Application of Deep Learning Architectures for Cyber Security

R. Vinayakumar, K. P. Soman, Prabaharan Poornachandran and S. Akarsh

Abstract Machine learning has played an important role in the last decade mainly in natural language processing, image processing and speech recognition where it has performed well in comparison to the classical rule based approach. The machine learning approach has been used in cyber security use cases namely, intrusion detection, malware analysis, traffic analysis, spam and phishing detection etc. Recently, the advancement of machine learning typically called as 'deep learning' outperformed humans in several long standing artificial intelligence tasks. Deep learning has the capability to learn optimal feature representation by itself and more robust in an adversarial environment in compared to classical machine learning algorithms. This approach is in early stage in cyber security. In this work, to leverage the application of deep learning architectures towards cyber security, we consider intrusion detection, traffic analysis and Android malware detection. In all the experiments of intrusion detection, deep learning architectures performed well in compared to classical machine learning algorithms. Moreover, deep learning architectures have achieved good performance in traffic analysis and Android malware detection too.

Keywords Machine learning · Deep learning · Intrusion detection · Traffic analysis · Android malware detection

1 Introduction

The digitization and rapid advancement of internet and its services has a huge impact on today's modern communication system for individuals and organization. At the same time the attacks to the internet and its services are rapidly growing and diverse in

R. Vinayakumar (✉) · K. P. Soman · S. Akarsh
Center for Computational Engineering and Networking (CEN),
Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India
e-mail: vinayakumarr77@gmail.com

Prabaharan Poornachandran
Centre for Cyber Security Systems and Networks, Amrita School of Engineering,
Amrita Vishwa Vidyapeetham, Amritapuri, India

A. E. Hassanien and M. Elhoseny (eds.), *Cybersecurity and Secure Information Systems*, Advanced Sciences and Technologies for Security Applications, https://doi.org/10.1007/978-3-030-16837-7_7

nature. Due to this cyber security plays an important role. The consistent increase in range of cyber threats and malwares obviously demonstrates that current countermeasures don't seem to be enough to defend it. The increasing number of cyber-attacks was a noteworthy point of concern at the World Economic Forum (WEF) 2016.[1] As indicated by its eleventh yearly worldwide global risks report, cyber-attacks are positioned in the rundown of prime ten threats in 140 economies ("The Global Risks" 2016). These issues demand the development of efficient solutions to cyber security.

Development of cyber security solutions has been remained as complex task. Most of the existing solutions are mostly based on signature based detection. A signature based detection system requires a human intervention to supervise and update the signature each and every time. This kind of systems fails in tackling the new variants of malware or cyber threats and entirely new malware or cyber threat, because the signature based system is entirely relying on a signature database. To allivate the problems of classical techniques of cyber security, research in artificial intelligence and more specifically machine learning is sought after [1, 2]. Machine learning is a method where the system is taught to distinguish between good and malicious file. For this, a machine has to be first trained with a set of features obtained from benign and malicious samples. Deep learning is a sub module of machine learning. It is also termed as deep neural networks (DNNs) [3]. Deep learning architectures have achieved remarkable results in numerous supervised and unsupervised long-standing artificial intelligence tasks related to natural language processing (NLP), image processing, speech recognition and many others [3]. Recent advance in optimization and technologies of parallel and distributed computing have made to train large scale data. Deep learning takes inspiration from how the brain works. Specifically, the deep learning architectures can understand the meaning of the data when it sees a large amount of data and tune it's meaning whenever new data appears. Thus, it doesn't need any domain expert knowledge assistance to understand the importance of new input. The advantage of deep learning is that it learns hierarchical feature representation by passing the information to more than one hidden layer.

Applying deep learning architectures to cyber security is a significant task to enhance the cyber-attack and malware detection rate [4–19]. In this work, intrusion detection, traffic analysis and Android malware detection cyber security use cases are considered. In intrusion detection, the performance of classical machine learning algorithms and deep learning architectures are evaluated in detail. For both the traffic analysis and Android malware detection, the application of deep learning architectures is used. Deep learning architectures have an added advantage to analyze the big data of security artifacts in comparison to classical machine learning algorithms. Both classical machine learning algorithms and Deep learning architectures are parameterized, thus the optimal performance depends on the best parameters. Finding optimal parameters in deep learning has been remained as a significant task in recent days. In this work, we run various experiments to choose right parameters for both classical machine learning algorithms and deep learning architectures. The rest of the paper is organized as follows. Section 2 provides background knowledge

[1] https://www.weforum.org/events/world-economic-forum-annual-meeting-2016.

of classical machine learning algorithms and deep learning architectures. Section 3 includes cyber security use cases. Section 4 contains conclusion and future works.

2 Background Knowledge

Generally, classical machine learning and modern machine learning called as deep learning architectures are two sub classes of artificial intelligence. Applying machine learning techniques to cyber security domain has been considered as a vivid area of research in recent days. This section provides the mathematical details of classical machine learning algorithms and deep learning architectures. In this work, classical machine learning are implemented using scikit-learn[2] and deep learning architectures are implemented using TensorFlow[3] with Keras.[4] Experiments with classical machine learning algorithms are run on CPU enabled machines and deep learning architectures are run on GPU enabled machines.

2.1 Classical Machine Learning (CML)

Machine learning is a field in artificial intelligence which trains computers to learn like people. The goal of machine learning is to make ensure a machine can learn and automate tasks without the intervention of human interference. The most commonly used classical machine learning algorithms are discussed below.

2.1.1 Logistic Regression (LR)

An idea obtained from statistics and created by David Cox in 1958, logistic regression is like linear regression, yet it averts misclassification that may occur in linear regression. This happens because of the freedom between the features of linear regression. While logistic regression use the log-odds of the probability of an event which is a linear combination of independent or predictor variables. Not at all like linear regression, which has results under '0' and more prominent than '1', logistic regression results are basically either '0' or '1'. This is acquired by utilizing the non-linear activation function, *sigmoid* to predict the output. The probability for information to be characterized into a specific class relies upon at least one independent features of the information. The efficacy of logistic regression is mostly dependent on the size of the training data.

[2] https://scikit-learn.org/stable/.
[3] https://www.tensorflow.org/.
[4] https://keras.io/.

2.1.2 Naive Bayes (NB)

Naive Bayes (NB) classifier makes use of Bayes theorem with the fundamental assumption of independence of features. This means the presence of a feature in a class doesn't affect other features. Naive Bayes classifier most likely outperforms when feature dimension is high and easy to build. The independence assumptions in Naive Bayes classifier overcome the curse of dimensionality.

2.1.3 Decision Tree (DT)

A Decision tree has a structure like flow charts, where the top node is the root node and each internal node signifies a test on a feature of the information. Each leaf node holds a class label. It is easy to visualize and understand and can deal with a wide range of information. The algorithm might be biased and may end up unstable since a little change in the information will change the structure of the tree.

2.1.4 K-Nearest Neighbor (KNN)

K-Nearest Neighbor (KNN) is a non-parametric approach which stores all the possible cases and using similarity measure i.e. distance function classifies other new cases. It is computationally expensive and requires larger memory because it stores the entire training data.

2.1.5 Ada Boost (AB)

Ada Boost (AB) learning algorithm is a technique used to boost the performance of simple learning algorithms used for classification. Ada Boost constructs a strong classifier using a combination of several weak classifiers. It is a fast classifier and at the same time can also be used as a feature learner. This can be mostly used in imbalanced data analysis tasks.

2.1.6 Random Forest (RF)

Random forest (RF) is an ensemble tool which builds a decision tree by combining a subset of observations and variables. In order to get stable predictions, Random forest unites several decision trees. The predictions from Random forest are more likely better than individual decision tree. It uses the concept of bagging to create several minimal correlated decision trees.

2.1.7 Support Vector Machine (SVM)

Support Vector Machine (SVM) belongs to the family of supervised machine learning techniques, which can be used to solve classification and regression problems. SVM is a linear classifier and the classifier is a hyper plane. It separates the training set with maximal margin. The points near to the separating hype plane are called support vectors and they determine the position of hyper plane. If the training data set is not linearly separable then it can be mapped to high-dimensional space using kernels where it is assumed to be linearly separable.

2.2 Modern Machine Learning

Deep learning is a modern machine learning which has the capability to take raw inputs and learns the optimal feature representation implicitly. This has performed well in various long standing artificial intelligence tasks [3]. Most commonly used deep learning architectures are discussed below in detail.

2.2.1 Deep Neural Network (DNN)

An artificial neural network (ANN) is a computational model influenced by the characteristics of biological neural networks. Feed forward neural network (FFN), Convolutional neural network and Recurrent neural network (RNN) are belongs to a family of ANN. FFN forms a directed graph in which a graph is composed of neurons named as mathematical unit. Each neuron in ith layer has connection to all the neurons in $i+1$th layer. That's the reason hidden layers are called as fully connected. Each neuron of the hidden layer denotes a parameter h that is computed by

$$h_i(x) = f({w_i}^T x + b_i) \tag{1}$$

$$hi_i : R^{d_i - 1} \rightarrow R^{d_i} \tag{2}$$

$$f : R \rightarrow R \tag{3}$$

where $w_i \in R^{d \times d_{i-1}}$, $b_i \in R^{d_i}$, d_i denotes the size of the input, f is a non-linear activation function, $ReLU$.

FFN passes information along connections from one node to another without forming a cycle. Thus it can't give importance to the past values. Multi-layer perceptron (MLP) is one type of FFN that contains 3 or more layers, specifically one input layer, one or more hidden layer and an output layer in which each layer has many neurons, called as units in mathematical notation (see Fig. 2). The hidden layers are chosen based on hyper parameter tuning method. Generally classical multi-layer perceptron uses *sigmoid* non-linear activation function. In Fig. 1 the derivative of

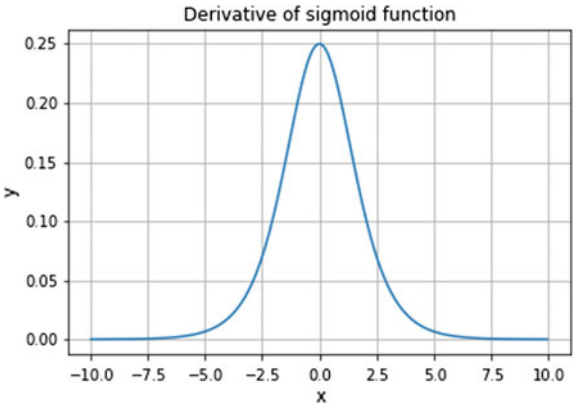


Fig. 1 Derivative of *sigmoid* non-linear activation function

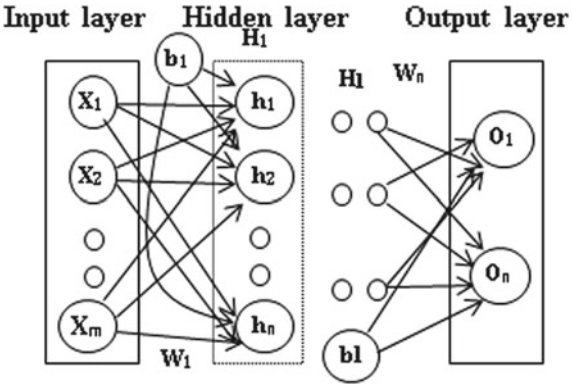


Fig. 2 Architecture of Deep neural network with l hidden layers, all connections are not shown, inputs $x = x_1, x_2, \ldots, x_{m-1}, x_m$ and $O = O_1, O_n$

sigmoid non-linear activation function is almost zero for high positive and negative values in the domain. This leads to vanishing gradient. This problem is overcome by using *ReLU* non-linear activation function. The mathematical representation of *ReLU* is given below.

