NETWORK INTRUSION DETECTION USING DEEP LEARNING

Dr. Suseela S
Assistant Professor,
School of Computer Science & Engineering .
Vellore Institute of Technology ,
Chennai, INDIA.
suseela.s@vit.ac.in

Sheshi Kiran Reddy Mandla,
School of Computer Science & Engineering,
Vellore Institute of Technology ,
Chennai, INDIA.
mandlasheshi.kiran2020@vitstudent.ac.in

Uppanapalli Lakshmi Sowjanya ,
School of Computer Science & Engineering,
Vellore Institute of Technology ,
Chennai, INDIA.
lakshmi.sowjanya2020@vitstudent.ac.in

Amaravadi Dheeraj,
School of Computer Science & Engineering
Vellore Institute of Technology,
Chennai, INDIA.
amaravadi.dheeraj2020@vitstudent.ac.in

***Abstract*—This A very common problem in Digital world is about Security issues of any kind of confidential information. Saving and retrieving data in digital world is easy, but what about the security of the information? Here's where we concern about Cyber Security measures and methods. One such measure is Intrusion Detection System. This system detects the intruder who tries to take over the control on website/organization's data and alarms / notifies the administrator or security officer.**

But how can IDS detect malicious activities? That's where we implement Deep Learning concepts i.e. Deep Neural Networks. They mimic the neurons in human brain and thus have the capability to detect the malicious intrusions in millions of normal intrusions. We mostly concentrated on balancing the data set , CIC-DoHBrw2020 data set and our concern is mostly on predicting between the malicious and benign records.So we have done a comparative study between the imbalanced data set and then the data set which we balanced using the SMOTE technique and the results have shown that the balanced data set has the better results..

Keywords—confidential, Deep Learning, CNN, SMOTE technique

I. INTRODUCTION

The main intention of the project is to introduce a Network Intrusion Detection System using a well-known and more powerful technique i.e., Neural Networks in Deep Learning. As NIDS has many attributes that has to be clearly studied/learned in order to build an efficient DL model that can predict intrusions. Scope of this project is to develop a deep learning model using Neural Networks that can predict Intrusions into an organization or a network.This CNN model can be installed in a networks' security system,such that, it can take the values of attributes in dataset of every system that tries to enter the network and if the prediction is Malicious, an alarm can be triggered to notify the security officers or the administrators.We can develop an ML model on IDS, it can detect attacks based on the data and features we considered during feature extraction in model building. But what about some new and strange Intrusion attack?

Here is where Deep Learning is required. Deep Learning exceeds Machine Learning due to it's capability of analyzing raw data and feature extraction.

Deep learning consists of various networks such as (CNNs),(RNNs), (DBNs) and (RBMs) which are having different capabilities and properties. In this paper we mainly concentrated on the CNN method and also on its optimizers rather than comparing between the different models.Apart from that we also thought of using the balanced data set and its importance and showcase on how it helps in increasing the accuracy.

II. RELATED WORK

The work in [1] mainly focuses on the machine learning algorithms to develop an intrusion detection system and the anamoly based detection has nearly 0.95 accuracy>The first part of the system uses the random forest which has best working accuracy and for the second part the one class SVM has achieved an accuracy of 0.80.

In [2], XGBoost and catboost achieved the near-perfect predictive results, 0.999995 and 0.9992 respectively. After XGBoost and catboost, Random Forest is the one with high accuracy (i.e., 0.92 - 2.3 prediction time). But the training and predicting time of these models are nearly 10-12seconds which is very high to it's application. i.e., by the time packet is considered malicious, it might enter the network and may start it's effect.

The experiment done on SMBR-Extracellular Data in [3] show that this is better than any other models used and also achieved precision of 1.0 and the f1s cores are nearly equal to 1(0.93).The (SMBR-XGBDT) which uses the SMOTE , BORUTA , XGBoost algorithm has been suggested in [3].

This paper [4] uses several machine learning algorithms and the experiments in [4] have proved that the random forest and the gradient boosting best fits for the data used and achieved the highest accuracy

In paper [5], they used machine Learning algorithms and "CIC-DoHBrw-2020" data set. They proposed that classifiers work better for detecting the DNS attacks.The random forest classifier and XGBoost classifier works best for the proposed sytem and the "Gated Reccurent Unit" has achieved the minimum error rate.

