A Novel Intrusion Detection System Based on Random Oversampling and Deep Neural Network

Nilamadhab Mishra

[1]Ph.D Scholar, Biju Patnaik University of Technology, Rourkela, Odisha, India,
nilamadhab76@gmail.com

Sarojananda Mishra

[2]Professor, Indira Gandhi Institute of Technology, Sarang , Dhenkanal, Odisha, India
sarose.mishra@gmail.com

Bhaskar Patnaik

[3]Professor, Nalla Malla Reddy Engineering College ,Hyderabad , Telengana, India
bhaskar7310@gmail.com

Abstract

This study proposes a new anomaly-based intrusion detection technique (IDT) that can successfully detect all types of attacks with comparatively higher accuracy. Since imbalanced datasets have the potential to compromise the performance of any classifier, the proposed study emphasizes on pre-processing of the imbalanced data using a random over-sampling algorithm. The sampled data are further subjected to a deep neural network, which is trained by different high-end optimizers, like; as Adam, Adadelta, Adagrad, Adamax, Nadam, RMSprop, and SGD to verify the efficiency of the model on testing with each optimizer. The IDT model has been evaluated using the famed KDD Cup 1999 dataset. It is observed that the proposed IDT outperforms many other state-of-the-art IDTS modelled on the same dataset in terms of accuracy to detect all attack types, namely dos, normal, probe, r2l, and u2r attack types.

***Keywords*: Intrusion Detection Technique; Imbalanced dataset; Deep Neural Network; Random oversampling.**

1. Introduction

The Internet has become the way of life for people worldwide irrespective of geographical regions, demography, and economic status of the person or country. All sorts of possible transactions of varied natures and types are now being carried out using the internet. Businesses and commerce have got digitized and the world has become a common marketplace. E-Commerce has completely taken over physical commercial activities. In this context, it is of paramount importance that these activities must have a secure platform, which means the internet should provide a secure environment. Despite the tremendous growth in internet technology, there lie many technological issues. Keeping the net secured by preventing its breaches by hackers remains a perennial problem. As the security system gets advanced, the hackers come out with newer systems of hacking. A weak server helps the hacker to intrude service provider's system [Abraham et al. (2007)]. Usually, hackers use attack types, such as DoS, Probe, U2R, and R2L to intrude into a service provider's system disrupting services or stealing vital information. In the above context, a reliable, robust Network Intrusion Detection System (NIDS) is so vital to be kept in place by the service provider in its system.

In a broader sense NIDS, which forms an essential component of the security infrastructure of a system (Fig.1), identifies hacking/malicious activities through the recognition of anomalies in incoming packets from the network traffic or recognizing bad patterns, or a combination of both approaches.

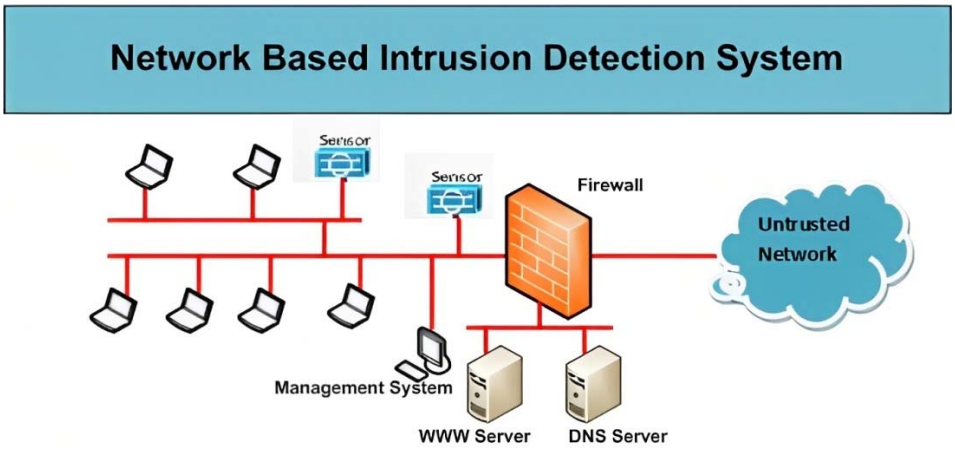

Fig-1 Network Based Intrusion Detection System

Accordingly, NIDSs are classified as anomaly-based, signature-based, and hybrid-type NIDSs (Fig.2). The types of attacks that a NIDS attempts to prevent are generally, denial of service attacks, and port scans on a computer network or the computer itself. Some attacks take shape silently getting originated and staying within a local network or being heisted inside the network with an out-of-the-network remote source. The NIDS is used efficiently in systems, like a firewall, to identify and prevent known sources of attack.

Based on the mode of operation NIDSs can also be classified as online and offline NIDS, often referred to as inline and tap mode, respectively. While the former is a rule-based real-time system, the latter deals with stored data and adopts certain decision-making logic to identify an attack.

