International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Stacked CNN-BiLSTM Model with Majority Technique for Detecting the Intrusions in Network

[1]Swati Mirlekar, [2]Dr. Komal Prasad Kanojia, [3]Dr. Bharti Chourasia

Submitted: 17/09/2023 **Revised**: 30/10/2023 **Accepted**: 14/11/2023

Abstract:The Internet of today is composed of almost 500,000 distinct networks. It is a challenging process to identify the attacks in every network connection according to the sorts of attacks they use since various attacks may have different connections, & the no. of attacks may range anywhere from a few to hundreds of network connections. Using a DNN (Deep Neural Network) technique to identify unknown attack packages is the primary objective of this research study. This will be accomplished by using an advanced Intrusion Detection System (IDS) that has excellent network performance. To conduct the assessment of metrics, UNSW-NB15 & NSL-KDD datasets are employed. This model makes use of LSTM & CNN to provide more accurate forecasts by concentrating more intently on the features of a successful earthquake. Initially, to reduce the quantity of noise in the majority group, we employ the One-Side Selection (OSS) technique. Then, to broaden the diversity of our samples, we employ the Synthetic Minority Oversampling Technique (SMOTE). This method of creating a balanced dataset dramatically reduces the amount of time required for training the model while allowing it to fully understand the characteristics of minority samples. Next, we use stacked CNN-biLSTM to extract spatial and temporal features, and then we use this information to build a deep stacked network model, which we stacked on top of one another. This proposed model can achieve remarkable accuracy in both datasets leaving a gap that is discussed at the end of the paper.

Keywords*: intrusion detection; stacked CNN-biLSTM; SMOTE; NSL-KDD, UNSW-NB15*

1. Introduction

The rate of cybercrime surged with the exponential evolution of technology & broad use of internet networks around the globe. As per ISTR (Internet Security Threat Report), in 2015 there were around 430 million newly discovered forms of malware, 362 of which were crypto-ransomware [1]. According to estimates, the total value of cybercrime was 1.5 trillion dollars in 2018. In 2019, A fact that can be said with complete assurance is that no company, no matter how big or little, is safe from cyberattacks. Cyberattacks are more sophisticated, stealthy, and targeted than they have ever been before [2]. Consequently, there is a need for the ongoing development of security measures.

Due to the ever-increasing frequency of sophisticated cyberattacks, network security research is a rapidly expanding field of study in computer networking. The intrusion detection systems, often known as IDSs, are intended to prevent intrusions & secure the systems, data, and computer systems from unauthorized access. IDSs can distinguish between intrinsic & extrinsic intrusions in computer networks of the organization and also may set off an alert if a security breach occurs inside a network belonging to that organization [3]. An important aspect of the concept of intrusion is that it results in malignant, externally induced functional breaches. The basic objective of IDSs is to identify a wide range of intrusions, including attacks that have not been discovered in the past; to find and adapt to unknown attacks, and to detect and recognize intrusions promptly [4].

Scholars have discovered the deployment of Deep Learning (DL) and Machine Learning (ML) methodologies to create an IDS capable of meeting the requirements for such a system. The techniques of deep learning (DL) and machine learning (ML) both seek to extract usable information from huge datasets [5]. Over the past ten years, network security has become increasingly popular, largely due to the development of extremely potent units for graphic processing (GPUs) [6]. Two possible uses for the potent artificial intelligence and deep learning approaches include learning advantageous qualities from network traffic and predicting normal and abnormal behaviours based on learned patterns. To derive relevant information from network data, the ML-based IDS heavily relies on the application of features [7]. However, due to the data's inherent structure, DL-based Intelligent Data Systems can autonomously learn complex features from unstructured information [8].

IDS can be implemented in two main ways: using signatures and anomaly detection [9]. Signature-based identification differs from anomaly-based identification in that it determines whether an activity is fraudulent or not

[1]Assistant Professor, Department of Electronics & Communication Engineering, St. Vincent Pallotti College of Engineering & Technology,Nagpur, Maharashtra
swati.mirlekar@gmail.com
[2]Associate Professor, Department of Electronics & Communication Engineering, RKDF Institute of Science and Technology Bhopal, M.P
komal44@gmail.com
[3]HOD, Department of Electronics & Communication Engineering, RKDF Institute of Science and Technology
Bhopal, M.P
chourasia3012@gmail.com

using an established collection of rules or indications from a system target database, whereas anomaly-based identification comprehends attacks based on uncommon trends in user behaviour [10]. An attack may be identified if there are users who engage in unusual or suspicious behavior.

The use of behavior in anomaly-based detection IDS provides the highest performance to identify attack activities since it may be used with several machine learning & data mining technologies [11]. These approaches intelligently detect & offer a new viewpoint on the many forms of attacks that are now being carried out throughout the world's computer networks. On the other hand, the use of the ML approach in IDS is still plagued by some issues. When using ML techniques, the most difficult problem to solve is figuring out how to construct a suitable model that accurately represents the dataset [12].