This paper [6] presents new methodology to classify the DNS using the explainable AI solution . The stacked RF method has got the high true positive rate and the f1 score.It also highlights the importance of having to know the ranking of the features.

The work done in [7] is mainly focused up on the deep learning model which uses the LSTM model and the BiLSTM

model and they have done a comparative study between the both.The BiLSTM (0.994 test accuracy) model has performed better than the LSTM model(0.952 test accuracy) and it has been used to generally classify DNS over HTTPS traffic with deep learning on the CIC Dataset-DoHBrw-2020.

The researchers in [8] had used the several machine learning approaches for the detection process and has used the layered approach that is for the layer-1 they have finalized on the Random Fine Trees to classify between the Doh and Non-DoH traffic with accuracy of 99.42 % and Ada-Boost Trees to define between the malicious and benign DoH with accuracy of nearly 100.00% among the SVM , Ada-Boost and Random Fine Trees model on the CIC Dataset-DoHBrw-2020.

In [9] , the research mainly focuses on the importance of the RGPQR (Rational Quadratic Gaussian Process Regression) in reducing the false alarm rates up to 71.73% and a reduction of 67.61% on CIC-DoHBrw-2020 data set.

The [10] shows the importance of sampling and the use of this sampling methods on the data has increased the accuracy from nearly 4 % to 30 % .In this paper they have used the CIC Dataset-DoHBrw-2018 and they have got the highest accuracy for the Ada-Boost method (99.32% accuracy) by using the oversampling method that is increasing the data size of the minority samples to balance the data-set.

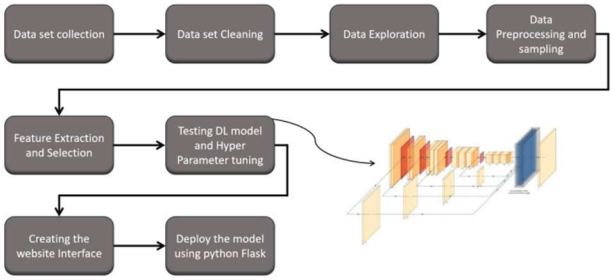
The research in the [11] mainly focuses upon the sampling of the imbalanced data-set and the emphasizes on the importance of sampling.They have used the MCMC approach , GANS approach and oversampling method to the CIC-IDS2018 data-set for balancing the data-set.However the logistic regression without balancing the data-set approach outperformed the GANS and MSMS approach with accuracy of 0.88.

Paper	Dataset	Algorithm used	Accuracy achieved
[1] A Machine Learning IDS for Known and Unknown Anomalies	CIC-IDS2018	Supervised (Random Forest) ; Unsupervised (1-CLASS SVM)	Random Forest - 95% ; 1-CLASS SVM - 80%
[2] USING MACHINE LEARNING FOR INTRUSIONDETECTION SYSTEMS	CIC-IDS2018	Naive Bayes ; Logistic Regression ; SVM (linear kernel) ; CSVM (RBF kernel) ; Random Forest ; xgboost ; xgboost with Active Learning ; xgboost	catboost - 0.9992 ; xgboost - 0.9992 ; xgboost with Feature Engineering - 0.999995
[3] Improve the Application of XGBDT in Network Abnormal Traffic Detection	CIC-DoHBrw-2020	(SMBR-XGBDT) Smote Algorithm & Boruta algorithm ; XGBoost	Precision rates of DOH., Non-DOH,Benign-doH, Malicious-19H, 100%, 98%, and
[4] Detecting Malicious DNS over HTTPS Traffic Using Machine Learning	CIC-DoHBrw-2020	Naive Bayes (NB), Logistic Regression (LR), Random Forest (RF), K-Nearest Neighbor (KNN), and Gradient	Random Forest & Gradient Boosting - 100%
[5] Classifying DNS Tunneling Tools For Malicious DoH Traffic	CIC-DoHBrw-2020	Random Forest , XGBoost , GRU , LSTM	Random Forest - 99.11% ; XGBC - 99.22% ; Low error rate - GRU -
[6] An Explainable AI-Based Intrusion Detection System for DNS Over HTTPS (DoH) Attacks	CIC-DoHBrw-2020	Stacked Random Forest, Explainable AI	Precision Of 99.91%
[7] Generalized Classification of DNS over HTTPS Traffic with Deep Learning	CIC-DoHBrw-2020	LSTM and BiLSTM	BiLSTM model has got highest accuracy (0.994)
[8] A Lightweight Double-Stage Scheme to Identify Malicious DNS over HTTPS Traffic Using a Hybrid Learning Approach	CIC-DoHBrw-2020	AdaBoost , Random Forest , SVM	Layer-1: Random Forest - 99.42% - (DoH vs Non-DoH) LAYER-2: Adaboost - 100% (Benign vs Malicious)
[9] RQGPR: Rational Quadratic Gaussian Process Regression for Attack Detection in the SCADA Networks	CIC-DoHBrw-2020	RQPGR (Rational Quadratic Gaussian Process Regression) algorithm	Reduction in alarm rate to nearly 72%
[10]Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset	CIC-IDS2018	Sampling method(SMOTE), and AdaBoost , Random Forest , Decision Trees , KNN , Gradient Boosting , Linear Discriminant Analysis	Highest Accuarcy for AdaBoost after sampling which is 99.32%
[11] Using Markov Chain Monte Carlo Algorithm for Sampling Imbalance Binary IDS Datasets	CIC-IDS2018	GANS Approach ,Markov Monte Carlo Algorithm , Logistic regression	Highest Accuracy For imbalanced dataset with Logistic regression 0.88