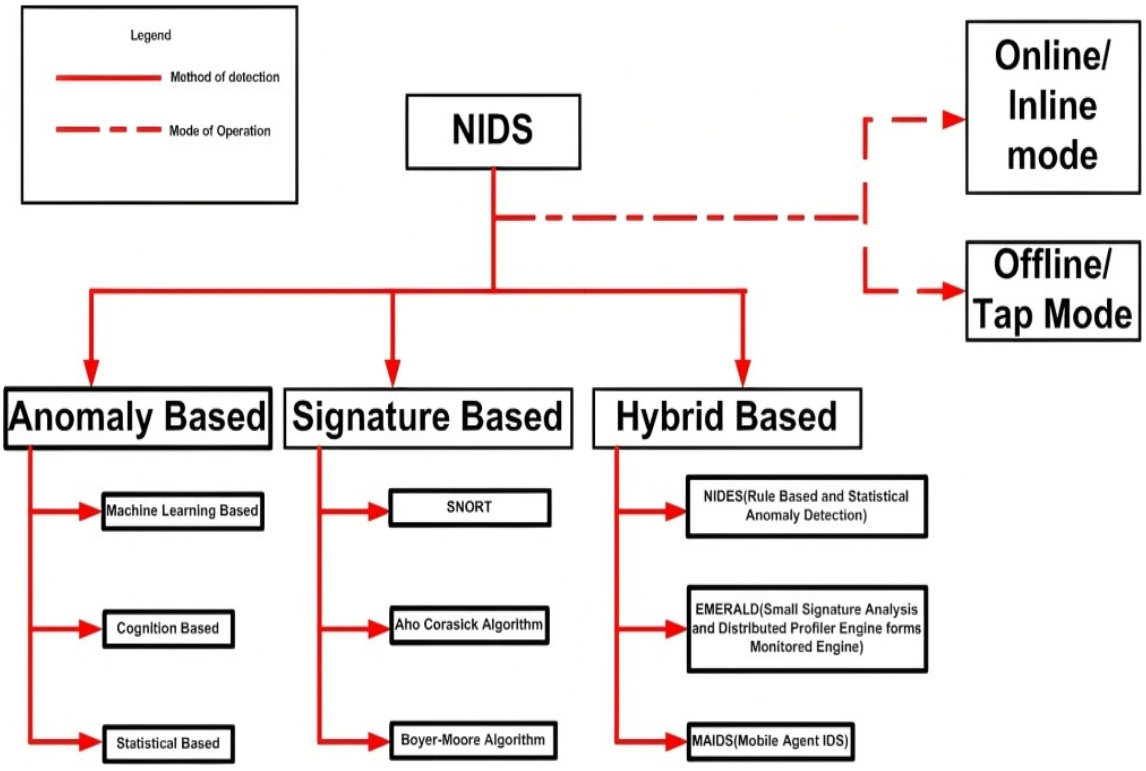

Fig-2. Classification of NIDS

NIDS(Anomaly-based) [Anomaly-based IDS looks for unknown attack types.]		**NIDS(Signature-based)** [Signature-based IDS is preconfigured and programmed attack patterns.]	
PROS	**CONS**	**PROS**	**CONS**
Models are designed to use machine learning to compare trustable behaviours with new behaviours thereby flagging out strange or unusual ones.	Possibility of accidental flagging of priorly unknown but admissible behaviours leading to compromised actions.	The signatures are defined on familiar intrusions. The users can easily identify the nature of intrusions that the IDS is designed for by scanning through the signature database.	Coping up with huge traffic is challenging as each incoming packet of data needs to be scanned with all signatures available in the database for comparison, a process which could be time-consuming increasing the detection time.
Based on the premise of standard and predictable network behaviour and type of traffic is simple enough to be recognized as good or bad.	Traffic behaviour, and not the quantum of traffic, is taken into scrutiny. Hence it may cause an error in flagging the appropriate traffic while running on a network with a non-standard configuration.	Gets operational soon after installation. Possibility of false detections for clearly defined attacks.	For foolproof intrusion detection, the IDS must have signatures articulated for every possible network attack. This means the signature database needs to be continually or periodically updated against newer attack types.
Most suitable while looking for possibilities of attack before their incidence.	Need to be deployed and distributed widely over the network. The machine learning algorithm employed needs guided training of the administrator.	This pragmatic approach is focused on specific attacks and is very accurate at lowering the rate of false positives.	The system is less accurate, leading to an increased rate of false positives.

Table-1: Difference between Anomaly-based and Signature-based NIDS

Recent literature is abounding with IDTs based on machine learning algorithms. There are many proposed IDT models which are based on classical machine learning algorithms such as KNN, SVM, etc., which incidentally exhibit low accuracy and are dependent on manual intervention in the design of the traffic features. In the era of big data, this manual exercise is humanly impossible and hence can be considered obsolete. In this context, the authors in [Su et al. (2020)], [Fu et al. (2022)] have proposed an intrusion detection model based on a combination of BLSTM (Bidirectional Long Short-term memory) and attention mechanism, wherein the key features of the data traffic get automatically extracted through screening of the network flow vectors obtained from BLSTM model. Stressing upon the importance of traffic classification as an all-important step before network anomaly detection, the authors in [Wang et al. (2017)] have proposed a taxonomy of traffic classification followed by the application of an IDT based on a convolutional neural network. In the proposed method the raw unprocessed data is considered as input to the adopted AI classifier not banking upon hand-designed features. In this kind of representation learning approach, as the authors themselves opine, the tuning of parameters of CNN was not addressed. Also, the generalization capability of the used ML model is not tested. Besides the proposed method is limited by the fact that its capability to identify unknown malware traffic is not validated and is left to be addressed as future scope of work. However, the IDT proposed by the authors in [Ding and Zhai (2018)] also employ CNN using the whole of the NSL-KDD dataset, which promises to be performing better than the traditional ML methods, such as Random Forest (RF) and SNM, besides certain deep learning methods, such as DBN and Long Short Term Memory (LSTM). The authors in [Zhao et al. (2017)] have proposed an IDT which addresses the problem of high dimensional data, which is so typical of network traffic, by first reducing the raw data into low dimension dataset using a deep belief network (DBN). Further, the learning performance of the proposed IDT model is enhanced by particle swarm optimization to optimize the number of nodes per hidden

layer. Finally, PNN is used to classify the low-dimensional data. The authors in [Tao et al. (2018)] have proposed an SVM-based IDT, wherein the processes of selection of useful features, tuning of parameters, and weights are optimized using characteristics of both the genetic algorithm and SVM. The authors in [Ren et al. (2019)] improve the detection by constructing a multi-level random forest model to detect network anomalous behaviour. The IDT proposed in [Tama et al. (2019)] employs PSO, ACO, and GA for feature size reduction working on the whole datasets of NSL-KDD and UNSW-NB15. It is followed by feature selection using reduced error pruning tree (REPT) classifier. Finally, the intrusion type classification is achieved by an ensemble of two classifiers i.e., rotation forest and bagging. A signature-based IDT has been proposed by authors in [Wisanwanichthan and Thammawichai (2021)] in an attempt to detect all types of attacks. As certain sparsely happening attacks, like Remote2Local (R2L) and User2Root (U2R), have stark differences in their patterns as compared to the commonly happening attack types, the authors opine that a single ML classifier is not helpful. Hence the proposed IDT adopts a two-layered hybrid approach, wherein Naive Bayes in layer 1 detect DoS and Probe, and SVM in layer 2 detect R2L and U2R serving the desired objective.

