NJESR/December-2022/Volume-4/Issue-12 E-ISSN-2582-5836
DOI-10.53571/NJESR.2022.4.12.88-106

An Analysis On The Use Of Deep Learning In Cyber Security

Jose James[*1], Dipu Jose[*2]
Department Of Computer Engineering[*1&*2]
Government Polytechnic College
Muttom[*1], Nedumangadu[*2]

(Received:10November2022/Revised:20November2022/Accepted:30November2022/Published:30December2022)

Abstract

Until now, cutting-edge Deep Learning (DL) techniques have been widely used in speech recognition and image processing. In a similar vein, there has been some work done by DL on cybersecurity. In this survey, we focus on intrusion detection, malware detection, phishing/spam detection, and website defacement detection, all of which have recently been proposed as DL methods in cybersecurity. The most common DL models and algorithms are first briefly defined. The four main modules of a general DL framework for cybersecurity applications are then presented and discussed. The focus area, methodology, model applicability, and feature granularity are all examined in conjunction with related papers, which are then summarized and analysed. The possible research topics that could be taken into consideration to improve various cybersecurity applications using DL models are discussed in the concluding remarks and future work. A literature review of deep learning (DL) applications for cyber security is the subject of this survey paper. Deep auto encoders, restricted Boltzmann machines, recurrent neural networks, generative adversarial networks, and a number of other DL techniques are all briefly described in this tutorial. Then, we talk about how security applications use each of the DL methods. Malware, spam, insider threats, network intrusions, false data injection, and malicious domain names used by botnets are just some of the attack types we cover.

Keywords: Deep learning, Neural Networks, Digital Analytics, Profound Learning, Profound Brain Organizations, Auto Encoders In Depth, Profound Conviction Organizations, Limitations On Boltzmann Machines, Cyber Security Applications

Introduction

Since the Internet is now a part of everyone's life, it's vulnerable to a variety of threats because of its widespread interconnectedness. Cyberspace is home to a variety of security risks, including jailbreaking, two-faced malware, network intrusions, and more. A security arms race has developed as a result of these threats. In order to safeguard computers, networks, and software applications from malware infections and network intrusions, numerous security firms around

the world are concentrating on the development of novel technologies. The underlying network and computers are safeguarded from unauthorized access, destruction, malfunction, and modification by two conventional security systems known as host security systems and network security systems, respectively. Firewalls, Intrusion Detection Systems (IDSs), and antiviruses are examples of integrated security modules that aid in monitoring a system or network and issue an alert in the event of malicious activity. In order to deal with network attacks and identify malicious activities in computer network traffic, it is believed that intrusion detection is a necessary security mechanism. It aids in the discovery, determination, and identification of unauthorized use, duplication, alteration, and destruction of information and information systems [1]. It plays a crucial role in information security technology.

In general, IDSs can be broken down into three categories: hybrid, anomaly detection, and misuse detection Pre-defined malicious activity signatures are used by misuse detection techniques to identify intrusions. As a result, they are only used to detect known attacks. Anomaly detection methods, on the other hand, identify malicious activities based on deviations from normal patterns and define normal patterns. Thus, zero-day attacks can be detected by anomaly-based detection techniques. Hybrid techniques make use of both methods for detecting misuse and anomalies. Hybrid approaches aim to increase detection rates of known intrusions while reducing false positives from unknown attacks[2].

The detection of malware is of the utmost importance due to the serious security issues and threats it has recently posed to Internet users. Fraudsters have taken advantage of these intrusive software programs, such as worms, viruses, Trojans, botnets, ransomware, and so on, to carry out a variety of security attacks on computer systems. These attacks put the confidentiality and integrity of the data that is communicated in question, as well as the availability of the services that are provided by the underlying infrastructure, in jeopardy.

To protect computers and legitimate users from malware attacks, numerous vendors, such as Kaspersky, Symantec, Microsoft, McAfee, and Invincea, have developed anti-virus products. Malware is typically detected by these vendors using signature-based techniques. Although signature-based methods are effective in some ways, they are unable to identify zero-day malware, which may be disguised by malware authors. The phenomenal volume of everyday malware creation requires concocting precise robotized frameworks to identify and characterize malware.

Due to their numerous advantages over other traditional Machine Learning (ML) methods, DL algorithms have established a crucial role in solving complex problems with the rapid expansion of applications in areas such as image recognition, natural language processing, bioinformatics, speech recognition, and bioinformatics. DL is characterized as a few multifaceted ML calculations that are strong at learning significant level deliberations of mind boggling enormous scope information. Typically, DL algorithms use a lot of non-linear hidden layers to learn feature representations, making feature engineering automatic. As a result, when new forms of malware or network attacks emerge, there is no need to spend more time or money hiring engineers to re-engineer the features. Different cybersecurity businesses can update their IDSs and malware detection systems with DL at no cost.

Several DL models that have been applied to the areas of intrusion detection, malware detection, phishing/spam detection, and website defacement detection in the literature are the subject of this survey, which examines the application of DL to cybersecurity. Supposedly, this is the primary overview that presents a nitty gritty writing survey of network safety underlining notable DL calculation portrayals. Although some research on ML and Data Mining (DM) techniques for malware or intrusion detection has been done[2, 3, 4, 5], only a few of these studies provide an overview of DL methods for intrusion detection[5,6], and there is no survey on DL techniques for malware or phishing detection.