III. PROPOSED ARCHITECTURE

A. DATASET

Data set chosen for this project to be worked on is [12].We received / retrieved the data-set to work on from the website[12]. Total_CSVs zip file is unzipped and l2_maliciousandl2_benign datasets are considered for model building.These two datasets are merged to make one united categorical data set. This categorical data set is the one which is considered from start: Data Pre-processing to end: Building various neural network models.



B. CNN ARCHITECTURE

A convolutional neural network (CNN) is a type of deep learning model that is commonly used in image and video processing tasks, but can also be applied to other types of data. The basic idea behind CNNs is to learn and extract features from raw input data such as images or text by applying multiple convolution filters and combining layers. When predicting the results of a data set where every attribute is of equal importance, a CNN model can be used to automatically learn the most important features of the input data. To do this, the input data is first converted into a suitable form that can be fed into the network, such as a series of image sets or vector sequences. The CNN model then applies a series of convolution filters to the input data that search the input data for specific features or patterns. These filters can be thought of as little windows that read the input and calculate the dot product between the filter weights and the input data values at each location. The resulting output is a functional map showing the presence or absence of a filter feature or pattern anywhere in the input data. After the convolutional filters, levels of aggregation are used to sample the feature maps and reduce their size, making the model more computationally efficient and less prone to overfitting. Finally, the resulting feature maps are smoothed into a vector and fed into one or more fully connected layers that perform a series of linear operations to produce the final model result. Based on the results of the CNN model, various outcomes, such as binary classification or regression, can be predicted. When predicting outcomes from a dataset where every attribute is equally important, a CNN model can learn to extract key characteristics from input data and use them to make accurate predictions.

The model used here is the CNN architecture consists of the following layers:

1. Conv1D layer with total of 64 filters, kernel size equals 3, same padding, and ReLU activation function.
2. MaxPooling1D layer with a pool size of 2 to down sample the feature maps.
3. Flatten layer to convert the 3D output of before layers into 1D vector.
4. Dense layer with 128 units and ReLU activation function.
5. Dropout layer has dropout rate of 0.5 to prevent overfitting.
6. Dense output layer with 2 units and sigmoid activation function for binary classification.

The total number of trainable parameters in this model is 180,866. Please note that this architecture assumes an input shape of (44,1) for the input data.

C. STRATIFIED K-FOLD CROSS VALIDATION

Stratified K-Fold Cross-Validation is a technique used to estimate the performance of the machine learning model. It's exactly like a Stratified K Fold Sampling, but rather than use it for training and testing purposes, it is being used to assess the model's performance. In order to ensure that each fold has almost the same amount of samples from each class, the data set is divided into K equalized folds in the Stratified K Fold Cross Validation. The model shall be trained on K1 folds and assessed in the remaining ones.