The method of data training affects the quality of the ML model that is ultimately produced. Through data preparation steps like reduction of data and selecting features, it is possible to get high-quality training data. The process of choosing which features to employ is known as features selection, and it depends on how significant each generated feature's data label is [13]. Eliminating data or occurrences that don't match with the remainder of the data is known as data cleansing. These are frequently referred to as "outlier data" [14].

2. Review of the Literature

A brief summary of the research on hybrid NIDS is presented in this section, or (Network Intrusion Detection System) so far. In addition, this part of the article explains the benefits of using a HIDS (Hybrid intrusion Detection System) as opposed to a standard IDS. In addition, many methods of ML are familiarized, and the discussion then turns to the benefits of picking certain ML techniques.

2.1 State-of-the-Art NIDSs

An adequate model can be constructed using a hybrid ML approach proposed in this research. This approach blends the FS technique (representing supervised learning) alongside the DR method (representing unsupervised learning). Initially, it uses RFE's (Recursive Feature Elimination's) component importance decision tree-based methodology to zero in on the most pertinent and crucial features, and then it employs LOF's (Local Outlier Factor's) technique to zero in on the most anomalous or outlier data. According to experimental findings, the suggested strategy not only performs better than the bulk of other studies on the data set from the NSL-KDD but also achieves the maximum accuracy (99.89 percent) for determining R2L and sustains this accuracy for various types of assaults. Because of this, its performance is more consistent than that of the others. More difficulties are encountered while working with the UNSW-NB15 dataset, which has binary classes [15].

In this study, they present a new framework for intrusion detection that is depending upon techniques of feature selection & ensemble learning. For the initial stage of dimensionality reduction, the CFS-BA heuristic approach is advised because it chooses the ideal subset based on the association between characteristics. Formerly, they provide an ensemble method that is a combination of the C4.5 algorithm, the Random Forest (RF), as well as the Forest by Penalizing Attributes (Forest PA) algorithm. The algorithms include Random Forest (RF) and Forest by Penalising Attributes (Forest PA). Finally, to identify strikes, the odds distributions from multiple base learners are combined using a voting system. The study's results, which were gathered using the datasets NSL-KDD, CIC-IDS2017, and AWID, show that the suggested CFS-BA-Ensemble method outperforms similar and advanced techniques in a number of ways [16].

In the present research, they suggest ML-IDS (Machine Learning Based Intrusion Identification System), which can find attacks on IoT networks. The primary objective of this study is on using IDS algorithms that are guided by using artificial intelligence with the IoT. In the very first stage of this study's method, the characteristics of the UNSW-NB15 dataset have been adjusted utilising the minimal level-maximum (min-max) method, idea of normalisation to limit how much info disappeared from the experimental data. This dataset is comprised of nine separate attack types constructed from a mix of contemporary attacks & ordinary network traffic. Using PCA (Principal Component Analysis), Dimensionality Reduction was accomplished in the succeeding stage. In the end, they employed six different proposed ML models to conduct the investigation. The results of their experiments were looked at in terms of validating datasets, precision, memory, accuracy, AUC, kappa, F1-score, and Mathew Correlation Coefficient (MCC). Findings were additionally compared to studies that had already been done and found to be comparable, with an accuracy of 99.9% and an MCC of 99.97%.

The NSL-KDD the data set which consists of 41 features but is reduced to 10 in this study, is employed with the ANOVA-PCA approach. It is verified & assessed using 3 different kinds of supervised classifiers, including K-NN (K-Nearest Neighbour), Random Forest (RF), as well as Decision Tree (DT). The results are produced by employing a variety of performance indicators, and then they are contrasted with outcomes of other feature selection methods, like NCA (Neighbour Component Analysis) and ReliefF. The findings demonstrated that the proposed approach was easy to understand, much quicker in terms of computation in comparison to previous approaches, and successful in achieving a classification accuracy of 98.9 percent [17].

In this study [18], a framework for implementing network intrusion detection that integrates data mining classification algorithms & association rules is presented. The KDD99 intrusion dataset has been used as the basis for some experiments, which have then been analyzed to determine the effectiveness of different ML classifiers. Some different data mining methods, including NB, DT, SVM, decision tables, K-NN algorithms, & ANN are the subject of their research. Using a KDD99 dataset anomaly detection, this research is also concerned with the process of associating attack rules with network audit data to determine them. To enhance the detection rate of IDS, the emphasis is on performance measures that measure both FN & FP results. The comparison of the outcomes of each algorithm that was put into the experiments demonstrates that DT is the most effective algorithm since it has the lowest FP rate (0.009) as well as the highest accuracy (0.992).

