Exploiting Incremental Classifiers for the Training of an Adaptive Intrusion Detection Model

Marwa R. Mohamed[1], Abdurrahman A. Nasr[2], I. F. Tarrad[3] and Salah R. Abdulmageed[2]
(Corresponding author: Marwa R. Mohamed)
Department of Communication Engineering, University of Helwan[1]
Cairo, Egypt
Department of Computers and Systems Engineering, University of Al Azhar[2]
Department of Communication Engineering, University of Al Azhar[3]
(Email: marwa.rizk@chi.edu.eg)
(Received Dec. 6, 2017; revised and accepted June 18, 2018)

Abstract

Due to the fact that network data is dynamic in nature, the demand for adaptive Intrusion Detection System (IDS) has increased for smart analysis of network data stream. An intrusion detection system is a component of the information security and its main aim is to detect abnormal activities of the network and tries to prevent suspicious data streams that might lead to network security breach. However, most IDS poverty to the capability to detect zero-day or previously unknown attacks. As such, two types of IDS have been contemplated for detecting network threats, namely, signature-based IDS and anomaly detection system. The former depends on stored signatures in a database (thus, its name) to detect intrusions, whereas the latter develops a model based on normal system or network behavior, with the aim of detecting both recognized and novel attacks. The two types of intrusion detection systems confront many problems comprehensive; continuous learning, scalability, a high rate of false alarm, and inability to work in the online model. Here, an Adaptive Intrusion Detection Model (AIDM) is proposed. Such model is an intelligent and learnable anomaly detection model that overcomes the problems of traditional anomaly detection systems namely, high false alarm, real-time learning, and scalability. In this paper, AIDM exploited and studied a set of different incremental machine learning classifiers for intelligent detection and analysis of network data streams is carried out. Such incremental classifiers are Non-Nearest Generalized Exemplar (NNGE), Incremental Naïve Bayes (INB), Hoffeding Trees (HT), Instance-Based K- Nearest Neighbor (IBK) and Radial Basis Function Neural Network (RBFNN). Besides that, we utilized Deep Learning 4 Java Multilayer Perceptron (DL4JMLP) classifier for a deep learning approach. Furthermore, a comparison of results between seven machine learning classifiers has been performed to choose the best classifier result capable of recognizing the incoming unknown attacks from the network traffic. These classifiers are incremental in nature such that it can learn network data streams in real-time without the need for redeployment of network infrastructure. AIDM is evaluated using three different datasets collected from Defense Advanced Research Projects Agency (DARPA), Kyoto University and the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS), the training model was obtained by the aforementioned data mining techniques. The evaluation of AIDM indicated promising and improved results with the deep learning classifier (DL4JMLP) when compared with above-mentioned Incremental classifiers and the recent best known related work for detecting anomalous network traffic.

Keywords: Anomaly Detection; Deep Learning; Incremental Classifiers; Intrusion Detection System; Machine Learning

1 Introduction

Intrusion Detection System (IDS) is one of the most popular technology for securing dynamic network environments which used for monitoring, detecting and responding to any potential presence of abnormal activities by both internal and external intruders. IDS composed of methods and techniques that automate the process of detecting attacks occurring over the network, alert the system administrators to any abnormal events and provide a report for any loss of confidentiality [34].

An IDS is categorized according to detection of attacks into two techniques; anomaly and signature-based detection. The signature (misuse) detection technique [32] used to detect attacks by comparing current activities to a list

of observed signatures. It is more successful in detecting known attacks but it produces high false negative alarm with novel attacks. Anomaly detection technique [14] depends on building a dynamic normal behavior model for a user or a network by using machine learning methods wherefore this technique is effective for detecting unknown attacks where any deviation from normal behavior model indicates as an intrusion. The Host-based and Network-based techniques are two primary types of detection sources for IDS [22]. Host-Based Intrusion Detection System (HBIDS) analyzes a certain user profile or system events to detect attacks whereas Network-Based Intrusion Detection System (NBIDS) analyzes the flow of packets on the network.

Intelligent intrusion detection systems based on machine learning classifiers must be fast in monitoring and analyzing such evolved streams in real time. Streaming data generated from computer networks are changing continuously over time and infinite in incoming with a high speed. These characteristics need a number of challenges associated with mining data streams [20] thus these systems should correctly detect unlabeled instances by training them with some labeled samples from each category in dataset to build a learnable model able to classify incrementally unseen samples without any need to previous data and should update itself to accommodate new data arrive over time.

Incremental learning (IL) [37] is a method of machine learning and a classification task which used to build a learnable model for effective classification during training from the dynamic network. Incremental learning technique processes the new instances without any need to train the model where the model updated automatically after receiving each new instance. Incremental learning plays a vital role in anomaly detection system where normal profiles are updated automatically according to any changes occur without a need to train the model using previously instances. The structure of incremental learning classifiers has the ability to detect novel intrusions and handle concept drift in a dynamic network that is changed over time.

Latterly, deep learning approach has achieved revolutionize in the field of artificial intelligence (AI) and become a step toward building independent systems with a higher-level [35].

Deep learning approach has the ability to learn valuable features from a huge amount of unlabelled data stream by emulating the used methods to process data and create patterns help in decision making. We are seeking by using this approach overcome the challenges of traditional intrusion detection systems which uses typical classifiers [6].

Some of the large-scale attacks can be considered acts of cyber war [28], where attacks against any telecommunications infrastructure in every government could cause damage and palpable effects *e.g.* in Turkey 2008, how cyber attacks able to hide the blast in an oil pipeline from the control room thus it was unable to respond immediately by hacking security cameras.

Despite that, IDSs become widespread as a security measure thus it is substantial to know the drawbacks of traditional systems [33] to enhance the performance of IDS. The most limitations of traditional IDS [23] are, they usually produce a large numbers of false reports thus many real attacks are ignored due to the unpredictable activities for both user profile and computer networks, need training examples for system events to describe normal behaviors patterns and inability to analysis the encrypted packets in the network to detect any malicious activities.

In this paper, we are focused on the second type of IDS namely, the anomaly detection which uses different machine learning algorithms for demarcating the region between normal and anomaly behaviors. The incremental method is broadly classified as a single shot and has the ability to evolve the structure of classifiers to process new instances and classes without having to be completely retrained.

To this end, we proposed an Adaptive Intrusion Detection Model (AIDM) based on based on machine-learning techniques [36] and feature extraction which competent to detect zero-time attacks and distinguish between normal and anomaly instances in the computer networks with keeps the knowledge that had been learned previously from old examples.

The rest of paper is structured as follows: Section 2 highlights the related works done. Section 3 discusses the incremental and deep learning classifiers. Section 4 describes our Adaptive Intrusion Detection Model (AIDM) while Section 5 demonstrates the implementation of AIDM with a description of the datasets used and discusses the experimental results. Section 6 presents a comparative study with recent related works. Section 7 summarizes the proposed model.

2 Related Work

Many research papers are published regarding the incremental classifiers for detecting the network attacks in a huge amount of data streams because of they able to build a learnable model for new and unknown attacks [10]. Recently, deep learning approach has become a renowned technique in machine learning due to the developments are done in the field of artificial intelligence. Therefore, most of the researchers utilize the deep learning in the field of intrusion detection systems to improve the performance of the detection and overcome the shortages in the traditional systems where they focus on various offline algorithms that conducted on the most popular KDD-Cup 99 benchmark dataset to survey and evaluate the performance of IDS. KDD-Cup 99 dataset has a lot number of attributes which supportive for testing feature selection techniques. KDD-Cup 99 dataset is prepared and managed by MIT, Lincoln Labs by DARPA Intrusion Detection Evaluation Program. Presently, literature works use some public datasets such as NSL-KDD, Kyoto 2006+,

and UNSW-NB15 to evaluate IDS.

Popoola *et al.* [24] proposed an effective feature selection technique for NID using discredited differential evolution (DDE). The results of the proposed technique presented that the model is able to identify 16 features capable of classifying the connections in the training and testing NSL-KDD dataset where achieved 99.92% classification accuracy for the training set and 88.73% accuracy for new attacks moreover it helped in reducing the training and testing time by using (C4.5) classifier.

Amrita *et al.* [25] proposed a hybrid feature selection approach - Heterogeneous Ensemble of Intelligent Classifiers (HYFSA-HEIC) to maximize the accuracy with low false alarm for intelligent lightweight network intrusion detection system which classifies input traffic into intrusion or normal. The ensemble method was built using 6 features and has been evaluated using both datasets KDD-Cup99 and Corrected Test where HYFSA-HEIC combined the hybrid feature selection approach (HyFSA) and a heterogeneous ensemble of intelligent classifiers (HEIC). The Heterogeneous ensemble employed five diverse accurate intelligent classifiers are, NB, NN (SGD), RIPPER, C4.5 and RF where their decisions were combined by utilizing majority voting of elementary combiner based on algebraic combination rule. The results showed that the proposed approach outperformed other methods with a reduction in the training and testing time by 50.79% and 55.30% respectively.