$$f\,(x) = max\,(0, x) \tag{4}$$

The derivative of *ReLU* non-linear activation function is 1 if $x > 0$ and 0 if $x < 0$. The gradient has a constant value for $x > 0$, this reduces the chance of the gradient to vanish. Since the gradient of *ReLU* is constant, it boosts the learning during training the neural network. In *ReLU* for $x \leq 0$ the gradient is 0, the zero gradient is an advantage since it allows sparsity. If the units with zero gradient is more than the connections, the network become more sparse. In the case of *sigmoid* non-linear activation function, the chance of generating non zero values is more which makes the connection representation in neural network dense while training. Sparsity always has an advantage over dense representations. Additionally, *ReLU* speeds up the training process in compared to other non-linear activation functions. Stacking hidden layers

on top of each other is named as deep neural network (DNN). DNN with l hidden layers is mathematically defined as follows

$$H(x) = H_l(H_{l-1}(H_{l-2}(\cdots(H_1(x))))) \tag{5}$$

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{6}$$

$$\tanh(z) = \frac{e^{2z}-1}{e^{2z}+1} \tag{7}$$

$$softmax(Z)_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{8}$$

2.2.2 Autoencoder

Autoencoder is implemented similar to other neural network, but in this case it is trained to learn the input itself [3]. Number of input and output dimensions will be same in the case of Autoencoder, which means that the network is built in such a way that it should be able to reconstruct the input data. Autoencoder are the neural network typically made for the purpose of dimensionality reduction. The architecture of an Autoencoder is similar to the multi-layer perceptron; it contains an input layer, one or more hidden layers and finally the output layer, see Fig. 3. If the Autoencoder is contains multiple hidden layers, then the features extracted from one layer are further processed to different features and these features should be capable of reconstructing the data. In usual classification methods, some particular features are selected initially and then calculated for all data points and then it is fed to classification algorithm as input. However, different features are extracted from different layers in Autoencoder since it is following unsupervised approach and this can be used as features and passed

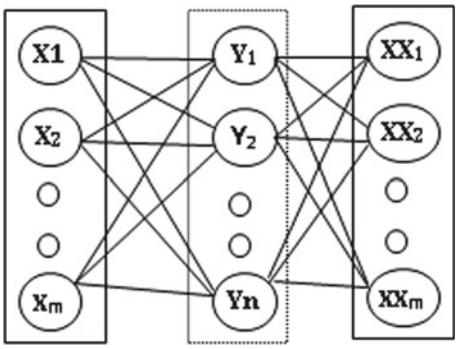


Fig. 3 Structure of Autoencoder

into other classical machine learning algorithms or deep learning architectures such as convolutional neural network (CNN), recurrent neural network(RNN), long short term memory (LSTM) and convolutional neural network-long short term memory (CNN-LSTM).

For example, consider the architecture shown in Fig. 3. Here, the input to the network is a N length vector X, which is reconstructed as XX. The idea is to select weights and biases to produce Y such that the error between $X1$ and XX is as small as possible. Y is constructed from $X1$ via the transformation

$$Y_{1\times m} = f(X_{1\times n} W_{n\times m} + b_{1\times m}) \tag{9}$$

where w and b represents the weights and biases corresponding to the first layer and f is some non-linear activation function. Similarly, in the next layer X is reconstructed from Y as XX via the transformation

$$XX_{1\times n} = f(Y_{1\times m} W1_{m\times n} + b1_{1\times n}) \tag{10}$$

where $w1$ and $b1$ are the corresponding weights and biases and f is the non-linear activation function. When Y and X are of the same dimensions, the obvious choice of w and $w1$ becomes the identity matrices with unity functions as the non-linear activation function. But when they are of different dimensions, the network is forced to learn newer representations of the input X via some transformations [3]. Autoencoder implementations need not be restricted to one layer; there can be multiple layers of different dimensions. In such cases, the features extracted from one layer are further processed to newer features which are still capable of reconstructing the data. Keeping the dimension of Y smaller than X results in learning some compressed encoding of the input and has found many applications in different fields [3].

2.2.3 Convolutional Neural Network (CNN)

Convolutional neural network (CNN) is a variant of classical MLP, most generally used in image processing applications. CNN is applied on various dimensions of data. In this work, we used convolution on 1D data. Convolution 1D is also named as temporal convolution. They have the capability to capture the spatial structure exists in the data. Generally, the CNN network consists of convolution1D, pooling1D and fully connected layer for 1D data. The purpose of convolution layer is to capture the optimal features from the input matrix. Convolution uses a linear operation i.e. filters that slide over rows of input matrix. The features that are extracted from each filter will be grouped into new feature set, called as feature map. The number of filters and the length will be chosen by following hyper parameter tuning method. This in turn uses non-linear activation function, *ReLU* on element wise. Next pooling (max, min or average) is applied on the feature map. This is a non-linear down sampling operation that helps to reduce the parameters and controls over fitting. The reduced feature sets are passed to the fully connected layer for classification. This contains a

connection from a neuron to every other neuron. Instead of passing the pooling layer features into fully connected layer, it can also be given to recurrent layers such as RNN, LSTM and GRU to capture the sequence related information of data.

2.2.4 Recurrent Structures

Recurrent neural networks (RNNs) are a family of neural networks that operate on sequential data [3]. Classical neural network assumes that all inputs and outputs are independent of each other. Generally, it takes input from two sources, one is from present and the other from past. The past information is stored in the self-recurrent loop typically called as recurrent. Given an input sequence $X = (x_1, x_2, \ldots, x_T)$, the transition function for RNN model can be mathematically represented as follows

$$h_t = g_n(w_{xh}X_t + w_{hh}h_{t-1} + b_h) \tag{11}$$

$$o_t = g_n(w_{oh}h_t + b_o) \tag{12}$$

where x_t denotes an input vector, g_n denotes non-linear activation function, h_t denotes hidden state vector, o_t denotes output vector and terms of w and b denotes weights and biases respectively.

RNN results in vanishing and error gradient when it is memorized to remember information for long time steps [3]. To reduce the vanishing and gradient issue, gradient clipping was used [3]. Later, LSTM was proposed [3]. This has a memory block instead of simple unit in RNN that helps to store information. A memory block has a memory cell controlled by input, output and forget gates. The gates and cell state together provides interactions. The main function of gate is to control the information of memory cell. These gates help the LSTM network to remember information for longer duration than the RNN. Given an input sequence $X = (x_1, x_2, \ldots, x_T)$, the transition function for LSTM model can be mathematically represented as follows

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{13}$$

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{14}$$

$$c_t = \tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{15}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{16}$$

$$h_t = o_t * \tanh(c_t) \tag{17}$$

where x_t denotes an input vector, h_t denotes hidden state vector, c_t denotes cell state vector, o_t denotes output vector, i_t denotes input vector and f_t denotes forget state vector and terms of w and b denotes weights and biases respectively.

LSTM is typically complex network. Recently minimized version of LSTM, gated recurrent unit (GRU) is introduced [3]. It is similar to LSTM but it is more computationally efficient than LSTM because it has only two gates; update and reset gate. In GRU, forget and input gates functionality found in LSTM are combined to form an update gate. The update gate characterizes the amount of past memory to be kept in GRU. Recently, RNN variant identity RNN was proposed which initialize the appropriate RNNs weight matrix using an identity matrix or its scaled version, and use *ReLU* as non-linear activation function to handle vanishing and exploding gradient issue [3].

2.2.5 Statistical Measures

In this work to evaluate the efficacy of classical machine learning algorithms and deep learning architectures, we have considered the various statistical measures. These statistical measures are estimated based on True positive (TP), True negative (TN), False positive (FP) and False negative (FN). TP is the number of attack connections correctly classified, TN is the number of normal connections correctly classified, FP is the number of normal connections incorrectly classified and FN is the number of attack connections correctly classified. The measures are defined as follows

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100, \tag{18}$$

$$Precision = \frac{TP}{TP + FP} \times 100, \tag{19}$$

$$Recall = \frac{TP}{TP + FN} \times 100, \tag{20}$$

$$F1\text{-}score = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \times 100, \tag{21}$$

$$TPR = \frac{TP}{TP + FN}, \tag{22}$$

and

$$FPR = \frac{FP}{FP + TN}. \tag{23}$$

3 Cyber Security Use Cases

3.1 DeepID: Detailed Analysis of Classical Machine Learning Algorithms and Deep Learning Architectures for Intrusion Detection

Intrusion detection system (IDS) is an important tool used to distinguish between the normal and abnormal behavior on the computer systems and networks. This has become an indispensable part of computer systems and networks due to the rapid increase in cyber-attacks, cyber threats and malwares. Identifying an optimal algorithm to enhance the attack detection rate has been considered as a significant task in the field of network security research. Most commonly used approaches for intrusion detection are rule based and classical machine learning based techniques. Machine learning based algorithms have the capability to detect new or variants of cyber-attacks, cyber threats and malwares in compared to rule based algorithms. The classical machine learning algorithms required feature engineering and feature representation phase to learn the hidden characteristics to distinguish between the normal and abnormal behaviors on the computer systems and networks. An advanced model of machine learning, deep learning completely avoids the feature engineering and feature representation phase by learning the optimal features by itself. To leverage the application of deep learning architectures to intrusion detection, in this sub module the efficacy of various deep learning architectures are evaluated. For comparative study, the existing various classical machine learning algorithms are evaluated. In all the experiments, the deep learning architectures have obtained superior performance in comparison to the classical machine learning algorithms.

3.1.1 Introduction

Information and communication technologies (ICT) systems have become more prevalent for increasingly powerful technologies in modern society. The constantly evolving paradigm of ICT systems pose new security challenges to the security researchers. At the same time attacks to ICT systems are common in recent days and it has been exists since the birth of the computers. These attacks are distinct and complex in nature. Due to the constantly evolving complex and diverse threats to the ICT systems, network security has been a vivid area of research for security researchers. Researchers are intensively studied and found various approaches namely firewalls, encryption and decryption techniques of cryptography, anomaly detection, intrusion detection and others. Intrusion detection system (IDS) is one of the feasible methods to attack those malicious attacks in an efficient way. This has been remained as a largely studied area since from 1980, a seminal work by John Anderson on the "Computer Security threat monitoring and surveillance" [20]. Recent advancement in technologies enabled peoples to use applications based on wireless

sensor networks (WSNs). Following, the various challenges and security issues in WSNs is discussed in detail by [50–60]. Undoubtedly, development of robust and efficient IDS is a perennial problem for the last years.