This article [19] closes the discrepancy by looking at how common set of characteristics works in a range of internet settings and attack scenarios. Using two distinct sets of features, NetFlow and CICFlowMeter, the accuracy of recognition was examined in three significant datasets, including CSE-CIC-IDS2018, BoT-IoT, and ToN-IoT.The results show that the NetFlow set of features is superior than other options for making it easier for ML models to correctly spot a wide range of network threats. Due to the difficulty of learning models, an AI method called SHAP (SHapley Aggregate exPlanations) has been used to clarify and comprehend how ML models decide how to put things into categories. Shapley, for metrics for two common sets of characteristics, has been looked at across many datasets to figure out how much each feature added to the general ML prediction.

This article is split into five parts. The first part is an introduction, which talks about the history of the subject and the main point of this study. The 2nd section consists of work that is linked to this topic and includes multiple other pieces of research that are relevant to this research. The proposed procedure is broken down and discussed in the 3rd section. In the 4th section, the experimental findings, together with a performance evaluation and comparison to those of previous studies, are addressed. The last part of this analysis is the study's conclusion.

3. Proposed Methodology

3.1 Problem identification

Research is now being carried out on new technologies for the automated identification of aberrant system usage. In addition to this, Denning reported the construction of an intrusion-detecting model, which he said may be used as a foundation for a general-purpose intrusion-detection system [12]. Since that time, people in the industry have devised and implemented a variety of algorithms to automate the process of network identification. They have also been continuously searching for ways that are more precise, quicker, and scalable for this goal. By 2020, it's projected that there will be more than 26 billion connected devices thanks to the "IoT" and big data eras [13]. As a result of this tendency, both the variety and the total no. of cybersecurity problems are anticipated to grow. These obstacles include false alarm rate, poor detection rate, unbalanced data sets, & response time. To solve these issues, this research proposes a neural network framework followed by data over and undersampling technique for balancing the imbalanced dataset.

3.2 Proposed methodology

We suggest a new way to handle large-scale datasets with class imbalances that combines SMOTE and CNN-biLSTM (called SMO-CNN-biLSTM) to boost the number of minority classes that are found while keeping a high level of productivity. The primary function of the imbalance processing module is to resample the training dataset to decrease the amount of bias introduced to experimental findings by the imbalance that was present in the initial dataset. SMOTE "synthesizes" minority class samples, which results in a rise in the total number of minority class samples. After that, we build a flow-based intrusion identification model called SMO-CNN-biLSTM. It combines imbalanced class processing with CNN, and we study how different parameters affect the model's success. The majority of samples are also made less important by using the one-side picking method. Finally, we create a multiple-layer layered CNNLSTM framework for the binary categorization of the datasets in the categorization decision module. The datasets are divided into two groups using this model. There are 130441 samples from the majority class and 18076 samples representing the minority class.

3.3 Datasets

UNSW-NB15: Analysis, Fuzzers, denial of service (DoS), exploits, backdoors, surveillance, general attacks, worms, and shellcode are the nine categories of attacks that are included in this dataset. A complete set of 49 features and labels for classes are produced using tools like Argus and Bro-IDS as well as the construction of twelve algorithms. The UNSW NB15 has a list of these traits in the csv file. The training set has 175 343 records, whereas the testing set contains 82 343 records. Data of both attacks and regular types are present in both sets [20].

NSL-KDD: Every single item in this data collection contains a 42-dimensional feature, which is composed of 38-dimensional digital traits, a 3-dimensional symbolic feature, and a label indicating the record's traffic type. While Label has four different attack data types (Probe, DoS, R2L, and U2R), its main focus is on conventional data. The training data set (KDDTrain+) as well as the test set (KDDDTest+) from the NSL-KDD dataset, along with

the associated kinds and figures of cases, were employed as the training set and validation set, respectively, in the experiments presented in this article [21].

3.3 Preprocessing

This process is done using the following steps:

- Null values removal
- Categories data in 5 classes
- Encoding all 5 classes

The encoding process is done by a single hot encoder. One of the goals of one-hot encoding is to change data in order to get a forecast that is more precise and to get it ready for a programme. Using a one-hot method, we change every class value into a new categorical column and then give each of those columns a binary value of either 0 or 1.

3.4 Convolutional Neural Network (CNN)

LeCun is recognized for developing a well-known deep neural network called CNN. Because convolution kernels only scan in one direction along the depth direction of logging curves, our research makes use of a 1-D CNN. A standard one-dimensional CNN is seen in Figure 1. A total of three layers make up CNN: a pooling layer, a convolution layer, and a fully associated layer. By executing convolution operations and pooling operations on input data, CNN can extract implicit features from those data. After that, extracted features are combined and supplied into a layer that is fully connected. Last but not least, a neuron's output is made nonlinear by using an activation function of some kind.