Shallow Learning Vs. Deep Learning

Artificial neural networks (ANNs) are AI calculations propelled by the focal sensory system. When McCulloch and Pitts [1] presented a mathematical model based on a biological neuron in a 1943 study, they were first thought of. Hebb[2] and Rosenblatt[3] later put this into practice when they developed supervised learning through the creation of perceptron and unsupervised learning through self-organized learning, respectively. They have a few layers of neurons connected by adaptive weights (Figure 1), and the connections between the layers of the network that are next to them are typically complete. According to the universal approximation theorem for ANNs, a multi-layer perceptron (a type of ANN) with only one hidden layer can approximate any continuous function that maps real number intervals to some real number output interval. Because of this, most of the early research on ANNs focused on networks trained through back-propagation with just one hidden layer because an ANN with one hidden layer can produce any non-linear continuous function[4]. The term "shallow learning" refers to networks with only one

hidden layer. There are shallow network architectures with and without supervised supervision. To learn a task, supervised learning uses labels (ground truth); Performing a machine learning task without labels is known as unsupervised learning. Feature extraction is performed separately rather than as part of the network in shallow learning.

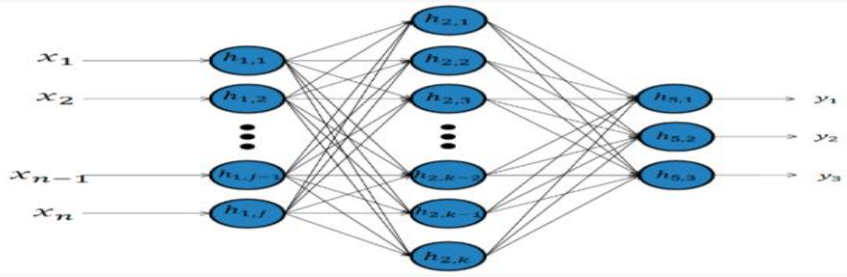

Figure 1. Shallow Neural Network.

The first computer implementation of DL occurred in 2006[9], making it a much more recent endeavour. There are numerous meanings of DL and profound brain organizations (DNNs). According to a straightforward definition, DL is a collection of machine learning algorithms that attempt to learn at multiple levels, which correspond to various levels of abstraction (Figure 2). The levels compare to unmistakable degrees of ideas, where more elevated level ideas are characterized from lower-level ones, and a similar lower-level ideas can assist with characterizing numerous more significant level ideas[5]. The first few layers of the deep network are responsible for performing feature extraction. DL architectures can be unsupervised, supervised, or hybrid. Because they only have one hidden layer, shallow neural networks can't learn the higher-level concepts that deep neural networks can because they can't do advanced feature extraction. This is also true for other algorithms used in machine learning. However, in order to train DL models in a reasonable amount of time, DL methods necessitate greater computational power—sometimes multiple graphical processing units (GPUs). A common person can now easily create DL models thanks to two developments. The first is the increased availability of graphics processing units (GPUs), which make computation significantly faster. The second is that a DL model's layers can be trained separately from one another[6]. As a result, optimizing a large model with millions of parameters can be done in smaller, more manageable steps that use significantly less resources.

The number of hidden layers is the primary distinction between shallow and deep networks; While shallow neural networks only have one hidden layer, DL architectures have multiple

hidden layers. A weighted sum of the units in the layer before it is applied to a nonlinear function in an ANN or DNN. There are a variety of nonlinear functions that can be utilized; However, the rectified linear unit (ReLU), which is simply f(z) = max(z.0), and the sigmoid function, softmax function, hyperbolic tangent function, are the most prevalent. ReLUs were first proposed in the 1970s[7], but they weren't widely used until 2009[8]. The various deep learning (DL) techniques utilized in cyber security are discussed in this section. For each technique, significant papers on methodology are cited.

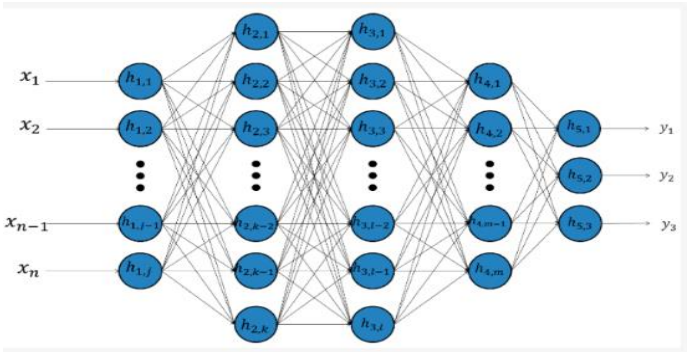

Figure 2. Deep Neural Network.

Profound Conviction Organizations

An original paper by Hinton[9] presented Profound Conviction Organizations (DBNs). They belong to a class of DNNs that are made up of multiple layers of hidden units that connect the layers but not the units in each layer. Unsupervised training is used to train DBNs. They are typically trained by individually adjusting weights in each hidden layer to reconstruct the inputs.

Deep Auto Encoders

A type of unsupervised neural network, the deep auto encoder takes a vector as its input and tries to match its output to that vector. One can create a data representation with either a higher or lower dimensionality by taking the input, altering the dimensionality, and then reconstructing the input. These kinds of brain networks are amazingly flexible in light of the fact that they learn packed information encoding in a solo way. They can also be trained one layer at a time, which reduces the amount of computing power needed to create an efficient model. The network is utilized for data encoding (also known as feature compression) when the hidden layers have a dimensionality that is lower than that of the input and output layers (as shown in Figure 3). By training an auto encoder to reconstruct the input from a noisy version of the input (Figure 4), a denoising auto encoder, an auto encoder can be designed to remove noise and be more robust[9].

It has been demonstrated that this method is more robust and generalizable than standard auto encoders.