Mainly Intrusion detection is of two types, network based IDS (N-IDS) and host based IDS. N-IDS uses network behaviors where as host based IDS uses sensors logs, system logs, software logs, file systems, disk resources and others. This work focuses on N-IDS. The aim of N-IDS is to distinguish between the abnormal behaviors on the system from the normal network behavior. This has become an indispensable part of ICT systems and networks. Early solutions to intrusion detection are based on signatures. While a new attack happens, a signature must be updated to effectively detect the attacks. Signature based intrusion detection completely fails to detect the new or as well as the variants of the existing attack. To avoid human intervention and tackle new attacks, cyber security demands the flexible and interpretable integrated network security solutions to the ICT systems. Machine learning techniques have been predominantly used in cyber security for intrusion detection [2]. Over the years, the development of network based intrusion detection systems (N-IDS) to identify the unforeseen and unpredictable attacks has been a main challenge in network security research. To analyze and classify network traffic events without prior knowledge on the attack signature, researchers have studied and adopted various machine learning classifiers. Extensive research on machine learning results in a new area called 'deep learning'. Applying deep learning techniques towards solving various complex tasks has been a trend and impressive this year. This has seen a remarkable performance in various tasks in natural language processing, image recognition, speech processing and many more [3]. The applications of deep learning architectures are transformed into intrusion detection by [7, 17, 21]. Following in this sub module, various deep learning architectures are evaluated for network intrusion detection with the publically available data set NSL-KDD. Additionally, the various classical machine learning are evaluated for comparative study.

3.1.2 Related Work

Applying machine learning solutions to the development of efficient ID has being a vivid area of research. Researchers have introduced various methods and this section reviews the largest study to date related to machine learning and deep learning solutions to ID. In addition, the major drawbacks of KDD-Cup-99 dataset are discussed.

A major concern in ID research is the availability of adequate data to train an effective model is very less. This might be one of the significant reasons why it is difficult to develop a reliable ID platform for detecting the unknown malicious activities. Though a few data sets exist, each suffers from its own disadvantages. These data sets can only be used to check the benchmark of the various classifiers and not used in real time IDS. One among them is the de facto standard KDD-Cup-99. This was the preprocessed form of KDD-Cup-98 tcpdump traces and used in the third International Knowledge Discovery and Data Mining Tools Competition. For transforming the KDD-Cup-98 tcpdump traces into feature set, Mining Audit data

for automated models for ID (MADAM ID) is used [22]. The task of KDD-Cup-99 challenge was expected to develop a predictive model that can classify the network connection into benign or other diverse 28 attacks into 'DoS', 'Probe', 'R2L' and 'U2R' category.

Totally 24 teams were participated and the winning entries of first three teams have largely benefitted from by using the variants of decision tree. The performance of first three entries in terms of statistical measures are negligible due to they have only slight variations in detection rate. The 9th winning entry team has used 1-Nearest neighbor classifier. The large difference in detection rate among the 24 entries found between 17th and 18th entry. This shows that the first 17 entries were powerful and details about the submitted results are outlined in [23]. The KDD-Cup-99 challenge data set and its results have remained as a base line work and after that various approaches have been found. Most of them have used only the 10% data or a few data with mixing custom process for training and testing with the aim to benchmark the various classifiers. In [24, 25] discussed the various machine learning solutions that are applied on KDD-Cup-99.

Other few methods have used feature engineering for dimensionality reduction [2] and other few studies have reported good results with using the custom built-in data sets [2] and others have used to know the effectiveness of newly introduced machine learning classifiers [2]. However, these published results are not directly comparable to the published results of KDD-Cup-99 challenge.

In [26] introduced PNrule that use P-rules to anticipate the existence of the class and N-rule to anticipate the non-existence of the class. PNrule has showed good performance in comparison to previously published KDD-Cup-99 challenge results except to 'U2R' category. The importance of feature engineering in ID with KDD-Cup-99 was discussed by [27]. They discussed the importance of each feature that was found in KDD-Cup-99. Zhang et al. [28] used the Random forest classifier with the hybrid detection as the combination of anomaly and misuse detection. They were able to show experiments with higher detection rate for misuse detection in comparison to the KDD-Cup-99 results. With hybrid classifier, they reported the performance of the system may be enhanced. Li [29] discussed the applicability of genetic algorithm to ID by taking the benefit from the temporal and spatial information to detect the complex and diverse threats. Kolias et al. [30] used ant colony optimization techniques including descriptive statistics for comparing their results with the previously published results. Al-Subaie and Zulkernine [31] had used Hidden Markov model (HMM) to model the temporal patterns of TCP/IP packets with the sequential relationship between the events of normal and malicious. They were claimed that modeling the temporal patterns of sequence of TCP/IP connection have enhanced the performance in comparisons to the other structural pattern recognition techniques. A very first time, the concept of CNN introduced for ID [32].

Initially, the applicability of neural network mechanisms for ID was very less due to the fact that hard to train. Most commonly used methods are deep belief network (DBN), Restricted Boltzmann machine (RBM), autoencoder. In [33] proposed DBN based IDS and evaluated on KDD-Cup-99 data set. The proposed model performed better than the SVM and ANN. In [34] used MLP to detect an attack and categorize

into attack category. To find out the optimal MLP network, they have followed hyper parameter tuning method. In [35] used the hybrid of ANN and SVM and showed that the hybrid method performed better than the individual classifier. In [36] used the RNN to extract rules for attack patterns and these patterns were used to detect intrusions. In [37] proposed hybrid of RBM-SVM to identify network anomalies.

Due to the advancement in optimization methods and easy access of GPU has made deep learning architectures to train over parallel and distributed computing platforms. In [38] proposed a two level classifier for intrusion detection. In first level sparse autoencoder was used for feature extraction and followed by classical machine learning for classification. In [39] represented the network data in the form of sequence and applied RNN with Hessian-Free optimization method. Following, in [40] employed the LSTM. A detailed analysis of LSTM to intrusion detection is done using KDD-Cup-99 data set by [21].

3.1.3 Drawbacks of KDD-Cup-99

A detailed report on major shortcomings of KDD-Cup-99 challenge data is outlined by [42]. The most common well-known problem discussed by many authors is that the data set is not a complete symbolize of real-world network traffic. Though, with harsh criticisms KDD-Cup-99 is used in many research studies to understand and know the effectiveness of various machine learning classifiers. Tavallaee et al. [42] authors reviewed a detailed analysis of the information of tcpdump data and reported non-uniformity and simulated artifacts in KDD-Cup-99 challenge data set. They attempted to improve the performance of network anomaly detection between the KDD-Cup-99 and mixed KDD-Cup-99. They identified the network attributes; remote client address, TTL, TCP options and TCP window are very small in KDD-Cup-99 data in comparison to the real world network traffic data. Even KDD-Cup-98 suffers from same issues of KDD-Cup-98 due to the KDD-Cup-99 was the subset version of KDD-Cup-98.

A report by Tavallaee et al. [42] have briefly reported the reason behind the less performance in attacking the network connections related to the low frequency attacks; 'R2L' and 'U2R' category. They reported the reasons behind achieving the best performance in classifying the attacks belongs to either 'R2L' or 'U2R' category. However, by including a few connection records related to 'R2L' and 'U2R' category to the existing KDD-Cup-99 may results in higher detection rate. However, they lack in showing the performance through the experiments of machine learning classifier on the mixed data, so remained as a statement. In [42] reported 'snmpgetattack' have similarity in attacks related to 'R2L' and normal connection records. As a result, the machine learning classifier performs poorly in classifying these normal and 'R2L' category with respect to 'snmpgetattack'.

Initially, DARPA/KDD-Cup-98 have missed to show the performance of classical IDS. To overcome this issue, Brugger and Chow [41] have used DARPA/KDD-Cup-99 tcpdump traces as an input data to assess the performance of Snort IDS. The system results in a very poor performance for the attacks belongs to 'DoS' and

'Probe' category, mainly due to the system have adopted the fixed signature sets as a mechanism. In contrast to high frequency attacks categories such as 'DoS' and 'Probe', the Snort system have showed good performance for 'R2L' and 'U2R' attacks category.

Though many issues exists in KDD-Cup-99 challenge dataset, still widely been used for benchmarking the various machine learning classifier to network ID. To fix the built-in issue, researchers [42] introduced a most refined version of KDD-Cup-99 data set called as NSL-KDD. They constructed NSL-KDD by removing the redundant records in both the train and test connection records and completely removed the connection records that are exist in index 136,489 and 136,497. This avoids the classifier not to be biased towards the frequently occurring connection records.

3.1.4 Data Sets for Network Intrusion Detection System (N-IDS)

The availability of existing data as open source for evaluating the effectiveness of various machine learning classifier for N-IDS is very less due to privacy issues in network traffic. To benchmark the effectiveness of CNN and its variants, we used DARPA/KDD-Cup-99 challenge, refined version of KDD-Cup-99 challenge; NSL-KDD and most recent intrusion data set; UNSWNB-15. In the following, we provide the details of methods employed in collecting TCP/IP packets and mechanisms used by them to label each TCP/IP packets in all three IDS data sets.

1. **KDD-Cup-99**: KDD-Cup-99 was collected in air force base local area network (LAN), MIT Lincoln laboratory in 1998. The network containing 1000s of UNIX machines during the data set collection. At a time 100s of user's access the network and the data was continuously collected for nine weeks. This data was then divided into two parts, particularly the last two weeks data set was used for testing and the rest used for training. The data set was in the form of tcpdump. During the data set collection, they have injected the attacks to UNIX machines using Solaris, UNIX, Linux, Windows NT and SunOS. They had driven 300 attacks. These attacks were grouped into 32 unique attacks and 7 unique attacks categories. This data set was used in KDD-Cup-98 and KDD-Cup-99 competitions. The detailed submitted systems of both of these competitions are discussed in detail by [23]. In KDD-Cup-99 competitions, the raw tcpdump data was preprocessed and converted into connection records using Mining Audit data for automated models for ID (MADMAID) framework. The detailed information of the data set is reported in Table 1. KDD-Cup-99 composed of 41 features and 5 different classes, 4 are attacks category and other one is normal. Training data consists of 24 attacks whereas testing data consists of additional 14 attacks. These data are grouped into 4 different categories. The connection records of preprocessed data contain feature information of TCP/IP. These are extracted from both the sender and receiver side with a specific protocol. Each traffic flow carries 100 bytes of information. Among 41 features, 34 features are continuous and 7 features are

Table 1 Description of 10% of KDD-Cup-99 and NSL-KDD data set

Attack category	Description	Data instances - 10 % data			
		KDD-Cup-99		NSL-KDD	
		Train	Test	Train	Test
Normal	Connection records without attacks	97,278	60,593	67,343	9710
DoS	Adversary puts network resources down, so a legitimate user cannot be able to access network resources	391,458	229,853	45,927	7458
Probe	Adversary obtain the statistics of computer network or a system	4107	4166	11,656	2422
R2L	Illegal access from unknown remote computer	1126	16,189	995	2887
U2R	Obtaining the password for root or super user	52	228	52	67
Total		494,021	311,029	125,973	22,544

discrete valued. Features in the range [1–9] are basic features, [10–22] are content features, [23–31] are traffic features with a particular time window and host based features in the range [32–41]. The information of each feature categories are discussed below.