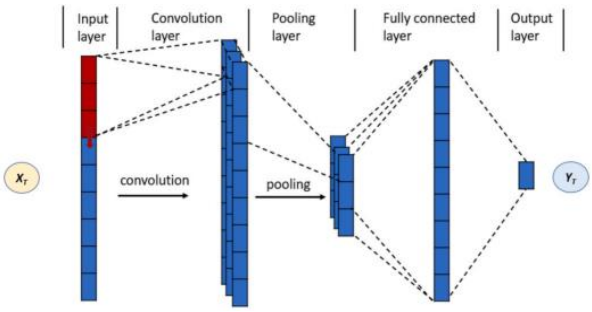

Fig. 1. Typical 1-D CNN architecture.

The convolution layer is a crucial component of CNN [22]. Every convolutional layer has several convolutional kernels that are convolved with information that is input to uncover previously hidden features and create feature maps. To produce an output of the convolutional layer, feature maps are first processed using a non-linear activation function. The convolutional layer is best described in the following manner:

$$ci = f(w_i^* x_i + b_i) \quad (1)$$

Here, x_i represents the convolution layer's input value, while c_i represents the ith produce featured map, wi represents the weight matrix, * represents the dot product, bi represents a bias vector, and f(.) represents the activator function. The function that activates the network of CNNs is frequently the rectified linear unit, commonly referred to as the ReLU function. Here is a description of the ReLU in terms of science.

$$ci = f(h_i) = \max(0, h_i) \quad (2)$$

Where hi is a feature map, which is part of a model that is built using convolutional methods. Decreasing the size of those with map features and preventing them from being overly well-fitted are the two objectives of the pooling procedure. Max pooling is one of the most common ways to pool. It is done by using equations (3) and (4) to find the maximum number of the area that has been given.

$$\gamma(c_i, c_i - 1) = \max(c_i, c_i - 1) \quad (3)$$

$$p_i = \gamma(c_i, c_i - 1) + \beta_i \quad (4)$$

The maximal pooling sub-sampling function, γ (·), is used in this case. β_i stands for prejudice. p_i is the result of the maxpooling layer. After the convolutional and pooling steps, the feature maps are sent to an entirely linked layer, which makes the final result a vector, as shown below:

$$y_i = f(t_i p_i + \delta_i) \quad (5)$$

Here, y_i is output vector, δ_i is bias, & t_i is weight matrix.

3.5 Bidirectional LSTM network

The LSTM architecture utilizes memory cells for keeping long-term historic data and a gate device to control how this information is used. A typical LSTM unit comprises 3 kinds of gates: input gate i_t, forget gate f_t, & output gate o_t. Figure 1 displays these 3 gates. At every single gate, the condition of cells in memory is changed by point-by-point multiplying and sigmoid function processes. All gates obtain the input x_t from the present state and send out $h_{t-1}$from the secret state of the previous layer. The forget gate defines what data should be deleted and what data should be retained. The sigmoid function is a function that

switches between the input that is being used right now, x_t, and a previously hidden state, h_{t-1}. Therefore, the output of the forget gates ranges from 0 to 1. If a number is close to 0, the data will be thrown away. If not, the closer to one the number is, the more info will be kept. Here's the method for figuring out forget gate:

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f) \quad (6)$$

Where stands for a sigmoid activating function, W for the weight of a gate unit, as well as b for the bias of a gate unit. The sigmoid function uses the current input x_t and the previous secret state h_{t-1} as its inputs. The input gate decides which bits are changed when 0 to 1 is switched on and off. In this context, zero means insignificance and one means paramount importance. The equation for the input to the gate is as follows:

$$i_t = \sigma(W_i . [h_{t-1} - 1, x_t] + b_i) \quad (7)$$

1) SMO-CNN-BiLSTM

Specifically, we lay out the framework and core features of the recommended approaches to earthquake prediction. In order to extract features more efficiently and increase the accuracy of predictions, we introduce a unique CNN-BiLSTM-SMOTE intrusion detection technique by combining these three networks into a single architecture.

CNN is used to derive spatial characteristics from raw data as well as retrieved characteristics of space, which are then sent to the BiLSTM network as input. The BiLSTM is used in a sequence-acquiring phase to learn long-term time data. With the help of the algorithm's trait inputs, the output block uses different weights to emphasise the most crucial details and help the model make better decisions. In the end prediction, the output layer and all layers that are entirely linked are put in the forecasting block. Each of these parts has parameters that can be instructed, such as the size of the filter, the function of loss, the amount of neurons, and the kernels. The best way to adjust these parameters may lower the error in prediction of the suggested method.

4. Results Illustrations

Experiments employed TensorFlow on Windows as backend, encoded using Python and Keras. Table 1 depicts the environment of the simulation system. In the present investigation, the training rate for the neural network model was 0.001. The experiment has repeated a total of 100 times, with a batch size of 128 for each round, and the weight inactivation rate for dropping out of the regularisation technique was set to 0.5. The Optimizer used in CNN is Adam

with categorical_crossentropy as loss function. The dataset is in training testing data at 80:20 then splitting training data again in training and validation at 80:20 respectively.