Boujnouni *et al.* [8] explored a new intrusion detection system (IDS) based on information gain criterion to estimate the quality of the attributes in the dataset. The proposed model works sequentially: Firstly preprocessing the intrusion dataset NSL-KDD to exclude varying resolution and ranges and then the novelty model takes a decision whether network traffic is an attack or normal based on SSPV-SVDD as classifier and SMO as a solver. The new IDS with the improved version of Support Vector Domain Description (SVDD) called SSPV-SVDD achieved 77.5% novelty detection rate.

Jayaswal *et al.* in [13] presented an effective and flexible NIDS based on deep learning approach by using self-taught learning (STL). The proposed model has two stages: In the first stage, a sparse auto-encoder used for the feature learning and in the second stage a soft-max regression used for classification NIDS. The evaluation of the model is conducted on the processed NSL-KDD dataset. The implementation used two types of classification 2-class and 5-class on the tested data. The model achieved 88.39% accuracy rate for the 2-class classification whereas SMR achieved 78.06%. For the 5-class classification, the f-measure values are 75.76% and 72.14% for STL and SMR respectively.

Belouch *et al.* in [4] introduced a proposed model based on two-stage and used a classification Reduced Error Pruning Tree (RepTree) algorithm for network intrusion detection system. In the first stage, the incoming traffic is firstly classified according to its protocol TCP, UDP and other, to normal or attack then a preprocessing applied for each subset. In the second stage, in case the traffic was an attack, a pre-trained multiclass classifier identifies the type of attack to provide the best response. The proposed model is evaluated using two datasets are NSL-KDD and UNSW-NB15. Feature selection technique was performed for each protocol to reduce the size of datasets by select features. The proposed model achieved a quite same accuracy of 88.85% with the combination of NBTree + RandomTree classifier with an advantage in training and testing time performance for the NSL-KDD dataset. For UNSW-NB15 dataset the model achieved the best accuracy of 89.95% with Decision tree classifier.

Jabbar *et al.* [12] demonstrated that the novel ensemble classifier (RFAODE) for intrusion detection system which built using two well -known algorithms RF and AODE is efficient for classifying the traffic to normal or malicious. The model outperforms classifiers namely, na?ve Bayes, J48, and PART with 90.51% accuracy using Kyoto dataset.

Shone *et al.* [29] presented a novel deep learning technique for intrusion detection system where they proposed a non-symmetric deep auto-encoder (NDAE) for unsupervised feature learning. The model was implemented using tensor Flow and evaluated using two datasets KDD Cup-99 and NSL-KDD. For KDD Cup '99 dataset the evaluation using 5-class proved that the proposed model was able to offer an average accuracy of 97.85% and a 3.8% improvement on its own accuracy for NSL-KDD dataset using 13-class.

Pattern recognition and increasing the time of learning became one of the main problems for most traditional intrusion detection systems that based on typical machine learning and signature-based approach. These traditional systems have limitations *e.g.* low detection rate for zero-day attacks, the rapid change in attacks behavior and arriving huge amount of data. For all of these issues, we proposed an Adaptive Intrusion Detection Model (AIDM) for network traffic classification to continue the work done from the previous related works. In addition, comparing the accuracy between seven classifiers has been done with a report on results.

3 Learning Classifiers

Different of effective IDS focus on incremental and deep learning approaches due to the ability to increase the model's performance and adapt to new samples without missing its existing knowledge. The intrusion network detection or the network flow classification can be formalized as shown: a set of instances in the dataset $D = \{S1, S2, \cdots, Sn\}$, where each instance has a labeled category class. For building a trained model, some labeled instances are used to learn the model how to link each instance to one category. For this purpose, we evaluated AIDM using seven different classification techniques able to handle data streams with high dimensional features for picking up the most effective classifier for building the

proposed model.

3.1 NNGE (Non-Nested Generalized Exemplars)

NNGE is a nearest-neighbor learner [15] that expands generalized exemplars in memory depending on the distance between a set of exemplars using Euclidean distance function. The generalized exemplars are a group of samples explained as rules which reduce the classification time without any effect on the accuracy. NNGE able to classify sequence, multiclass data, and data with a different format. NNGE learns incrementally by classifying then generalizing a new example where an example is generalized and combined with the nearest exemplars having the same class. The generalization process of an exemplar is aborted if NNGE classifier checks that exemplar is similar to any exemplar in the area of feature space and may conflict with a new exemplar. Learning process of NNGE classifier builds a group of generalized exemplars where a hyper-rectangle covers a group of exemplars $\{H^1, H^2, \cdots, H^k\}$. In case of a hyper-rectangle covers only one exemplar, it is considered to be a non-generalized exemplar. Each example in training instances passing through three steps: the first step is the classification, where finding the hyper-rectangle H^k that is nearest to training instances E^j by calculating the distance $D(E, H)$ between an example and the hyper-rectangle H as in Equation (1):

$$D(E,H) = \sqrt{\sum_{i=1}^{n}(w_i \frac{(d(E_i,H_i))}{E^{max} - E^{min}})^2} \quad (1)$$

Where $E^{max} - E^{min}$ are the ranges of values of attribute I over the training set, H_i is the interval $[H_i^{min}, H_i^{max}]$ which based on the attribute type. The second step is the model adjustment, where the hyper-rectangle H^k splits when covers conflicting example. This step applied when the hyper-rectangle covers an example of a different class. The third step is a generalization, where extend H^k to cover E^j where it extends the closest hyper-rectangle H^k that has the same class. This extension can be done only if the new hyper-rectangle doesn't overlap with other hyper-rectangles having a different class (See Algorithm 3.1).

3.2 Naïve Bayes (NB)

NB [21] uses Bayesian classification method which useful in understanding a lot of learning classification algorithms. NB based on Bayes theorem considers one of the simplest probabilistic classifiers. This classifier has the ability to handle high streaming data, use data streaming as supervised classification and effective for various real applications due to its incremental nature. NB classifies new data streams by finding the highest posterior probability where it can predict whether X belongs to the class C_i.

The posterior probability $P(C_i\ x)$ of class C_i defined in Equation (2):

$$P(c_i\ x) = P(x\ C_i)p(C_i)/p(x) \quad (2)$$

Where c is the predicted class and x is the instance. NB classifier is an independence assumption where the probability of one feature is unrelated to affect the probability of the other. It has two phases namely, a training phase and testing phase. In training phase, NB classifier calculates the probabilities of each attribute and stores this probability then, it calculates all time taken for the probabilities for each attribute. In the testing phase, the time taken to calculate the probabilities for each attribute is proportional to a number of attributes (See Algorithm 3.2).

Algorithm 3.1 Non-Nested Generalized Exemplars

$H \leftarrow \Phi$
for j $\in$ { 1, ….L} **do**
  if $H = \Phi$ **then**
    $H \leftarrow H \cup E^j$
  else
  Find H^k ε H such that $D(H^k, E^j) \leq D(H^q, E^j)$
  for all $H^q \in H$
    if $D(H^q, E^j) = 0$ **then**
      if $class(H^k) \neq class(E^j)$ **then**
      $H_k \leftarrow$ Split (H^k, E^j)
      end if
  else
    $H' \leftarrow$ Extend (H^k, E^j)
      if H' overlaps with conflicting hyperrectangles **then**
      $H \leftarrow H \cup \{E^j\}$
    else
      $H \leftarrow H \setminus \{H^k\} \cup \{H'\}$
      end if
    end if
  end if
end for

Algorithm 3.2 Naïve Bayes

Let training set of instances with class labels $c_1, c_2, \ldots, c_k$, n-dimensional vector X = {x_1 , x_2 ,… … , x_k} and n attributes, A_1 , A_2 ,… …, A_n , respectively.
For all training instances **do**
Calculate P(x \ c_j)
end for
  For all 1 ≤ j ≤ m, j≠ I **do**
  Calculate P(c_j) and p(x) for all classes
  Calculate P(c_j \ x) = P (x \c_j) p(c_j) / p(x)
    If P (c_j \ x) > P (c_i\x), **then** X $\in$ c_j having the highest posterior probability.
    end if
  end for

3.3 HT (Hoeffding Tree)

HT is a decision tree method [19] used for classifying data stream and this method considers the most efficient clas-

sification technique in data mining. Most existing classification techniques are very sensitive to stream ordering and take time-consuming with a huge amount of data streams. HT is one of the most suitable classifiers for data streaming which it gives results the same extent to batch classifiers and the learning process has constant time per instance. HT uses Hoeffding bound metrics which solved one of the most critical problems in the process of choosing a node which determined how many examples in a subset of the training dataset require for each node.

