithm-5da3db9aea95 (accessed on 22 June 2022).
8. Matt, B. A One-Stop Shop for Principal Component Analysis. Available online: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c (accessed on 28 June 2022).
9. Ashiku, L.; Dagli, C. Network Intrusion Detection System using Deep Learning. Procedia Comput. Sci. 2021, 185, 239–247.
10. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
11. Protic, D. Review of KDD Cup '99, NSL-KDD and Kyoto 2006+ Datasets. Vojnoteh. Glas. 2018, 66, 580–596.
12. Moustafa, N. The UNSW-NB15 Dataset. Available online: https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed on 28 February 2022).
13. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization; Canadian Institute for Cybersecurity (CIC): Fredericton, NB, Canada, 2018; pp. 108–116.
14. Chen, L.; Kuang, X.; Xu, A.; Suo, S.; Yang, Y. A Novel Network Intrusion Detection System Based on CNN. In Proceedings of the 2020 Eighth International Conference on Advanced Cloud and Big Data (CBD), Taiyuan, China, 5–6 December 2020; pp. 243–247.
15. Gautam, R.K.S.; Doegar, E.A. An Ensemble Approach for Intrusion Detection System Using Machine Learning Algorithms. In Proceedings of the 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 11–12 January 2018; pp. 14–15.

16. Al-Yaseen, W.L.; Othman, Z.A.; Nazri, M.Z.A. Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. Expert Syst. Appl. 2017, 67, 296–303.
17. Kanimozhi, P.; Victoire, T.A.A. Oppositional tunicate fuzzy C-means algorithm and logistic regression for intrusion detection on cloud. Concurr. Comput. Pract. Exp. 2022, 34, e6624.
18. Chen, Y.; Yuan, F. Dynamic detection of malicious intrusion in wireless network based on improved random forest algorithm. In Proceedings of the 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), Dalian, China, 14–16 April 2022; pp. 27–32.
19. Jabez, J.; Muthukumar, B. Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach. Procedia Comput. Sci. 2015, 48, 338–346.
20. Kurniawan, Y.; Razi, F.; Nofiyati, N.; Wijayanto, B.; Hidayat, M. Naive Bayes modification for intrusion detection system classification with zero probability. Bull. Electr. Eng. Inform. 2021, 10, 2751–2758.
21. Chauhan, N. Naïve Bayes Algorithm: Everything You Need to Know. Available online: https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html#:~:text=One%20of%20the%20disadvantages%20of,all%20the%20probabilities%20are%20multiplie (accessed on 9 September 2022).
22. Gu, J.; Lu, S. An effective intrusion detection approach using SVM with naïve Bayes feature embedding. Comput. Secur. 2021, 103, 102158.
23. Pan, J.-S.; Fan, F.; Chu, S.C.; Zhao, H.; Liu, G. A Lightweight Intelligent Intrusion Detection Model for Wireless Sensor Networks. Secur. Commun. Networks 2021, 2021, 1–15.
24. Xiao, Y.; Xing, C.; Zhang, T.; Zhao, Z. An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks. IEEE Access 2019, 7, 42210–42219.
25. Zhang, X.; Chen, J.; Zhou, Y.; Han, L.; Lin, J. A Multiple-Layer Representation Learning Model for Network-Based Attack Detection. IEEE Access 2019, 7, 91992–92008.
26. Yu, Y.; Bian, N. An Intrusion Detection Method Using Few-Shot Learning. IEEE Access 2020, 8, 49730–49740.
27. Gao, X.; Shan, C.; Hu, C.; Niu, Z.; Liu, Z. An Adaptive Ensemble Machine Learning Model for Intrusion Detection. IEEE Access 2019, 7, 82512–82521.
28. Marir, N.; Wang, H.; Feng, G.; Li, B.; Jia, M. Distributed Abnormal Behavior Detection Approach Based on Deep Belief Network and Ensemble SVM Using Spark. IEEE Access 2018, 6, 59657–59671.
29. Wei, P.; Li, Y.; Zhang, Z.; Hu, T.; Li, Z.; Liu, D. An Optimization Method for Intrusion Detection Classification Model Based on Deep Belief Network. IEEE Access 2019, 7, 87593–87605.
30. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep Learning Approach for Intelligent Intrusion Detection System. IEEE Access 2019, 7, 41525–41550.
31. Shone, N.; Ngoc, T.N.; Phai, V.D.; Shi, Q. A Deep Learning Approach to Network Intrusion Detection. IEEE Trans. Emerg. Top. Comput. Intell. 2018, 2, 41–50.

32. Yan, B.; Han, G. Effective Feature Extraction via Stacked Sparse Autoencoder to Improve Intrusion Detection System. IEEE Access 2018, 6, 41238–41248.
33. Khan, F.A.; Gumaei, A.; Derhab, A.; Hussain, A. A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection. IEEE Access 2019, 7, 30373–30385.
34. Andresini, G.; Appice, A.; Mauro, N.D.; Loglisci, C.; Malerba, D. Multi-Channel Deep Feature Learning for Intrusion Detection. IEEE Access 2020, 8, 53346–53359.
35. Ali, M.H.; Mohammed, B.A.D.A.; Ismail, A.; Zolkipli, M.F. A New Intrusion Detection System Based on Fast Learning Network and Particle Swarm Optimization. IEEE Access 2018, 6, 20255–20261.
36. Liang, D.; Liu, Q.; Zhao, B.; Zhu, Z.; Liu, D. A Clustering-SVM Ensemble Method for Intrusion Detection System. In Proceedings of the 2019 8th International Symposium on Next Generation Electronics (ISNE), Zhengzhou, China, 9–10 October 2019; pp. 1–3.
37. Wisanwanichthan, T.; Thammawichai, M. A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM. IEEE Access 2021, 9, 138432–138450.
38. Elhefnawy, R.; Abounaser, H.; Badr, A. A Hybrid Nested Genetic-Fuzzy Algorithm Framework for Intrusion Detection and Attacks. IEEE Access 2020, 8, 98218–98233.