4.1 Experimental Results On Nslkdd

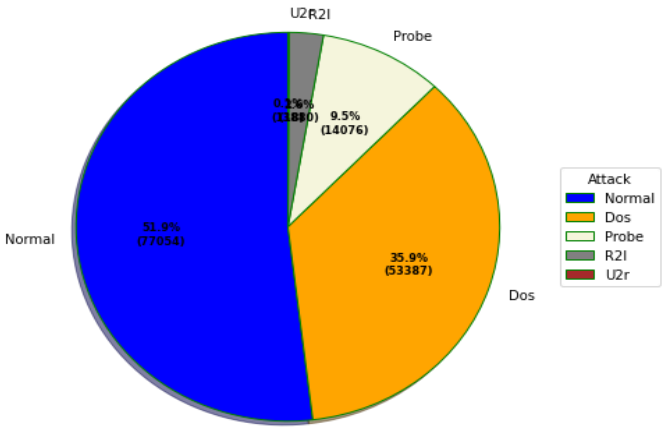

Fig. 2. Before Hybrid Sampling

Figures 2 and 3 are the visualization of datasets before sampling and after sampling. Data sampling is a statistical analysis technique utilized to choose, transform, & examine a representative group of data points to detect patterns & trends within a greater data collection. Before applying sampling, a model is not able to make differences between attack and normal classes of the NSL-KDD dataset.

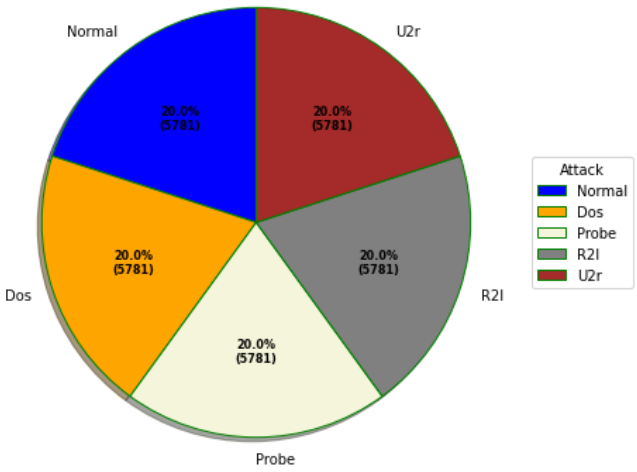

Fig. 3. After Hybrid Sampling

Figure 3 clearly shows that SMOTE has to resample the datasets and divided the dataset into 5 classes.

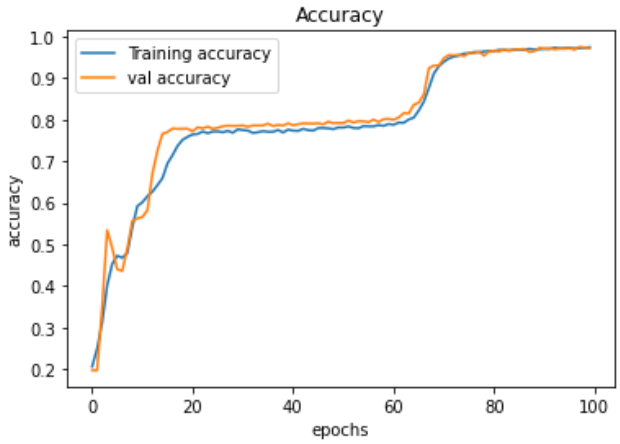

Fig. 4. The Accuracy Graph of NSL-KDD

There are one hundred epochs in the training & testing activity. No. of passes through the training dataset that the ML algorithm has gone through is denoted by the number of epochs that have passed since the beginning of the machine learning algorithm's execution. In most cases, different datasets will be bundled together into batches (especially when the amount of data is very large).

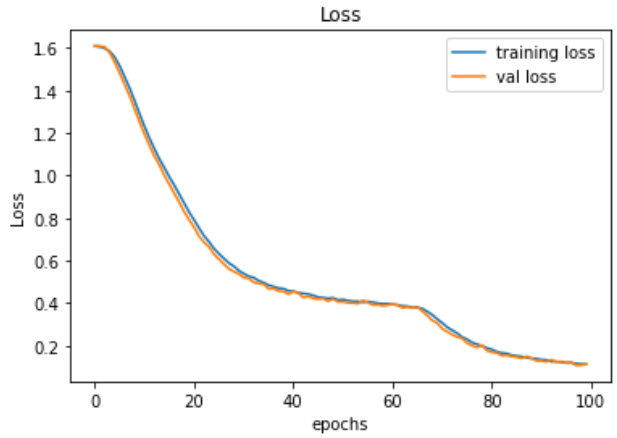

Fig. 5. The loss Graph of NSL-KDD

Also, the model has a high level of accuracy, with a training accuracy of 0.9766 and a testing accuracy of 0.9717. Training accuracy means that the same pictures are used for training as well as testing, whereas test accuracy implies that the trained model accurately finds images that were not used in training.