If r is an actual random variable (r is the information gain of an attribute) with range R and n is the observation of this variable Hoeffding Bound states that with probability $1-\delta$, the actual mean r is at least $\bar{r}-\epsilon$, where δ defined by a user and ϵ can be expressed as in Equation (3):

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n_l}} \tag{3}$$

Let $G(X_i)$ be the heuristic measure used to choose test attributes, X_a be the attribute with highest observed G after seeing n examples and X_b the second-best attribute. Let the difference between two observed heuristic values $\Delta G = \bar{G}(X_a) - \bar{G}(X_b) \geq 0$ with a given desired δ. Hoeffding bound states that X_a is the best choice with probability $(1-\delta)$ for the number of examples (n) that have been seen at the node. In another expression we can say that X_a is the best attribute with probability $1-\delta$, if the observed difference of information gain (heuristic measure) $\Delta\bar{G} > \epsilon$ (See Algorithm 3.3).

3.4 Instance-based k-nearest neighbor Ibk (KNN)

IBK [26] is a k-nearest neighbor learner and kind of a simple instance-based learning algorithm It classifies new instance-based by determining the K-nearest neighbors on Euclidean distance metric which defined in Equation (4):

$$d(x_i, x_j) = \sqrt{\sum_{r-1}^{r=n} [a_r(x_i) - a_r(x_j)]^2} \tag{4}$$

Where $d(x_i, x_j)$ is the distance between the two instances x_i and x_j, $a_r(x)$ is the attribute value of instance x. The nearest neighbors used for classifying new instance and the relation between the time taken in classify new instance is linearly increased with the number of training instances saved in the classifier thus it called Lazy learning classifier. The new classification for new instance can be found easily by comparing the new instance with the stored trained dataset. IBK classifier has limitation *e.g.* computational complexities for finding K-nearest neighbor instances and the processing time for no longer optimal with a huge number of instances because we need to calculate all similarities between all training instances. We can solve this problem by reducing the dimensions of the features or using small datasets and dependency on the training set where building KNN (See Algorithm 3.4).

Algorithm 3.3 HT (Hoeffding Tree)

Let *HT* be a tree with a single leaf l1 (the root)
for all training examples **do**
  Sort example into leaf *l* using *HT*
  Update sufficient statistics in *l*
  Increment n_l, the number of examples seen at *l*
  if n_l mod n_{min}= 0 **and** examples are seen at *l* not all of the same class **then**
  Compute $\bar{G}_l(X_i)$ for each attribute
  Let X_a be attribute with highest $\bar{G}_l$
  Let X_b be attribute with second-highest $\bar{G}_l$
  Compute Hoeffding bound = ε = $\sqrt{\frac{R^2 \ln(1/\delta)}{2n_l}}$
  if $X_a \neq X_\Phi$ **and** ($\bar{G}_l(X_a)$- $\bar{G}_l(X_b$ > ε or ε < τ) **then**
  Replace *l* with an internal node that splits on X_a
  for all branches of the split **do**
  Add a new leaf with initialized sufficient statistics
  end for
  end if
  end if
end for

Algorithm 3.4 Instance-based k-nearest neighbor Ibk (KNN)

Input: K, the number of nearest neighbors; D, the set of the test sample; T, the set of the training sample
output: L, the label set of the test sample
L = {}
For each d in D and each t in T **do**
// Neighbors (d) return the K nearest of d
  Neighbors (d) = {}
  If |Neighbors (*d*)| < k **then**
// closest (d,t) return the K closest elements of t in d
  Neighbors (d) = closest (d,t) ∪ Neighbors (d)
  end if
  If |Neighbors (*d*)| < k **then**
Break
L = test class (Neighbors (d)) ∪ L
end for
test class (S) return the class label of S

3.5 Radial Basis Function Neural Network (RBFNN)

RBF learner [18] is a function based classifier and the feed-forward artificial neural network. RBF is an alternative to a public MLP classifier but it is superior to MLP in the fast training process and it simple in its structure.

RBF uses Gaussian transfer functions instead of sigmoidal functions which used by MLP learner. RBFNN

classifies instances by measuring the distances between centers of hidden and inputs neurons and it consists of three layers: an input layer, which feeds the feature into the network. The number of input neuron is determined by the number of input vector; hidden layer calculates the outcome of the basis functions and used the normalized Gaussian function as the radial basis function which expressed as in Equation (5):

$$\phi_j(X) = \phi(X - X_j) = \exp(\frac{1}{2\delta^2}|X - X_j|^2) \quad (5)$$

Where δ is the width of the Gaussian and K represents the number of units in the hidden layer, X_j is denoted the center of the ith node; and the output layer, which calculates the linear combination of basis functions as in Equation (6):

$$Y = f_j(x) = [\sum_{i=1}^{k} w_i \phi_i(x)] \quad (6)$$

Where $f(x)$ is the final output, $\phi_j(X)$ denotes the radial function of the jth hidden node and w_i denotes the hidden-to-output weight corresponding to the ith hidden node (See Algorithm 3.5).

Algorithm 3.5 Radial Basis Function Neural Network (RBFNN)

1: Identify set of distinct nodes available on all the routes
2: **For** each node identified
3: **Create** a neuron and assign parameters
4: **Input** = $\{X_1, X_2, \ldots, X_{mo}\}$ where the suffix mo represents the number of assigned parameters as input
5: **end**
6: **Generate** topology from neuron attributes
7: **Compute** paths available
8: Compute squared Euclidean distance of neurons
9: $\|X - X_j\|^2 = \sum_{k=1}^{mo} (X_k - X_{jk})^2$
10: **Perform** Gaussian function for the hidden layer
11: $\Phi_j(X) = \Phi(X - X_j) = \exp(\frac{1}{2\delta^2}\|X - X_j\|^2)$, j =1,2, …, k where K represents the number of units in the hidden layer
12: **Update** the neural network
13: **Add** routes to optimum route set
14: **Repeat** the above steps until all of the samples are processed

3.6 Online Accuracy Updated Ensemble (OAUE)

Online accuracy updated ensemble is an ensemble algorithm [30] where it is an improved version of accuracy weighted ensemble. OAUE classifier has a weighted pool of classifiers to predict the class of incoming sample where it predicts the new instances by aggregating all predictions using a rule called weighted voting. The new classifier is added to the ensemble after a segment of instances is classified and it replaced by the weakest ensemble member in performance. OAUE is updated incrementally when the performance of each classifier is evaluated using expected prediction error for the most recent instances. The classifier with the poorest performance is replaced from the pool of classifiers and then the weights adjusted according to accuracy (See Algorithm 3.6).

Algorithm 3.6 Online accuracy updated ensemble (OAUE)

Input : **S**: data stream of examples, d: window size, k: number of ensemble members, m: memory limit
ε: an ensemble of k weighted incremental classifiers
ε ←Φ;
Ć← new candidate classifier;
for all examples $x^t \in S$ **do**
  calculate the prediction error of all classifiers $C_i \in \varepsilon$ on x^t;
  if t > 0 and t mod d = 0 **then**
    if |ε|< k **then**
      ε ← ε ∪ {Ć};
    else
      weight all classifiers $C_i \in \varepsilon$ and Ć using (5);
      substitute least accurate classifier in ε with Ć;
    end if
    Ć← new candidate classifier;
    if memory_usage(ε) > m **then**
      prune (decrease size of) component classifiers;
    end if
  else
    incrementally train classifier Ć with x^t;
    weight all classifiers $C_i \in \varepsilon$ using (5);
  end if
  for all classifiers $C_i \in \varepsilon$ **do**
  end for
end for

3.7 Deep Learning 4 Java Multilayer Perceptron (DL4JMLP)

Deeplearning4J (DL4J) is a Java-based toolkit for configuring deep neural networks which consist of multiple layers. These layers are organized by using hyper-parameters which utilize to learn the neural networks. DL4J treats the tasks of loading data and training the algorithms as separate processes and works with a lot of different data types such as images, CSV, ARFF, plain text and Apache Camel Integration. DL4J uses for retrieving the data in a format suited for training a neural net model which uses multiple passes and each pass is called an epoch. DL4J gives data to the scientists and tools developers to build a deep neural network on a high level by using concepts

like layer. DL4J employs a builder pattern to build the neural net declaratively. The neural networks are typically trained by using batches of the training data where the updates to the weights and biases of the neural network effect on the outputs of each node of the network. Deep Learning 4 Java Multilayer Perceptron (DL4Jmlp) is a straightforward wrapper which consists of three layers: the Input layer, the hidden layer which uses ReLU (Rectified Linear Unit) activation function and it defined as in Equation (7):

$$F(x) = \max(x, 0). \tag{7}$$

Where $f(x)$ has output 0 if the input is less than 0 and the output is raw if the input is greater than 0. The output layer uses the softmax activation function which squashes the outputs of each unit between 0 and 1and it represented in Equation (8):

$$\sigma(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{k} e^{x_k}} \tag{8}$$

Where x is a vector of the inputs to the output layer, j indexes the output units, and $j = 1, 2, \cdots, K$ (See Algorithm 3.7).