References	Year	Algorithm	Main Contribution	Field
[Shapoorifard and Shamsinejad (2017)]	2017	K-MEANS, KNN	Joint ventures of K-MEANS clustering and KNN classifier increase detection accuracy.	Machine Learning
[Wang et al. (2017)]	2017	CNN	Used of CNN improved the detection of traffic data.	Deep Learning
[Zhao et al. (2017)]	2017	DBN, PNN	DBN and PNN model to reduce the ambit of the data with DBN and classified using PNN.	Deep Learning
[Tao et al. (2018)]	2018	SVM and GA	Weights of SVM features, selection, and parameters optimized by genetic algorithm.	Machine Learning
[Ding and Zhai (2018)]	2018	DBN, LSTM	Proposed CNN-based IDS for better accuracy and type of intrusion.	Machine Learning and Deep Learning
[Ren et al. (2019)]	2019	Random forest	Abnormal network behaviour have detected by the multilevel random forest model	Machine Learning
[Tama et al. (2019)]	2019	PSO, ACA , GA	Proposed hybrid feature selection and two-stage meta classifier.	Hybrid features selection technique
[Su et al. (2020)]	2020	CNN and LSTM	CNN and LSTM models are proposed to detect each attack type.	Deep Learning
[Wisanwanichthan and Thammawichai (2021)]	2021	Naive Bayes classifier and SVM	A double-Layered Hybrid Approach (DLHA) have proposed.	Machine Learning
[Fu et al. (2022)]	2022	CNN and ADASYN	Proposed a model DLNID for better performance classification	Deep Learning

Table-2: Snapshot of the literature survey

2. Background:

2.1 *Random Over Sampling:*

To weed out the possible presence of an imbalance in a given dataset various sampling techniques are invariably used to expect better classification accuracy from the designed model. In this work, a hybrid sampling method based on oversampling and a neural network has been adopted for the above purpose.
A minority class of dataset is the one that is underrepresented and oversampling technique is used to duplicate these results for a more balanced amount of positive results in the training dataset.
In the above context, random oversampling encompasses improving the training data with multiple replicas of some of the minority classes. In place of replicating all samples in the minority class, some have been haphazardly chosen with replacement.

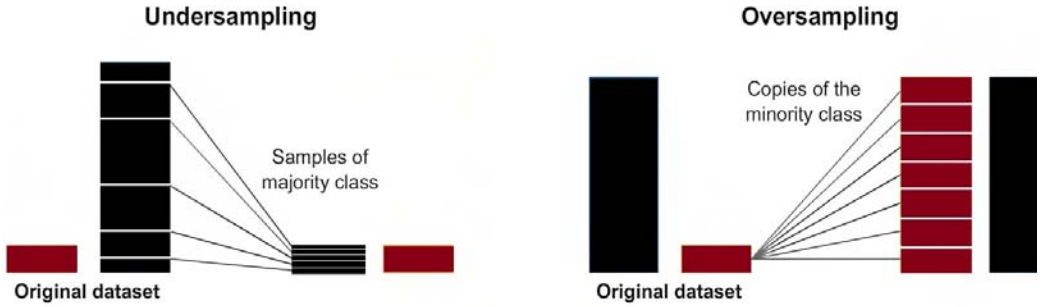

Fig-3. Under sampling vs. Oversampling [Nour AI-Rahman (2021)]

2.2 *Deep Neural Network (DNN)*

Neural networks, which serve as the all-important component in developing machine learning algorithms, essentially imitate the functioning of human brains. Neural networks comprise the systematic interconnection of layers of nodes, capable of processing the input data to provide the desired output through appropriate training procedures. The training process could be either supervised or unsupervised. However, the novelty of these neural networks lies in their capability to recognize hitherto unknown inputs and provide correct solutions just as the human brain does. The algorithms executing such objectives are known as artificial neural networks. A neural network essentially consists of an input layer (the raw information is fed at this layer), one or multiple hidden layer(s) (that determines the activity of each hidden unit through weighted inputs), and an output layer (that provides the outcome depending upon the activities of the weighted hidden units). Neural networks, however, are marred with certain deficiencies, such as (i) their dependence on hardware architecture for enhanced processing power, (ii) through providing a meaningful solution, it remains silent on their way of functioning, (iii) undefined rule for the design of ANN, (iv) heavily reliant on the ability of the user to translate the problem in hand to numerical values before being fed to the designed ANN, and (v) it's prone to error based on the sample means adopted for training the ANN. A Convolutional Neural Network (CNN) in this context can be considered a better option. It is a feed-forward neural network that is generally used to detect and classify objects in an image. CNN also has its share of drawbacks. It is significantly slower due to its inbuilt operational steps, such as a max pool. The training takes considerable time due to the multiple layers of CNN. It requires a sizeable amount of data for training purposes. Recurrent Neural Networks (RNN) are another class of neural networks that are useful in modelling sequence data. RNNs are comparatively slow, difficult to train, inefficient in processing very long sequential data, and susceptible to errors such as exploding and vanishing.
In the context of the above deliberation on the cons of different ANN versions, a deep neural network (DNN) can be viewed as a refreshingly better alternative to many neural network algorithms or structures. Its acceptability can be gauged from its wide range of applications in the field varied as pattern image recognition, detection of fraud, critical analysis on stock and news, self-manoeuvred vehicles, medical technology, etc. With adequate data, a DNN gets trained very well. It has multiple layers, and each layer performs to execute providing a specific solution, which makes it distinctively superior to other ANN models. For an instance, in the process of image recognition, if the first layer identifies edges and lines of a given image, the second layer may detect the eye/ear/nose and so on. Deep learning is found more useful as it does not follow a rigid data structure.
Additional advantages of DNN can be listed as:

- Can handle complex problems such as audio and image processing, due to its inherent ability to extract useful features from the inputted data, thereby eliminating the need for a feature extraction process.