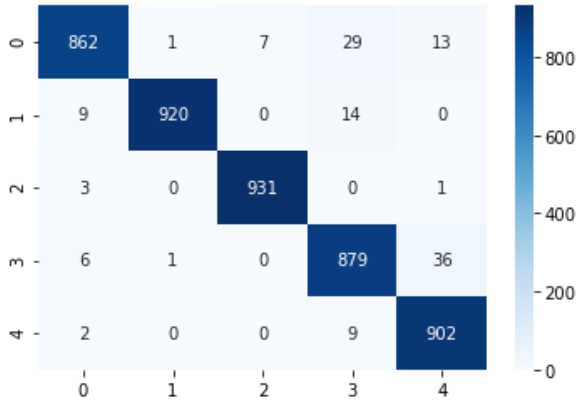

Fig. 6. Confusion Matrix of NSL-KDD

The confusion matrix is made up of two predicted classes and two real classes. Class 0 says conditions are normal, however, class 1 within the UNSW-NB15 dataset and DoS attacks Probe, U2R, and R2L activity in the NSL-KDD dataset indicate attacks. When an attack is successfully predicted as an attack, this is called a "True Positive," or "TP." False positive, or FP, occurs when a regular occurrence is mistakenly thought to be an attack. When an ordinary occurrence is correctly predicted as normal, this is called a True Negative (TN). When an attack is inaccurately predicted as normal, this is called a false negative (FN).

In Figure 6, the model has successfully predicted the true positive values which means that the model has truly identified the number of attacks in every class.

	precision	recall	f1-score	support
0	0.98	0.95	0.96	912
1	1.00	0.98	0.99	943
2	0.99	1.00	0.99	935
3	0.94	0.95	0.95	922
4	0.95	0.99	0.97	913
accuracy			0.97	4625
macro avg	0.97	0.97	0.97	4625
weighted avg	0.97	0.97	0.97	4625

Fig. 7. Performance Measures of NSL KDD

Figure 7 shows how well the NSLKDD dataset did in terms of precision, F1 score, and recall. From this number, we can determine that the model's average training accuracy is 0.9766, precision is 0.9721, recall is 0.9717 and F1 score is 9717, and validation accuracy is 0.9717.

4.2 EXPERIMENTAL RESULTS ON UNSW-NB15

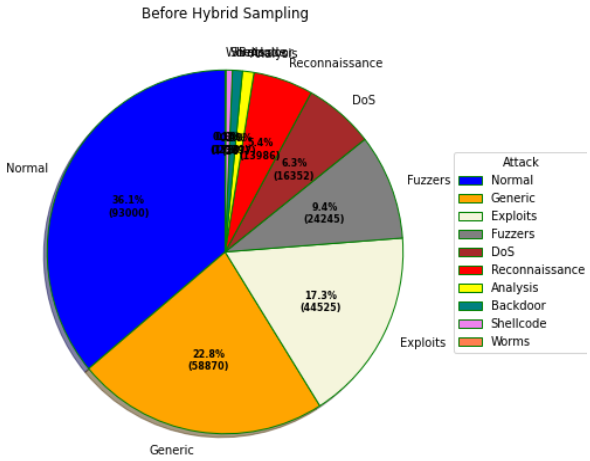

Fig. 8. Before Hybrid Sampling

Figures 8 and 9 show the sampling effect of the dataset before sampling and after sampling. Through sampling, information about the population is collected depending upon statistics of a subset of the population (sample), as opposed to examining every person. Before applying sampling (SMOTE), the dataset is highly imbalanced as can be seen in Figure 10.

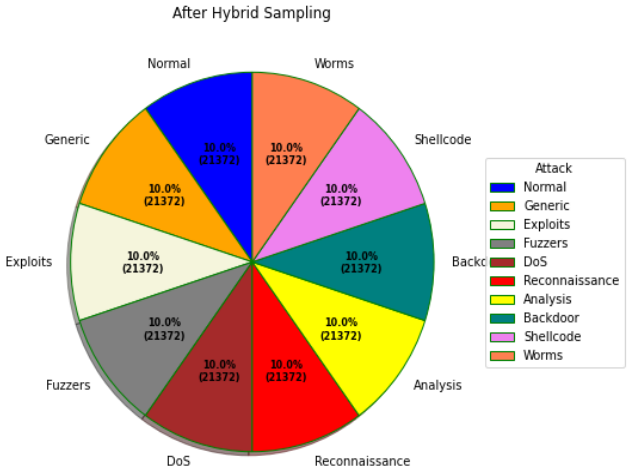

Fig. 9. After Hybrid Sampling

After the sampling process, a balanced class distribution of UNSW-NB15 is obtained. Prior to constructing a model, sampling can be done by making copies of examples from the training dataset's minority class. This may be used to balance class distribution but provides no more data to the model. In Figure 9, attacks in the dataset are classes that after applying SMOTE are evenly divided for a better classification process.