Basic features: the information of packet headers, TCP segments, and UDP datagram were extracted from each session in packet capture (pcap) files by using tcpdump tool.

Content features: with domain knowledge, features were extracted from the payload in each pcap files. Researchers have used many approaches as feature engineering for payload in the last years and recently [43] has introduced a deep learning method without including the feature engineering method. The method performed well even for detecting unknown traffic. 'R2L' and 'U2R' attacks doesn't follow the sequential pattern due to they involve in a single connection. Content feature helps to identify these 'R2L' and 'U2R' attacks.

Time-window of traffic features: features related to 'same host' and 'same service' was extracted in a specific time window of two seconds. However, a few 'Probe' attacks takes more than two seconds of time window. So, 100 connections were considered for both 'same host' and 'same service' and termed as connection-based traffic features.

Two forms of data available in KDD-Cup-99 challenge. (1) Full data set (2) 10% data set. The full data set has many 'DoS' connections compared to others and they were constructed 10% data set by choosing a 'DoS' connection records in a time-window of five seconds. The 10% data of train and test have distinct probability

distribution of connection records and test data has attacks that were not exists in train data. A description of KDD-Cup-99 10% data is placed in Table 1.

2. **NSL-KDD**: NSL-KDD is the refined version of KDD-Cup-99 challenge data set that is contributed by [42]. They used 3 step processes. One was to completely removing the duplicate connection records of KDD-Cup-99 in order to protect classifier from the biased state during training. In addition, the invalid records particularly 136,489 and 136,497 were removed. Second was to select various types of connection records with the aim to effectively detect the attacks in testing. The third one aimed at reducing the false detection rate by balancing the number of connection records between the training and the testing. This remained as an effective data set for misuse or anomaly detection. Though the data set can't be used in real-time network intrusion detection but can be used to effectively benchmark the newly introduced classifiers. The description of NSL-KDD is placed in Table 1. From Table 1, we conclude the NSL-KDD data set is good in compared to KDD-Cup-99. Even NSL-KDD contains more number of 'DoS', 'normal', 'Probe' and 'R2L' connection records. Thus the machine learning classifier gives more importance to these data. In order to overcome, we have added 100 connection records to 'U2R' during training. The train data consist of 1,25,973 connection records and test data consist of 22,544 connection records. Each connection records consist of a vector defined as

$$CV = (f1, f2, \ldots, f41, cl) \tag{24}$$

where f denotes features of length, 41 and each feature, $f \in R$ and cl denotes class label of length 1.

3.1.5 Proposed Architecture - Deep-ID

The proposed architecture namely Deep-ID is shown in Fig. 4. Deep-ID is composed of an input layer, 4 hidden layers and an output layer. Hidden layers are LSTM layers, a network which has more than one LSTM layer are called as stacked LSTM, shown in Fig. 5. Detailed configuration details of proposed LSTM network for binary class and multi class is shown in Tables 2 and 3 respectively. The Deep-ID input layer contains 41 neurons, 4 LSTM layers where each LSTM layer composed of 32 memory blocks. Dropout and batch normalization is used in between the LSTM layers to avoid overfitting and increase the speed of training. When the networks trained without dropout and batch normalization, the network ends up in overfitting and took larger time for training. The penultimate layer of Deep-ID follows fully connected layer. The full connected layer composed of 1 neuron, *sigmoid* non-linear activation function and binary cross entropy as loss function for classifying the connection record into either normal or attack and 5 neurons, *softmax* non-linear activation function and categorical cross entropy for classifying the connection record

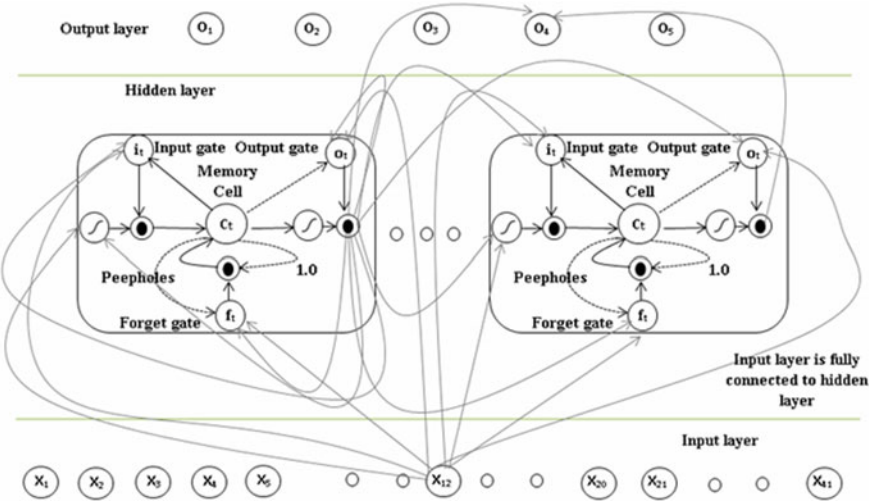
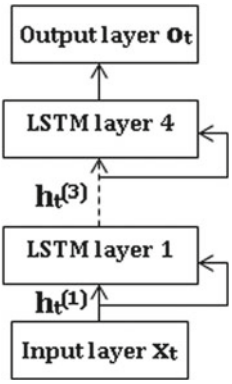


Fig. 4 Deep-ID - All connections and inner units are not shown

Fig. 5 Stacked LSTM



into normal and categorizing an attack into its attack categories. Binary cross entropy and categorical cross entropy is defined as follows

$$loss(pd, ed) = -\frac{1}{N}\sum_{i=1}^{N}[ed_i \log pd_i + (1 - ed_i)\log(1 - pd_i)] \tag{25}$$

$$loss(pd, ed) = -\sum_{x} pd(x)\log(ed(x)) \tag{26}$$

where *pd* is predicted probability distribution, *ed* is expected class, values are either 0 or 1. To minimize the loss of binary cross entropy and categorical cross entropy we used *adam* optimizer via backpropogation.

Table 2 Detailed configuration details of proposed LSTM for classifying connection record as either normal or attack

Layer (type)	Output shape	Param #
lstm_1 (LSTM)	(None, None, 32)	9472
batch_normalization_1	(Batch (None, None, 32)	128
dropout_1 (Dropout)	(None, None, 32)	0
lstm_2 (LSTM)	(None, None, 32)	8320
batch_normalization_2	(Batch (None, None, 32)	128
dropout_2 (Dropout)	(None, None, 32)	0
lstm_3 (LSTM)	(None, None, 32)	8320
batch_normalization_3	(Batch (None, None, 32)	128
dropout_3 (Dropout)	(None, None, 32)	0
lstm_4 (LSTM)	(None, None, 32)	8320
batch_normalization_4	(Batch (None, None, 32)	128
dropout_4 (Dropout)	(None, None, 32)	0
dense_1 (Dense)	(None, 1)	33
activation_1 (Activation)	(None, 1)	0
Total params: 34,977		
Trainable params: 34,721		
Non-trainable params: 256		

Table 3 Detailed configuration details of proposed LSTM for classifying connection record as either normal or attack and categorizing an attack into attack categories

Layer (type)	Output shape	Param #
lstm_1 (LSTM)	(None, None, 32)	9472
batch_normalization_1	(Batch (None, None, 32)	128
dropout_1 (Dropout)	(None, None, 32)	0
lstm_2 (LSTM)	(None, None, 32)	8320
batch_normalization_2	(Batch (None, None, 32)	128
dropout_2 (Dropout)	(None, None, 32)	0
lstm_3 (LSTM)	(None, None, 32)	8320
batch_normalization_3	(Batch (None, None, 32)	128
dropout_3 (Dropout)	(None, None, 32)	0
lstm_4 (LSTM)	(None, None, 32)	8320
batch_normalization_4	(Batch (None, None, 32)	128
dropout_4 (Dropout)	(None, None, 32)	0
dense_1 (Dense)	(None, 1)	165
activation_1 (Activation)	(None, 1)	0
Total params: 35,109		
Trainable params: 34,853		
Non-trainable params: 256		

3.1.6 Results and Observations

In this work, various classical machine learning algorithms and deep learning architectures are trained on NSL-KDD data set. During training to monitor the validation accuracy, the train data is randomly divided into 70% train and 30% valid data sets. The validation data helps to monitor the train accuracy. During testing the trained model are evaluated using the test data of NSL-KDD. The detailed reports are reported in Tables 4, 5, 6, 7 and 8. To identify the optimal parameters for all deep learning architectures, hyper parameter tuning approach is followed. The experiments are done on the following cases

- Full Binary classification: Classifying the connection record as either normal or attack using full feature set.
- Minimal Binary classification: Classifying the connection record as either normal or attack using minimal feature set.
- Full multi class classification: Classifying the connection record as either normal or attack and categorizing an attack into 'DoS', 'Probe', 'U2R' and 'R2L' categories using full feature set.

Table 4 Detailed test results of MLP, RNN, LSTM, GRU and IRNN on minimal feature sets of NSL-KDD for binary classification

Architecture	Accuracy	Precision	Recall	F1-score	TPR	FPR	Loss
LSTM 1 layer	0.92	0.849	0.991	0.914	0.9918	0.151	0.59
LSTM 2 layers	0.964	0.931	0.990	0.959	0.9917	0.069	0.26
LSTM 3 layers	0.967	0.932	0.995	0.962	0.996	0.068	0.08
LSTM 4 layers	0.978	0.958	0.992	0.975	0.994	0.416	0.17
RNN 1 layer	0.946	0.891	0.995	0.940	0.996	0.108	0.39
RNN 2 layers	0.951	0.901	0.996	0.946	0.996	0.099	0.16
RNN 3 layers	0.968	0.931	1.0	0.964	0.999	0.687	0.07
RNN 4 layers	0.989	0.976	1.0	0.988	0.999	0.024	0.01
GRU 1 layer	0.939	0.885	0.987	0.933	0.9889	0.115	0.48
GRU 2 layers	0.940	0.880	0.995	0.934	0.9959	0.119	0.36
GRU 3 layers	0.973	0.944	0.996	0.969	0.9967	0.056	0.09
GRU 4 layers	0.989	0.974	1.0	0.987	0.9998	0.258	0.02
IRNN 1 layer	0.926	0.859	0.992	0.921	0.993	0.141	0.86
IRNN 2 layers	0.896	0.814	0.984	0.891	0.985	0.186	0.51
IRNN 3 layers	0.914	0.838	0.992	0.909	0.993	0.162	1.04
IRNN 4 layers	0.996	0.997	0.995	0.996	0.994	0.003	0.07
MLP 1 layer	0.799	0.717	0.879	0.790	0.889	0.283	2.53
MLP 2 layers	0.811	0.701	0.879	0.817	0.977	0.299	1.16
MLP 3 layers	0.861	0.766	0.977	0.859	0.978	0.234	2.17
MLP 4 layers	0.866	0.768	0.988	0.864	0.988	0.232	1.97