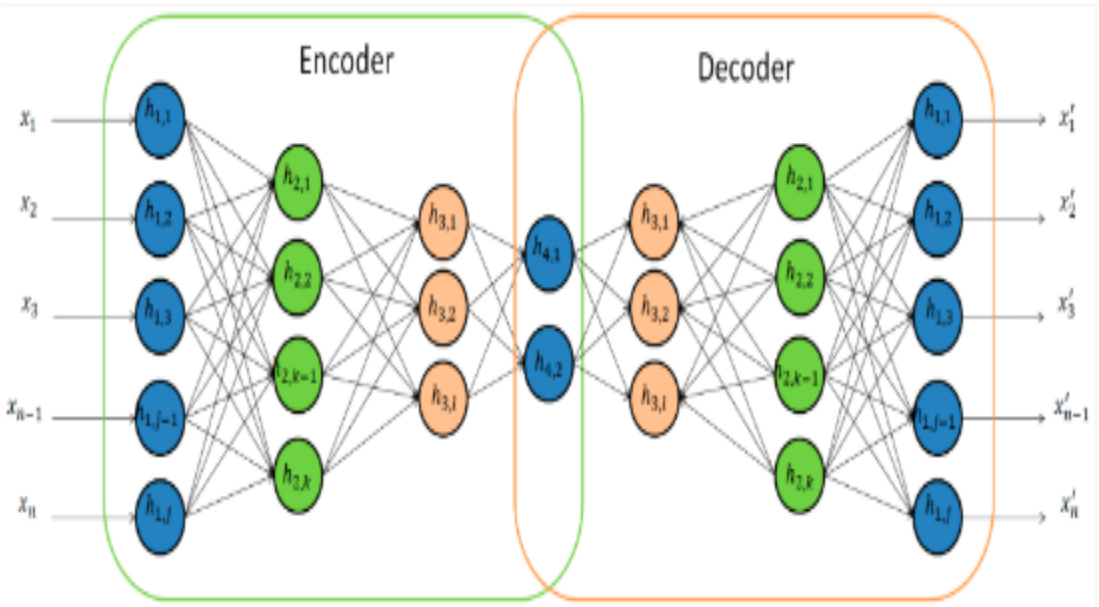

Figure 3. Deep Autoencoder.

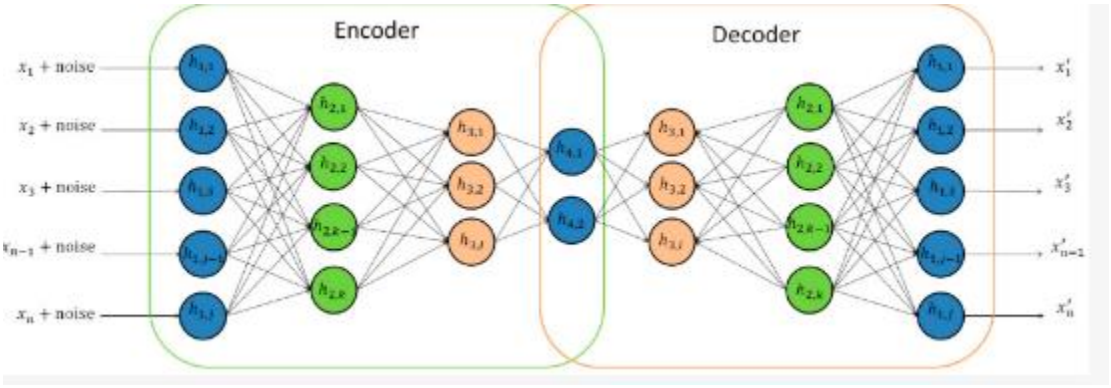

Figure 4. Denoising Auto Encoder.

Figure 5 depicts stacked auto encoders, which compress information by employing multiple layers of sequentially trained auto encoders[10]. The finished stacked auto encoder with a classification layer is shown in Figure 5a. It is made by making an auto encoder, as shown in Figure 5b. After that, the inputs from Figure 5b are used to construct the auto encoder shown in Figure 5c. A classification layer is added to these after they have been trained together. Denoisingauto encoders can also be stacked, just like regular auto encoders[9].

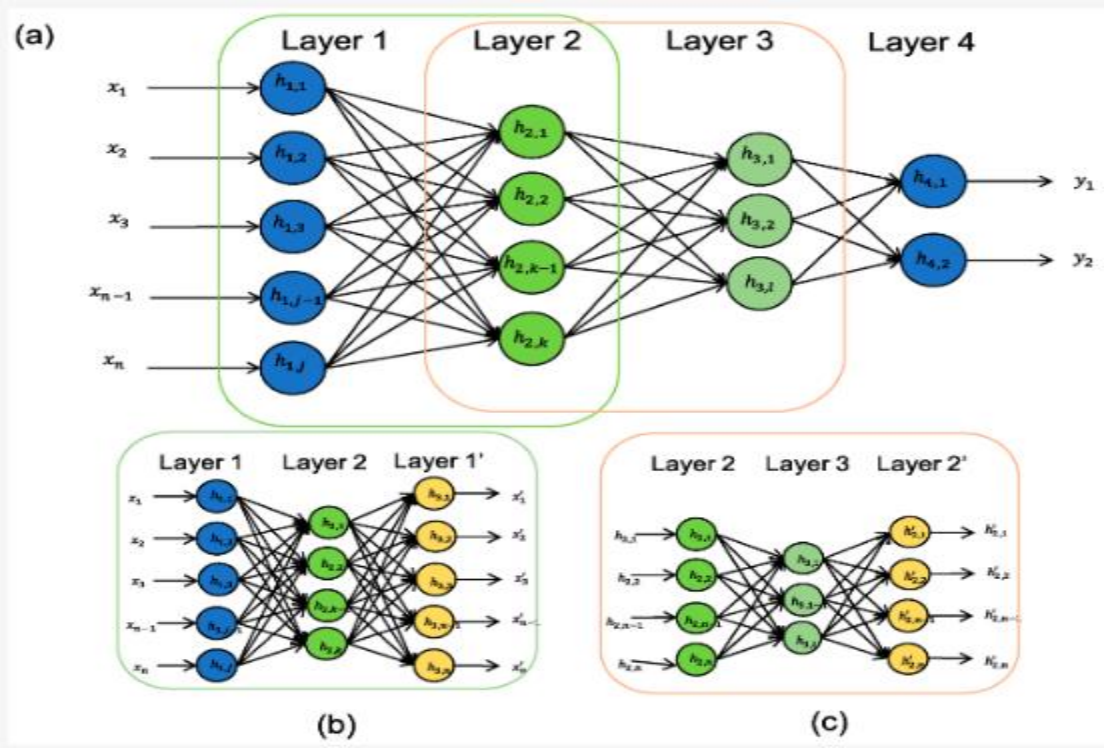

Figure 5. Stacked Auto Encoder With A Classification Layer. (**a**) Stacked Autoenoder; (**b**) Auto Encoder For Layer 2; (**c**) Auto Encoder For Layer 3.