It will be repeated K times, using all of the folds for evaluation on a single occasion. For an estimation of model performance, the results shall be averaged across all K folds. The advantages of synthesized K Fold Cross Validation are that it provides a more precise estimate of the model's performance compared to other methods such as simplified K Fold Cross Validation.

For this reason, it ensures that every fold is distributed in line with the same classes to avoid discrimination and oversimplification. When dealing with imbalanced datasets in which individual classes do not have equal number of samples, the use of Stratified KDLLF Cross Validation is particularly useful. In such cases, using a representative sample of data can help to ensure an accurate assessment of the model and may lead to more accurate prediction by use of K Stratified XML Cross Fold Validation. One of the strongest techniques for evaluating machine learning models' performance, with particularly in view of balancing datasets, is systematic K.Fold cross validation.

D. ADAM

Adam is another optimization algorithm that can be used in CNN models to efficiently train them and improve their performance in predicting outcomes from a dataset where every attribute is equally important. Here are some key points about Adam's uniqueness and how it works:

- This means that it not only uses a moving average of past gradients like momentum, but also adapts the learning rates of each weight parameter based on the statistics of the gradients.
- The learning rates in Adam are adapted by computing a separate adaptive learning rate for each weight parameter based on estimates of the first and second moments of the gradients.
- The first moment estimate is a moving average of the gradients, which serves as an estimate of the gradient mean.
- The second moment estimate is a moving average of the squared gradients, which serves as an estimate of the gradient variance.
- By using these estimates, Adam is able to automatically adapt the learning rates for each weight parameter in a

way that is sensitive to the magnitude and direction of the gradients.

- Overall, Adam is a powerful optimization algorithm that can be used to train CNN models more efficiently and effectively, especially when every attribute of the input data is equally important.
- Its ability to adapt the learning rates of each weight parameter based on estimates of the gradients can help the model to converge more quickly and produce more accurate results.

E. Adagrad

Adagrad is an optimization algorithm that can be used with CNN models to train them more effectively and efficiently. By predicting results from a dataset where each attribute is equally important, Adagrad can help ensure that the model learns the most important features of the input data and converges to a good solution. Here are some key points about the uniqueness of Adagrad and how it works:

- Adagrad is a gradient-based optimization algorithm, i.e. it works by calculating the gradients of the loss function with respect to the model parameters and adjusting them in a direction that minimizes the loss.
- This means that parameters with large gradients have a lower learning capacity, while parameters with small gradients have a higher learning capacity.
- Learning rate matching in Adagrad is done using a parameter specific learning rate, calculated by dividing the initial learning rate by the sum of squares of the historical gradients for that parameter.
- This means that parameters with large historical gradients have a slower learning rate, while parameters with small historical gradients have a higher learning rate.
- This is because the parameter-specific learning rate can be adjusted separately for each feature, so the model learns the most important features faster and more accurately.
- Instead, the learning rate is automatically adjusted for each parameter based on its historical gradients.

F. RMSProp

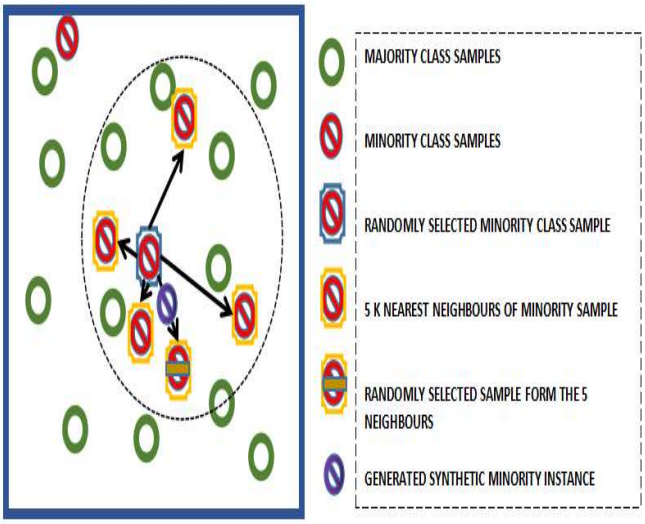
RMSProp is an optimization algorithm that can be used in CNN models to improve the efficiency and effectiveness of the training process. When predicting outcomes from a dataset where each attribute is equally important, RMSProp can help the model learn the most important features and converge to a good solution. Here are some key points about RMSProp and its uniqueness:

- Unlike Adagrad, RMSProp uses the method of "moving average of squared gradients" to normalize the learning rate, which helps prevent the learning rate from becoming a very larger value or a smaller value as training progresses.
- One of the main differences between RMSProp and other optimization algorithms is that it uses a parameter-specific learning rate that adapts based on the moving average of the squared gradients of each parameter.
- This is because parametric learning rates can help the model avoid getting stuck in narrow valleys of the loss function.
- One potential disadvantage of RMSProp is that it can be sensitive to the choice of hyper-parameters, such as the initial learning rate and the decay rate of the moving average.
- Overall, RMSProp is a powerful optimization algorithm that helps CNN models learn more efficiently and effectively, especially when dealing with large datasets or datasets with high-dimensional inputs.
- Its parametric learning rate adaptation, moving average of squared slopes, and robustness to local optima make it a popular choice for many deep learning applications.

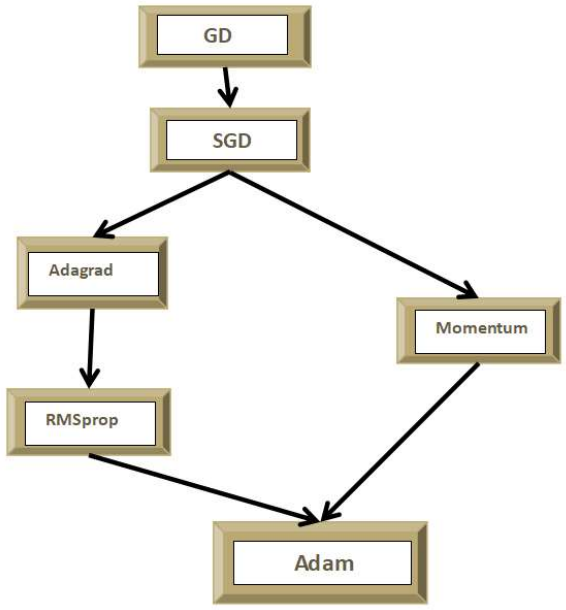
G. SMOTE TECHNIQUE

The SMOTE algorithm is a technique used in machine learning to address imbalanced classification problems, where one class has significantly fewer samples than the other. It generates synthetic samples for the minority class by creating new ones that are linear combinations of existing samples. To apply SMOTE, a sample is selected from the minority class, and its k nearest neighbors (where k is a user-defined parameter) are identified. One of these neighbors is randomly chosen, and a new sample is generated by interpolating between the selected sample and the chosen neighbor. This process is repeated until the desired number of synthetic samples has been created.

SMOTE is effective in increasing the number of minority class samples and improving classifier performance for imbalanced classification tasks. However, in some cases, it can produce noisy or redundant samples, which may negatively impact classification performance. Therefore, it's crucial to carefully determine the appropriate value for the k parameter and evaluate classifier performance with and without SMOTE oversampling to determine its usefulness for a given data set and task.



IV. PROPOSED METHODOLOGY



A. *Points that support Adagrad and RMSProp and Adam model's high accuracies.*

1) Adagrad:

- Adagrad is a proven optimization algorithm that has been used successfully in many deep learning applications.
- Adagrad is well-suited to deal with sparse gradients, which can be a common issue in deep learning models with large numbers of parameters.
- Adagrad adjusts the learning rate for each parameter based on its past gradients, which can help the model learn efficiently and effectively.
- Adagrad is simple to implement and does not require much hyper parameter tuning, making it an attractive option for many deep learning applications.
- One potential drawback of Adagrad is that it may accumulate too much gradient history over time, leading to a diminishing learning rate and slow convergence.

2) RMSProp:

- RMSProp is an adaptive optimization algorithm that can help deep learning models learn more efficiently and effectively.
- RMSProp uses the method of "moving average of squared gradients" to normalize the learning rate, which helps prevent the learning rate from becoming a larger value or small value .
- RMSProp is less prone to getting stuck in local optima than some other optimization algorithms, such as stochastic gradient descent.
- RMSProp can help the model learn more quickly by adapting the learning rate based on the gradients of each parameter.
- One potential drawback of RMSProp is that it can be sensitive to the choice of hyper parameters, such as the initial learning rate and the decay rate of the moving average.

