International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 10, No.1 (Dec-2021)
http://dx.doi.org/10.12785/ijcds/1001123

Exploring Deep Neural Network Capability for Intrusion Detection Using Different Mobile Phones Platforms

Nouf Fahad Alharbi[1] and Nabil M. Hewahi[2]

[1] *Cyber Security Program, University of Bahrain, Bahrain*
[2]*Department of Computer Science, University of Bahrain, Bahrain*

Received 24 Apr. 2021, Revised 31 Jul. 2021, Accepted 13 Nov. 2021, Published 28 Dec. 2021

Abstract: Network intrusion activities started since the network typologies become available for public, the target devices were computers, servers, switches, routers and so on, but with the fever of smartphones spreading among the public, these smartphones have become a target for hacker attacks. Network Intrusion Detection System (IDS) is a device, software or both used by network administrator to monitor the network security and recognize any malicious activity or organization's network policy violation. Deep neural network is the famous technology in deep learning where it is an artificial neural network with multiple hidden layers, and it simulates the human brain and it's the central nervous system in thinking and detecting pattern. Mobile phones are widely used today, with different platforms which known now as smartphones, these smartphones belong to many manufacturing and require operating system. This research explores deep neural networks capability for intrusion detection using different mobile phones platforms, the research experiments five deep neural networks approaches, Fully Connected Deep Neural Networks (FCDNN), Convolutional Neural Networks (CNN), Convolutional Neural Networks followed by Fully Connected Neural Networks (CNN&FCDNN), Convolutional Neural Networks followed by Fully Connected Neural Networks and then followed by Convolution Neural Network (CNN&FCDNN&CNN), and finally Fully Connected Deep Neural Networks followed by Convolution Neural Networks (FCDNN&CNN).Two data sets have been used for testing the approaches, Android data set and NSL-KDD data set. The proposed approaches have been compared with each other's and also compared with traditional Machine Learning (ML) approaches. The results show that CNN&FCDNN&CNN is the best deep learning approach where it achieved accuracy of 0.863 and 0.997 for the two data sets Android and NSL-KDD respectively. The accuracy results also show that the best approach is the random forest where it achieved 0.88 and 0.998 for Android and NSL-KDD data sets respectively. Deep neural networks show that they are good machine learning candidates for problems similar to mobile phones intrusion detection systems.

Keywords: Intrusion Detection System, Convolutional Neural Networks, Fully Connected Deep Neural Network.

1. INTRODUCTION

Network intrusion detection system (IDS) is a device, software or both used by network administrator to monitor the network security and recognize any malicious activity or organization's network policy violation [1].

An IDS monitor daily activities by targeting network traffics to identify the inconsistencies and abnormal behavior on a network to detect risks or attacks related to network security, like denial-of-service (DoS). An intrusion detection system also helps to locate, decide, and control unauthorized system behavior such as unauthorized access, or modification and destruction [2].

Network intrusion activities started since the network typologies become available for public, the targets devices were computers, servers, switches, routers and so on, but with the accelerated speed of smartphones spreading among the public, smartphones have become a target for hacker attacks.

Legacy protection systems like firewall detects external attacks, while modern IDS detect internal and external attacks [3].

Mobile phones are widely used today, with different platforms which known now as smartphones, these smartphones belong to many manufacturing and require operating system to manage the smartphone resources, the most popular operating systems for smartphones are Android, iOS, Windows phone OS, and Symbian.
Smartphones are real portable computers, with a high ability to connect to the Internet, perform mathematical operations and store data, making them able to run complex applications, which made them a target for

E-mail:20189506@stu.uob.edu.bh, nhewahi@uob.edu.bh

developers and researchers in the field of artificial intelligence and protection systems [4].

This report has six sections. Apart from introduction, section two presents the literature review, section 3 is devoted to the methodology we follow to tackle the problem. Section 4 is the experimentation details which describe the experiment's methodology, data sets exploring and experiment's parameters. Section 5 is focused on the obtained results and the discussion. Finally, section 6 is for the conclusion and future work.

A. Network Intrusion

Network intrusion is a term refers to any illegal activity against digital computer and its networks, were this illegal activity breaks the protection rules of the target digital network, digital computers include, servers, ordinary personal computers, workstations, laptops, routers, and smartphones which are the target of this study [5].

Network intrusion is a type of network attack, where attacker targets a data reside in one or more digital device within target network. This an unauthorized access allow attacker to steal or change data without any right.

Mobile platform like Android allows user to install third party applications, usually these applications weren't scanned with antivirus or with intrusion detection system, these applications which are installed through Google play are called un-trusted source applications.

Google play scan Android application before publish it to the public, and receive any report about suspicious activities from application's users, by this mechanism Google can refuse and remove un-trusted and malicious application from its application store.

Mobile application permissions are roles which are given to installed application to access hardware devices like camera, microphone, accelerometer sensor, GPS, and others, and built-in Android operating system like contacts, sending and receiving SMS and so on.

Google decided at its Android operating system which is called Android 6.0 Marshmallow to give user the ability to accept or refuse permissions request from installed application at running time which is known as on-fly permission granted [6].

Studies focus on Android permissions to determine malicious application use two types of analysis methods [6]:

1. Static analysis: where researchers study the application's code without executing it especially where the risk of executing the application is high.
2. Dynamic analysis: this analysis is during application execution where researchers study the application behavior within network.

In the light of the previous information about network intrusion attacks and its dangerous on different mobile phones platforms, and the powerful of deep machine learning like deep neural network in classifying new patterns of network intrusion attacks, this paper focuses of investigating the efficiency of DNN on detecting intrusions on various mobile phones OS platforms.

B. Intrusion Detection Systems

Network Intrusion detection system (IDS) is a device, software or both using by network administrator to monitor the network security and recognize any malicious activity or organization' s network policy violation [7].

An IDS monitors daily activities by targeting network traffics to identify the inconsistencies and abnormal behavior on a network to detect risks or attacks related to network security, like denial-of-service (DoS). An intrusion detection system also helps to locate, decide, and control unauthorized system behavior such as unauthorized access, or modification and destruction [8].

Smartphones are real portable computers, with a high ability to connect to the Internet, perform mathematical operations and store data, making them able to run complex applications, which made them a target for developers and researchers in the field of artificial intelligence and protection systems [9].

The intrusion detection system has monitor server which can be machine learning based or human administrator based with non-intelligent software, computers are connected to the internal network with switch, the internal intrusion detection system module is connected to the internal network to detect internal intrusions.

The network is protected from external traffic with firewall which is supported by external detection system module which now in many organization uses machine learning and large database to detect network intrusion [10].

Researchers in network intrusion detection focus on network layer, and application layer traffic which known as TCP/IP, mobile platforms and others computer platforms share many traffic types at network layer, the difference on application layer where mobile programming differs from desktop programming. Data sets that collected from computer networks are larger and contain many fields over than that collected from mobile networks [11].

1) General classification of intrusion detection system

Fig. 1. Illustrates the classifications of IDS. As noted from the figure, this classification includes various ML approaches. Nowadays, ML is becoming one of the very well- known approaches used in IDS.