Table 5 Detailed test results of classical machine learning algorithms for binary class classification (LR - Logistic regression, NB - Naive Bayes, KNN - K-Nearest Neighbors, DT - Decision Tree, AB - Ada Boost and RF - Random forest)

Algorithm	Accuracy	Precision	Recall	F1-score	TPR	FPR
LR	0.692	0.590	0.932	0.723	0.410	0.908
NB	0.703	0.601	0.925	0.728	0.904	0.399
KNN	0.782	0.669	0.975	0.794	0.971	0.331
DT	0.720	0.62	0.905	0.736	0.890	0.380
AB	0.774	0.662	0.970	0.787	0.965	0.338
RF	0.728	0.622	0.935	0.747	0.920	0.378
SVM-linear	0.788	0.677	0.971	0.798	0.968	0.323
SVM-rbf	0.795	0.687	0.963	0.802	0.960	0.313

Table 6 Detailed test results of MLP, RNN, LSTM, GRU and IRNN on minimal feature sets of NSL-KDD for multi class classification

Architecture	Normal		DoS		Probe		U2R		R2L		Accuracy	Loss
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR		
LSTM 4F	0.9999	0.013	0.937	0.106	0.812	0.065	0.164	0.001	0.216	0.002	0.862	0.81
LSTM 8F	1.0	0.076	0.773	0.045	0.901	0.106	0.313	0.002	0.303	0.003	0.827	1.21
LSTM 12F	0.997	0.124	0.747	0.0176	0.898	0.093	0.45	0.001	0.429	0.002	0.832	2.20
RNN 4F	0.999	0.003	0.925	0.0591	0.842	0.109	0.388	0.003	0.192	0.001	0.859	0.65
RNN 8F	1.0	0.064	0.807	0.041	0.77	0.103	0.358	0.008	0.388	0.003	0.835	1.22
RNN 12F	0.996	0.111	0.809	0.022	0.879	0.088	0.493	0.001	0.344	0.002	0.84	1.20
GRU 4F	0.999	0.021	0.909	0.106	0.797	0.068	0.343	0.001	0.232	0.003	0.853	0.95
GRU 8F	0.998	0.062	0.833	0.081	0.742	0.085	0.194	0.0007	0.338	0.001	0.833	1.05
GRU 12F	0.996	0.0926	0.7802	0.024	0.848	0.0763	0.4477	0.001	0.529	0.0149	0.849	1.63

F denotes Features and all architectures 4 hidden layers

Table 7 Detailed test results of classical machine learning algorithms for multi class classification (LR - Logistic regression, NB - Naive Bayes, KNN - K-Nearest Neighbors, DT - Decision Tree, AB - Ada Boost and RF - Random forest)

Algorithm	Normal		DoS		Probe		U2R		R2L		Accuracy
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	
LR	0.926	0.468	0.641	0.12	0.171	0.020	0.0	0.0	0.0	0.0	0.635
NB	0.2686	0.083	0.761	0.165	0.539	0.02	0.806	0.258	0.365	0.104	0.4777
KNN	0.976	0.334	0.726	0.031	0.589	0.0375	0.179	0.0001	0.085	0.017	0.741
DT	0.972	0.350	0.754	0.02	0.732	0.033	0.299	0.002	0.007	0.002	0.754
RF	0.976	0.399	0.7680	0.014	0.609	0.019	0.254	0.0003	0.001	0.0	0.747
AB	0.936	0.37	0.796	0.147	0.11	0.009	0.0	0.0	0.0	0.0	0.685

Table 8 Detailed test results of MLP, RNN, LSTM, GRU and IRNN on minimal feature sets of NSL-KDD for multi class classification

Architecture	Normal		DoS		Probe		U2R		R2L		Accuracy	Loss
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR		
LSTM 1 layer	0.985	0.124	0.782	0.007	0.968	0.121	0.403	0.0009	0.164	0.002	0.814	1.38
LSTM 2 layers	0.986	0.126	0.782	0.007	0.978	0.095	0.493	0.026	0.134	0.001	0.812	1.34
LSTM 3 layers	0.991	0.048	0.800	0.0158	0.919	0.099	0.388	0.006	0.394	0.025	0.845	1.26
LSTM 4 layers	0.994	0.053	0.841	0.012	0.934	0.071	0.299	0.0003	0.680	0.001	0.896	0.76
IRNN 1 layer	0.995	0.0836	0.828	0.073	0.839	0.113	0.0	0.0	0.002	0.004	0.799	2.04
IRNN 2 layers	0.974	0.232	0.774	0.015	0.854	0.091	0.0	0.0	0.023	0.001	0.776	2.20
IRNN 3 layers	0.976	0.149	0.777	0.059	0.816	0.065	0.0	0.0	0.343	0.007	0.812	1.66
IRNN 4 layers	0.976	0.149	0.777	0.059	0.816	0.0645	0.0	0.0	0.343	0.007	0.783	3.49
RNN 1 layer	0.988	0.129	0.777	0.004	0.973	0.112	0.029	0.002	0.206	0.003	0.818	2.44
RNN 2 layers	0.978	0.09	0.781	0.012	0.944	0.127	0.478	0.006	0.206	0.009	0.814	1.61
RNN 3 layers	0.980	0.075	0.772	0.008	0.97	0.114	0.448	0.002	0.325	0.022	0.828	2.01
RNN 4 layers	0.985	0.113	0.780	0.014	0.939	0.104	0.433	0.001	0.33	0.002	0.83	2.34
GRU 1 layer	0.983	0.102	0.771	0.020	0.891	0.131	0.075	0.008	0.174	0.003	0.801	2.37
GRU 2 layers	0.99	0.101	0.839	0.005	0.988	0.09	0.388	0.002	0.305	0.001	0.855	1.49
GRU 3 layers	0.980	0.113	0.775	0.014	0.934	0.103	0.627	0.002	0.364	0.003	0.831	2.04
GRU 4 layers	0.980	0.112	0.784	0.009	0.966	0.109	0.328	0.003	0.293	0.002	0.828	2.24
MLP 1 layer	0.975	0.0928	0.768	0.0315	0.775	0.127	0.0	0.0	0.212	0.027	0.789	2.91
MLP 2 layers	0.982	0.091	0.777	0.014	0.922	0.111	0.0	0.0	0.261	0.026	0.817	2.89
MLP 3 layers	0.972	0.254	0.759	0.023	0.785	0.082	0.0	0.0	0.035	0.001	0.764	2.64
MLP 4 layers	0.973	0.390	0.658	0.010	0.732	0.056	0.0	0.0	0.0	0.0	0.721	2.11

- Minimal multi class classification: Classifying the connection record as either normal or attack and categorizing an attack into 'DoS', 'Probe', 'U2R' and 'R2L' categories using minimal feature set.

In all the test cases, the deep learning architectures performed well in comparison to the classical machine learning. Moreover, the performance obtained by deep learning is superior over classical neural network, MLP. The performances obtained by all the deep learning architectures are almost closer to each other. Thus voting methodology can be employed in order to increase the attack detection rate. Source code for all deep learning architectures and classical machine learning are available publically.[5]

3.1.7 Conclusion and Future Works

In this work, the performance of classical machine learning algorithms and deep learning architectures are evaluated for intrusion detection with the publically available benchmark data set. In all the experiments, deep learning architectures have performed well in comparison to the classical machine learning algorithms. These

[5] https://github.com/vinayakumarr/Network-Intrusion-Detection.

experiments only suggest the proof of concept, applying the same on real time intrusion detection will be remained as one of the significant direction towards future work. The raw data which are captured through different sensors are unlabeled and uncategorized. These data are ideally suited for deep learning architectures. In real world network, the data collected from ICT systems is huge generally undergo big data challenges. Big data Analytics has become an important paradigm for many domains mainly cyber security. Big data analytics help to process and analyze and get insights from very large volume of network traffic data sets. A key advantage of deep learning for big data analytics is the analysis and learning of massive amounts of unsupervised data, making it a valuable tool for big data analytics where raw data is largely unlabeled and uncategorized.

3.2 DeepTrafficNet: Deep Learning Framework for Network Traffic Analysis

The network traffic is increasing exponentially. Identifying and monitoring network traffic is of significant task towards identifying the malicious activities. Most of the existing systems for network traffic identification are based on hand crafted features and they are inaccurate and easily evadable. Extracting optimal hand crafted features in feature engineering requires extensive domain knowledge and it is time consuming approach. These methods are completely invalid for unknown protocol and applications. Thus, this work proposes DeepTrafficNet, a scalable framework based on Deep learning which replaces the manual feature engineering with machine learned ones and these are harder to be fooled. The performance of various deep learning architectures are evaluated for 3 different network traffic use cases such as Application network traffic classification, Malicious traffic classification and Malicious traffic detection with the public and privately collected data set. The performances obtained by various deep learning architectures are closer and moreover, combination of convolutional neural network (CNN) and long short term memory (LSTM) pipeline performed well in all the 3 network traffic use cases. This is due to the fact that the CNN-LSTM has the capability to capture both spatial and temporal features.

3.2.1 Introduction and Existing Methods for Traffic Analysis

The Internet is widely popular nowadays. Primarily every communication happens through Internet. It is the interconnection of many computer networks and uses TCP/IP suite to link the devises worldwide. Internet protocol suite is a framework described by the Internet standards. This is a model architecture that divides methods into a layered system of protocols. Identifying the traffic in networks is an important task in Internet. Most commonly used approaches are port based, signature based and statistical features based network traffic identification. Port based method is the

initially employed method which relies on predefined specific port numbers [44]. As new protocols which have been designed every day without following the rule of port registration, so the rate of error in protocol identification is growing higher. In the year 2002 on wards, signature based method was used. The signatures are hexadecimal values or it can be a sequence of strings which varies according to each application [45]. This is a simple method and the performance relatively high when compared to the port based protocol classification. The error rate is lesser than 10% and these methods are more effective [9]. When specification of a protocol changes or a new one is designed, it is time consuming to start over for finding valuable signatures. Deep packet inspection (DPI) is one more technique used for network traffic classification. nDPI is publically available which is based on DPI. This can detect standard application effectively and generates issues with rare application and encrypted applications. Statistical features and machine learning based classification is an another method [46]. These are more popular in recent days. This relies on the statistical features in transmission of the traffic, such as the time interval between packets, packet size, repeating pattern, and so on. Then these features are fed into classical machine learning algorithms like Naive Bayes, Decision tree, Support vector machine and Neural network [47]. These methods can be executed in real time or near real time, but the key point is that the time and experience that we need to select the appropriate features which is to be fed into machine learning algorithm. The attacks related to port abuse, random port usage and tunneling is increasing rapidly day by day. These attacks are not fully able to eradicate which causes massive security problems in networks. These systems require extensive domain knowledge and more importantly they are not accurate.