Algorithm 3.7 Forward Propagation Algorithm for DL4JMLP

Input : x_i (I = 1.2…..m)
Output: $\hat{y}_i$
For i from 1 to m **do**
  $t_l = w_{hxxi} + w_{hhhi-1+bh}$
  $h_i = rleu\ (t_{il})$
  $s_i = w_{yhhi+bh}$
  $\hat{y}_i = SoftMax\ (s_i)$
end for

DL4Jmlp uses the most popular Stochastic Gradient Descent (SGD), as an optimization to the neural network [3]. Which known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization and iterative method for minimizing an objective function that is written as a sum of differentiable functions (See Algorithm 3.8).

Algorithm 3.8 Stochastic Gradient Descent (SGD) used in DL4JMLP

Input : Training data S, regularization parameters λ, learning rate η, initialization σ
Output: Model parameters $\theta = (w_0, \mathbf{w}, \mathbf{V})$
$w_0 \leftarrow 0$; $\mathbf{W} \leftarrow (0,\ldots, 0)$; $\mathbf{V} \sim \mathcal{N}(0, \sigma)$;
repeat
For $(\mathbf{x}, y) \in S$ **do**
  $w_0 \leftarrow -\eta \left(\frac{\partial}{\partial w_0} \iota\ (\hat{y}\ (\mathbf{x} \mid \theta), y) + 2\ \lambda^0 w_0\right)$;
  For $i \in \{1,\ldots, p\} \wedge x_i \neq 0$ **do**
    $w_i \leftarrow w_i - \eta \left(\frac{\partial}{\partial w_i} \iota\ (\hat{x}\ (\mathbf{x} \mid \theta), y) + 2\ \lambda^{w}_{x(i)} w_i\right)$;
    For $f \in \{1,\ldots,k\}$ **do**
      $v_{i,f} \leftarrow v_{i,f} \left(\frac{\partial}{\partial v_{i,f_i}} \iota\ (\hat{y}\ (\mathbf{x} \mid \theta), y) + 2\ \lambda^{v}_{f,x(i)} v_{i,f}\right)$;
    end for
  end for
end for
until stopping criterion is met;

4 The Proposed Model

The main strategy of the proposed AIDM, Figure 1, is to utilize the study of the incremental and deep learning methodologies for classifying the stream of the computer networks. This methodology depends on selecting the best machine learning classifier which optimizes the efficiency of anomalous detection and we compared the model with the traditional intrusion detection systems. AIDM examines all events circulating in the network by observing abnormal activities and it consists of two phases: the off-line and the on-line phases. In the off-line (training dataset) phase, the model is trained by machine learning classifiers to be more familiarize and learnable for activities exist in the network flow where labeled processed training records are fed into AIDM to construct learnable model able to be tested.

In online (testing dataset) phase, new unlabelled and labeled samples are fed into the model to predict each unlabelled samples based on the extracted trained model from the off-line phase. AIDM encompass three stages as shown in Figure 1 are, dataset collection stage, preprocessing engine stage and classification stage. In the dataset collection stage, datasets samples are collected with known labels are fed into off-line phase whereas a mixed collection of known and unknown labels are fed into the on-line phase. AIDM evaluated using three datasets are, KDD Cup 99 and UNSW-NB15 and real Kyoto 2006+. In the preprocessing engine stage, offline and online network datasets are processed using a pre-processing engine where transformation, feature extraction, and normalization techniques applied for the datasets in order to correctly identify records over which the attacks resemble. In classification stage, AIDM trained using different classifiers on labeled datasets to build a trained model able to use for new datasets. For choosing the best results for AIDM a comparison between the classifiers results is carried out.

4.1 Pre-processing Engine Stage

Data pre-processing [9] considers the first and important step for both the online and the offline phases which used to analyze and transform the datasets to a suitable form to go through learning phase and data analysis. The flow of different network traffic sources is processed by using a sub model in the pre-processing engine by using a Detector for Number of Features in Dataset (DNFD) where a proper feature extraction and transformation (coding) techniques are applied based on the number of features in each dataset. The following section provides a concise definition of the used three techniques of the preprocessing engine.

4.1.1 Data Transformation

Some machine learning algorithms handle numeric features to guarantee the best performance thus we converted the symbolic and text values to numerical values

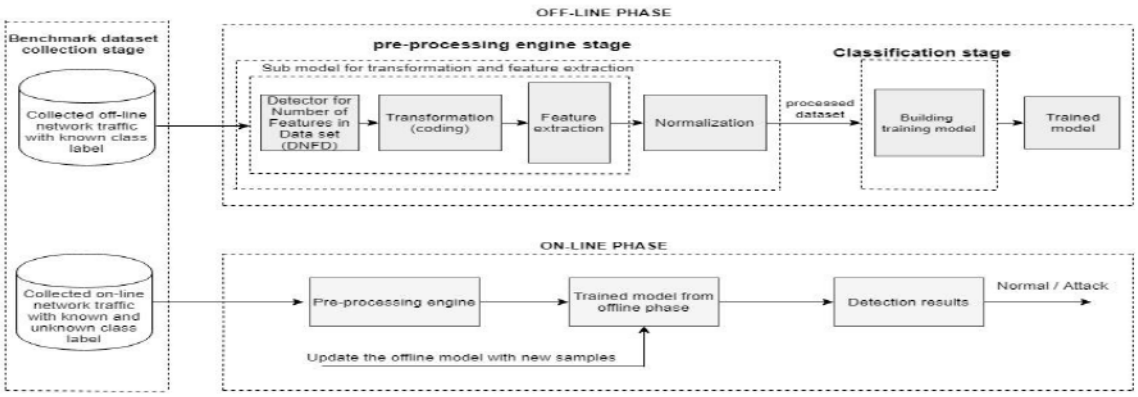

Figure 1: The structure of an Adaptive Intrusion Detection Model (AIDM)

```
0,tcp,http,SF,228,2878,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,4,0,0,0,0,1,0,0,87,255,1,0,0.01,0.01,0,0.01,0,0,Normal
0,tcp,http,SF,225,873,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,97,255,1,0,0.01,0.01,0,0.01,0,0,Normal
0,tcp,http,SF,225,873,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,3,0,0,0,0,1,0,0,97,255,1,0,0.01,0.01,0,0.01,0,0 Normal
0,udp,private,SF,28,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,91,91,0,0,0,0,1,0,0,163,91,0.56,0.02,0.56,0,0,0,0,0,DOS
0,udp,private,SF,28,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,92,92,0,0,0,0,1,0,0,164,92,0.56,0.02,0.56,0,0,0,0,0,DOS
0,udp,private,SF,28,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,93,93,0,0,0,0,1,0,0,165,93,0.56,0.02,0.56,0,0,0,0,0,DOS
0,udp,private,SF,28,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,94,94,0,0,0,0,1,0,0,166,94,0.57,0.02,0.57,0,0,0,0,0,DOS
0,tcp,ftp_data,SF,334,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,9,21,1,0,1,0.19,0,0,0,0,R2L
0,tcp,other,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,509,1,0.22,0,0.76,1,0,1,0,255,1,0,1,0,0,0.16,0,0.82,1,PROBE
0,tcp,other,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,509,1,0.22,0,0.76,1,0,1,0,255,1,0,1,0,0,0.14,0,0.84,1, PROBE
0,icmp,eco_i,SF,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0.75,0,0,1,0,0,1,0,1,0,0,0,52.16,1,0.82,0, PROBE
176,tcp,telnet,SF,1559,2732,0,0,0,3,0,1,4,1,,0,1,0,0,0,0,0,1,1,0,0,0,0,1,0,0,8,8,1,0,0.12,0,0,0,0,0.12,0.12,U2R
61,tcp,telnet,SF,2336,4194,0,0,0,3,0,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1,0,0,9,9,1,0,0.11,0,0,0,0,0.11,0.11,U2R
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,509,1,0.26,1,0.73,0,0,1,0,255,1,0,1,0,0,0.2,1,0.78,0,PROBE
```

Figure 2: Random records taken from the KDD-CUP99 dataset

where we gave a number to each symbol repeated in the feature, for example, 1 gave to the feature that has a greater number of repeatedly then, less repeatedly takes 2, *etc.* Table 1 shows an example for symbols converted to numerical values based on the previous random set of the KDD-CUP99 dataset in Figure 2. KDD CUP 99, UNSW-NB15 and Kyoto 2006+ datasets have symbolic values; where KDD CUP 99 dataset contains three symbols features are, Protocol type, Service, and Flag features; UNSW-NB15 dataset has three symbols features are protocol, service and state features; Kyoto 2006+ dataset has two symbols features are service and flag. The transformation (coding) technique depends on the DFND used in the sub model which distinguished between the numbers of features in the datasets and the coding table is applied according to the number of features for each dataset.