A type of encoder called a sparse auto encoder has more hidden nodes than input and output layers, but only a portion of the hidden units are activated at once[11]. This is represented by punishing enacting extra hubs.

Restricted Boltzmann Machines

The fundamental components of DBNs are restricted Boltzmann machines (RBMs), which are two-layer, bipartite, undirected graphical models in which data can flow in both directions rather than just one[11]. RBMs, like auto encoders, can be trained one layer at a time and are unsupervised. The input layer occupies the first layer. The hidden layer is located in the second layer (Figure 6). There are no intra-layer associations (i.e., between hubs in a similar layer); However, full connectivity means that every node in the hidden layer is connected to every node in the input layer.

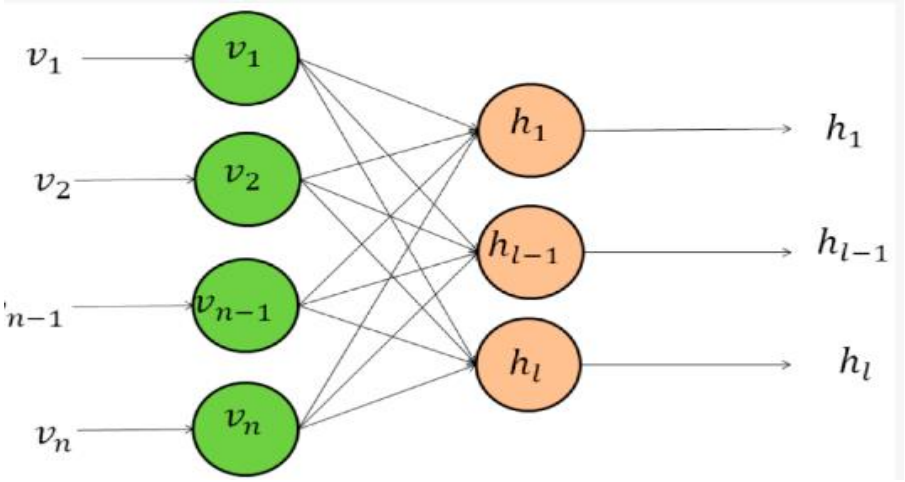

Figure 6. Restricted Boltzmann Machine.

In most cases, the input and hidden layers only use binary units for their units. Using a lot of math from statistical mechanics, the network is trained to minimize the "energy" function, which measures the compatibility of the model. The model is being trained to find the functions and, consequently, the hidden state that reduce the system's energy. Additionally, RBMs are probabilistic, which means that they assign probabilities rather than specific values. Notwithstanding, the result can be utilized as highlights for another model. By feeding binary input data forward through the model, the model is trained. After that, the model processes it in reverse to reassemble the input data. The weights are then updated by calculating the system's energy. Until the model converges, this procedure is repeated. Likewise, to auto encoders, RBMs can be stacked to frame various layers to make a more profound brain organization. Stackable RBMs are the name given to these.

Metrics

The authors of the papers cited in the following sections used a variety of classification metrics, which are described in this section. A model performing a binary classification task can be measured using a variety of metrics. These measurements incorporate exactness, accuracy, review, misleading positive rate, F1 Score, and region under the bend (AUC) and large numbers of the measurements have more than one name. These assessment measurements are gotten from the four qualities tracked down in the disarray grid (Table 1), which depends on the determined anticipated class versus the ground truth.

Table 1. Confusion matrix.

		Predicted Class	
		Malicious	**Benign**
Actual Class (ground truth)	Malicious	True Positive (TP)	False Negative (FN)
	Benign	False Positive (FP)	True Negative (TN)

Accuracy (acc) or Proportion Correct: the ratio of correctly classified examples to all items. The usefulness of accuracy is lower when the classes are unbalanced (i.e., there are a significantly larger number of examples from one class than from another). However, it does provide useful insight when the classes are balanced.

$$acc = \frac{TP + TN}{TP + TN + FP + FN}.$$

Positive Predictive Value (PPV) or Precision (p): The ratio of items correctly classified as class X to all items that were classified as class X.

True positive rate and precision are given equal weight in this particular version of the F-function.The AUC (area under the curve): The sum of the area under a receiver operating characteristic (ROC) curve, which is a plot of the false positive rate in comparison to the true positive rate that is produced by varying the classification thresholds.

Calculating accuracy for problems with multiple classes is simple; However, metrics like precision, recall, FPR, F1 Score, and AUC cannot be easily calculated (TP and TN do not exist for three-class problems, for example). Recall, precision, etc. can be determined for a problem with more than three classes by turning it into a two-class problem (i.e., all versus one), where the metrics for each class are calculated. For multiclass problems, accuracy is usually the only thing used.

It is essential to keep in mind that it is not possible to compare the developed models based on the accuracy or any other metrics they obtained because each of the subsequent papers uses a different dataset (or sometimes a different subset of a given dataset). If the authors of both publications used the same training dataset and testing dataset, then this comparison would be valid.