- It is equipped with the feature of automated tasks so that it can make predictions fast using Keras and Tensorflow.
- As it does parallel processing, computing overheads are largely reduced.
- As models get trained by a huge quantity of data, the performance of these models is expectedly better.
- High-Quality Predictions comparable to the functioning of human brains are achieved through tireless training.

2.2.1 *Characteristics of Deep Learning*

Given below are the characteristics of Deep Learning:

2.2.1.1 Training of DNN

Training of any neural network is the most crucial and most daunting task as it is time taking in terms of process configuration and computational complexity involved therein. DNN is no exception in this regard. The training of DNN is accomplished to identify the parameters by updating its weights in response to the errors that the model makes on the training dataset. Updating is carried out continuously reducing the error till there is an end to the process and/or the model is found to be sound enough.

Deep Learning Optimizers

The theoretical developments of the advanced optimization tools used to tune the parameters of the DNN algorithm employed in the proposed IDT model are enumerated as follows [Ruder (2016)] , [Andrew (2017)] , [Krish (2012)] , [Jason (2022)] . Before delving into the details of these advanced optimizers, it is pertinent to know the weaknesses of the base optimizers, such as Gradient Descent (GD), Stochastic Gradient Descent (SGD), and mini-batch GD, which merits the choice of advanced optimizers in this study.

In GD the training data in its entirety is used to update the weight and bias, and while dealing with huge traffic of data the training process becomes both slow and computationally intensive. SGD on the other hand uses single records for updating parameters, requiring forward and backward propagation for every record. Also, the global minima are attained through a noisy path. The mini-batch GD in the above contexts can be considered as a cross between GD and SGD, wherein training is done with a batch of records, and not the entire records, to update the parameter, thereby both reduced computational complexity and faster convergence are obtained, albeit not so smooth path for global minima as that of GD. Fig. 4 enumerates how smooth or how oscillatory the loss is concerning the number of epochs [Gunand (2022)].

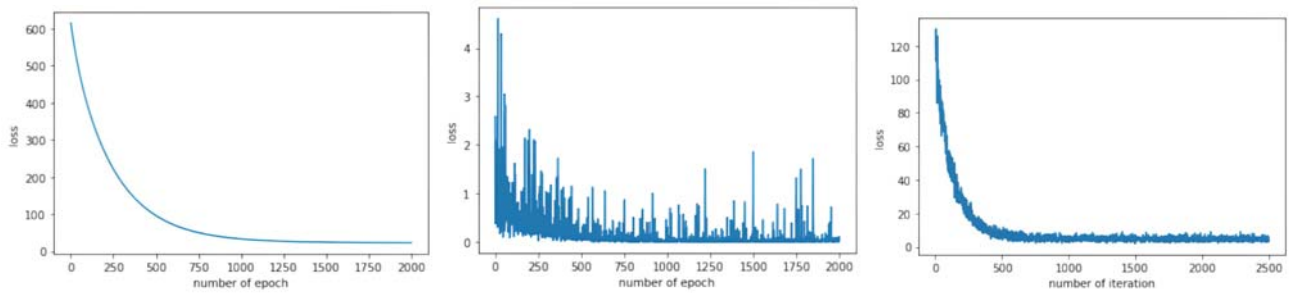

Fig. 4: Loss vs number of the epoch (a) Gradient Descent (b) Stochastic Gradient descent (c) Mini-batch Gradient Descent [Gunand (2022)].

It may be noted that "Exponentially Weighted Averages" (EWA) is popularly used to weed out the noise and smoothen that data that is sequentially noisy. The new sequentially generated data with decreasingly less noise are obtained using the equation (1).

$$V_t = \beta V_{t-1} + (1-\beta)S_t \quad (1)$$

For three sequences, equation (1) can be elaborated as:

Let s1, s2, and s3 are the noisy data generated at times t1, t2, and t3 respectively;

$$t_1 \Rightarrow V_1 = \beta V_0 + (1-\beta)s_1 \quad (2)$$

$$t_2 \Rightarrow V_2 = \beta V_1 + (1-\beta)s_2 \quad (3)$$

$$t_3 \Rightarrow V_3 = \beta V_2 + (1-\beta)s_3 \quad (4)$$

Where, V_1, V2, and V_3 is the new sequence obtained after de-noising of s1, s2, and s_3.

Now V3 on being expanded,

$$V_3 = \beta V_2 + (1-\beta)s_3 \leftrightarrow substitude\ V_2$$
$$V_3 = \beta(\beta V_1 + (1-\beta)s_2) + (1-\beta)s_3 \leftrightarrow substitude\ V_1$$
$$V_3 = \beta\big(\beta(\beta V_0 + (1-\beta)s_1)_{)} + (1-\beta)s_2\big) + (1-\beta)s_3 \leftrightarrow\ V_0 = 0$$
$$V_3 = \beta\big(\beta\big((1-\beta)s_1\big) + (1-\beta)s_2\big) + (1-\beta)s_3$$

β being the hyperparameter ranging between 0 and 1 .

At t=3 more weightage is given to s3 (the most recently generated data), followed by s2, and so on. This is how the sequence of noisy data is smoothened and for reasons obvious works better for long sequences.

I. SGD with momentum

Considered an improvisation over SGD, the algorithm employs EWA to compute the gradient and the resultant gradient is used to update the parameter. While the equation in SGD for weights updation is;

$$W_t = W_{t-1} - \eta \frac{\partial L}{\partial W_{t-1}} \quad (5)$$

$$b_t = b_{t-1} - \eta \frac{\partial L}{\partial b_{t-1}} \quad (6)$$

The relevant equation to update weights in SGD with momentum is given as;

$$w_t = w_{t-1} - \eta V_{dw_t} \quad (7)$$

$$\text{Where } V_{dw_t} = \beta V_{dw_{t-1}} + (1-\beta)\frac{\partial L}{\partial w_{t-1}} \quad (8)$$

$$b_t = b_{t-1} - \eta V_{db_t} \quad (9)$$

$$\text{where } V_{db_t} = \beta V_{db_{t-1}} + (1-\beta)\frac{\partial L}{\partial b_{t-1}} \quad (10)$$

The intended momentum has been added to the gradient function by making the concerned gradient dependent upon its preceding gradient and so on, resulting in acceleration of the SGD for faster convergence and reduced oscillation.