In recent days, the application of deep learning is used for traffic identification. Deep learning has the capability to learn the optimal features on taking raw input samples [3]. Payload bytes information is passed into deep learning architectures [43]. In [43] showed the first thousand bytes is sufficient to effectively identify protocols. In this literature, they collected TCP flow data from internal network and the full payload is extracted from every packets. Then they joined the payload bytes for every TCP session. A byte is represented by an integer from 0 to 255. It was then normalized [0, 1] scale. The length of each payload sequence was 1,000. They picked up about 0.3 million records after pruning data for experiments. The number of protocol types in the data taken was 58. The fully extracted payloads of the packets collected were made into an image format. This means that the each byte was represented as a pixel. Deep learning has performed well in computer vision tasks and leveraged the same to network traffic analysis [43]. Since the Deep learning methods have the advantage of automatic feature extraction and selection this will be the best method for information security. In [43] proposes a method of feature extraction by using stacked Autoencoder. The main advantage of using Autoencoder is that we can feed both supervised and unsupervised data; of course the labeled data will give more precise features. Along with that another advantage is that, this Autoencoder method will achieve the goal of dimensionality reduction also. Here the features are mapped to new space and the redundant information is also filtered. Following, the main aim of this work are set as follows

- Application of deep learning techniques is leveraged for traffic identification tasks such as application traffic classification, Malicious traffic classification, and Malicious traffic detection.
- To handle data in real-time, a highly scalable framework is constructed which has the capability to collect very large amount of data and the capability can be increased by adding additional resources to the existing system architecture.

3.2.2 Description of Data Set

There are two types of data sets are used, one is privately collected data set named as AmritaNet and second one is UNSW-NB15. UNSW-NB15 data set is used in this work [48]. This is composed of 1 million bi-directional flows of application and Malicious traffic. A network flow is a communication between two end points on a network. These end points exchange packets during the conversation. Packets composed of two sections. One is header section which provides information about the packet and second one is payload which contains the exact message. A flow can be unidirectional or bi-directional. The data set was provided in pcap format from 22nd of January 2015 and 17th of February 2015. The samples of train and test data sets are disjoint. The most commonly used 5 applications are considered and remaining flows are labeled as unknown. There are three different data set is formed, one is for Application network traffic classification, second one is for Malicious traffic classification and third one is for Malicious traffic identification. The data samples from 22 Jan 2015 are used for training and data samples from 17 Feb 2015 is used for testing and validation. The application protocols are DNS, FTP, Mail, SSH, BGP and unknown. The malicious classes are 'DoS', Fuzzers, Exploits and other malicious. For Application network traffic classification, train data of DNS, FTP, Mail, SSH, BGP, unknown contains 2,600, 1,200, 2,801, 1,641, 901 and 600 samples respectively and test data of DNS, FTP, Mail, SSH, BGP, unknown contains 1,200, 700, 1,350, 754, 352, 300 respectively. For Malicious traffic classification, train data of 'DoS', Fuzzers, Exploits, other malicious contains 2,600, 3,554, 4,053, 2,754 and test data of 'DoS', Fuzzers, Exploits, other malicious contains 1,100, 1,432, 1,578, 2,034. For Malicious traffic detection, train data of malicious, non-malicious contains 12,961, 11,042 and test data of malicious and non-malicious contains 2,408, 2,147 samples. AmritaNet composed of 52,010 network flows for training and 35,100 for testing. Train composed of 30,000 legitimate network flows and 22,010 Malicious traffic flows and test composed of 12,010 legitimate network flows and 10,000 Malicious traffic flows.

3.2.3 Proposed Architecture - DeepTrafficNet

An overview of proposed architecture, DeepTrafficNet for application traffic classification, Malicious traffic classification, and Malicious traffic detection is shown in Fig. 6. We developed a scalable framework to handle large amount of network

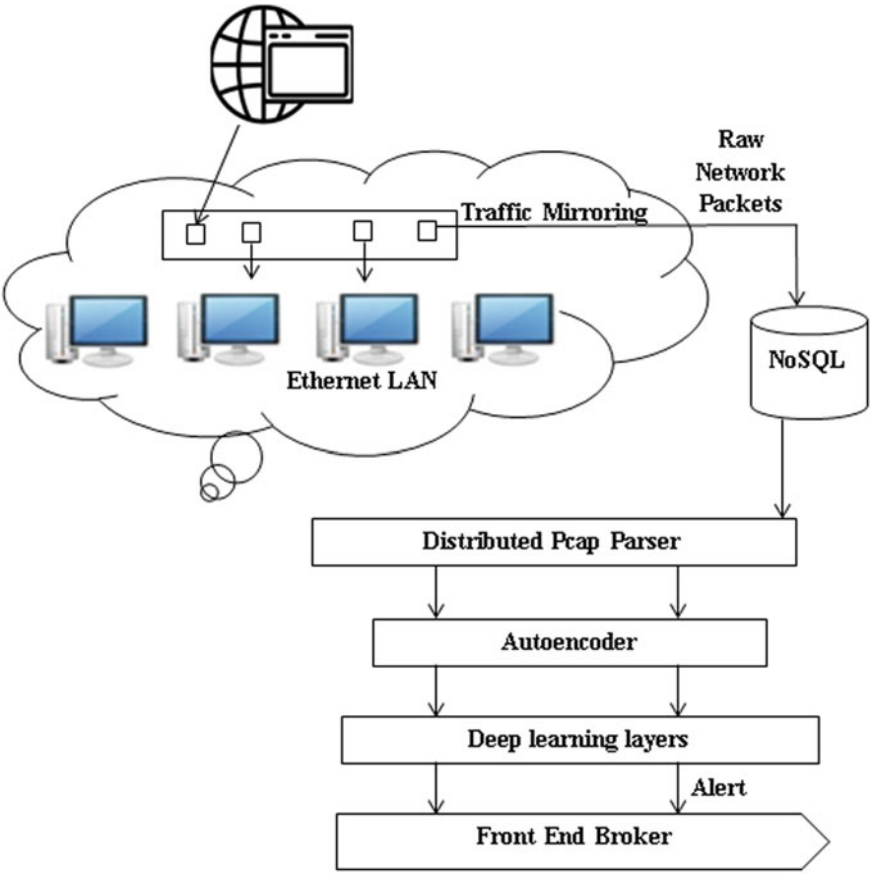


Fig. 6 Proposed architecture - DeepTrafficNet

traffic data and used distributed preprocessing, distributed data base and deep learning architectures for analysis [9]. The framework composed of data set collection, preprocessing, classification section. In this work, we have used publically avilable data but the proposed framework has the capability to collect data inside an Ethernet LAN. In preprocessing, by following the method of [43], data set is formed. Each flow contains information of length 1,000. This is passed into classification section. Classification composed of Autoencoder and deep learning architectures. Autoencoder facilitates to capture the important features. Finally, these features are passed into deep learning architectures for classification. By running 3 trials of experiments, the Autoencoder feature dimension is set into 512, the learning rate is set to 0.02, *adam* as an optimizer, batch size to 64 and 128 hidden units for recurrent structures. For CNN, 64 filters, filter length 3 and maxpooling length 2 is used. CNN contains two fully connected layers, one is with 64 followed by dropout 0.02 and another one is fully connected layer for classification. In CNN-LSTM network, LSTM composed

of one hidden recurrent layer containing 50 memory blocks. Last fully connected layer used *softmax* non-linear activation function with categorical cross entropy loss function.

3.2.4 Results and Observations

The data set is passed into Autoencoder. It contains 7 hidden layers. Input layer contains data of 1024 dimension and mapped into 896, 768, 640, 512, 640, 768, and 896. The data dimension is reduced into 128. During optimization, mean squared error is used as objective function to calculate the reconstruction error between the input and output layer. This is solved using gradient descent optimizer. The newly formed feature vector of length 512 served as input for other deep learning architectures. The data set is randomly divided into training and testing data set. 70% of the data is used for training and 30% is used for testing. Each deep learning architecture is trained on the train data and tested on the test data set. The detailed results for Application network traffic classification, Malicious traffic classification and Malicious traffic detection are reported in Table 9. To know the effectiveness of Autoenocder, two sets of experiments are run on Data set 1. First experiment is with Autoencoder followed by different deep learning architectures and second experiment is with only deep learning architectures. The detailed results are reported in Table 10. Experiments with Autoencoder for feature learning showed best performances in compared to the deep learning architectures without Autoenocder. This is primarily due to the fact that Autoencoder facilitates to capture the optimal features which can be effectively used to distinguish between the legitimate and Malicious traffic.

3.2.5 Conclusion and Future Works

In this work, the applications of deep learning architectures are employed for network traffic identification. The data set is collected in a private environment. Autoencoder is used for feature learning and followed by CNN, LSTM, RNN and CNN-LSTM for classification. These methods have been employed for Application network traffic classification, Malicious traffic classification and Malicious traffic detection. Most of the methods performed well, and they have only marginal differences among them in all the different tasks. Thus to improve the performances, voting approach can be used. Moreover, the proposed method works well on automatic feature learning which completely avoids the classical feature engineering methods. In this work, we considered only less number of protocols and applications with very simple deep learning architecture. Moreover, the inner details of Autoencoder and deep learning architectures are not explained in detail. These are remained as directions towards future works.

Table 9 Summary of test results for UNSW-NB15

Application network traffic classification	CNN		RNN		LSTM		CNN-LSTM	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
DNS	1.0	0.921	1.0	0.924	1.0	0.948	1.0	0.918
FTP	0.958	0.992	0.976	1.0	0.974	1.0	0.984	1.0
Mail	0.876	0.965	0.889	0.973	0.886	0.973	0.889	0.982
SSH	0.999	0.999	0.998	0.999	0.999	0.997	0.991	0.998
BGP	0.999	0.956	0.998	0.912	0.991	0.942	0.998	0.992
unknown	0.801	0.804	0.812	0.803	0.852	0.841	0.885	0.741
Malicious traffic classification								
DoS	0.9950	0.9996	0.9981	0.9997	0.998	1.0	0.881	1.0
Fuzzers	0.9781	0.9992	0.9988	1.0	0.991	1.0	1.0	0.994
Exploits	0.9997	0.9998	0.884	0.956	0.992	0.894	0.997	0.991
Other malicious	0.804	0.814	0.821	0.841	0.856	0.803	0.894	0.991
Malicious traffic detection								
Accuracy	0.814		0.852		0.874		0.892	

Table 10 Summary of test results for Data set 1

Architecture	Without Autoencoder	With Autoencoder
CNN	0.784	0.795
RNN	0.798	0.812
LSTM	0.827	0.833
CNN-LSTM	0.841	0.864

3.3 Deep-Droid-Net (DDN): Applying Deep Learning Approaches for Android Malware Detection

This sub module presents deep learning based method for static Android malware detection. Deep learning architectures implicitly learn the optimal feature representation from raw opcode sequences extracted from disassembled Android programs. The various numbers of experiments are run for various deep learning architectures with Keras Embedding as the opcode representation method. The Keras embedding facilitates to preserve the sequence order of opcode and to learn the semantic and contextual similarity. The performance of various deep learning architectures are closer and the combination of convolutional neural network (CNN) and long short term memory (LSTM) showed highest accuracy 0.968.