3) Adam:

- Adam is an advanced optimization algorithm that combines the benefits of Adagrad and RMSProp.
- Adam uses both a moving average of past gradients and a moving average of past squared gradients to adapt the learning rate for each parameter.
- Adam is well-suited to deal with sparse gradients, making it a good choice for deep learning models with large numbers of parameters.
- Adam is computationally efficient and requires relatively little hyper parameter tuning, making it an attractive option for many deep learning applications.
- One potential drawback of Adam is that it can be sensitive to the choice of hyper parameters, such as the initial learning rate and the decay rates of the moving averages.

B. *DATA PRE-PROCESSING*

The first step in processing the data involved combining two datasets, one labeled as l2-benign and the other as l2-malicious, that were assigned to me. However, this posed a challenge when splitting the data into training and testing sets because the order of the entries could bias the results. To address this, we randomly shuffled the combined data set so that both labels were evenly distributed throughout.

The combined data set contained both categorical and numerical features. The former included features such as SourceIP, DestinationIP, SourcePort, and DestinationPort, while the latter included features such as Duration, FlowSentRate, FlowReceived, and FlowBytesRate. To handle the categorical features, we analyzed each feature and selected a suitable threshold, categorizing values outside the threshold as "other". One-hot encoding was then applied to these features to feed them into the neural network. For the numerical features, we applied different methodologies such as Median, Standard Scaling, and Sc_Fit_Transform, to normalize the values and avoid bias in the data set.

SAMPLE TYPE	NUMBER OF SAMPLES BEFORE SAMPLING PROCESS	NUMBER OF SAMPLES AFTER SAMPLING PROCESS
MALWARE	269643	269643
BENIGN	19807	203479
TOTAL	289450	473122

Since the data set had significantly more records for the l2-malicious label than for the l2-benign label (269643 versus 19807), we used SMOTTEN oversampling method to create a balanced data set of total 473122 records. Random sampling was then used to combine the two datasets into one, resulting in a total of 473122 records.

Total number of records totally added to the **183672.**

C. MODEL BUILDING

a) To begin the data analysis, the necessary libraries and packages were imported into Jupyter notebook. Next, we checked for any invalid Python data types or non-numeric values in the data, ensuring that the array contained only numerical data.Since the given data set was highly unbalanced as it had significantly more records for the l2-malicious label than for the l2-benign label (269643 versus 19807), we performed sampling to prevent overfitting.We used SMOTTEN oversampling method to create a balanced data set of total 473122 records. Random sampling was then used to combine the two datasets into one, resulting in a total of 473122 records.

In the CNN model, we added two layers - Convolution1D and MaxPooling1D - and used different optimizers such as Adam(50 Epochs), RMSProp(50 Epochs), and Adagrad(50 Epochs). We applied 2-fold cross-validation and binary cross-entropy as the metric for loss. Additionally, we performed hyperparameter tuning for parameters such as number of hidden units, number of stacked units, loss function, value of dropouts, number of epochs, folds variations, and choice of metrics, in order to obtain a final model with improved accuracy.

Since we got adam as the best result among the three optimizers , we wanted to improve it by some more training so we trained it with 100 epochs and achieved the best accuracy.

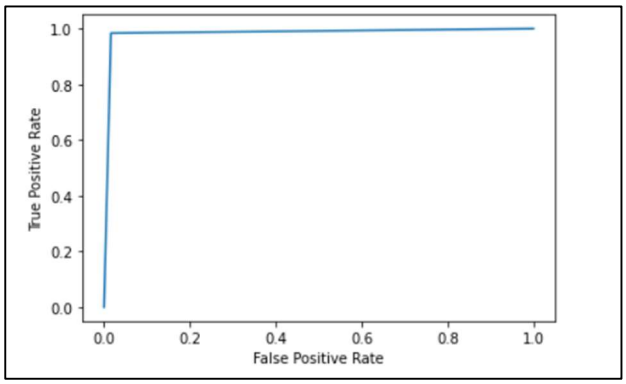
To evaluate the performance , we used various performance metrics such as precision, recall, F1-score, and support, and also plotted the ROC curve to calculate the area under the curve (AUC).