Cyber Security Datasets For Deep Learning

The Knowledge Discovery and Dissemination (KDD) 1999 dataset is one of the most widely used datasets for intrusion detection[12]. More than 4 million records of network traffic were compiled into this dataset for the 1999 KDD Cup competition. However, there is no raw traffic

data in the dataset. Instead, 41 features based on basic type, content type, and traffic type have been pre-processed into the raw pcap data. There are 22 distinct types of attacks in the dataset, which are divided into four families: probing, unauthorized access to local super-user privileges (U2R), unauthorized access from a remote machine (R2L), and denial of service (DoS). However, a review conducted by[13] revealed a number of issues with the dataset. The synthetic nature of the network and attack data, dropped data due to overflow, and ambiguous attack definitions were all major contributors to this. Additionally, the dataset was biased by a large number of redundant records. A new dataset known as NSL-KDD, which is another dataset that is frequently utilized for network intrusion detection, was proposed by[14] as a result of these flaws.

Raw packet data can be found in a few datasets; The CTU-13 dataset, on the other hand, is used the most frequently[15]. Raw pcap files for background, normal, and malicious data are included in this dataset. When compared to the KDD 1999 and NSL-KDD datasets, the advantage of raw pcap files is that individuals can perform their own pre-processing, allowing a wider range of algorithms to be utilized. Additionally, there is no simulated dataset in the CTU-13 dataset. There is ground truth, the unknown traffic originates from a large network, and the botnet attacks are real. It is a diverse dataset because it combines seven distinct botnet families and 13 distinct scenarios with varying numbers of computers.

For the purpose of detecting the domain generation algorithm (DGA), there are three significant sources of data. Although a variety of feature extraction methods have been described in previous ML papers, the DL algorithms almost entirely rely on domain names. This is reflected in that the three essential wellsprings of information for DGA identification are simply space names. Since the Alexa Top Sites[10] dataset contains as many as one million domain names, it is typically used as a source of benign domain names. OSINT[16] and DGArchive[12] provide the malicious domain names. The OSINT DGA feed from Bambenek Counseling was generally utilized in light of the fact that it contains DGA spaces from 50 distinct DGAs and incorporates in excess of 800 thousand pernicious area names. Alternately, more than 30 reverse-engineered DGAs can be accessed through DG Archive, which can be utilized to generate malicious domain names on an internal network.

Datasets with malware are much more prevalent. The most popular apps in the Google Play Store are the most common source of normal data in malware experiments. Despite not being

guaranteed to be malware-free, these apps are the most likely to be malware-free due to Google's vetting and their widespread availability. Additionally, the Virus Total service may be utilized to screen them at times. Malware can be found in a number of datasets, including Microsoft, Contagio, Comodo, the Genome Project, Virus Share, Virus Total, and DREBIN. Because that is where they obtained normal data, the datasets that contain malicious data and the data from the Google Play Store share some similarities.

Malware datasets are typically saved as raw program files. When it comes to feature extraction and processing, this makes for a tremendous amount of adaptability. There are 2123 applications in the Genome Project dataset, 1260 of which are malicious and belong to 49 distinct malware families. Similar to the Virus Share and Virus Total datasets, which provide an ever-changing list of new malware types, this is a continuously updated repository of malware files. Another large dataset with 22,500 malicious and 22,500 benign raw files is the Comodo dataset. With 250 malicious files, the Contagio dataset is significantly smaller than the others. There are 120,000 Android applications in the DREBIN dataset, 5000 of which are malicious, which is a highly imbalanced dataset. These crude information records can be handled in various ways including as paired documents, as Programming interface calls extricated utilizing a device like Cuckoo sandbox, or different strategies. The Microsoft dataset is the only significant dataset that did not provide raw files. The Microsoft dataset, which was created for a kaggle competition, consists of 10,868 hexadecimal and assembly-representative labeled malware binary files belonging to nine distinct malware families: Obfuscator, RAmnit, Lollipop, Kelihos_ver3, Vundo, Simda, and Kelihos_ver1. Gatak and ACY

In the DL literature, numerous malware datasets are based on publicly accessible malware databases. Contagio, Comodo, the Genome Project, Virus Share, or DREBIN, and the Google Play Store as a source of benignware were the most prevalent of these. Others drew on internal resources that were kept secret from the general public. Then, features were mostly extracted through dynamic or static analysis. Binary versions of the software were the source of other features.

Additionally, the Computer Emergency Readiness Team (CERT) Insider Threat Dataset v6.2 was a substantial synthetic dataset for insider threat detection. Over 130 million events from 516 days of system logs are included in this dataset, 400 of which are malicious. Email datasets are challenging to get in light of the fact that they are especially difficult to access because of

security concerns. However, EnronSpam, SpamAssassin, and LingSpam are some common email corpora. Other studies on cyber security that used DL were frequently novel and did not use standard datasets; they weren't made public because they were made internally. These studies were frequently the only ones conducted in those subject areas and relatively recent.

Cyber Applications Of Deep Learning Methods

Malware

It is becoming increasingly difficult to defend against malware attacks using standard strategies because of their increasing frequency and variety. The development of generalizable models for the autonomous detection and classification of malware is made possible by DL. This can protect organizations or individuals from small-scale attackers who use known malware or large-scale attackers who use novel malware.

Detected

Malware can be detected in a variety of ways. The second study improved on the first by developing DL-based detectors of malicious Android applications using features from static and dynamic analyses. Three specific sources were used to specifically select the features: static examination of dynamic behaviours as well as sensitive application program interfaces (APIs) and required permissions. By parsing AndroidManifest.xml and classes.dex, the static-based features are derived from the installation.apk file. This details the APIs used and the required permissions. By collecting data from DroidBox, an Android application sandbox, the dynamic behaviour features are derived from dynamic analysis. A DBN with two hidden layers that took these features as input had an accuracy of 96.76 percent, a TPR of 97.84 percent, and a FPR of 4.32 percent. Numerous designs were tried and a two-stowed away layer DBN was viewed as the best. They tested random forests, logistic regression, naive Bayes, and support vector machines (SVMs), and their results were superior to these.