3.3.1 Introduction and Existing Methods for Android Malware Detection

The modern amelioration in mobile technology has paved the way for colossal growth in smart phones as being saved as 'tiny computers'. The IDC report,[6] shows that Android shines among the other mobile platforms as the most widely used mobile platform and it currently shines at the market with the market share of approximately 87.6% due to its ease of use, low-cost, open source, user-friendly and portability nature as the characteristics.[7] Smartphones with Internet services has changed people's daily life activities through numerous applications like social network apps, gaming apps, information exchange apps, cloud storage apps, financial transactions and banking apps etc. Though Android has all these advantages but these benefits have endangered the development of malware. TrendLabs announced that top ten installed malicious Android apps contained around 71,250 installations in 3Q 2012 security roundup.[8] Also, Cisco reported that Android has encountered 99% new malware apps in 2014.[9] The outpouring usage of smart phones over personal computers has paved the way for malware writers to shift their focus completely on creating malware for smart phones. Additionally distribution of malware on the smart phones is easier than on PC due to the lack of well-established security mechanisms like intrusion detection systems (IDS), firewalls, encryption anti-virus and other end-point based security measures available on PC.

As Android OS becomes more popular, the target on the Android OS will also grow. This situation has led to birth of so many Android malware. Signature and heuristic based methods fall in the rule based system. Rule based system depends on signature data base. This data base requires to be updated on and off by the domain experts as and when a new malware appears. It is a good solution to trace already existing malware but it cannot detect the variants of new malware. Although signature based and heuristic based detection is significant, application of self-learning systems for the analysis and detection of ever increasing Android malware is being studied. Self-learning system consists of data mining, machine learning and deep learning architectures. These techniques give fresh sensing capabilities for Android malware detection which can detect new types of malware. Further, these approaches are capable of detecting the variants of already existing malware or even entirely new malware as well.

Application of machine learning techniques to detect Android malware is becoming popular as it has high scope for research. Its methods greatly depend on the feature engineering, feature selection and feature representation methods. The set of features with a corresponding class is used to train a model to create a separating plane between the benign and malicious apps. The separating plane helps to trace malware and it categorizes into respective malware family. Machine learning

[6] http://www.idc.com/.

[7] https://securelist.com/.

[8] www.trendmicro.com.

[9] http://www.cisco.com/web/offer/gistty2asset/Cisco2014ASR.pdf.

algorithms works on a set of features. Static and Dynamic analysis are two most commonly used methods for feature extraction from Android OS.

Static analysis collects set of features from apps by unpacking or disassembling them without the runtime execution and dynamic analysis monitor the run-time execution behavior of apps such as system calls, network connections, memory utilization, power consumption, user interactions, etc. The hybrid analysis is a two-step process. In this initially static analysis is performed before the dynamic analysis which results in less computational cost, lower resource utilization, light-weight and less time-consuming in nature. This hybrid analysis method is mostly used by anti-virus providers for the smart phones to enhance the malware detection rate.[10]

Machine learning, neural networks and deep learning are all identical terms that find place in the discussion about artificial intelligence. There terms appear to be some confusion, in fact deep learning is a sub field of machine learning which evolved from neural networks. It simply duplicate the way human brain functions, that is, processing data, creating patterns while making decisions. The classical machine learning creates algorithms with data in a linear way, but deep learning consists of several neurons which are interrelated like a web and it handles data through a non-linear way.

Deep learning is applied in various problems found in natural language processing, computer vision, robotic planning etc. It has exhibited high performance in artificial intelligence tasks by taking raw input samples as input [3]. Recently, this has been applied towards static Android malware detection [49]. These methods have extracted the opcode sequence representation from .apk files and passed to embedding layer to learn the semantic information among them and again passed into deep learning layers to learn the optimal feature representation. These features were classified using the fully-connected layer with non-linear activation function. Following in this sub module

- The performance of various deep learning architectures are evaluated for static Android malware detection.
- This work proposes Deep-Droid-Net (DDN) which uses deep learning and NLP text representation to learn the optimal feature representation from the benign and malicious apps. The proposed method doesn't relies on feature engineering and it is more robust in an adversarial environment in compared to classical machine learning based Android malware detection.

3.3.2 Details of Data Set and Opcode Representation Using Keras Embedding

There is no just once source that you can refer to while listing out the malware families. It was necessary that there were a variety of different samples that needed to be collected. Over the time as malware have grown it has become increasingly difficult

[10] http://www.avg.com/us-en/avg-software-technology.

for people even after collaborating with one another to keep track of the growing infection of malware. The malicious Android apps are selected from drebin data set.[11] The benign applications are crawled from Google play store. The benign samples are rechecked in virusTotal to find out the benign apps are malware free. Additionally, the malicious samples are collected privately. The train data set composed of 4,000 benign samples and 3,841 malware samples and test data composed of 2,410 benign samples and 2,104 malware samples.

The Android apps, .apk are disassembled to produce .smali using Baksmali.[12] Each .apk files are transformed to more than one .smali files. This .smali file has human readable Dalvik byte code information. Then only raw opcodes are extracted from each .smali files and combined together to form a single sequence of opcodes. The opcode sequence $X = \{x_1, x_2, \ldots, x_n\}$ are transformed into one-hot vectors, where x_n is the one-hot vector for the opcode in the sequence. In Dalvik, only 218 default defined codes. Each one-hot vector is multiplied with a weight matrix $W_E \in R^{D \times k}$ to project each opcodes in X into an embedding space.

$$p_i = x_i W_E \tag{27}$$

p denotes program representation and it is a projection of all opcodes in X. W_E denotes weight matrix which is initialized randomly at initial time and later updated by backpropogation.

3.3.3 Proposed Architecture - Deep-Droid-Net (DDN)

An overview of proposed christened Deep-Droid-Net (DDN) model is shown in Fig. 7. This composed of 3 sections. One is embedding layer which helps to transform the opcode into numerical representation. Embedding layer size is set to 64. During backpropogation, embedding layer learn the proper weights which best facilitates to distinguish between the benign and malicious apps. The embedding layer features are passed into different deep learning layers such as RNN, LSTM, CNN CNN-RNN and CNN-LSTM models. For all these models the learning rate is set to 0.01, batch size to 64 and *ADAM* as an optimizer. Convolution layer composed of 64 filters of length 3, pooling length of 2. RNN and LSTM has 64 units and memory blocks respectively. CNN contains 2 fully connected layers, the first fully connected layer contains 64 units and followed by 1 unit with *sigmoid* as non-linear activation function for classification. RNN and LSTM contains only one fully connected layer with *sigmoid* non-linear activation function for classification. In CNN-RNN and CNN-LSTM, the reduced features of CNN layers are passed again into RNN and LSTM layer respectively instead of fully connected layer. The RNN and LSTM layer contains 32 units and memory blocks followed by a fully connected layer with *sigmoid* non-linear activation function for classification. All the deep learning

[11] https://www.sec.cs.tu-bs.de/~danarp/drebin/.

[12] https://github.com/JesusFreke/smali.

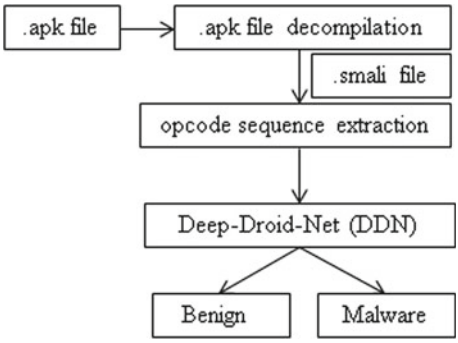


Fig. 7 Overview of Deep-Droid-Net (DDN)

models used binary cross entropy as loss functions. This is mathematically defined as follows

$$loss(p, e) = -\frac{1}{N}\sum_{i=1}^{N}[e_i \log p_i + (1 - e_i)\log(1 - p_i)] \tag{28}$$

where p is a vector of predicted vector for testing data set, e is a vector of expected class label, values are either 0 or 1.

3.3.4 Results and Observations

Initially, the data set is randomly split into two sets, 70% for training and 30% for testing. To evaluate the performance of deep learning architectures various trials of experiments are run. The hyper parameter technique is not followed for selecting the best parameters for deep learning architectures and embedding layer. For each model, the results in terms of Accuracy, Precision, Recall and F1-score are reported in Table 11. The CNN-LSTM model has performed well in comparison to the other models.

Table 11 Detailed result of various deep learning methods

Model	Accuracy	Precision	Recall	F1-score
RNN	0.896	0.814	0.984	0.891
CNN [49]	0.946	0.891	0.995	0.940
LSTM	0.951	0.901	0.996	0.946
CNN-RNN	0.946	0.891	0.995	0.940
CNN-LSTM	0.968	0.931	1.0	0.964

3.3.5 Conclusion

This sub module has proposed Deep-Droid-Net (DDN) which is a combination of CNN and LSTM model for static Android malware detection. DDN uses Keras embedding which facilitate to transform opcode to numeric representation. This completely avoids classical manual feature engineering methods which are followed for classical machine learning algorithms. The performance of various deep learning architectures such as LSTM, CNN and CNN-LSTM are evaluated. CNN-LSTM performed well in comparison to the CNN and LSTM. This is primarily due the reason that it can extract the long-range features related to sequence and temporal information of opcode using CNN and LSTM respectively. The proposed methods act as a general method and can be applied on any other malware detection with a small change to the network architecture.

4 Conclusion and Future Works

Machine learning and deep learning approaches are largely used in various domains and in recent days, these approaches are being used in cyber security also. Thus evaluating various algorithms on cyber security is an important task which helps to identify the adequate algorithm which achieves best performance. Deep learning is a complex model of machine learning which have the capability to obtain optimal feature representation by itself. To leverage the application of deep learning architecture, in this work we have evaluated the performances of various deep learning architectures on cyber security use cases such as intrusion detection, network traffic analysis and Android malware detection. Deep learning architecture have achieved superior performance on all the use cases and additionally outperformed the classical machine learning algorithms on both the binary and multi class classification in intrusion detection. The reported results of deep learning architectures are further enhanced by carefully following hyper parameter tuning approach. Deep learning is an early stage, thus the performance has to be evaluated in an adversarial environment. This is remained as one of the significant direction towards future work in evaluating the deep learning architectures for cyber security use cases. From our study, we found that the deep learning architectures can be used along with the classical machine learning and rule based algorithms to enhance the detection rate of cyber-attacks, cyber threat and malware.

Acknowledgements This research was supported in part by Paramount Computer Systems and Lakhshya Cyber Security Labs. We are grateful to NVIDIA India, for the GPU hardware support to research grant. We are also grateful to Computational Engineering and Networking (CEN) department for encouraging the research.