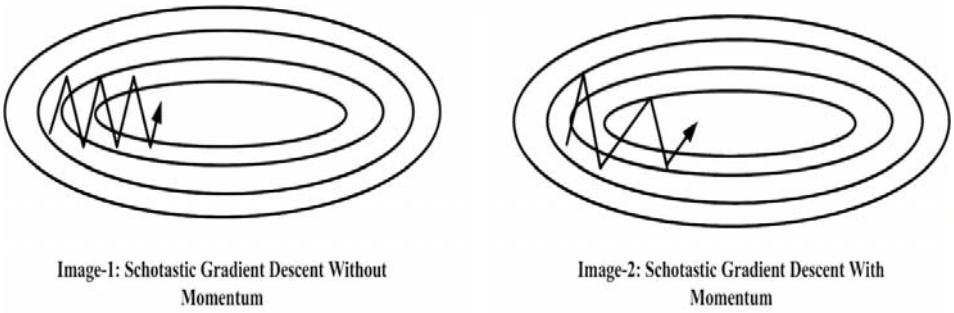

Image-1: Schotastic Gradient Descent Without Momentum

Image-2: Schotastic Gradient Descent With Momentum

Fig.5. Image-1: SGD without momentum, Image-2: SGD with momentum [Ruder (2016)]

The above picture shows how the convergence happens in SGD with momentum vs SGD without momentum.

II. Adagrad (Adaptive Gradient Algorithm)

The adagrad, unlike the afore-discussed optimizers, uses different learning rates for each parameter base on iteration and has no application of momentum. By using different learning rates, higher learning rates can be ascribed to sparsely available features as compared to that densely available ones. And this provides adagrad with the ability to encompass the sparse features with a low rate of occurrence.
The defining equation for adagrad is given as:

$$\text{Adagrad} \Rightarrow w_t = w_{t-1} - \eta_t \frac{\partial L}{\partial w_{t-1}} \quad (11)$$

$$\text{Where } \eta_t = \frac{\eta}{\sqrt{\alpha_t + \varepsilon}} \quad (12)$$

$$\epsilon \text{ is a small} + \text{ve number to avoid divisibility by 0}$$

$$\alpha_t = \sum_{i=1}^{t} \left(\frac{\partial L}{\partial w_{t-1}}\right)^2 \text{ summation of gradient square} \quad (13)$$

Equations 12 & 13 indicate that as "t" increases "α" also increases leading to a decrease in the learning rate "η". This happens to be one of the major drawbacks of adagrad as it makes the model stop learning once the learning rate closes upon zero.

III. Adadelta

Adadelta attempts to overcome the drawback of adagrad by using Exponentially Weighted Averages over Gradient. The "α" with exponentially weighted averages of squared gradients "S*dw*" is enumerated in equation (14) below.

$$\eta_t = \frac{\eta}{\sqrt{S_{dw_t} + \epsilon}} \; where \; S_{dw_t} = \beta S_{dw_{t-1}} + (1-\beta)\left(\frac{\partial L}{\partial w_{t-1}}\right)^2 \quad (14)$$

$$\in \text{ is a small} + \text{ve number to avoid divisibility by 0}$$

IV. Adam optimizer

Adam takes in the best of both the optimizers "SGD with momentum" and "Ada delta", proving to be the most popular optimizer in many cases. The relevant weight and bias Scholastic Gradient descent updations formula for adam is represented in equations 15 & 16.

$$W_t = W_{t-1} - \frac{\eta}{\sqrt{S_{dw_t} - \epsilon}} * V_{dw_t} \quad (15)$$

$$b_t = b_{t-1} - \frac{\eta}{\sqrt{S_{db_t} - \epsilon}} * V_{db_t} \quad (16)$$

2.2.1.2 Huge Amount of Resources

DNN requires additional resources such as advanced Graphical Processing Units (GPUs) to process large quantities of jobs. As DNN handles large volume work, it generally has to pre-process proportionately large amounts of structured or unstructured data like big data thus requiring longer processing time.

2.2.1.3 Large Amount of Layers in Model

A DNN structure has multiple or sometimes many layers, each layer executing a small part of the problem. The output of each such layer serves as input to the next layer, thereby summing the small solution of the previous layer with that of the present. Ultimately the final all these small solutions from the hidden layers are summed up in the softmax layer to present a broader classification of the output of the DNN structure.

2.2.1.4 Optimizing Hyper-parameters

Hyper-parameters like no of epochs, Batch size, No of layers, and Learning rate, need to be tuned for enhanced and desired model accuracy. Tuning of hyper-parameters also takes care of overfitting and under-fitting issues.

2.2.1.5 Cost Function

The cost function presents the performance of a model in terms of prediction accuracy. In each iteration of DNN, the cost function is minimized as compared to its previous iterations. Depending on the objective the DNN models

can be designed using a gamut of algorithms, such as Mean absolute error, Mean Squared Error, Hinge loss, Cross entropy etc. to name a few.

3. Simulation and Results:

An imbalanced data problem is a type of problem in which at least one of the classes is very rare and has less proportion. The researchers have provided different solutions to such problems. Data level, algorithm-level, and a mixture of these two known as hybrid approaches are a few of them. In a data-level approach resampling, such as oversampling, under-sampling, and hybrid sampling is performed. While in algorithm-level approaches, different cost-sensitive methods, increasing the layers of neural networks, creating fake data points, such as images, speeches, etc. are performed. In hybrid methods, the data-level and the algorithm-level approaches are mixed to get better results. In this experiment, an oversampling and a hybrid sampling method with a neural network have been implemented.

The Random oversampling method has experimented with the data level approach for oversampling. While the amalgamation of random oversampling and random undersampling has been implemented for the hybrid sampling. Random oversampling is a method in which the minority classes get oversampled based on different sampling strategies. In this case, the sampling strategy used was a minority which means the least minor class was oversampled based on the next least minority one and so on. Random under-sampling is a method in which the majority of classes get under-sampled as required.