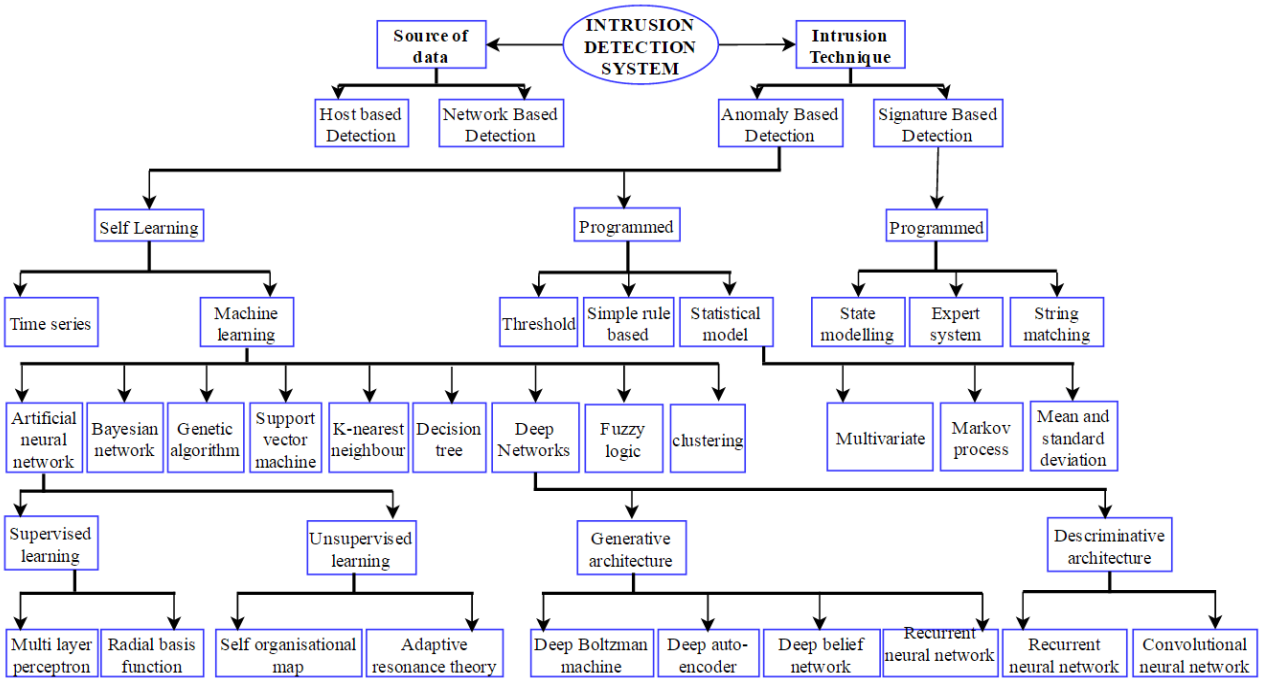

Figure 1. General classification of intrusion detection system [10]

C. Deep Neural Networks

Deep Neural Network (DNN) is an Artificial Neural Network (ANN) with multiple hidden layers between the input and the output layers, DNN can model very complex nonlinear relationship between the input and the output layers [12]. There are many structures or topologies of DNNs. In this section, we shall only talk about the neural network structures which we use as a base in our paper, these are Fully connected deep neural network (FCDNN) and Convolution Neural Networks (CNN).

A multilayer perceptron (MLP) has three or more layers. It utilizes a nonlinear activation function (mainly hyperbolic tangent or logistic function) that lets it classify data that is not linearly separable. Every node in a layer is connected to every node in the following layer making the network fully connected as shown at the Fig.2 [13]. MLP works based on weight adjustment of the neural network, the process is performed based on a well-known algorithm called backpropagation algorithm. The learning process includes training the neural network with a given data set by keep modifying the weights until an acceptable error is achieved (if possible), the neural network can be then tested through unseen data set.

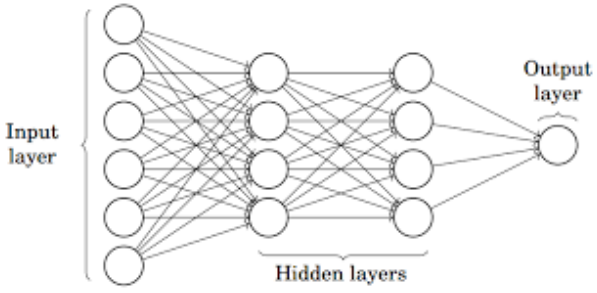

Figure 2. Multilayer perceptron (MLP) structure

A convolutional neural network (CNN) consists of one or more convolutional layer, each convolution layer is usually followed by a pooling layer. These two layers can be repeated as requested by the neural network developer. The output of these repeated layers is injected to a flatten layer which can be followed by a number of fully connected layers reaching to the output layer. Convolution layer uses what so called a kernel to map the input to the size of the convolution layer. The pooling layer makes the feature map in the convolution layer smaller by applying some sort of pooling like max pooling, min pooling or average pooling. Fig.3. Depicts the structure of CNN. CNN is well-known structure for image recognition and allows the neural network to be deeper with much fewer parameters [14].

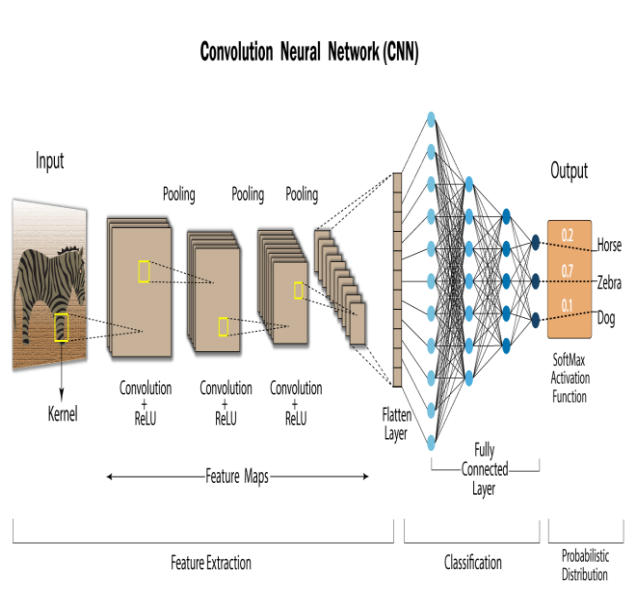

Figure 3. CNN example [14]

D. Problem Statement

Our main purpose is to explore the efficiency of various DNN structures to detect and classify external activities coming to various mobile phone platforms as "intrusion" or "no-intrusion". We need to compare the results of DNN different structures with each other in detecting intrusions for mobile platforms. Also, results obtained by various machine learning approaches are compared with our various experimented DNN structures. Among the experimented structures, we propose a new DNN structure that will be also tested. The work will be limited to the data sets, Android data set and NSL-KDD data set.

2. LITERATURE REVIEW

In this chapter we present various types of network attacks related to computer networks and mobile networks. We also discuss various methods proposed by researchers to detect intrusions and focus more on those related to machine learning, especially neural networks.

A. Network Attacks

This section presents the most well-known network attacks.

1) Denial of service

Denial of Service (DoS) is a very common attack; it is famous for its simple implementation where the main purpose of this attack is flooding a server with requests to fill its network adapter buffer with trash packets, this causes inability of the server to respond to actual requests from real clients. In this type of attack, usually the attacker is a single device [17]. In general, DoS is responsible of deny authorized user to reach its legal resources over the network like servers, data storage, database warehouse, printers, and so on, attacker need only to know the address of victim resource to attack it.

2) Distributed denial of service(DDoS)

The main difference between this type of attack and the DoS is that in this type the number of attackers is large, where multiple devices corporate to accomplish the attack. DDoS attack is more advanced from the DoS, and in many situations the original attacker distributes this attack over victim devices as Trojan virus program that run without knowledge of victim use. The victim device acts as an attacker and can infect another device. In many mobile networks, this type of attack targets the routing protocols to break routes between nodes in mobile ad hoc network (MANET) networks [18].

3) Probe attacks

The attacker in probe attacks tries to scan and discover detailed information about target server, by sending requests over TCP protocol, attacker searches for backdoor and drawbacks in protection systems like firewalls [12]. There are many applications and tools that can generate network traffic of probe attacks for research purpose like [19]:

- Saint
- Satan
- Ipsweep
- Mscan
- Portsweep

In this work we focus on network intrusion detection system.

4) Privilege escalation attacks

Privilege escalation attack is an advance type of attack where attackers try to discover the network security weakness or security vulnerabilities to access very sensitive data like passwords and credit card numbers [13].

In business, privilege escalation attack occurs when an attacker gets ability to access an employee's account, bypasses the proper authorization channel giving himself permissions and authorization to access data he/she does not have rights to access it. After cracking the employee's account, the attacker can create new

functions or insert worms or Trojan and create backdoors.