When compared to static features, which are simple to obfuscate, dynamic features tend to be more reliable. Since the software is run in a sandbox, features like API calls are frequently utilized. Pascanu et al. is one example of this[18], who created a classification method for malware detection that incorporates logistic regression, multilayer perceptron (MLP), and RNNs. To predict the next API call, the RNN is trained unsupervised. After max-pooling the feature vector to prevent it from potentially reordering temporal events, the output of the hidden layer of this RNN is fed into the classifier. The hidden state from the middle of the sequence and the final

hidden state are utilized to guarantee that the features contain temporal patterns. The FPR was 1%, and the TPR was 71.71 percent.

Kolosnjaji and co.[13] discovered malware by employing CNNs and RNNs. One-hot encoding is used to turn the list of API kernel call sequences into binary vectors. A method for storing categorical data in a format that is easier for machine learning is one-hot encoding. The DL algorithm, which consists of a CNN and a RNN (with an LSTM and a softmax layer), is trained with this data. This model accomplishes an exactness of 89.4%, accuracy of 85.6%, and review of 89.4%.

Tobiyama and other built a malware detector that used an RNN to extract features from API calls time series data. An image containing these features is then processed using a CNN to determine whether it is malicious or normal. The CNN has two convolutional layers and two pooling layers, and the RNN makes use of an LSTM. Two completely connected layers come next. They were able to achieve an AUC of 0.96 despite the small dataset they used.By preprocessing the Windows Portable Executable (PE) files, Ding, Chen, and Xu extracted the n-grams and created a DBN with the operational codes (opcodes in machine language). There were three hidden layers in the DBN. There were 10,000 unlabelled files and 3,000 malicious files in the dataset. When pre-trained with unlabelled data, the DBN model performs better than SVMs, decision trees, and k-nearest neighbours clustering. The best DBN had an accuracy rate of 96.7%.

McLaughlin and others additionally, a feature-free, engineering-free detector was developed using the opcodes found in malware files. McLaughlin and others[8] processed the raw opcode data with an embedding layer before feeding it into a CNN with two convolution layers, one max pooling layer, a fully connected layer, and a classification layer. They achieved recall of 95 percent and 85 percent, an F1 Score of 97 percent and 78 percent, and accuracy of 98 percent and 80 percent on various datasets. Similar to the decline in non-DL methods, the significant difference between the first and second datasets is probably due to a significant increase in the variety of malware in the second dataset.

Hardy and co. also built a DL malware detector with API calls. For this job, they used autoencoders and a sigmoid classification layer, and they got 95.64 percent right. Xu et al., Benchea and Gavriluț, Hou and co., Zhu and co., furthermore, Ye et al. utilized RBMs The datasets and approaches used varied in determining success with these. Hardy and others, Hou and co., and Ye as well all made use of the Comodo Cloud Security Center dataset, which had an

accuracy of 96.6% or a TPR of 97.9%. Using a custom dataset, Benchea and Gavriluţ achieved a true positive rate of 90.1% and an accuracy of 99.72 percent. Xu and co. on a Google Play Store and VirusSharedataset, achieved an accuracy of 93.4 percent. Zhu and co. used a dataset that included data from the Google Play Store, Genome, DREBIN, and VirusTotal, and they got an F1 Score of 95.05 percent. The raw software binaries, on the other hand, can be utilized as features. The software binaries were transformed into two-dimensional entropy histograms by Saxe and Berlin[11]; a feature vector based on the import address table of the input binary file; and numerical fields taken from the portable executable packaging of the binary. This was accomplished without having to manually sort, unpack, or filter the software. A standard DNN for classification was then trained using these features. Using dropout layers and parametric rectified linear or sigmoid activation functions, these features are used to train a four-layer neural network, which includes an input layer, two hidden layers, and an output layer. Saxe and Berlin [9] followed this with a Bayesian alignment model to give a likelihood that a given record is malware. Because the classifier cannot be assumed to have a standard distribution, this is based on a prior of the ratio of malware to benignware and the DNN's error rate, using an Epanechnikov kernel for kernel density estimation. They assert that they have reached a level of success that could be used in real life: a 95% identification rate and a 0.1% FPR.

To distinguish refined malware, network conduct based techniques are required as they key on the coordinated order and control (C2) traffic from the malware. Due to the demands on resources, it is impossible for humans to thoroughly examine all new malware samples for an extended period of time. Shibahara et al.[9] proposed a technique for deciding if network-based unique examination ought to be applied to organize information, and when it ought to be suspended, in view of organization conduct, explicitly when the malware stops C2 action. Their approach's central concept was centred on two features of malware communication: the alteration in the common latent function (i.e., the outcomes that were unplanned or unexpected) and the communication purpose. Malware communications share these characteristics with natural language. In order to achieve high classification performance, they used the recursive tensor neural network (RSTNN), which improved the performance of recursive neural networks (RSNN) by using a tensor to calculate high-order composition of input features. Their proposed method reduced analysis time by 67.1% when tested on 29,562 malware samples, with a precision of 97.6%, recall of 96.2%, and F1 Score of 96.9%.

Malware frequently needs to communicate with C2 servers located on external networks. Mizuno et al. used network traffic's HTTP headers to[9] identified malicious software-generated traffic with a 97.1% accuracy and a 1.0% FPR. A DNN with two hidden layers was used to achieve this.Among the many significant advantages of mobile edge computing (MEC), location awareness services and cloud computing capabilities are two examples. However, due to the mobile device's vulnerability when connecting to an edge computing device, this new computing paradigm raises potential security concerns. Chen, Zhang, and Maharjan[3] trained a DBN with an unsupervised hidden layer of RBMs, followed by a classification layer, using a dataset consisting of 500 malicious and 5000 benign applications from an MEC environment. Depending on the proportion of normal to malicious data, this method outperforms softmax, decision trees, SVMs, and random forests with accuracies of between 91 and 96 percent. However, without additional metrics like true positive and false negative rates, it is difficult to interpret these numbers.Ransomware, also known as cryptovirology, is a growing problem due to the large number of variations and relative ease with which new variations can be created through minor augmentations. Hill and Bellekens[4] classified cryptographic primitives in compiled, binary executables with the help of dynamic convolutional neural networks (DCNNs) in order to combat these cryptovirological augmentations. A DCNN is like a regular CNN; However, instead of utilizing a maximum pooling layer with a particular dimension, it makes use of k-max pooling, where k scales with input length to accommodate inputs of various lengths. An accuracy of 91.3 percent was achieved by employing a DCNN with an embedding layer, 11 convolutions, and k-max pooling layers.