Here in this experiment, we have to use KDD-Cup 1999 dataset, the most popular dataset, particularly recommended for being used for building network intrusion detectors. A standard set of auditable data, including a wide range of simulated intrusions into a military network environment, is contained in this database. Attacks fall into four main categories:

- DOS: denial-of-service.
- R2L: unauthorized access from a remote machine.
- U2R: unauthorized access to local super user (root) privileges.
- probing: surveillance and another probing.

The dataset got divided into three parts, viz. train, test and validation in a ratio of 62:20:16. The training dataset was used to train the model; the test dataset is the unseen data on which the model is evaluated; the validation dataset is the data that doesn't participate in any of those but validates if there are any discrepancies in training algorithm while the training process is holding. The training data was then oversampled and under-sampled as mentioned in the previous paragraph. After the resampling, it was fed to a Neural Network which had 4 layers. An input layer that took the features and labels as inputs, two hidden layers that helped in training, and an output layer that told the probability of a class being itself. The implementation was coded by python, using Anaconda 3.0. The workflow diagram of the work done is given in Fig-6.

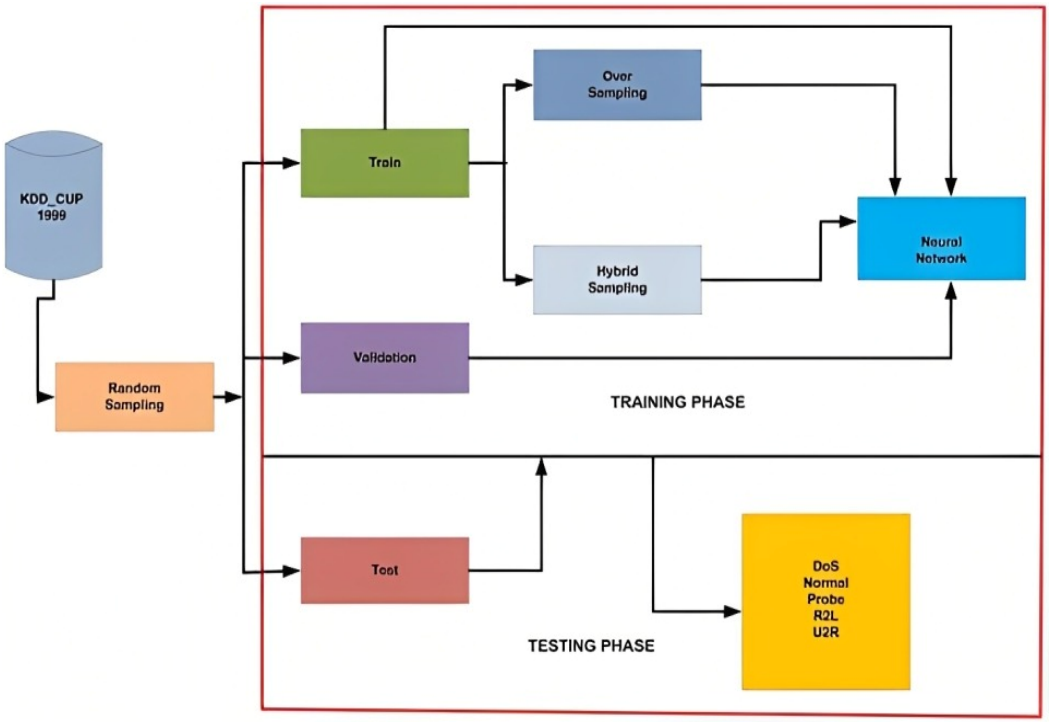

Fig. 6. Workflow diagram

3.1 *Hyper-parameter tuning*

The neural network got some hyper-parameter tuning to enhance the results. A hyper-parameter tuning is a process in which a set of optimal hyper-parameters undergoes trial and the best performing one is chosen. In this case, we used seven different optimizers as a part of the hyper-parameter tuning which are Adam, Adadelta, Adagrad, Adamax, Nadam, RMSprop, and stochastic gradient descent (SGD).

3.2 *Accuracy performance Measure indices*

To measure the accuracy, different indices we have to use are given below:

$$Overall\ Accuracy = \frac{\sum_{i=1}^{n=5} \mathrm{TP}_i}{N}$$

$$Accuracy_i = \frac{TP_i}{N}$$

$$precision = \frac{TP}{TP + FP}$$

$$\mathrm{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Where, **TP** – Truly predicted (+)ve values; **TN** – Truly predicted (-)ve values; **FP** – Falsely predicted (+)ve values; **FN** – Falsely predicted (-)ve values

F1 score, which is generally used along with accuracy to present the performance of a given model tested with the imbalanced dataset, is given as:

$$F1 = 2 * \frac{Precision_i * Recall_i}{Precision_i + Recall_i}$$

In this proposed study the model exhibits an F1 score of value more than 87%. The confusion matrix of the best-performing model can be seen below. The model is <tensorflow.python.Keras.optimizer_v2.adamax.Adamax object at 0x0000018585932A30>.

77895	3	1	0	408
1	19353	3	4	110
0	24	784	0	2
0	14	0	166	28
0	3	0	0	6

Table-3 Confusion Matrix

3.3 *Classification Report:*

	precision	**recall**	**f1-score**	**Support**
DoS	1.00	0.99	1.00	78307
normal	1.00	0.99	1.00	19471
R2L	0.99	0.97	0.98	810
U2R	0.98	0.80	0.88	208
probe	0.01	0.67	0.02	9
accuracy			0.99	98805
macro avg	0.80	0.88	0.77	98805
weighted avg	1.00	0.99	1.00	98805

Table-4: Classification report

4. Datasets

4.1 *Data Analysis*

As stated earlier KDD99 dataset has been used for experimentation in this proposed work. It consists of a training set and a test set. In this dataset 494021 rows × 43 columns are available. Due to highly correlated values of the columns, we dropped 7 columns from the dataset. Now it contains 494021 rows × 35 columns. After feature selection, we group the attack type which contains 170 attack types. The dataset has a total of 494021 rows × 35 columns of dimensional features, one of which is a classification label, and the rest are feature labels. For multi-classification, the classification labels are divided into five categories, i.e., Dos, normal, R2L, U2R, and probe.