Attacker on the internet can use SQL injection to get sensitive data over web or change/delete records using some SQL skills like append or 1=1 in some websites without SQL injection protection mechanisms. Other Privilege escalation attack types and tools that generate the corresponding network traffic of the attack include [21]:

- Loadmodule
- buffer_overflow

B. Related work

In [5], authors proposed a new method of deep learning approach for network intrusion detection. The proposed work consists of two-stages of self-taught learning (STL), where stage one is the Unsupervised Feature Learning (UFL), which is a good presentation of features learning from unlabeled data x_u. The output from stage one is applied to labeled data x_l, then these labeled data are used to build the final classification model (stage two). Fig.4. Illustrates self-taught learning model [5].

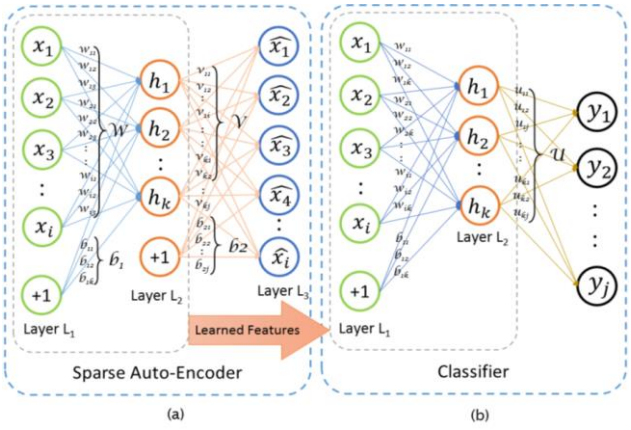

Figure 4. self-taught learning block diagram [5]

The first stage consists of sparse auto-encoder neural network from previous research [17], the sparse auto-encoder neural network consists of one input layer, one hidden layer, and one output layer, where the input and output layers contain the same number of nodes N or neurons, and the hidden layer contains K number of nodes.

To calculate the weights parameters values, the stage uses the back-propagation algorithm by minimizing the lose function.

Authors evaluate their proposed solution on existing data set NSL-KDD data set, the proposed algorithm is tested to classify different number of classes of TCP attacks, the attacks contain many categories:

- Denial of Services (Dos)
- Probing attack
- User-to-Root (U2R) attack
- Root-to-Local (R2L) attack.

NSL-KDD is used as a classification data set, where each record is labeled as normal traffic or attack type label with 23 class different labels. The data contains 41 features, and authors extracted more features by applying 1-2-n encoding method to convert nominal attributes to discrete attributes, and finally they get 121 features. Authors used 10-fold cross-validation technique over the training data to evaluate the classification accuracy of self-taught learning (STL) for 2-class, 5-class, and 23-class. They compared the STL performance with the soft-max regression (SMR) when applied directly to the data set without the stage one which is the features learning. Their proposed solution is better than the traditional SMR with accuracy equals to 98% for all types of classifications.

DroidLight is one of active research in mobile devices' security applications, which is a Lightweight Anomaly-based intrusion detection system for smartphone devices [22], it can detect zero-day malware which means when a new malware attacks the target network of mobile phones, the authors designed their algorithm based on one class cluster with statistics analysis probability distribution function based. Fig.5. illustrates the proposed application.

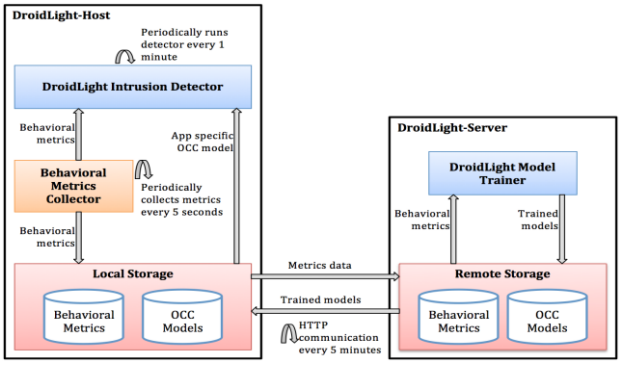

Figure 5. DroidLight Architecture [22]

The proposed algorithm assumes one class which is the normal or benign network traffic from and to the smartphone. During daily network activity the model will build and train this one class, the model has its mean and deviation which is known as probabilistic distribution model. After building the model, the algorithm checks every network traffic and if any incoming activity has significant deviation from the mean of the model, then

the proposed algorithm will detect a network intrusion. To perform the experiments, authors developed three malware applications which act as real malicious applications, these three applications are:

- DroidDDoS
- DroidThief
- DroidHijack

Authors performed realistic evaluation of DroidLight (i.e. the evaluation was performed on a real device while a real user was interacting with it). Results evaluation demonstrate that DroidLight can detect smartphone malwares with accuracy ranging from 93.3% to 100% while imposing only 1.5% total overhead on device resources.

A research in [11] extracted an Android Data set features and apply traditional machine learning classification algorithms on the generated data set. The obtained results accuracy is as shown in Table 1. In our paper we shall use the Android data set as one of the data sets in our experimentation.

TABLE 1. MACHINE LEARNING PERFORMANCE METRICS [11]

Algorithm	**Accuracy**	**f1-score**
Naive Bayes	0.43	0.59
Random Forest	0.88	0.85
KNN. K=4	0.83	0.79
SVM	0.51	0.65
Logistic regression	0.58	0.2
Decision Tree	0.86	0.83

In [11], the random forest and decision tree, the classic machine learning techniques provide the highest accuracy while the naïve bayes algorithm is the worst, the authors selected only 10 features because many features have null and undefined data.

HIDROID is a proposed prototype for Android devices to detect and prevent intrusion depending on behavior host-based, where there is no need to centralize data storage and cloud-based service [23]. The proposed prototype collects information in real run time as shown in Table 2.

TABLE 2. COLLECTED INFORMATION [23]

Index	Feature	description
1	CPU usage	Overall CPU consumption %
2	Memory usage	Overall memory consumption in KB
3	Cashed memory	The occupied memory size for cash in KB
4	Bit rate	The average of transfer data over internet in kilo bytes per second (KB/S)
5	Packet rate	The average number of communicated packet is second
6	Battery drain	Battery consumption or drop in %
7	Battery temperature	The battery temperature in Celsius degree C^{o}
8	No. running processes	The total number of running processes at any given time
9	No. running services	The total number of running background services at any given time
10	No. TCP open sockets	The total number of open sockets at any given time
11	No. sent SMS	The total number of sent SMS to unknown receivers
12	No. outgoing calls	The total number of outgoing calls in history
13	No. out SMS	The total number of sent
14	No. installed applications	The total number of installed applications
15	Screen status	Screen status at any given time which is binary 0 for off, 1 for on.

As shown in Fig.6. HIDROID app consists of the following main components:

1. Run Time Data Acquisition.
2. Run Time Data set Generation.
3. Feature Normalization.
4. Detection Engine.
5. Intrusion Probability Assessment.
6. Alert Manager.
7. Prevention Engine.

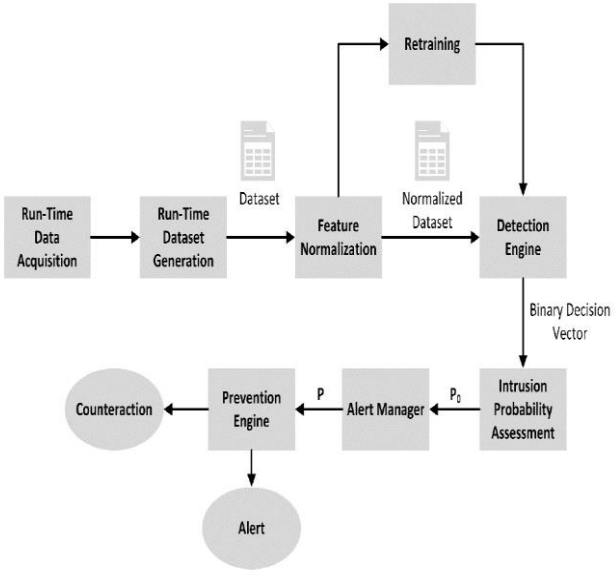

Figure 6. HIDROID structure [23]

In this research research authors used K-means and Gaussian algorithms with boundary parameter varies from 0.7 to 1 and obtained results shown in Table 3.