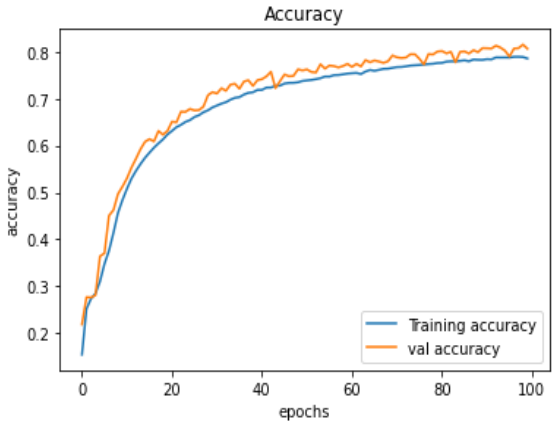

Fig. 10. The accuracy Graph of UNSW-NB15

Figures 10 and 11 are graphs that show how the suggested model works. The accuracy graph (fig. 10) shows how well our model's predictions match up with the real data. Figure 11 shows the model's training loss, which shows how effectively a model fits the training data, and Figure 12 shows the loss during validation, which shows the way successfully the model suits new data.

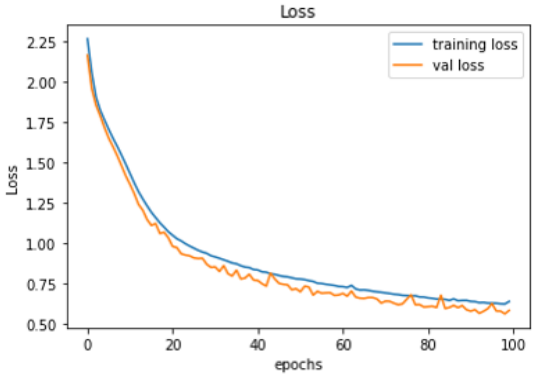

Fig. 11. The loss Graph of UNSW-NB15

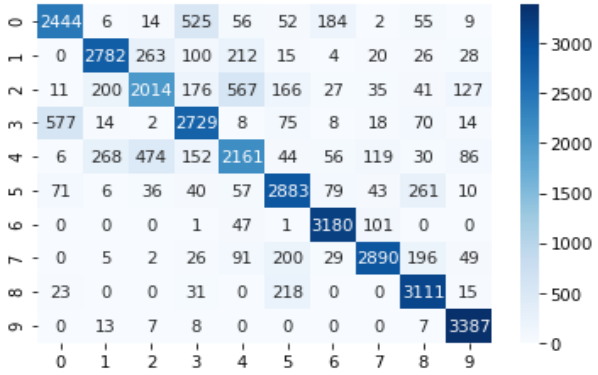

Fig. 12. Confusion Matrix of NSL-KDD

In Figure 12, the model has successfully predicted the true positive values which means that the model has truly identified the number of attacks in every class.

	precision	recall	f1-score	support
0	0.78	0.73	0.75	3347
1	0.84	0.81	0.83	3450
2	0.72	0.60	0.65	3364
3	0.72	0.78	0.75	3515
4	0.68	0.64	0.66	3396
5	0.79	0.83	0.81	3486
6	0.89	0.95	0.92	3330
7	0.90	0.83	0.86	3488
8	0.82	0.92	0.86	3398
9	0.91	0.99	0.95	3422
accuracy			0.81	34196
macro avg	0.80	0.81	0.80	34196
Weighted avg	0.80	0.81	0.80	34196

Fig. 13. Performance Measures of UNSW-NB15

Figure 13 depicts the performance matrix of the UNSW-NB15 dataset for recall, precision, and F1-score. The model's total training accuracy is 0.8173, its precision is 0.8042, its recall is 0.8066, its F1 score is 0.8038, and its validation accuracy is 0.9717, as shown in the figure.

Table 1 is the comparison of performances of both datasets based on various performance parameters. From the table, it can be reviewed that the model can efficiently predict the output in the method works well in the NSL-KDD dataset but not so well in the UNSW dataset. For the second and third strikes, NSL achieved 100% accuracy in both recall and precision.

TABLE I. COMPARISON TABLE OF PERFORMANCE MEASURES OF BOTH DATASETS

Attacks	Precision		Recall		F1-score	
	NSL	UNSW	NSL	UNSW	NSL	UNSW
0	0.98	0.78	0.95	0.73	0.96	0.75
1	1.00	0.84	0.98	0.81	0.99	0.83
2	0.99	0.72	1.00	0.60	0.99	0.65
3	0.94	0.72	0.95	0.78	0.95	0.75
4	0.95	0.68	0.99	0.64	0.97	0.66

All models were trained for 100 epochs, & statistics and classification results are provided in Table.