V. RESULTS & FIGURES

a) ADAM (50 EPOCHS)

```
Training accuracy: 98.46366047859192
Test accuracy: 98.4000027179718
```

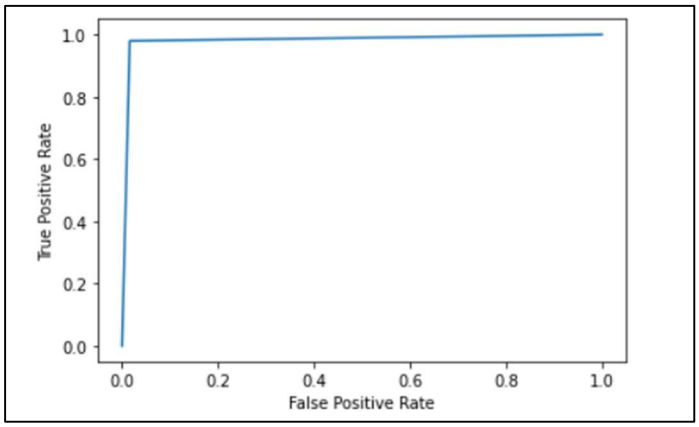
	precision	recall	f1-score	support
0	0.98	0.98	0.98	48236
1	0.98	0.98	0.98	46389
micro avg	0.98	0.98	0.98	94625
macro avg	0.98	0.98	0.98	94625
weighted avg	0.98	0.98	0.98	94625
samples avg	0.98	0.98	0.98	94625



b) ADAGRAD (50 EPOCHS)

```
Training accuracy: 98.2515037059784
Test accuracy: 98.19498062133789
```

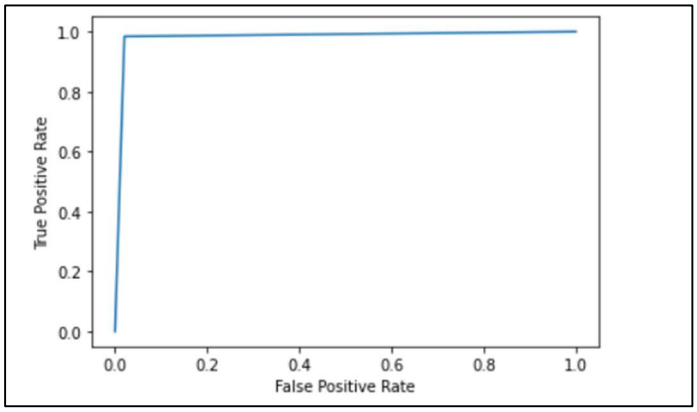
	precision	recall	f1-score	support
0	0.98	0.98	0.98	48236
1	0.98	0.98	0.98	46389
micro avg	0.98	0.98	0.98	94625
macro avg	0.98	0.98	0.98	94625
weighted avg	0.98	0.98	0.98	94625
samples avg	0.98	0.98	0.98	94625



c) RMSPROP (50 EPOCHS)

```
Training accuracy: 98.2094943523407
Test accuracy: 98.18229675292969
```

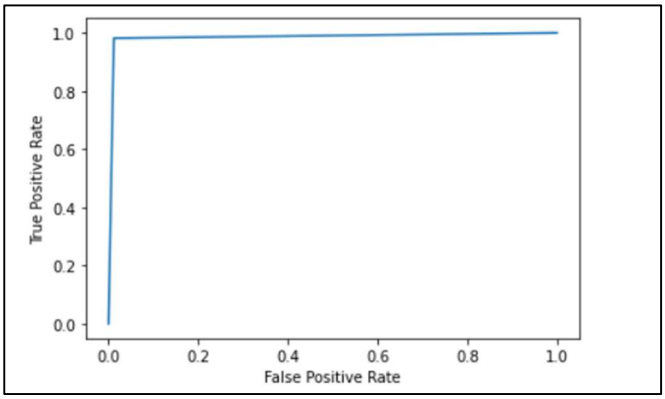
	precision	recall	f1-score	support
0	0.98	0.98	0.98	48236
1	0.98	0.98	0.98	46389
micro avg	0.98	0.98	0.98	94625
macro avg	0.98	0.98	0.98	94625
weighted avg	0.98	0.98	0.98	94625
samples avg	0.98	0.98	0.98	94625



d) ADAM (100 EPOCHS)

```
Training accuracy: 98.59126806259155
Test accuracy: 98.50885272026062
```

	precision	recall	f1-score	support
0	0.98	0.99	0.99	48236
1	0.99	0.98	0.98	46389
micro avg	0.99	0.99	0.99	94625
macro avg	0.99	0.99	0.99	94625
weighted avg	0.99	0.99	0.99	94625
samples avg	0.99	0.99	0.99	94625