TABLE 2. HIDROID PERFORMANCE [23]

Algorithm	Accuracy	TPR	FPR	Boundary
K-means	0.9087	0.836667	0.019333	0.9
Gaussian	0.9143	0.857667	0.029	0.95

A study focused on instruction detection in connected and autonomous vehicles has been given in [24]. The study proposed CNN to detect intrusion in ethernet-based network. The study detects the audio video streaming injection attacks, and obtained F1-score equals to 0.9704 and recall equals to 0.9949.

Another study that applied CNN for network intrusion detection is in [25]. The proposed work modeled the network traffic over TCP/IP as time-series. The target data set was KDD Cup 99. Authors developed many approaches CNN-based like CNN_LSTM, CNN_RNN, and the results showed that the CNN can be used in network intrusion detection [25].

Newly research proposed a new CNN approach called CNN-MCL that detected anomalies and can learn low-level malicious or abnormal feature's [26]. The proposed design is shown in Fig.7, where data input vector projected into MCL layer with 11x11 dimensions for each data sample and then feed CNN1 which feeds another CNN2.

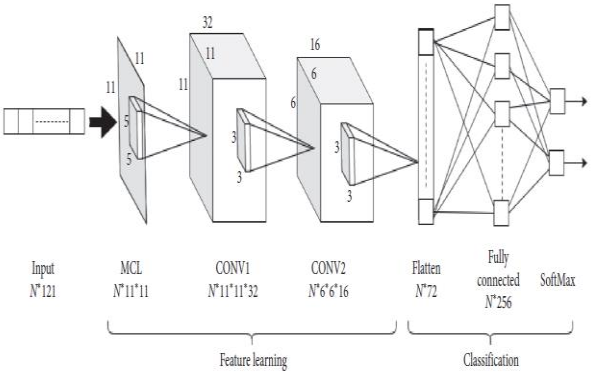

Figure 7. Design of proposed architecture [26]

In Riyaz and Ganapathy[27], a new approach for network intrusion detection in wireless network using CNN was proposed, the study targeted the KDD data set, but the proposed approach developed a new features selection algorithm before feed the CNN. The research got overall accuracy 98.88%. The proposed design is as shown in Fig. 8.

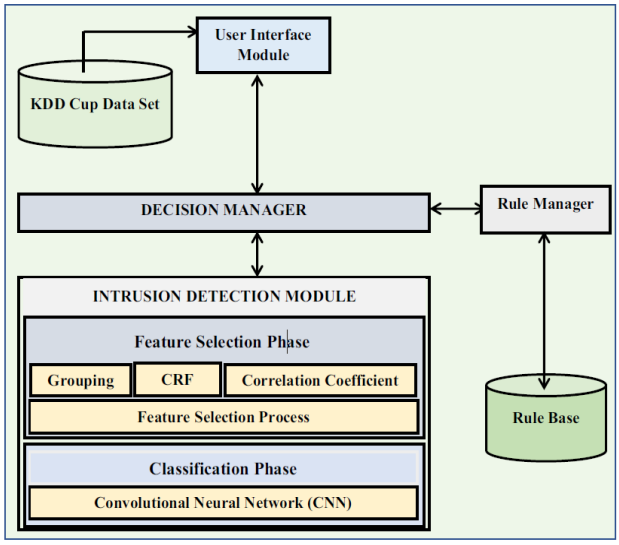

Figure 8. Proposed system architecture as in [27].

In most of the previous studies researchers used various approaches of machine learning to include fully connected neural networks and convolutional neural networks to detect intrusions. The main difference between our proposed work and others are, first, most of the works are not related to mobile platforms and the second, some of these approaches are tested on certain types of intrusions where others are using one proposed approach. In our work we try to explore various structures of DNN to detect intrusions and notice their performance on all the existing types of intrusions as one data set for mobile platforms.

3. METHODOLOGY

Many machine learning approaches have been tested for network intrusion detection, most of these algorithms belong to classical group, where human need to train the algorithm or method over many data sets to solve the problem scope. On the other hand, deep learning is a modern field in Artificial Intelligent (AI), which tries to find methods and algorithms related to deep neural networks to enhance the computer ability in self learning. In deep learning, machine simulates the human in thinking and learning to solve problems without human intervention.

The proposed solution is creating multiple combination of deep neural network typologies, and apply the new hybrid deep neural networks on the selected data sets and compare the results with previous results.

A. Proposed hybrid deep neural networks

Multiple hybrids and pure deep neural networks were developed, to view the results and recognize the best deep neural network topology by applying each of them on the two selected data sets (Android data set and NSL-KDD). All the structures presented in the paper like the Fully Connected Deep Neural Networks (FCDNN) and Convolution Neural Network (CNN) are used as basis for various hybrid structures. We use these structures in the hybrid topologies because they have shown good performance in research specially in problems related to IDS as given in section 2.

1. FCDNN
2. CNN
3. CNN followed by FCDNN
4. CNN followed by FCDNN followed by CNN
5. FCDNN followed by CNN

All the fully connected layers are using Rectified Liner Unit (RELU/relu) activation function, which is very common and well-known in deep neural networks [30]. For the output layer in all the structures we used sigmoid activation function [31]. To the knowledge of the researchers, the 4th structure (CNN followed by FCDNN followed by CNN) has never been used by any other researchers. Fig. 9. illustrates the proposed methodology activity diagram.

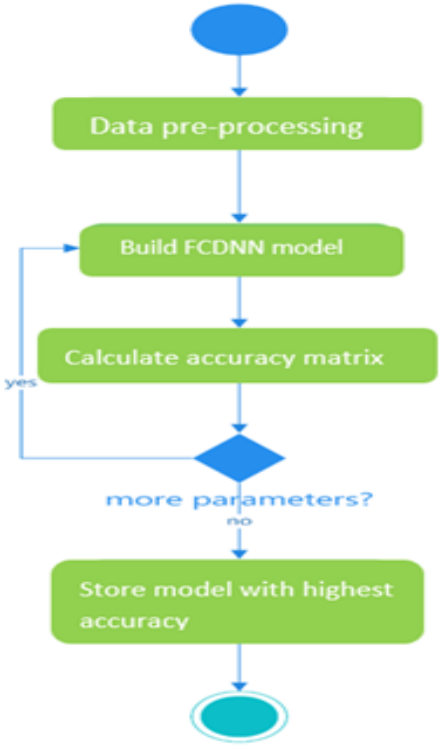

Figure 9. Proposed methodology activity diagram

1) FCDNN

The first deep neural network topology is four dense deep neural layers as shown in Fig. 10. The used structure is:

1. Input layer
2. 4 hidden layers
3. Flatten layer
4. Dropout layer
5. Output layer

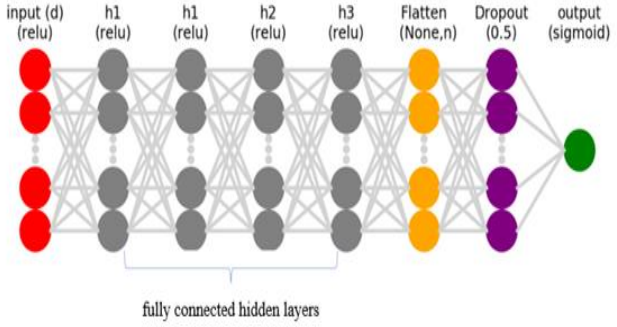

Figure 10. FCDNN proposed approach

The fully connected hidden layer is similar to the hidden layer in ANN, on another hand, flatten layer is responsible of transforming feature matrix into a single column. In this structure the flatten layer is similar to hidden layer. It is worth mentioning that dropout is not a layer as such, it is an operation to dropout some nodes randomly from the layer to overcome the problem of over fitting. We shall refer to it as a layer in all structures we use.