5. Conclusion

Within the research community, ML-based NIDSs have shown exceptional performance in the area of attack detection. Nonetheless, operational deployments have been quite limited. This inability to turn study into practical deployments was partly caused by failing to look closely enough at a common set of features throughout many datasets and not explaining the results of categorization. In the next section, we address a few of the things we think could help IDS detection work better generally. In the first step of the process, we create a balanced dataset for training models by combining OSS & SMOTE. It could cut down on the time spent developing the model, and it helps to some extent with the normal problem of bad training from uneven data. Also, a good way to make network information available for complex, multidimensional cyber-attacks is set up for a deep hierarchical network project that has been suggested. Then, use CNN and BiLSTM's hierarchical network framework to put the input data into clusters. Using the good things about deep learning, the algorithm effortlessly pulls out features by learning at different levels over and over again. We utilized both the NSL-KDD and UNSW-NB15 data to test how well the suggested method works so we could wrap up this work.

In the future, methods for modifying the cutoff value for identifying data outliers should be improved, and unbalanced data in various categories ought to be monitored. Also, for future work, we will consider applying our proposed model NIDS to fog computing.

References

[1] I. T. Union, "Internet Security Threat Report," 2019. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1353485805001947.

[2] S. B. Casey Cane, "33 Alarming Cybercrime Statistics You Should Know in 2019," 2019. [Online]. Available: https://securityboulevard.com/2019/11/33-alarming-cybercrime-statistics-you-should-know-in-2019/.

[3] H. J. Liao, C. H. Richard Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*. 2013, doi: 10.1016/j.jnca.2012.09.004.

[4] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. 1998.

[5] R. Prasad and V. Rohokale, "Artificial Intelligence and Machine Learning in Cyber Security," 2020.

[6] J. Lew *et al.*, "Analyzing Machine Learning Workloads Using a Detailed GPU Simulator," 2019, doi: 10.1109/ISPASS.2019.00028.

[7] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J. Big Data*, 2015,

doi: 10.1186/s40537-014-0007-7.

[8] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," 2016, doi: 10.1109/ICCSN.2016.7586590.

[9] W. M. Jacob NM, "A Review of Intrusion Detection Systems," *Glob J Comput Sci Technol.*, vol. 17, no. 3, pp. 54–83, 2017, doi: 10.4018/978-1-5225-8176-5.ch003.

[10] V. Jyothsna, V. V. Rama Prasad, and K. Munivara Prasad, "A Review of Anomaly based Intrusion Detection Systems," *Int. J. Comput. Appl.*, 2011, doi: 10.5120/3399-4730.

[11] J. Sen and S. Mehtab, "Machine Learning Applications in Misuse and Anomaly Detection," in *Security and Privacy From a Legal, Ethical, and Technical Perspective*, 2020.

[12] A. L'Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine Learning with Big Data: Challenges and Approaches," *IEEE Access*, 2017, doi: 10.1109/ACCESS.2017.2696365.

[13] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," 2015, doi: 10.1109/MIPRO.2015.7160458.

[14] O. Saini and S. Sharma, "A Review on Dimension Reduction Techniques in Data Mining," *Comput. Eng. Intell. Syst.*, vol. 9, no. 1, pp. 7–14, 2018.

[15] A. A. Megantara and T. Ahmad, "A hybrid machine learning method for increasing the performance of network intrusion detection systems," *J. Big Data*, 2021, doi: 10.1186/s40537-021-00531-w.

[16] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," *Comput. Networks*, 2020, doi: 10.1016/j.comnet.2020.107247.

[17] M. Nasser Mohammed and M. Mohamed Ahmed, "Data Preparation and Reduction Technique in Intrusion Detection Systems: ANOVA-PCA," *Int. J. Comput. Sci. Secur.*, 2019.

[18] N. A. Awad, "Enhancing network intrusion detection model using machine learning algorithms," *Comput. Mater. Contin.*, 2021, doi: 10.32604/cmc.2021.014307.

[19] M. Sarhan, S. Layeghy, and M. Portmann, "Evaluating Standard Feature Sets Towards Increased Generalisability and Explainability of ML-based Network Intrusion Detection," 2021.

[20] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015, doi: 10.1109/MilCIS.2015.7348942.

[21] I. Abrar, Z. Ayub, F. Masoodi, and A. M. Bamhdi, "A Machine Learning Approach for Intrusion Detection System on NSL-KDD Dataset," 2020, doi: 10.1109/ICOSEC49089.2020.9215232.

[22] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Trans. Neural Networks Learn. Syst.*, 2021, doi: 10.1109/tnnls.2021.3084827.