4.1.2 Feature Extraction

Feature extraction technique [2] is an amelioration process for detecting the abnormal events by selecting eclecticism set of features in the datasets, which are represented using different features and then, removing the features with less influence for achieving a high performance for IDSs. In data analysis for network flow classification we don't need all these features wherefore we need to pick up the most informative features which optimize the performance of machine learning algorithms. In AIDM we have been performed feature extraction technique [11] for each dataset by using information gain (IG) criteria according to the number of detected features by DNFD.

Table 1: Coding of KDD-CUP99 dataset features

classes feature		Protocol type feature		service feature		flag feature	
symbol	code	symbol	code	symbol	code	symbol	code
Normal	1	tcp	1	Private	1	SF	1
PROBE	2	udp	2	http	2	REJ	2
DOS	3	icmp	3	other	3		
U2R	4			telnet	4		
R2L	5			ftp_data	5		
				Eco_i	6		

Figure 3 shows a simple flow chart for the sub model in the preprocessing engine where the collected datasets pass through this sub model where DNFD detect the number of features in the datasets and according to the number of features in the datasets, the implementation of transformation (coding) and feature extraction techniques in a suitable manner for each dataset. Information gain (IG) criteria [1] considers one of the simplest feature selection methods used to rank all features in the datasets based on the entropy which measures the reduction in purity in an arbitrary collection of instances.

The entropy H(X) of variable Y can define as in Equation (9):

$$H(Y) = -\sum_{i} P(y_i) \log_2(p(y_i)) \tag{9}$$

Where $P(y_i)$ is the prior probability for variable Y. The entropy of variable X after observing value of another

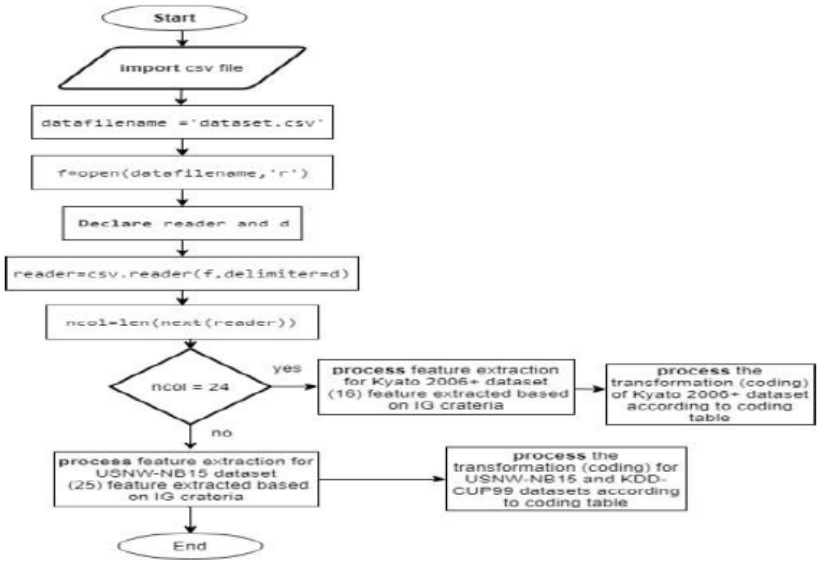

Figure 3: Flowchart of the sub model

variable Y can define as in Equation (10):

$$H(Y\backslash X) = -\sum_i P(x_i) \sum_i P(y_i\backslash x_i) \log_2(p(y_i\backslash x_i)) \quad (10)$$

Where $P(y_i\backslash x_i)$ is the posterior probability of X given Y. The information gain (feature ranking) about X provided by Y to pick up the most important features can be defined as in Equation (11):

$$IG(X/Y) = H(Y) - H(Y\backslash X). \quad (11)$$

We have been extracted 25 features from KDD-CUP99 and USNW-NB15 datasets and 18 features from Kyoto 2006+ dataset. Table 2 shows the optimal features that selected from the three datasets for AIDM.

Table 2: Selected features for the three datasets by IG

datasets	Selected features by using IG criteria
KDD-CUP99	F5, f3, f6, f33, f23, f37, f35, f40, f12,f34,f4,f41,f30,f24,f29,f38,f25, f36,f10,f27,f28,f32,f39,f31 and f13
USNW-NB15	F1,f8,f28,f13,f9,f29,f33,f12, f11,f10,f14,f18,f7,f2,f17,f26, f25,f27,f3,f6,f16,f20,f36,f5 and f19
Kyoto 2006+	F1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,f13, f14, f18,f20,,f22 and f23

The intrusion detection accuracy for the selected features by IG criteria [31] outperforms the all features for the three datasets as shown in Figure 4 which increased the effectiveness of all classifiers.

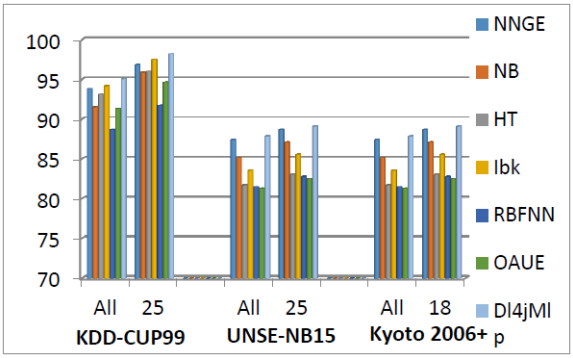

Figure 4: The accuracy of three datasets for all features and selected features

4.2 Data Normalization

The normalization technique [17] is the process of scaling each value of the attributes in the dataset into the specified new range [0, 1] or [-1, 1] which makes dataset acquires a particular property helpful in prediction purpose. Min-max normalization method converts v_i in the dataset to new value x_i using Equation (12):

$$x_i = \frac{v_i - \min(v_i)}{\max(v_i) - \min(v_i)} \quad (12)$$

Where v_i is the existing value of the attribute. The maximum and minimum values are taken over all values of the attribute normally, x_i is set to zero if the maximum is equal to the minimum.

4.3 Data Classification

The classification stage used two experiments to illustrate the importance of the feature extraction technique. The

first experiment, AIDM classified the datasets without apply feature extraction technique and the results are recorded of each classifier respectively. The second experiment, AIDM classified the datasets with applying the feature extraction technique on the datasets where the feature extraction has been applied many times to reach the maximum accuracy for each dataset. we reached to the best number of features for each dataset as shown in Table 2. Figure 4 shows a comparison between the three datasets before and after applying the feature extraction technique which proved the importance of this technique in increasing the detection accuracy.

5 AIDM Implementation

AIDM implemented using: Java programming language with aid of an open source framework Massive Online Analysis (MOA), which used for data stream mining and big data processing, and the open source WEKA tool, which have a collection of machine learning algorithms for data mining and preprocessing tasks [16]. For training the proposed model we used 80,000 of processed datasets having known attacks and 10,000 of processed datasets having unknown and known attacks for testing the model.

The performance evaluation has been taken from the final prediction of the datasets classes. The experiments were performed on a laptop having configuration Intel© Core^{TM} i3 CPU M 380 2.53 GHz, 2.53GHz, 4.00GB of RAM and the operating system is windows 7 professional.

5.1 Description of KDD CUP 99, UNSW-NB15 and Kyoto 2006+ Datasets

The evaluation of AIDM conducted on three datasets namely, KDD CUP 99, USNW-NB15 and Kyoto 2006+. The following section presents a brief description of the datasets.