2) CNN

The second deep neural network topology is a multiple CNN as shown in Fig. 11. It has the following structure:

1. Input layer
2. 2 One-dimension CNNs
3. One-dimension max pooling layer
4. Flatten layer
5. Dropout
6. Output layer

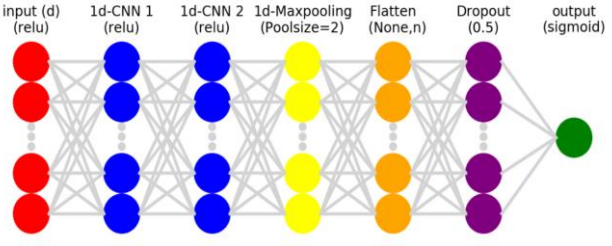

Figure 11. CNN proposed approach

3) CNN & FCDNN

The third proposed deep neural network is two CNNs followed by 4 layers of dense deep layers; the hybrid deep neural network is shown in Fig. 12, and has the following structure:

1. Input layer
2. 2 one-dimension CNNs
3. One-dimension max pooling layer
4. 4 layers dense of FCDNN
5. Flatten layer
6. Dropout layer
7. Output layer

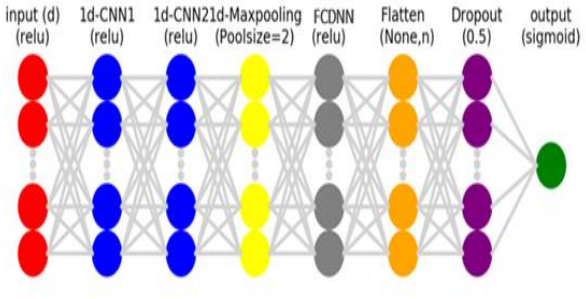

Figure 12. CNN & FCDNN approach

4) CNN & FCDNN & CNN

The fourth hybrid deep neural network is a combination between CNN followed by FCDNN, then again followed by CNN as shown in Fig. 13. It has the following structure:

1. Input layer
2. 2 one-dimension CNNs
3. One-dimension max pooling layer
4. 4 layers dense of FCDNN
5. Flatten layer
6. Dropout layer
7. Output layer

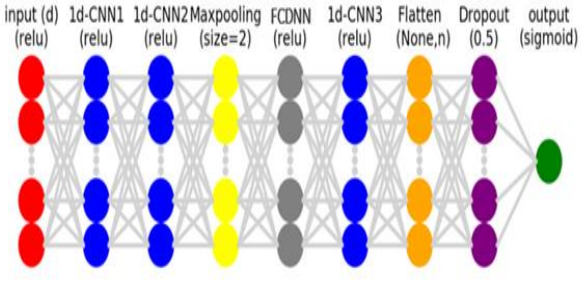

Figure 13. CNN & FCDNN & CNN approach

5) FCDNN & CNN

The fifth hybrid deep neural network is a combination of multiple dense deep neural network layers FCDNN followed by two CNNs as shown in Fig. 14. It has the following structure:

1. Input layer
2. 4 layers dense of FCDNN
3. 2 one-dimension CNNs
4. One-dimension max pooling layer
5. Flatten layer
6. Dropout layer
7. Output layer

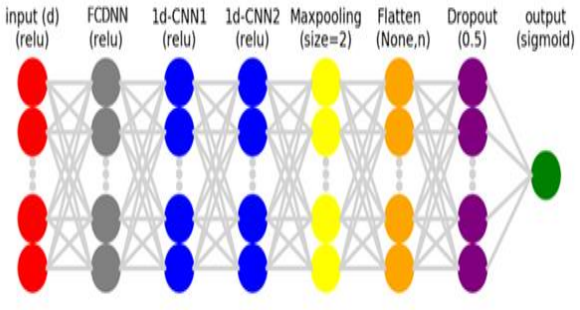

Figure 14. FCDNN & CNN approach

4. EXPERIMENTATION

The proposed intrusion detection system using mobile phones platform deep neural network classifier is trained and tested over the following data sets:

- NSL-KDD data set.
- DroidCollector Android data set.

A. NSL-KDD data set preprocessing

NSL-KDD is a significant upgrade of the KDD Cup 99 data set, where it solved many problems of the original KDD cup 99 data set [28]. Due to the lack of availability of data sets related to various mobile platform, we use NSL-KDD data set since it has several instances related to various mobile platforms without stating the related platform. The data set is a multiclass where it recognizes multiple types of attacks, but our work is about intrusion detection regardless of type of attack, so we need to convert the data set to binary where any sample with label differs from normal labeled is considered as malicious. The data set has been split to training and testing with 80% for training and 20% for testing (100778 for training and 25195 for testing), The data set has been scaled with standard scalar (in python).

B. DroidCollector Android data set preprocessing

The data set file called android_trafic.csv, it is a set of pcap files from the DroidCollector project integrated with 4705 benign and 3141 malicious applications [29]. We need to drop columns with null values which are duration, avg_local_pkt_rate, and avg_remote_pkt_rate. In addition, we need to remove duplicated data (source_app_packets.1).

According to [11], the best features for Android network traffic are:

- (R1): TCP packets, it has the number of packets TCP sent and got during communication.
- (R2): Different TCP packets, it is the total number of packets different from TCP.
- (R3): External IP, represents the number the external addresses (IPs) where the application tried to communicated
- (R4): Volume of bytes, it is the number of bytes that was sent from the application to the external sites
- (R5) UDP packets, the total number of packets UDP transmitted in a communication.
- (R6) Packets of the source application, it is the number of packets that were sent from the application to a remote server.
- (R7) Remote application packages, number of packages received from external sources.
- (R8) Bytes of the application source, this is the volume (in Bytes) of the communication between the application and server.
- (R9) Bytes of the application remote, this is the volume (in Bytes) of the data from the server to the emulator.
- (R10) DNS queries, number of DNS queries.

Standardization of a data set is a common requirement for many machine learning estimators. Typically, this is done by removing the mean and scaling to unit variance. However, outliers can often influence the sample mean / variance in a negative way. In such cases, the median and the interquartile range often give better results. The data set has been split to training and testing with 80% and 20% respectively (6265 for training, 1567 for testing).

1) Experiment Methodology

Every proposed approach is written in a separate Python file, there are five approaches deep neural networks, and two data sets. On the other hand, it is necessary to develop other machine learning algorithms and apply them to the used data sets to compare classic machine learning algorithms with the proposed deep neural network approaches.

The selected others algorithms are:

- Naive Bayes
- Random Forest
- k-nearest neighbors' algorithm (KNN), K=4
- Support vector machine (SVM)
- Logistic regression
- Decision Tree
- Gaussian with radial basis function (RBF) kernel, only for DroidCollector Android data set (for the NSL-KDD data set, the experiment could not be performed due to the limited system specifications).

Each proposed approach will be trained for 500 iterations.

The performance metrics for the proposed deep neural networks are:

- Accuracy for training
- Accuracy for testing
- F1-score for testing
- True positive (TP)
- False positive (FP)
- True negative (TN)
- False negative (FN)
- TP rate (sensitivity)

The metrics that are used to compare the proposed approaches with other machine learning algorithms are:

- Accuracy
- F1-score

Fig. 15. Illustrates the flowchart of experiment methodology:

1. Read data set
2. Data set pre-processing
3. Split data set (20% for testing, 80% for training)
4. Build the proposed approach model
5. Train the model
6. Test the model
7. Display model performance metrics

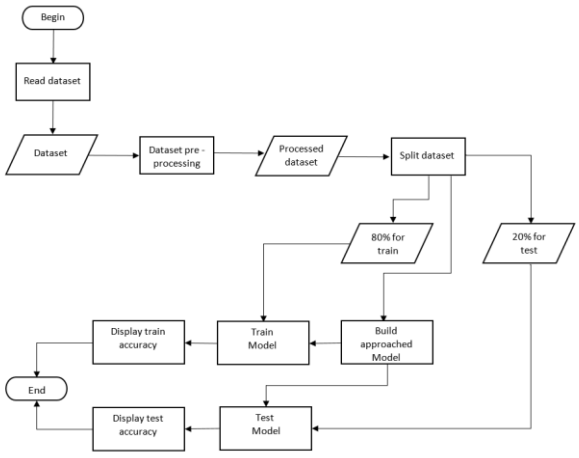

Figure 15. Experiment methodology flowchart

2) Experiment Parameters

In this section we present the best parameters used with various structures for different experimentations.

a) FCDNN

The first deep neural network topology is four dense deep neural layers with the following specification as shown in Fig. 16.