4.2 *Data Augmentation*

A much larger number of U2R and R2L samples comprises the test set as against a smaller number of the same samples in the training set, thus making it hard for the trained model to identify these two types of samples. And therefore, the proposed model employs the Random over Sampling algorithm, as enumerated in earlier subsections, to bring about parity in the otherwise imbalanced dataset, by duplication of the U2R and R2L samples present in the original training set. This process helps in boosting the generalization ability of the proposed model to a greater extent.

4.3 *Normalization*

Large discrepancies between the dataset's various dimensional features might result in issues like sluggish model training and minimal increase in accuracy, and therefore, to tackle this issue, Random Over Sampling and Hybrid Sampling was adopted.

5. Result Analysis

This section dwells upon the details of the experimental settings and the performance metrics picked up to assess the veracity of the proposed ITD model.

5.1 *Experimental Settings*

Specification of the Hardware used: Intel(R) Core(TM) i3-3220 CPU @ 3.30GHz
Operating System: Windows 10
Software: Python 3.7, and the sklearn library for model simulation

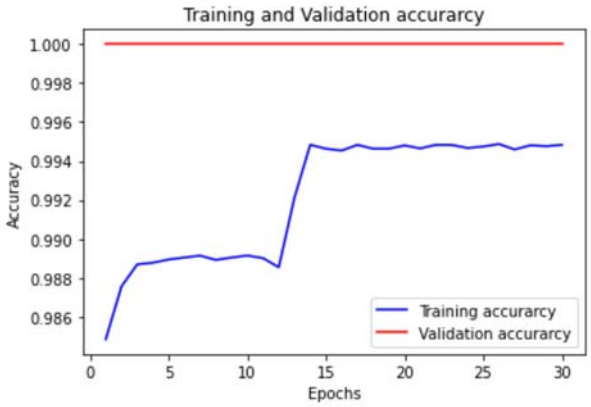

Fig-7: Training and validation accuracy

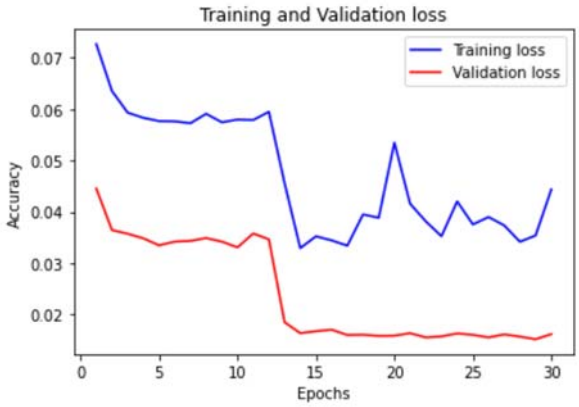

Fig-8: Training and validation loss

5.2 *Analysis of the result*

The experiment studied the performance of the proposed network on normal, Dos, Normal, R2L, U2R, and probe for binary and multi-classification experiments, respectively. When the network parameters were chosen, a high accuracy and F1 score could be achieved on the test set. The classification report shows the experimental results using the confusion matrix as in Table 3. The commendable classification performance of the proposed IDT model is quite evident as most samples were classified correctly, the values in diagonal being indicative of it. It can also be observed that had there been no data augmentation, both U2R and R2L would have been misclassified more, as these samples were more in number in the test set than in the training set.

5.3 *Model Comparison*

The supremacy of the proposed IDT over various similar state-of-the-art IDTs is established through the comparison made in terms of several performance matrices, and the same has been presented in Table 4. The proposed IDT exhibits Accuracy and F1 score as high as 99.00% and 89.65% respectively on the considered test dataset. In comparison to the commonly used ML algorithms, namely GAR-Forest or NB Tree, the proposed IDT has better accuracy and requires no manual feature extraction.

Different IDSs	Type of attack	ACC (%)	Pre (%)	Rec (%)	F1 (%)
Proposed IDS	dos	99	100	99	100
	normal	99	100	99	100
	probe	99	99	97	98
	r2l	99	98	80	88
	u2r	99	1	67	2
DLNID [Fu et al. (2022)]	u2r	90.73	86.38	93.17	89.65
DLHA [Wisanwanichthan and Thammawichai (2021)]	r2l, u2r	87.55	88.16	90.14	89.19
BAT-MC [Su et al. (2020)]	dos,normal,probe,r2l,u2r	84.25			
Autoencoder [Ieracitano et al. (2020)]	Normal, DoS, R2L and Probe	84.24	87	80.37	81.98
CNN [Gao et al. (2019)]	r2l, u2r	80.13			
Adaptive Ensemble [Tama et al. (2019)]	Probe, R2L and U2R	85.2	86.5	86.5	85.2
GAR-Forest [Kanakarajan and Muniasamy (2016)]	dos,normal,probe,r2l,u2r	85.06	87.5	85.1	85.1
CNN+BiLSTM [Jiang et al. (2020)]	dos,normal,probe,r2l,u2r	83.58	85.82	84.49	85.14
NB Tree [Tavallaee et al. (2009)]	dos,normal,probe,r2l,u2r	82.02			
SVM-IDS [Parvez and Farid (2014)]	dos,normal,probe,r2l,u2r	82.37			

Table-5: Performance comparison in respect of similar recently proposed IDSs

6. Conclusion

In this research work, an anomaly-based intrusion detection technique (IDT) has been proposed, which is capable of detecting all types of commonly encountered network attacks with greater accuracy. The model proposed to leverage the inherent capabilities of a deep neural network as a classifier, which is trained by different high-end optimizers, like; Adam, Adadelta, Adagrad, Adamax, Nadam, RMSprop and SGD to establish the efficiency of the model. The proposed model also incorporates the means to tackle the fallouts of imbalanced datasets through the adoption of random oversampling for pre-processing of the dataset. The IDT model has used the famed KDD Cup 1999 dataset. The proposed IDT outperformed many other state-of-the-art IDTS modelled on the same dataset in terms of accuracy to detect all attack types, namely dos, normal, probe, r2l, and u2r attack types. The accuracy and F1 score are observed to be as high as 99.00% and more than 87.00% respectively for DoS, Normal, Probe, R2L, and U2R. This study opens up the scope for the development of an IDT for real case implementation.