5.1.1 KDD-CUP99 Dataset

KDD-cup 99 benchmark dataset [5] was built based on DARPA'98 evaluation program which is a compressed TCP dump data of network traffic during 7 weeks. Each instance in KDD-cup 99 have 41 features with a labeled class indicates one of five values are, Normal, DOS (denial of service), PROBE (surveillance), U2R (user to root) and R2l (root to local). The KDD-CUP99 dataset may not be a good choice of the neoteric computer networks due to the scarcity in computer networking datasets but it is one of the most criterion realistic benchmark datasets for training and testing intrusion detection system. The KDD-CUP99 dataset has been divided into training dataset with 4,898,431 records and 311,027 records for the testing dataset.

5.1.2 UNSW-NB15 Dataset

The UNSW-NB15 benchmark dataset [7] was developed based on an IXIA tool which used to produce nine modernistic attacks beside to, assortment broad of normal activities. The UNSW-NB15 dataset has been divided into training dataset with 82,332 records and 175, 341 records for the testing dataset. The UNSW-NB15 dataset has 45 features with a label contains nine attacks namely, Reconnaissance, Shellcode, Exploit, Fuzzers, Worm, DoS, Backdoor, Analysis and Generic beside, a normal class. UNSW-NB15 and KDD-CUP99 datasets have six common features are, protocol type, service, duration, source, and destination.

5.1.3 Kyoto 2006+ Dataset

Kyoto 2006 + dataset [27] have been collected at Kyoto University from honey spot systems. Kyoto 2006 + dataset is a real traffic traces devoid of any manual labels which each connection has 23 features and a label contains three values where, '1' value means that the session was normal, '-1' means that known attack was observed in the session and '-2' means that unknown attack was observed in the session. Kyoto 2006+ and KDD-CUP99 datasets have 14 common features namely, Duration, Service, Source bytes, Destination bytes, Count, Same srv rate, Serror rate, Srv serror rate, Dst host count, Dst host srv count, Dst host same src port rate, Dst host serror rate, Dst host srv serror rate and flag beside to, 10 new features namely, IDS detection, Malware detection, Ashula detection, Label, Source IP Address, Source Port Number, Destination IP Address, Destination Port Number, Start Time and Duration. A statistics description for the datasets based on the number of instances, attributes, and classes shown in Table 3.

Table 3: Statistics description of three benchmarks

Datasets	instances	Attributes	classes
KDD CUP 99	10,000	42	5
UNSW-NB15	10,000	49	10
Kyoto 2006+	10,000	24	4

Table 4 represented the features for each dataset used in AIDM. We randomized selected the data of 27, 28, 29, 30 and 31 December 2015 where it has the latest updated data. The statistical description of classes used in the testing (on-line) phase for the three benchmark datasets represented in Table 5.

5.2 Performance Evaluation Metrics

AIDM evaluation has been done based on (1) The cumulative accuracy, which refers to the accuracy after classification of all chunks and it is the percentage of corrected instances classified to the total number of instances), (2) kappa statistics it measures the extent of convergence of

Table 4: The Features of the KDD-CUP99, UNSW-NB15 and Kyoto 2006+ datasets

KDD-CUP99 dataset				UNSW-NB15 dataset				Kyoto 2006+ dataset	
1	Duration	21	is_host_login	1	srcip	25	trans_depth	1	Duration
2	Protocol_type	22	is_guest_login	2	sport	26	res_bdy_len	2	Service
3	Service	23	count	3	dstip	27	sjit	3	Source bytes
4	Flag	24	srv_count	4	dsport	28	djit	4	Destination bytes
5	src_bytes	25	serror_rate	5	proto	29	stime	5	Count
6	dst_bytes	26	srv_serror_rate	6	state	30	ltime	6	Same srv rate
7	Land	27	rerror_rate	7	dur	31	Sintpkt	7	Serror rate
8	wrong_fragment	28	srv_rerror_rate	8	sbytes	32	dintpkt	8	Srv serror rate
9	urgen	29	same_srv_rate	9	dbytes	33	tcprtt	9	Dst host count
10	hot	30	diff_srv_rate	10	sttl	34	synack	10	Dst host srv count
11	num_failed_logins	31	srv_diff_host_rate	11	dttl	35	ackdat	11	Dst host same src port rate
12	logged_in	32	dst_host_count	12	sloss	36	is_sm_ips_ports	12	Dst host serror rate
13	num_compromised	33	dst_host_srv_count	13	dloss	37	ct_state_ttl	13	Dst host srv serror rate
14	root_shell	34	dst_host_same_srv_rate	14	service	38	ct_flw_http_mthd	14	Flag
15	su_attempted	35	dst_host_diff_srv_rate	15	sload	39	is_ftp_login	15	IDS detection
16	num_root	36	dst_host_same_src_port_rate	16	dload	40	ct_ftp_cmd	16	Malware detection
17	num_file_creations	37	dst_host_srv_diff_host_rate	17	spkts	41	ct_srv_src	17	Ashula detection
18	num_shells	38	dst_host_serror_rate	18	dpkts	42	ct_srv_dst	18	Label
19	num_access_files	39	dst_host_srv_serror_rate	19	swin	43	ct_dst_ltm	19	Source IP Address
20	num_outbound_cmds	40	dst_host_rerror_rate	20	dwin	44	ct_src_ ltm	20	Source Port Number
		41	dst_host_srv_rerror_rate	21	Stcpb	45	ct_src_dport_ltm	21	Destination IP Address
		42	label	22	dtcpb	46	ct_dst_sport_ltm	22	Destination Port Number
				23	smeansz	47	ct_dst_src_ltm	23	Start Time
				24	dmeansz	48	Attack_cat	24	Duration
						49	Label		

prediction with correct class using Equation (13):

$$K = (Po - Pe)/(1 - Pe). \tag{13}$$

Where Po indicate the relative observed convergence, Pe indicates the hypothetical probability of chance convergence and (3) running time defined as the time taken for execution the classification of each classifier in the proposed model.

Table 5: Classes statistical distribution for 10,000 samples

Dataset	Class	Number of instances in the testing phase
KDD-CUP 99	DOS	1,818
	PROBE	2,131
	R2L	999
	U2R	52
	Normal	5,000
UNSW-NB15	Analysis	654
	Backdoor	654
	DOS	654
	Exploits	654
	Fuzzers	654
	Generic	654
	Reconnaissance	654
	Shellcode	654
	Worms	654
	Normal	4,999
Kyoto 2006+	Normal	5,000
	Attack	5,000

5.3 Experimental Results and Discussion

In this paper, seven classifiers have been used to evaluate the effectiveness and performance of the proposed AIDM to identify unknown diverse attacks in the network traffic and overcome the problems in traditional IDS.

AIDM evaluation is based on the incremental and deep learning techniques for multiclass classification for the datasets. To guarantee the best performance for AIDM we utilized six incremental classifiers and DL4JMLP deep learning classifier consecutively for training the model. The evaluation of AIDM has conducted using the three benchmark datasets compare the results among each trained model by each classifier and learning classifiers proposed by beforehand researchers has been carried out to determine the efficiency AIDM which helpful for testing the model in on-line phase.

Through our results in the proposed model, it is shown that AIDM achieved high accuracy with using Dl4jMlp deep learning classifier and its effectiveness is superior to that of traditional machine learning classifiers in differentiate attacks for multiclass classification.

Final accuracy of AIDM conducted on 10,000 samples of known and unknown attacks from the three datasets the performance was measured for every 1000 instances between samples. AIDM evaluated using a prequential evaluation (Interleaved Test-Then-Train) which used for the real-time evaluation.

Such evaluation technique it is considered an evaluation for streaming data where each sample tests first before using in training.

Table 6 summarizes the classification results of the seven selection classifiers in regard to accuracy (%), kappa statistics and running time.

It clearly shown from Table 6 that the proposed model based on deep learning approach using the Dl4jMlp Classifier and feature selection technique using IG criterion, has achieved the best accuracy of 98.2%, 92.5% and 82.12% for the three datasets, KDD-Cup 99, UNSW-NB15 and Kyoto 2006+ respectively than other incremental classifiers used by AIDM.

It is notable from Table 6 that the classification accuracies for KDD-CUP4 99 and real Kyoto 2006+ datasets achieved higher evaluation than UNSW-NB15 dataset because of the complexity of the UNSW-NB15 dataset that has an assortment of the contemporary anomaly and normal behaviors which has been developed for evaluating NIDSs. Figures 5, 6 and 7 showed the prequential accuracy for the seven classifiers for KDD Cup 99, UNSW-NB15 and Kyoto 2006+ datasets respectively. The three previous figures demonstrated that the AIDM which combined with deep learning Dl4jMlp Classifier and based on feature selection performs the best efficiency with smooth curve compared to the other curves of the classifiers.