1. Input layer with data dimension d=10 for DroidCollector Android data set, and d=121 for NSL-KDD data set.
2. Layer 1 with 8 neurons, the input dimension is 10 for Android data set and 121 for NSL-KDD data set, and the activation function is relu .
3. Layer 2 with 100 neurons and the activation function is relu
4. Layer 3 with 100 neurons and the activation function is relu
5. Layer 4 with 100 neurons and the activation function is relu.
6. Flatten
7. Dropout layer
8. Output layer with one neuron and the activation function is sigmoid.

```
In [15]: model = Sequential()
         model.add(Dense(8, input_dim=10, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Flatten())
         model.add(Dropout(0.5))
         model.add(Dense(1, activation='sigmoid'))
```

Figure 16. FCDNN Model

b) CNN

The second deep neural network topology is multiple CNN networks with the following specification and as shown in Fig. 17.

1. Input layer with data dimension d=10 for DroidCollector Android data set, and d=121 for NSL-KDD data set.
2. One-dimension CNN network: filters=64, kernel size=3, activation function is relu, and the input vector dimension is (121,1) for NSL-KDD data set and (10,1) for Android data set.
3. One-dimension CNN network: filters=64, kernel size=3, and activation function is relu
4. One-dimension max pooling layer: with pool size =2
5. Flatten layer
6. Dropout laye
7. Output layer: one neuron and activation function is sigmoid.

```
In [21]: model=Sequential()
         model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(121,1)))
         model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
         model.add(MaxPooling1D(pool_size=2))
         model.add(Flatten())

         model.add(Dropout(0.5))

         model.add(Dense(1,activation='sigmoid'))

         model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Figure 17. CNN Model

c) CNN & FCDNN

The third proposed deep neural network is two CNN followed by 4 layers of dense deep layers FCDNN; the hybrid deep neural network has the following specification as shown in Fig. 18.

1. Input layer with data dimension d=10 for DroidCollector Android data set, and d=121 for NSL-KDD data set.
2. One-dimension CNN network: filters=64, kernel size=3, activation function is relu, and the input victor dimension is (121,1) for NSL-KDD data set and (10,1) for Android data set.
3. One-dimension CNN network: filters=64, kernel size=3, and activation function is relu
4. One-dimension max pooling layer: with pool size =2
5. 4 layers dense of FCDNN: 100 neurons and relu activation function
6. Flatten layer
7. Dropout layer
8. Output layer: one neuron and activation function is sigmoid

```
In [21]: model=Sequential()
         model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(121,1)))
         model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
         model.add(MaxPooling1D(pool_size=2))
         model.add(Dense(100, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Flatten())

         model.add(Dropout(0.5))

         model.add(Dense(1,activation='sigmoid'))
```

Figure 18. CNN & FCDNN Model

d) CNN & FCDNN & CNN

The fourth hybrid deep neural network is combination between CNN followed by FCDNN, then followed by CNN and has the following specification as shown in Fig. 19.

1. Input layer with data dimension d=10 for DroidCollector Android data set, and d=121 for NSL-KDD data set.
2. One-dimension CNN network: filters=64, kernel size=3, activation function is relu, and the input victor dimension is (121,1) for NSL-KDD data set and (10,1) for Android data set.
3. One-dimension CNN network: filters=64, kernel size=3, and activation function is relu
4. One-dimension max pooling layer: with pool size =2
5. 4 layers dense of FCDNN: 100 neurons and relu activation function
6. One-dimension CNN network: filters=64, kernel size=3, and activation function is relu
7. Flatten layer
8. Dropout layer
9. Output layer: one neuron and activation function is sigmoid

```
In [12]: model = Sequential()
         model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(10,1)))
         model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
         model.add(MaxPooling1D(pool_size=2))
         model.add(Dense(100, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
         model.add(Flatten())
         model.add(Dropout(0.5))

         model.add(Dense(1))
         model.add(Activation('sigmoid'))
```

Figure 19. CNN & FCDNN & CNN model

e) FCDNN & CNN

The fifth hybrid deep neural network is combination of multiple dense deep neural network layers FCDNN followed by two CNNs and has the following specification as shown in Fig. 20.

1. Input layer with data dimension d=10 for DroidCollector Android data set, and d=121 for NSL-KDD data set.
2. 4 layers dense of FCDNN: 100 neurons and relu activation function
8. One-dimension CNN network: filters=64, kernel size=3, and activation function is relu
3. One-dimension CNN network: filters=64, kernel size=3, and activation function is relu
4. One-dimension max pooling layer: with pool size =2
5. Flatten layer
6. Dropout layer
7. Output layer: one neuron and activation function is sigmoid

```
In [10]: model=Sequential()

         model.add(Dense(100, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Dense(100, activation='relu'))
         model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
         model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
         model.add(MaxPooling1D(pool_size=2))
         model.add(Flatten())

         model.add(Dropout(0.5))