References

1. Jordan MI, Mitchell TM (2015) Machine learning: trends, perspectives, and prospects. Science 349(6245):255–260
2. Buczak AL, Guven E (2016) A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun Surv Tutorials 18(2):1153–1176
3. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436
4. Vinayakumar R, Soman KP, Poornachandran P (2018) Evaluating deep learning approaches to characterize and classify malicious URLs. J Intell Fuzzy Syst 34(3):1333–1343
5. Vinayakumar R, Soman KP, Poornachandran P (2018) Detecting malicious domain names using deep learning approaches at scale. J Intell Fuzzy Syst 34(3):1355–1367
6. Vinayakumar R, Soman KP (2018) DeepMalNet: evaluating shallow and deep networks for static PE malware detection. ICT Express 4(4):255–258
7. Vinayakumar R, Soman KP, Poornachandran P (2017) Applying convolutional neural network for network intrusion detection. In: 2017 International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 1222–1228
8. Vinayakumar R, Soman KP, Poornachandran P (2017) Applying deep learning approaches for network traffic prediction. In 2017 International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 2353–2358
9. Vinayakumar R, Poornachandran P, Soman KP (2018) Scalable framework for cyber threat situational awareness based on domain name systems data analysis. In: Big data in engineering applications. Springer, Singapore, pp 113–142
10. Vinayakumar R, Soman KP, Poornachandran P (2017) Deep encrypted text categorization. In: 2017 International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 364–370
11. Vinayakumar R, Soman KP, Poornachandran P (2017) Evaluating effectiveness of shallow and deep networks to intrusion detection system. In: 2017 International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 1282–1289
12. Vinayakumar R, Soman KP, Poornachandran P (2017) Secure shell (ssh) traffic analysis with flow based features using shallow and deep networks. In: 2017 International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 2026–2032
13. Vinayakumar R, Soman KP, Poornachandran P (2017) Evaluating shallow and deep networks for secure shell (ssh) traffic analysis. In: 2017 International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 266–274
14. Vinayakumar R, Soman KP, Poornachandran P (2017) Long short-term memory based operation log anomaly detection. In: 2017 International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 236–242
15. Vinayakumar R, Soman KP, Poornachandran P (2017) Deep android malware detection and classification. In: 2017 International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 1677–1683
16. Mohan VS, Vinayakumar R, Soman KP, Poornachandran P (2018) Spoof net: syntactic patterns for identification of ominous online factors. In: 2018 IEEE security and privacy workshops (SPW). IEEE, pp 258–263
17. Vinayakumar R, Soman KP, Poornachandran P (2017) Evaluation of recurrent neural network and its variants for intrusion detection system (IDS). Int J Inf Syst Model Des (IJISMD) 8(3):43–63
18. Vinayakumar R, Barathi Ganesh HB, Anand Kumar M, Soman KP (2018) Deepanti-phishnet: applying deep neural networks for phishing email detection. Cenaisecurity@iwspa-2018, pp 40–50. http://ceur-ws.org/Vol-2124/paper9
19. Vinayakumar R, Soman KP, Poornachandran P, Mohan VS, Kumar AD (2019) ScaleNet: scalable and hybrid framework for cyber threat situational awareness based on DNS, URL, and email data analysis. J Cyber Secur Mobility 8(2):189–240
20. Anderson JP (1980) Computer security threat monitoring and surveillance. In: Technical report. James P Anderson co., Fort Washington, Pennsylvania

21. Staudemeyer RC (2015) Applying long short-term memory recurrent neural networks to intrusion detection. S Afr Comput J 56(1):136–154
22. Lee W, Stolfo SJ (2000) A framework for constructing features and models for intrusion detection systems. ACM Trans Inf Syst Secur (TiSSEC) 3(4):227–261
23. Lippmann RP, Fried DJ, Graf I, Haines JW, Kendall KR, McClung D, Weber D, Webster SE, Wyschogrod D, Cunningham RK, Zissman MA (2000) Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In: Proceedings DARPA information survivability conference and exposition, DISCEX'00, vol 2. IEEE, pp 12–26
24. Özgür A, Erdem H (2016) A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. PeerJ PrePrints 4:e1954v1
25. Bhuyan MH, Bhattacharyya DK, Kalita JK (2014) Network anomaly detection: methods, systems and tools. IEEE Commun Surv Tutorials 16(1):303–336
26. Agarwal R, Joshi MV (2000) PNrule: a new framework for learning classifier models in data mining. Technical Report TR 00–015. University of Minnesota, Department of Computer Science
27. Kayacik H, Zincir-Heywood AN, Heywood MI (2005) Selecting features for intrusion detection: a feature relevance analysis on KDD 99 intrusion detection datasets. In: Proceedings of the third annual conference on privacy, security and trust 2005, PST 2005, DBLP
28. Zhang J, Zulkernine M, Haque A (2008) Random-forests-based network intrusion detection systems. IEEE Trans Syst Man Cybern Part C Appl Rev 38(5):649–659
29. Li W (2004) Using genetic algorithm for network intrusion detection. In: Proceedings of the United States department of energy cyber security group, vol 1, pp 1–8
30. Kolias C, Kambourakis G, Maragoudakis M (2011) Swarm intelligence in intrusion detection: a survey. Comput Secur 30(8):625–642. https://doi.org/10.1016/j.cose.2011.08.009
31. Al-Subaie M, Zulkernine M (2006) Efficacy of hidden Markov models over neural networks in anomaly intrusion detection. In: 30th Annual international computer software and applications conference. COMPSAC 06., vol 1, pp 325–332. ISSN 0730-3157
32. Upadhyay R, Pantiukhin D Application of convolutional neural network to intrusion type recognition. https://www.researchgate.net
33. Gao Ni et al (2014) An intrusion detection model based on deep belief networks. In: 2014 Second international conference on advanced cloud and big data (CBD). IEEE
34. Moradi M, Zulkernine M (2004) A neural network based system for intrusion detection and classification of attacks. In: Paper presented at the proceeding of the 2004 IEEE international conference on advances in intelligent systems Theory and applications. Luxembourg
35. Mukkamala S, Sung AH, Abraham A (2003) Intrusion detection using ensemble of soft computing paradigms. In: Third international conference on intelligent systems design and applications, intelligent systems design and applications, advances in soft computing. Springer, Germany, pp 239–48
36. Xue J-S, Sun J-Z, Zhang X (2004) Recurrent network in network intrusion detection system. In: Proceedings of 2004 international conference on machine learning and cybernetics, vol 5, pp 2676–2679
37. Yang J, Deng J, Li S, Hao Y (2015) Improved traffic detection with support vector machine based on restricted Boltzmann machine. Soft Comput 21(11):3101–31112. https://doi.org/10.1007/s00500-015-1994-9
38. Javaid A, Niyaz Q, Sun W, Alam M (2015) A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI international conference on bio-inspired information and communications technologies (formerly BIONETICS), New York, NY, USA, 3–5 Dec 2015, pp 21–26. They also used recurrent network to preserve the state full information of malware sequences
39. Jihyun K, Howon K (2015) Applying recurrent neural network to intrusion detection with hessian free optimization. In: Proc, WISA
40. Kim J, Kim J, Thu,HLT, Kim H (2016) Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International conference on platform technology and service (PlatCon), Jeju, pp 1-5. https://doi.org/10.1109/PlatCon.2016.7456805

41. Brugger S, Chow J (2005) An assessment of the DARPA IDS evaluation dataset using snort. Tech. Rep. CSE-2007-1, Department of Computer Science, University of California, Davis (UCDAVIS)
42. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the second IEEE symposium on computational intelligence for security and defence applications
43. Wang Z (2015) The applications of deep learning on traffic identification. BlackHat USA
44. Touch J, Kojo M, Lear E, Mankin A, Ono K, Stiemerling M, Eggert L (2013) Service name and transport protocol port number registry. The Internet Assigned Numbers Authority (IANA)
45. Park BC, Won YJ, Kim MS, Hong JW (2008) Towards automated application signature generation for traffic identification. In: NOMS 2008-2008 IEEE network operations and management symposium. IEEE, pp 160–167
46. Zuev D, Moore AW (2005) Traffic classification using a statistical approach. In: International workshop on passive and active network measurement. Springer, Berlin, Heidelberg, pp 321–324
47. Tan KM, Collie BS (1997) Detection and classification of TCP/IP network services. In: Proceedings 13th annual computer security applications conference. IEEE, pp 99–107
48. Moustafa N, Slay J (2015) UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: Military communications and information systems conference (MilCIS). IEEE, pp 1–6
49. McLaughlin N, Martinez del Rincon J, Kang B, Yerima S, Miller P, Sezer S, Safaei Y, Trickel E, Zhao Z, Doupê A, Joon Ahn G (2017) Deep android malware detection. In: Proceedings of the seventh ACM on conference on data and application security and privacy. ACM, pp 301–308
50. Elhoseny M, Hassanien AE (2019) Mobile object tracking in wide environments using WSNs. In: Dynamic wireless sensor networks. Springer, Cham, pp 3–28
51. Elhoseny M, Hassanien AE (2019) Expand mobile WSN coverage in harsh environments. In: Dynamic wireless sensor networks. Springer, Cham, pp 29–52
52. Elhoseny M, Hassanien AE (2019) Hierarchical and clustering WSN models: their requirements for complex applications. In: Dynamic wireless sensor networks. Springer, Cham, pp 53–71
53. Elhoseny M, Hassanien AE (2019) Extending homogeneous WSN lifetime in dynamic environments using the clustering model. In: Dynamic wireless sensor networks. Springer, Cham, pp 73–92
54. Elhoseny M, Hassanien AE (2019) Optimizing cluster head selection in WSN to prolong its existence. In: Dynamic wireless sensor networks. Springer, Cham, pp 93–111
55. Elhoseny M, Hassanien AE (2019) Secure data transmission in WSN: an overview. In: Dynamic wireless sensor networks. Springer, Cham, pp 115–143
56. Elhoseny M, Hassanien AE (2019) An encryption model for data processing in WSN. In: Dynamic wireless sensor networks. Springer, Cham, pp 145–169
57. Elhoseny M, Hassanien AE (2019) Using wireless sensor to acquire live data on a SCADA system, towards monitoring file integrity. In: Dynamic wireless sensor networks. Springer, Cham, pp 171–191
58. Elhoseny M, Elleithy K, Elminir H, Yuan X, Riad A (2015) Dynamic clustering of heterogeneous wireless sensor networks using a genetic algorithm towards balancing energy exhaustion. Int J Sci Eng Res 6(8):1243–1252
59. Elhoseny M, Elminir H, Riad AM, Yuan XIAOHUI (2014) Recent advances of secure clustering protocols in wireless sensor networks. Int J Comput Netw Commun Secur 2(11):400–413
60. Riad AM, El-Minir HK, El-hoseny M (2013) Secure routing in wireless sensor networks: a state of the art. Int J Comput Appl 67(7)