The proposed model results as shown in Table 2 indicated promising results in terms of low computational time and high classification results for Kyoto 2006+ dataset where it achieved 92.39% accuracy with a low time of 2.85 sec by using DL4JMLP. It is noticed that the time is somewhat close for both datasets KDD-Cup 99 and UNSW-NB15 which they have the same number of selected features as mentioned in the pre-processing stage.

For the KDD-CUP99 dataset, the DL4JMLP classifier achieved a slight increase in efficiency with 97.9% than Ibk classifier which achieved 97.52%. And also, DL4JMLP classifier outperformed with a slight increase of 89.12% over NNGE which achieved 88.76% for the UNSW-NB15 dataset.

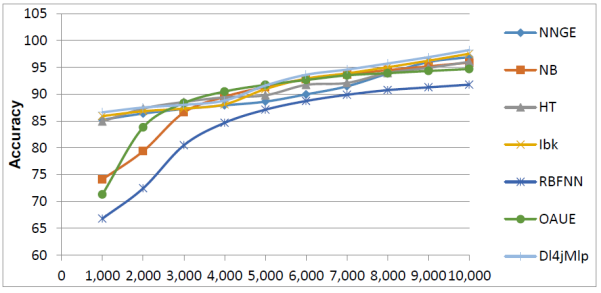

Figure 5: Prequential accuracy (%) for the KDD-CUP4 99 dataset

The accuracy of the seven classifiers has been recorded and illustrated as shown in the three previous figures for every 1000 instances between samples for the three

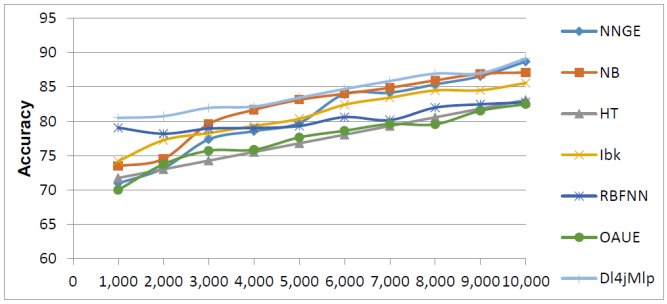

Figure 6: Prequential accuracy (%) for the UNSW-NB15 dataset

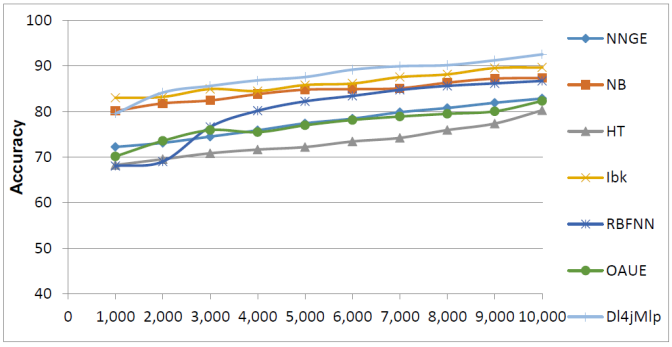

Figure 7: Prequential accuracy (%) for the Kyoto 2006+ dataset

datasets used, and indicated that AIDM with DL4JMLP classifier is a learnable model over time and with increase of the samples.

6 Comparative Study

In order to evaluate the performance of our proposed model, we have compared AIDM with some other state-of-the-art models. All evidence proved that AIDM which based on feature selection technique and the Dl4JMlp classifier is a promising model to compare with other state-of-art models. We have made a comparison between AIDM and the proposed model in [13] which runs on the NSL-KDD dataset due to that the two papers based on the deep learning technique.

AIDM helped to increase the detection rate of abnormal attacks by 97.9% accuracy for the KDD-CUP99 dataset and the other model has achieved 78.06% accuracy for the NSL-KDD dataset.

Another comparison has been done between the proposed model in [4] which based on feature extraction technique and runs on UNSW-NB15 and KDD-CUP99 datasets.

AIDM still achieved a high accuracy of 89.12% and 97.9% for UNSW-NB15 and KDD-CUP99 datasets respectively while the proposed model in [4] achieved an accuracy of 88.95% and 89.85% for UNSW-NB15 and KDD-

Table 6: Accuracy, Kappa statistic and running time (sec) for three datasets using six incremental classifiers

Mining data stream classifiers	KDD-CUP4 99 dataset			Kyoto 2006+ dataset			UNSW-NB15 dataset		
	Accuracy (%)	Kappa Statistics	Running time(sec)	Accuracy (%)	Kappa Statistics	Running time(sec)	Accuracy (%)	Kappa Statistics	Running time(sec)
NNGE	96.89	96.28	16.16	82.82	84.8	1.25	88.76	69.82	55.68
NB	95.87	74.99	1.62	87.31	8.16	0.20	87.09	47.92	2.56
HT	96.01	94.89	6.54	80.23	33.84	0.16	83.09	57.20	1.87
IBK	97.52	96.7	94.88	89.63	95.6	6.05	85.56	67.32	15.44
RBFNN	91.76	87.14	21.79	86.68	73.56	0.86	82.76	58.10	302.84
OAUE	94.68	91.82	54.04	82.28	38.52	17.50	82.5	30.21	0.50
Dl4jMlp	97.9	98.20	50.39	92.39	95.88	2.85	89.12	70.89	42.52

CUP99 datasets respectively.

Our AIDM enjoyed with higher performance compared with the other proposed models and this is demonstrated that our model is a learnable and predictive model. Depending on the comparisons with the other two models in [13] and [4], AIDM model based on feature selection technique achieved the best accuracy by the DL4JMLP classifier than other traditional IDS and state-of-the-art models for intrusion detection systems.

7 Conclusion

Most recent researches trend towards the creation of a proposed model for detecting the anomalous connections by using an efficient classification technique and supported by feature extraction technique from datasets. In this paper, an intelligent, effective and learnable model has been built based on machine learning techniques with the ability to reduce feature extracted from the three datasets by selecting the most relevant attributes using information gain method. The proposed model outperformed typical machine intrusion detection systems and incremental learning by using DL4JMLP deep learning classifier in addition feature extraction technique. The evaluation has been conducted on the processed three datasets, namely, KDD CUP 99, UNSW-NB15 and real Kyoto 2006+ datasets in the preprocessing engine, which has a skillful sub model which has ability to detect the number of features in the datasets and apply the proper techniques of feature selection and coding beside, a normalization process for the input datasets. AIDM was tested using 10,000 random sets of the three processed datasets using a classification stage which based on 6 incremental classifiers and one deep learning classifier, namely, Non-Nearest Generalized Exemplar (NNGE), Incremental Naïve Bayes (NB), Hoffeding Trees (HF), Instance-based k-nearest neighbor IBK (KNN), and Radial Basis Function Neural Network (RBFNN), Online Update Accuracy Ensemble (OUAE) and DL4JMLP deep learning classifier. AIDM has achieved a 97.9% high accuracy with DL4JMLP deep learning classifier compared with other incremental learning classifiers as mentioned in the experimental results and discussion section furthermore, it has fulfilled better results compared with the state-of-the-art models as shown in the comparative study section all of these positives indicate that AIDM is powerful, learnable and a good real-time model for classifying abnormal and normal traffics in the network flow.

References

[1] M. S. I. Ahmed, A. M. Riyad, R. L. R. Khan, M. J. K, and E. Shamsudeen, "Information based feature selection for intrusion detection systems," *International Journal of Scientific & Engineering Research*, vol. 8, no. 7, pp. 2362-2366, 2017.

[2] V. R. Balasaraswathi, M. Sugumaran, and Y. Hamid, "Feature selection techniques for intrusion detection using non-bio-inspired and bio-inspired optimization algorithms," *Journal of Communications and Information Networks*, vol. 2, no. 4, pp. 107-119, 2017.

[3] J. E. Ball, D. T. Anderson, and C. S. Chan, "A comprehensive survey of deep learning in remote sensing: theories, tools and challenges for the community," *Journal of Applied Remote Sensing*, vol. 11, no. 4, 2017.

[4] M. Belouch, "A two-stage classifier approach using reptree algorithm for network intrusion detection," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, pp. 389-394, 2017.

[5] B. Chakrabarty, O. Chanda, and M. Saiful, "Anomaly based intrusion detection system using genetic algorithm and K-centroid clustering," *International Journal of Computer Applications*, vol. 163, no. 11, pp. 13-17, 2017.

[6] M. M. U. Chowdhury, F. Hammond, G. Konowicz, C. Xin, H. Wu, and J. Li, "A few-shot deep learning approach for improved intrusion detection," in *8th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON'17)*, pp. 456-462, 2017.