         model.add(Dense(1,activation='sigmoid'))
```

Figure 20. FCDNN & CNN Model

5. EXPERIMENTAL RESULTS AND ANALYSIS

1) DroidCollector Android data set results

Table 4. Illustrates the performance metrics results for applying the proposed approaches deep neural networks on the DroidCollector Android Data set.

TABLE 4. DROIDCOLLECTOR ANDROID DATA SET PERFORMANCE RESUITS

	Proposed approaches				
Metrics	**FCDNN**	**CNN**	**CNN& FCDNN**	**CNN&FCDNN &CNN**	**FCDNN &CNN**
Training accuracy	0.945	0.917	0.94	0.961	0.941
Testing accuracy	0.862	0.833	0.861	0.863	0.853

F1-score	0.823	0.816	0.848	0.85	0.839
TP	2348	2061	2225	2242	2188
FP	287	226	168	172	201
TN	4404	4465	4523	4519	4490
FN	793	1080	916	899	953
TP rate (sensitivity)	0.747	0.656	0.708	0.71	0.70

Fig. 21. Illustrates the comparison of the five proposed approaches on performance metrices: training accuracy, testing accuracy, F1-score for testing, and TP rate or sensitivity.

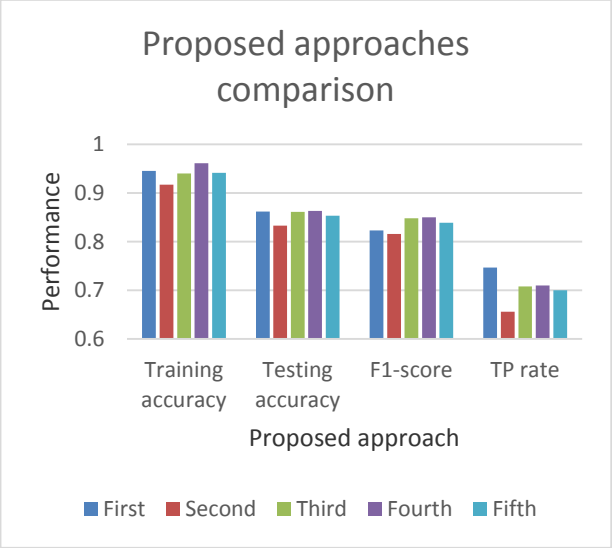

Figure 21. Proposed approaches comparisons

For the training accuracy, the fourth proposed approach which is CNN followed by FCDNN followed by CNN gets the highest value, and the worst approach is CNN approach. The second ranked approach is the FCDNN. However, all the approaches are performing reasonably well since all are above 90%.

For testing accuracy, the fourth proposed approach which is CNN followed by FCDNN followed by CNN gets again the highest value, and the worst approach is again CNN approach. The first approach FCDNN and the third one which is CNN followed by FCDNN get very closest values to the fourth approach. It is to be noticed that the ranking order for the accuracy for both training and testing is almost the same, especially for the highly ranked ones. This is actually expected, the ones perform well during the training are expected to perform also well during the testing.

For testing F1-score, the fourth proposed approach which is CNN followed by FCDNN followed by CNN gets the highest value, and the worst approach is again CNN approach. The third one which is CNN followed by FCDNN gets very closest values to the fourth approach (CNN&FCDNN&CNN).

For testing TP rate or sensitivity, the first proposed approach which is FCDNN gets the highest value, and the CNN&FCDNN&CNN approach get the second position. The worst approach is still the same which is CNN approach.

We can conclude that in DroidCollector Android data set the fourth approach CNN&FCDNN&CNN is the best one for DroidCollector Android data set. We can also conclude that FCDNN approach is having the best sensitive. In all metrics, CNN alone was the worst, but it should be clear that its metrics values are not so bad compared to other metrics values related to other approaches.

2) NSL-KDD data set results

Table 5 illustrates the performance metrics results for applying the proposed deep neural networks approaches on the NSL-KDD data set.

TABLE 5. NSL-KDD DATA SET PERFORMANCE RESULTS

	Proposed approaches				
Metrics	FCDNN	CNN	CNN&FCDNN	CNN&FCDNN&CNN	FCDNN&CNN
Training accuracy	0.995	0.9962	0.9962	0.9963	0.993
Testing accuracy	0.993	0.9968	0.993	0.997	0.934
F1-score	0.993	0.9968	0.993	0.997	0.934
TP	13475	13457	13371	13350	13308
FP	101	55	62	41	80
TN	11545	11657	11650	11630	11734
FN	74	26	112	92	73
TP rate (sensitivity)	0.994	0.998	0.991	0.993	0.994

Fig. 22 illustrates the comparison of the five proposed approaches on performance metrics: training accuracy, testing accuracy, F1-score for testing, and TP rate or sensitivity:

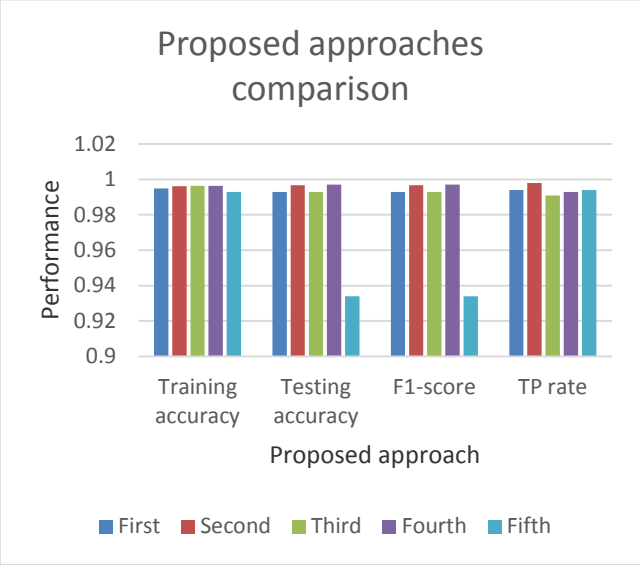

Figure 22. Proposed approaches comparisons

For the training accuracy, the fourth proposed approach which is CNN followed by FCDNN and then followed by CNN gets the highest value, and the worst approach is the fifth one which is FCDNN followed by CNN approach. The rest of the approaches get very close values to the highest one.

For testing accuracy, the fourth proposed approach which is CNN followed by FCDNN followed by CNN gets the highest value, and the worst approach is the fifth FCDNN followed by CNN approach. CNN approach gets very close value to the fourth approach.

For F1-score, the fourth proposed approach which is CNN followed by FCDNN followed by CNN gets the highest value, and the worst approach is the fifth approach FCDNN followed by CNN. CNN approach gets very close values to the fourth approach.

For testing TP rate or sensitivity, the second proposed approach which is CNN gets the highest value, and the worst approach is the third approach which is CNN followed by FCDNN approach.

We can conclude that the fourth approach CNN followed by FCDNN followed by CNN is the best approach. Also, it is to be noticed that CNN approach is having the best sensitivity, and its performance is very good comparing to its performance when Android data set is used. FCDNN is still performing well in both the data sets.

3) Proposed approaches and previous algorithms comparisons

Many algorithms were applied on the selected data sets, the following tables show the performance metrics accuracy and F1-score for the previous and the proposed approaches on the two data sets. Table 6 shows the results obtained by the proposed approaches and the previous machine learning approaches using Android data set [11], whereas Table 7 shows the same but using KDD-NSL data set. It is worth mentioning that the experiments on KDD-NSL data set using various machine learning approaches have been conducted by us for the sack of comparison study.

TABLE 6. PREVIOUS AND PROPOSED ALGORITHMS PREFORMANCE COMPARISONS ON ANDROID DATA SET

Previous Algorithms	**Accuracy**	**f1-score**
Naive Bayes	0.43	0.59
Random Forest	0.88	0.85
KNN. K=4	0.83	0.79
SVM	0.51	0.65
Logistic regression	0.58	0.2
Decision Tree	0.86	0.83
Gaussian with RBF kernel	0.789	0.773
Our Algorithms	**Accuracy**	**f1-score**
FCDNN	0.862	0.823
CNN	0.833	0.816
CNN+FCDNN	0.861	0.848
CNN+FCDNN+CNN	0.863	0.88
FCDNN+CNN	0.853	0.839

TABLE 7. PREVIOUS AND PROPOSED ALGORITHMS COMPARISONS ON KDD-NSL DATA SET

classic Algorithms	**Accuracy**	**f1-score**
Naive Bayes	0.8474	0.8395
Random Forest	0.998	0.998
KNN. K=4	0.9972	0.9972
SVM	0.9924	0.992
Logistic regression	0.9738	0.9737
Decision Tree	0.9985	0.9985
Our Algorithms	**Accuracy**	**F1-score**
FCDNN	0.993	0.993
CNN	0.9968	0.9967
CNN+FCDNN	0.993	0.993
CNN+FCDNN+CNN	0.997	0.997
FCDNN+CNN	0.9934	0.9934

For DroidCollector Android data set, random forest could outperform all the deep neural network approaches with small difference. The second best approach is the CNN followed by FCDNN followed by CNN, which is the best approach for those related to deep neural networks.

For NSL-KDD data set, most algorithms get high performance values with very small differences, except Naïve Bayes algorithm.

It is clear from the experiments that deep neural networks approaches are good competitive approaches with other approaches towards detection of intrusions for various mobile platforms. Also, CNN&FCDNN&CNN approach is a good approach and shows very good results. Adopting CNN&FCDNN&CNN in other various domains could be a potential work.

6. CONCLUSIONS AND FUTURE WORK

In this research, deep neural networks approaches have been developed to explore deep neural networks capability for intrusion detection for different mobile phones platforms using two data sets, Android data set and KDD-NSL data set. The proposed approaches are:

1. FCDNN with multiple hidden layers
2. CNN model
3. CNN followed by FCDNN
4. CNN followed by FCDNN followed by CNN
5. FCDNN followed by CNN