Dahl and co.[15] produced DNNs used for classifying malware by combining feature selection and random projections[17] to reduce the dataset's dimensionality. Using a modified version of Microsoft's production anti-malware engine, the initial dataset was created; the same engine that powers Microsoft Security Essentials [18] to extract features like tri-grams of API calls, distinct combinations of a single system API call and one input parameter, and null terminating patterns. 50 million features were produced by combining all feature collection parameter combinations. Using feature selection, this was reduced to 179,000. The feature space was then reduced to a few thousand using the sparse random projections method. The utilization of RBMs for the hidden layers was one of the various DNN architectures that were tested. The one-hidden-layer DNN architecture without RBMs performed the best, with a test error on malware type of 9.53

percent and a FPR of 0.35 percent and a test error on two classes of 0.49 percent and a FPR of 0.83 percent, respectively. In any case, the two-layer without RBMs didn't perform genuinely more terrible. Cordonsky et al.[9] used a DNN with nine layers, batch normalization, and dropout between layers to perform a similar test of malware classification. They were able to classify malware families with 97% accuracy using features derived from static and dynamic analysis. Cordonsky and others Additionally, it was discovered that the output of the DNN prior to the decision layer can be used to distinguish between novel and known family types in a two-dimensional visualization.

CNNs, on the other hand, can be used to classify malware. Classifying the binary files using a 2D CNN and transforming them into a 2D grayscale image is one option[10]. Treating the operation code as words and applying a 1D CNN with an embedding layer for classification is another strategy. The CNN without a pre-trained embedding layer had the highest accuracy of any CNN-based model, achieving 99.52 percent.

David and Netanyahu's[11] novel approach to malware classification made use of denoising auto encoders and DBNs trained on unlabeled data to generate malware signatures. The marks of the product were worked by taking the logs from a sandbox, and handling them utilizing n-grams by taking the most widely recognized 20,000 unigrams that show up just in the malware, and making a 20,000-highlight vector that distinguishes whether a given unigram showed up. This information was then used to pre-train an eight-layer DBN with denoising auto encoders. 30 numbers made up the final malware signature vector. The organization was prepared on the 1800 malware models, acquired from C4 Security, with six distinct kinds of malware, 300 models for each sort. An SVM was used to build the malware classifier after the features were computed, and 1200 malware examples were used for training. The results were promising, with an accuracy rate of 98.6%.By using an RNN autoencoder to convert the API call sequences into a low-dimensional feature vector, followed by a classification layer to identify the type of malware family, Wang and Yiu [12] improved upon the work of David and Netanyahu[11]. Wang and Yiu's[12] classification accuracy was 99.1% thanks to a bidirectional recurrent neural network autoencoder layer. In addition, they instructed a second classifier to correctly classify zero-day attacks by interpreting file access patterns. A 99.2% accuracy recurrent neural network autoencoder was the best model. In a similar vein, Yousefi-Azar et al.[13] tested various classification layers and constructed an autoencoder-based classifier with API calls. With an

accuracy of 96.3%, they discovered that an SVM classifier was the best. However, the accuracy of a unigram classifier using Xgboost was 98.2%.

For their features, Huang and Stokes[11] only used software dynamic analysis. The API and parameter stream, in addition to the raw executable file, provided the basis for the features. They were able to solve the binary malware detection problem with an error rate of 0.36 percent and the malware classification problem with an error rate of 2.94 percent using a DNN.

Grosse and others[15] further tested their DL model on adversarial samples generated using a method developed by Papernot et al. to classify malicious Android applications.[16] (a distinction from GANs). The reason for the work by Grosse et al. was to examine the DNN's generalizability. They utilized the application's static analysis-derived features from the 120,000 Android applications in the DREBIN dataset[16]. For classification, the DNN had two hidden layers and a softmax layer. Their initial DNN classifier had an accuracy of 95.93 percent to 98.35%, with FNR and FPR of 9.73 percent and 1.29 percent, respectively, depending on the training set's malware-to-beneficial ratio. Be that as it may, on the test dataset made out of ill-disposed models, the misclassification rate was somewhere in the range of 63.08% and 69.35%, contingent upon the proportion of malware to benignware in the preparation dataset. After that, they retrained the models using a variety of adversarial samples, and they found that the misclassification rate went down, but only slightly.

Conclusions

Cyber defenders' ability to write and deploy new signatures to detect these new attacks is outpaced by the rate at which attacks against cyber networks continue to advance. Cyber security applications can use neural network-based DL approaches to detect new malware variants and zero-day attacks thanks to advancements in ML algorithm development and this. In this survey paper, we discussed how DL techniques could be applied to a wide range of cyber security attacks that targeted host systems, application software, networks, and data. Additionally, we provided a comprehensive analysis of the known applications of DL techniques for detecting these cyberattacks. We talked about the DL architecture and the training process for a wide range of new and established approaches, from RNNs to GANs. The various types of attacks were treated separately in the current methods. The cascading connection of malicious activities throughout an attack lifecycle (such as breach, exploitation, command and control, data theft, etc.) should be taken into consideration in future research. We also talk about the various metrics

that are used to measure how well DL performs in cyber security applications. Be that as it may, the utilization of various datasets for preparing and testing didn't take into account fair correlation across the various methodologies as a whole. As a result, benchmark datasets are absolutely necessary to advance DL in the field of cyber security. We identified the need for approaches to be developed that take the adversary into consideration as to how they may use DL as a tool to subvert DL detection mechanisms and identified future research opportunities related to the development of new datasets to motivate work in developing new DL approaches for cyber security. As a result, the goal of this survey is to compile a useful body of research to inspire researchers to improve DL for cyber security systems.