Conflict of Interest Statement:

The author(s) declare(s) that there is no conflict of interest.

Reference:

[1] Abraham, A., Grosan, C., & Martin-Vide, C. (2007). Evolutionary design of intrusion detection programs. Int. J. Netw. Secur., 4(3), 328-339.
[2] Shapoorifard, H., & Shamsinejad, P. (2017). Intrusion detection using a novel hybrid method incorporating an improved KNN. Int. J. Comput. Appl, 173(1), 5-9.
[3] Wang, W., Zhu, M., Zeng, X., Ye, X., & Sheng, Y. (2017, January). Malware traffic classification using convolutional neural network for representation learning. In 2017 International conference on information networking (ICOIN) (pp. 712-717). IEEE.

[4] Zhao, G., Zhang, C., & Zheng, L. (2017, July). Intrusion detection using deep belief network and probabilistic neural network. In 2017 IEEE international conference on computational science and engineering (CSE) and IEEE international conference on embedded and ubiquitous computing (EUC) (Vol. 1, pp. 639-642). IEEE.
[5] Tao, P., Sun, Z., & Sun, Z. (2018). An improved intrusion detection algorithm based on GA and SVM. Ieee Access, 6, 13624-13631.
[6] Ding, Y., & Zhai, Y. (2018, December). Intrusion detection system for NSL-KDD dataset using convolutional neural networks. In Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence (pp. 81-85).
[7] Ren, J. D., Liu, X. Q., Wang, Q., He, H., & Zhao, X. (2019). An multi-level intrusion detection method based on KNN outlier detection and random forests. Journal of Computer Research and Development, 56(3), 566-575.
[8] Tama, B. A., Comuzzi, M., & Rhee, K. H. (2019). TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. IEEE access, 7, 94497-94507.
[9] Su, T., Sun, H., Zhu, J., Wang, S., & Li, Y. (2020). BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset. IEEE Access, 8, 29575-29585.
[10] Wisanwanichthan, T., & Thammawichai, M. (2021). A double-layered hybrid approach for network intrusion detection system using combined naive bayes and SVM. IEEE Access, 9, 138432-138450.
[11] Fu, Y., Du, Y., Cao, Z., Li, Q., & Xiang, W. (2022). A Deep Learning Model for Network Intrusion Detection with Imbalanced Data. Electronics, 11(6), 898.
[12] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.
[13] Andrew, Ng. (2017). DeepLearning.AI https://www.youtube.com/c/Deeplearningai/featured. 2017.
[14] Krish ,Naik. (2012): Youtube channel. https://www.youtube.com/channel/UCjWY5hREA6FFYrthD0rZNIw .
[15] Jason , Brownlee.(2022) Welcome to Machine Learning Mastery. https://machinelearningmastery.com/.
[16] Gunand ,Mayanglmbm.(2022) . multi_linear-Gradient-descent https://github.com/GUNAND12/multi_linear-Gradient-descent.
[17] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.
[18] Ieracitano, C., Adeel, A., Morabito, F. C., & Hussain, A. (2020). A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. Neurocomputing, 387, 51-62.
[19] Gao, X., Shan, C., Hu, C., Niu, Z., & Liu, Z. (2019). An adaptive ensemble machine learning model for intrusion detection. IEEE Access, 7, 82512-82521.
[20] Kanakarajan, N. K., & Muniasamy, K. (2016). Improving the accuracy of intrusion detection using gar-forest with feature selection. In Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015 (pp. 539-547). Springer, New Delhi.
[21] Jiang, K., Wang, W., Wang, A., & Wu, H. (2020). Network intrusion detection combined hybrid sampling with deep hierarchical network. IEEE Access, 8, 32464-32476.
[22] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009, July). A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE symposium on computational intelligence for security and defense applications (pp. 1-6). Ieee.
[23] Pervez, M. S., & Farid, D. M. (2014, December). Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. In The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014) (pp. 1-6). IEEE.
[24] Nour AI-Rahman AI-Serw (2021, February). Undersampling and oversampling: An old and a new approach. Web contents.

Authors Profile

Mr. Nilamadhab Mishra working as an Assistant Professor at G H Raisoni College of Engineering and Management, Pune, Maharastra. He has obtained his M. Tech from BPUT University India and pursuing Ph.D. in BPUT University Rourkela, Odisha India. He has more than 15 years of teaching experiences and published 6 international Journals, 5 conferences. His research interests are Network Security & Cryptography, Artificial Intelligence , Machine Learning, and Deep Learning

Dr.Sarojananda Mishra working as a Professor in the Dept. of Computer Science , Engineering and Application, Indira Gandhi Institute of Technology (IGIT) , Sarang , Dhenkanal, Odisha , India. He obtained his M.Tech from IIT Delhi, India and Ph.D. from Utkal University , Odisha India. He has more than 30 years of teaching and research experience. He has published in 60 international Journals, 35 conferences and many more publications in SCI and Scopus Index journals. His research interests are Fractals and Graphics, System Dynamics , MIS, Operation Research , Networking , Computer Programming. Many students obtained Ph.D degree and are continuing their Ph.D and M.Tech research work under his guidance.

Dr. Bhaskar Patnaik serving as Professor at NMREC, Hyderabad ,Telengana, India, has obtained his B.Tech (Electrical Engineering) from CET, OUAT, Bhubaneswar, India, followed by M.Tech (Computer Science and Engineering), and M.Tech (Power Electronics and Drives), all from BPUT, Rourkela, Odisha , India in the year 2006 and 2014 respectively. He has obtained his Ph.D. from BPUT , Rourkela , Odisha, India in the year 2022. He has ample numbers of publications in journals of high impact factor as well as book chapters to his credit. His research interest are Artificial Intelligence, Machine Learning, and their applications in the fields of smart microgrids.