The first Android data set is a small data set with only 10 data dimensions, whereas the second data set NSL-KDD is a large famous data set with 121 data dimensions. The results show that CNN followed by FCDNN followed by CNN is the best approach. To measure the performance of the proposed deep neural networks approaches, a comparative study has been conducted between the proposed approaches and various approaches of machine learning. Random forest obtained the best results; however, deep learning approaches show that they are good approaches and can very well compete with various other approaches for intrusion detection for mobile platforms. CNN&FCDNN&&CNN obtained accuracy of 0.863 and 0.997 for Android data set and NSL-KDD data set respectively, whereas the random forest approach obtained 0.88 and 0.998 for the same order. The obtained results encourage researchers to try CNN&FCDNN&CNN in other application domains. For future work, exploring deep neural network capability for intrusion detection on other mobile data sets with enough dimensions is very interesting research. Also, developing more hybrid deep neural networks approaches using recursive neural network and recurrent neural network with combination with CNN network. In addition, experimenting CNN&FCDNN&CNN structure with various problem domains can be a good research potential.

REFERENCES

[1] L. N. Tidjon, M. Frappier, and A. Mammar, "Intrusion Detection Systems : A Cross-Domain Overview," no. 1, p. 1.

[2] G. L. Scoccia, S. Ruberto, I. Malavolta, M. Autili, and P. Inverardi, "An investigation into Android run-time permissions from the end users' perspective," *Proc. - Int. Conf. Softw. Eng.*, pp. 45–55, 2018.

[3] J. Cheng, R. Xu, X. Tang, V. S. Sheng, and C. Cai, "An abnormal network flow feature sequence prediction approach for DDoS attacks detection in big data environment," *Comput. Mater. Contin.*, vol. 55, no. 1, pp. 95–119, 2018.

[4] C. Yang, W. Yang, and H. Shi, "DoS attack in centralised sensor network against state estimation," *IET Control Theory Appl.*, vol. 12, no. 9, pp. 1244–1253, 2018.

[5] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 2, no. 1, pp. 41–50, 2018.

[6] P. Zegzhda, D. Zegzhda, E. Pavlenko, and G. Ignatev, "Applying deep learning techniques for Android malware detection," *ACM Int. Conf. Proceeding Ser.*, 2018.

[7] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network Intrusion Detection for IoT Security Based on Learning Techniques," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[8] H. Alqahtani, I. H. Sarker, A. Kalim, S. M. Minhaz Hossain, S. Ikhlaq, and S. Hossain, *Cyber intrusion detection using machine learning classification techniques*, vol. 1235 CCIS, no. July. Springer Singapore, 2020.

[9] A. Ignatov *et al.*, "AI benchmark: All about deep learning on smartphones in 2019," *Proc. - 2019 Int. Conf. Comput. Vis. Work. ICCVW 2019*, pp. 3617–3635, 2019.

[10] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," *arXiv*, pp. 1–43, 2017.

[11] C. C. U. Lopez, J. S. D. Villarreal, A. F. P. Belalcazar, A. N. Cadavid, and J. G. D. Cely, "Features to detect android malware," *2018 IEEE Colomb. Conf. Commun. Comput. COLCOM 2018 - Proc.*, pp. 0–5, 2018.

[12] Z. Allaf, M. Adda, and A. Gegov, "A Comparison Study on Flush+Reload and Prime+Probe Attacks on AES Using Machine Learning Approaches," *Adv. Intell. Syst. Comput.*, vol. 650, pp. 203–213, 2018.

[13] W. Qiang, J. Yang, H. Jin, and X. Shi, "PrivGuard: Protecting Sensitive Kernel Data from Privilege Escalation Attacks," *IEEE Access*, vol. 6, pp. 46584–46594, 2018.

[14] J. Wen *et al.*, "Overview of classification of Alzheimer's disease," *Med. Image Anal.*, vol. 63, 2020.

[15] S. P. Singh, A. Kumar, H. Darbari, L. Singh, A. Rastogi, and S. Jain, "Machine translation using deep learning: An overview," *2017 Int. Conf. Comput. Commun. Electron. COMPTELIX 2017*, pp. 162–167, 2017.

[16] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, "An overview and comparative analysis of recurrent neural networks for short term load forecasting," *arXiv*, pp. 1–41, 2017.

[17] X. Ma, X. Fu, B. Luo, X. Du, and M. Guizani, "A design of firewall based on feedback of intrusion detection system in cloud environment," *2019 IEEE Glob. Commun. Conf. GLOBECOM 2019 - Proc.*, 2019.

[18] A. Kumar Khare, J. L. Rana, and R. C. Jain, "Detection of Wormhole, Blackhole and DDOS Attack in MANET using Trust Estimation under Fuzzy Logic Methodology," *Int. J. Comput. Netw. Inf. Secur.*, vol. 9, no. 7, pp. 29–35, 2017.

[19] J. Kaur, "Wired LAN and wireless LAN attack detection using signature based and machine learning tools," *Lect. Notes Data Eng. Commun. Technol.*, vol. 3, no. January 2018, pp. 15–24, 2018.

[20] H. Gu *et al.*, "DIAVA: A Traffic-Based Framework for Detection of SQL Injection Attacks and Vulnerability Analysis of Leaked Data," *IEEE Trans. Reliab.*, vol. 69, no. 1, pp. 188–202, 2020.

[21] N. Islam, S. M. Shamim, F. Rabbi, and S. I. Khan, *Firewall on Spanning Tree Protocol over*, no. March 2021. Springer Singapore, 2020.
[22] S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "DroidLight: Lightweight Anomaly-based Intrusion Detection System for Smartphone Devices," *ACM Int. Conf. Proceeding Ser.*, vol. Part F1656, no. April, 2020.
[23] J. Ribeiro, F. B. Saghezchi, G. Mantas, J. Rodriguez, and R. A. Abd-Alhameed, "HIDROID: Prototyping a Behavioral Host-Based Intrusion Detection and Prevention System for Android," *IEEE Access*, vol. 8, pp. 23154–23168, 2020.
[24] S. Jeong, B. Jeon, B. Chung, and H. K. Kim, "Convolutional neural network-based intrusion detection system for AVTP streams in automotive Ethernet-based networks," *Veh. Commun.*, vol. 29, 2021.
[25] R. Vinayakumar, K. P. Soman, and P. Poornachandrany, "Applying convolutional neural network for network intrusion detection," *2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017*, vol. 2017-Janua, pp. 1222–1228, 2017.
[26] L. Mohammadpour, T. C. Ling, C. S. Liew, and A. Aryanfar, "A Mean Convolutional Layer for Intrusion Detection System," *Secur. Commun. Networks*, vol. 2020, no. Ml, 2020.
[27] B. Riyaz and S. Ganapathy, "A deep learning approach for effective intrusion detection in wireless networks using CNN," *Soft Comput.*, vol. 24, no. 22, pp. 17265–17278, 2020.
[28] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *IEEE Symp. Comput. Intell. Secur. Def. Appl. CISDA 2009*, no. July, 2009.
[29] C. Urcuqui, J. Osorio, A. Navarro, and M. García, "Machine learning classifiers to detect malicious websites," *CEUR Workshop Proc.*, vol. 1950, no. October, pp. 14–17, 2017.
[30] A. F. M. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," *arXiv*, no. 1, pp. 2–8, 2018.
[31] A. C. Marreiros, J. Daunizeau, S. J. Kiebel, and K. J. Friston, "Population dynamics: Variance and the sigmoid activation function," *Neuroimage*, vol. 42, no. 1, pp. 147–157, 2008.

Nabil M. Hewahi obtained his PhD degree in Computer Science from Jawaharlal Nehru University, New Delhi, India in 1994, and M.Tech degree in Computer Science and Engineering from Indian Institute of Technology, Bombay, India in 1991. He is now with the university of Bahrain, Bahrain and the Islamic University of Gaza, Palestine. Dr. Hewahi is a full professor of Computer Science (Artificial Intelligence) since 2006. He published over 65 papers in well-known journals and conferences. His main research interest is intelligent systems including machine learning and knowledge representation.

Nouf Alharbi obtained her BSc degree in Computer Science from Princess Nourah University, Riyadh in 2015. Also obtained her Master's degree in Cyber Security from University of Bahrain, Bahrain in 2020.