References

[1]. Buczak, Anna L., and ErhanGuven. "A survey of data mining and machine learning methods for cyber security intrusion detection." IEEE Communications surveys & tutorials 18, no. 2 (2015): 1153-1176.

[2]. Mahdavifar, Samaneh, and Ali A. Ghorbani. "Application of deep learning to cybersecurity: A survey." Neurocomputing 347 (2019): 149-176.

[3]. Salloum, Said A., Muhammad Alshurideh, Ashraf Elnagar, and Khaled Shaalan. "Machine learning and deep learning techniques for cybersecurity: a review." In The International Conference on Artificial Intelligence and Computer Vision, pp. 50-57. Springer, Cham, 2020.

[4]. Alazab, Mamoun, and MingJian Tang, eds. Deep learning applications for cyber security. Springer, 2019.

[5]. Dixit, Priyanka, and Sanjay Silakari. "Deep learning algorithms for cybersecurity applications: A technological and status review." Computer Science Review 39 (2021): 100317.

[6]. Handa, Anand, Ashu Sharma, and Sandeep K. Shukla. "Machine learning in cybersecurity: A review." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 9, no. 4 (2019): e1306.

[7]. Xin, Yang, Lingshuang Kong, Zhi Liu, Yuling Chen, Yanmiao Li, Hongliang Zhu, Mingcheng Gao, HaixiaHou, and Chunhua Wang. "Machine learning and deep learning methods for cybersecurity." Ieee access 6 (2018): 35365-35381.

[8]. Sarker, Iqbal H., A. S. M. Kayes, ShahriarBadsha, HamedAlqahtani, Paul Watters, and Alex Ng. "Cybersecurity data science: an overview from machine learning perspective." Journal of Big data 7, no. 1 (2020): 1-29.

[9]. Martínez Torres, Javier, Carla Iglesias Comesaña, and Paulino J. García-Nieto. "Machine learning techniques applied to cybersecurity." International Journal of Machine Learning and Cybernetics 10, no. 10 (2019): 2823-2836.

[10]. Annamalai, Chinnaraji. "Factorials, Integers and Multinomial Coefficients and its Computing Techniques for Machine Learning and Cybersecurity." Available at SSRN 4190789 (2022).

[11]. Annamalai, Chinnaraji. "Computation of Combinatorial Geometric Series and its Combinatorial Identities for Cryptographic Algorithm and Machine Learning." (2022).

[12]. Ahsan, Mostofa, Kendall E. Nygard, Rahul Gomes, MdMinhaz Chowdhury, NafizRifat, and Jayden F. Connolly. "Cybersecurity threats and their mitigation approaches using Machine Learning—A Review." Journal of Cybersecurity and Privacy 2, no. 3 (2022): 527-555.

[13]. Annamalai, Chinnaraji. "Computation of Factorial and Multinomial Theorems for Machine Leaning and Cybersecurity." Available at SSRN 4205589 (2022).
[14]. Ghillani, Diptiban. "Deep Learning and Artificial Intelligence Framework to Improve the Cyber Security." Authorea Preprints (2022).
[15]. Abu Al-Haija, Qasem, MoezKrichen, and Wejdan Abu Elhaija. "Machine-learning-based darknet traffic detection system for IoT applications." Electronics 11, no. 4 (2022): 556.
[16]. Berghout, Tarek, Mohamed Benbouzid, and S. M. Muyeen. "Machine learning for cybersecurity in smart grids: A comprehensive review-based study on methods, solutions, and prospects." International Journal of Critical Infrastructure Protection (2022): 100547.
[17]. Chen, Dongliang, PawełWawrzynski, and Zhihan Lv. "Cyber security in smart cities: a review of deep learning-based applications and case studies." Sustainable Cities and Society 66 (2021): 102655.
[18]. Dixit, Priyanka, and Sanjay Silakari. "Deep learning algorithms for cybersecurity applications: A technological and status review." Computer Science Review 39 (2021): 100317.
[19]. Sarker, Iqbal H. "Deep cybersecurity: a comprehensive overview from neural network and deep learning perspective." SN Computer Science 2, no. 3 (2021): 1-16.
[20]. Ferrag, Mohamed Amine, OthmaneFriha, LeandrosMaglaras, Helge Janicke, and Lei Shu. "Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis." IEEE Access 9 (2021): 138509-138542.
[21]. Rodriguez, Eva, Beatriz Otero, Norma Gutierrez, and Ramon Canal. "A survey of deep learning techniques for cybersecurity in mobile networks." IEEE Communications Surveys & Tutorials 23, no. 3 (2021): 1920-1955.
[22]. Ahmed, KosratDlshad, and ShavanAskar. "Deep learning models for cyber security in IoT networks: A review." International Journal of Science and Business 5, no. 3 (2021): 61-70.
[23]. Geetha, R., and T. Thilagam. "A review on the effectiveness of machine learning and deep learning algorithms for cyber security." Archives of Computational Methods in Engineering 28, no. 4 (2021): 2861-2879.
[24]. Nikoloudakis, Yannis, IoannisKefaloukos, Stylianos Klados, Spyros Panagiotakis, EvangelosPallis, CharalabosSkianis, and Evangelos K. Markakis. "Towards a Machine Learning Based Situational Awareness Framework for Cybersecurity: An SDN Implementation." Sensors 21, no. 14 (2021): 4939.
[25]. Tran, Minh-Quang, Mahmoud Elsisi, Karar Mahmoud, Meng-Kun Liu, MattiLehtonen, and Mohamed MF Darwish. "Experimental setup for online fault diagnosis of induction machines via promising IoT and machine learning: Towards industry 4.0 empowerment." IEEE access 9 (2021): 115429-115441.