[7] K. Datasets, D. G. Mogal, S. R. Ghungrad, and B. B. Bhusare, "NIDS using machine learning classifiers on UNSW-NB15 and KDDCUP99 datasets," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 6, no. 4, pp. 533-537, 2017.

[8] M. El-Boujnouni and M. Jedra, "New intrusion detection system based on support vector domain description with information gain metric," *Interna-*

tional Journal of Network Security, vol. 20, no. 1, pp. 25-34, 2018.

[9] L. Gnanaprasanambikai and N. Munnusamy, "Data preprocessing and classification for traffic anomaly intrusion detection using NSLKDD dataset," *International Journal of Pure and Applied Mathematics*, vol. 119, no. 10, pp. 847-858, 2018.

[10] J. Henrydoss, S. Cruz, E. M. Rudd, M. Gunther, and T. E. Boult, "Incremental open set intrusion recognition using extreme value machine," in *16th IEEE International Conference on Machine Learning and Applications (ICMLA'17)*, pp. 1089-1093, 2017.

[11] V. K. Jabali, "Taxonomy of feature selection in intrusion detection system," *IJCSNS International Journal of Computer Science and Network Security*, vol. 17, no. 6, pp. 88-102, 2017.

[12] M. A. Jabbar, R. Aluvalu, and S. S. Reddy, "RFAODE: A novel ensemble intrusion detection system," in *7th International Conference on Advances in Computing & Communications*, vol. 115, pp. 226-234, 2017.

[13] A. Jayaswal, "Detecting network intrusion through a deep learning approach," *International Journal of Computer Applications*, vol. 180, no. 14, pp. 15-19, 2018.

[14] J. Josemila Baby and J. Jeba, "Survey paper on various hybrid and anomaly based network intrusion detection system," *Research Journal of Applied Sciences*, vol. 12. pp. 304-310, 2017.

[15] H. A. Kholidy, "Attacks detection in SCADA systems using an improved non-nested generalized exemplars algorithm," in *12th International Conference on Computer Engineering and Systems (ICCES'17)*, pp. 607-612, 2017.

[16] A. A. Kolpyakwar, M. G. Ingle, and R. V. Deshmukh, "A survey on data mining approaches for network intrusion detection system," *International Journal of Computer Applications*, vol. 159, no. 1, pp. 20-23, 2017.

[17] D. A. Kumar, S. Venugopalan, "The effect of normalization on intusion detection classifiers (Naïve Bayes and J48)," *International Journal on Future Revolution in Computer Science & Communication Engineering*, vol. 3, pp. 60-64, 2017.

[18] A. Mansourkhaki, M. Berangi, and M. Haghiri, "Comparative application of radial basis function and multilayer perceptron neural networks to predict traffic noise pollution in tehran roads," *Journal of Ecological Engineering*, vol. 19, no. 1, pp. 113-121, 2018.

[19] A. Muallem, S. Shetty, J. W. Pan, J. Zhao, and B. Biswal, "Hoeffding tree algorithms for anomaly detection in streaming datasets: A survey," *Journal of Information Security*, vol. 8, no. 4, pp. 339-361, 2017.

[20] L. R. Nair, S. D. Shetty, and S. D. Shetty, "Streaming big data analysis for real-time sentiment based targeted advertising," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 1, pp. 402-407, 2017.

[21] A. R. Onik and T. Samad, "A network intrusion detection framework based on bayesian network using wrapper approach," *International Journal of Computer Applications*, vol. 166, no. 4, pp. 13-17, 2017.

[22] M. Panda and M. R. Patra, "A comparative study of data mining algorithms for network intrusion detection," *Annals of Emerging Technologies in Computing*, vol. 2, no. 1, pp. 49-57, 2018.

[23] P. Parrend, J. Navarro, F. Guigou, A. Deruyver, and P. Collet, "Foundations and applications of artificial Intelligence for zero-day and multi-step attack detection," *EURASIP Journal on Information Security*, 2018.

[24] E. Popoola and A. Adewumi, "Efficient feature selection technique for network intrusion detection system using discrete differential evolution and decision tree," *International Journal of Network Security*, vol. 19, no. 5, pp. 660-669, 2017.

[25] K. K. Ravulakollu, "A hybrid intrusion detection system: integrating hybrid feature selection approach with heterogeneous ensemble of intelligent classifiers," *International Journal of Network Security*, vol. 20, no. 1, pp. 41-55, 2018.

[26] H. Shapoorifard, "Intrusion detection using a novel hybrid method incorporating an improved KNN," *International Journal of Computer Applications*, vol. 173, no. 1, pp. 5-9, 2017.

[27] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Software Networking*, vol. 6, no. 1, pp. 177-200, 2017.

[28] S. Sharma, J. Thakkar, and J. Patel, "A survey on supply chain cyber security," in *National Conference on Latest Trends in Networking and Cyber Security*, pp. 77-79, 2017.

[29] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, pp. 1-10, 2017.

[30] V. T. Slavko Gajin, "Ensemble classifiers for supervised anomaly based network intrusion detection," in *13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP'17)*, pp. 13-19, 2017.

[31] P. Sudha, "Feature selection techniques for the classification of leaf diseases in turmeric," *International Journal of Computer Trends and Technology*, vol. 43, no. 3, pp. 138-142, 2017.

[32] J. Surana, J. Sharma, I. Saraf, N. Puri, and B. Navin, "A survey on intrusion detection system," *International Journal of Engineering Development and Research*, vol. 5, no. 2, pp. 960-965, 2017.

[33] H. Tanaka, "Effectiveness and weakness of quantified/automated anomaly based IDs," *International Journal of Network Security & Its Applications*, vol. 9, no. 6, pp. 1-11, 2017.

[34] C. J. Ugochukwu, E. O. Bennett, and P. Harcourt, "An intrusion detection system using machine learning algorithm," *International Journal of Computer*

Science and Mathematical Theory, vol. 4, no. 1, pp. 39-47, 2018.

[35] R. Vargas, A. Mosavi, R. Ruiz, "Deep learning: A review," *Advances in Intelligent Systems and Computing*, vol. 5, July, 2017.

[36] L. Wang and R. Jones, "Big data analytics for network intrusion detection: A survey," *International Journal of Networks and Communications*, vol. 7, no. 1, pp. 24-31, 2017.

[37] J. Zhong, Z. Liu, Y. Zeng, L. Cui, and Z. Ji, "A survey on incremental learning," in *5th International Conference on Computer, Automation and Power Electronics (CAPE'17)*, no. Cape, pp. 166-174, 2017.

Biography

Marwa R. Mohamed was born in 1982 in Egypt, she received her B.S.C degree in Electrical Engineering in 2005 from the Faculty of Engineering, Helwan University, Egypt. She had worked as a system administrator at Gulf English School (GES)-Cairo. She has successfully completed the Networking security and Management track granted through MCIT Scholarship at Raya Academy for a total of 1200 hours. She was ranked within the top 20% of trainers, accordingly, she has taken part from 04/02/2007 to 04/05/2007 and she has a certificate from Raya with this period.

Abdurrahman A. Nasr is a lecturer of software engineering, Computer, and System Engineering Department, Faculty of Engineering, Al-Azhar University at Cairo. He received his M.Sc. and Ph.D. degrees in electrical engineering from Al-Azhar University in 2012, and 2014 respectively. His fields of interest include artificial intelligence, stochastic process, machine learning, data mining, mathematics and operating systems.

I. F. Tarrad was born in 1959 in Egypt, He received his B.S.c degree in Electrical Engineering in 1984 from the Faculty of Engineering, Al-Azhar University, Egypt. He had worked as a demonstrator (teaching and research assistant) at the Al-Azhar University. He received his Master of Science degree in communication engineering 1989 and Ph.D. from Hungarian Academy of the Sciences Technical University of Budapest in 1996. In 1996 he was appointed as Lecturer at Department of Communication Engineering, Al-Azhar University, his research activities are within mobile, Computer Network, and digital communication.

Salah R. Abdulmageed received his M.S. and Ph.D. in Systems and Computers Engineering from Al-Azhar University in 2002 and 2005, respectively. Since 2017, he is a professor and head of Systems and Computers Engineering Department at Al-Azhar University. He performed his postdoctoral research in 2007 and 2008 in the Computer Science and Engineering Department, School of Engineering, the Southern Methodist University at Dallas, TX, USA. He was a software development manager of TEMPO (Tool for Extensive Management and Performance Optimization) project in Cairo University and Vodafone Egypt asan industrial partner in 2014 and 2015. His research interests include Mobile Computing, Cellular Networks, Sensor Networks, Cognitive Radio Networks, Vehicular Ad-hoc Networks, Big Data and Data Analysis, Internet Services and Applications.