We have selected the "Adam - 100 epochs" model because it gives us the high accuracy and also the the f-score value near 1 (~0.98) proves that the model is best trained, neither under-fitted or over-fitted.

OPTIMIZER	NO OF EPOCHS	TRAINING ACCURACY (%)	TEST ACCURACY (%)
Adam	50	98.46366048	98.40000272
Adagrad	50	98.25150371	98.19498062
RMSprop	50	98.20949435	98.18229675
Adam	**100**	**98.59126806**	**98.50885272**

VI. CONCLUSION

A deep neural network consisting of stacked Convolution1D and MaxPooling1D layers, together with appropriate changes in optimizers and epochs, has been successfully created. In order to avoid bias in the model & to feed the data to the algorithm for training, appropriate preprocessing & sampling has been carried out. For improved accuracy, the model has been tested by using a twofold cross validation method and tuning with Hyperparameters.On the same data set of 50 epochs, Adam optimizer showed better results than we could have expected. Even after the number of epochs had increased to 100, it showed a considerable increase in accuracy which was almost 99% (Adam - 100 Epochs).

REFERENCES

[1] F. Aguiló–Gost, E. Simó–Mezquita, E. Marín–Tordera and A. Hussain, "A Machine Learning IDS for Known and Unknown Anomalies," 2022 18th International Conference on the Design of Reliable Communication Networks (DRCN), Vilanova i la Geltrú, Spain, 2022, pp. 1-5, doi: 10.1109/DRCN53993.2022.9758010.

[2] Q.-V. Dang, "Using Machine Learning for Intrusion Detection Systems", Comput. Inform., vol. 41, no. 1, pp. 12–33, Apr. 2022.

[3] F. Binhao, H. Hong and Z. Ziyun, "Improve the Application of XGBDT in Network Abnormal Traffic Detection," 2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT), Chongqing, China, 2021, pp. 193-200, doi: 10.1109/ICESIT53460.2021.9696640.

[4] S. K. Singh and P. K. Roy, "Detecting Malicious DNS over HTTPS Traffic Using Machine Learning," 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT), Sakheer, Bahrain, 2020, pp. 1-6, doi: 10.1109/3ICT51146.2020.9312004.

[5] R. Alenezi and S. A. Ludwig, "Classifying DNS Tunneling Tools For Malicious DoH Traffic," 2021 IEEE Symposium Series on Computational Intelligence (SSCI), Orlando, FL, USA, 2021, pp. 1-9, doi: 10.1109/SSCI50451.2021.9660136.

[6] T. Zebin, S. Rezvy and Y. Luo, "An Explainable AI-Based Intrusion Detection System for DNS Over HTTPS (DoH) Attacks," in IEEE Transactions on Information Forensics and Security, vol. 17, pp. 2339-2349, 2022, doi: 10.1109/TIFS.2022.3183390.

[7] L. F. Gonzalez Casanova and P. -C. Lin, "Generalized Classification of DNS over HTTPS Traffic with Deep Learning," 2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Tokyo, Japan, 2021, pp. 1903-1907.

[8] Abu Al-Haija, Qasem & Alohaly, Manar & Odeh, Ammar. (2023). A Lightweight Double-Stage Scheme to Identify Malicious DNS over HTTPS Traffic Using a Hybrid Learning Approach. Sensors. 23. 3489. 10.3390/s23073489.

[9] L. A. Chijioke Ahakonye, C. Ifeanyi Nwakanma, J. M. Lee and D. -S. Kim, "RQGPR: Rational Quadratic Gaussian Process Regression for Attack Detection in the SCADA Networks," 2022 27th Asia Pacific Conference on Communications (APCC), Jeju Island, Korea, Republic of, 2022, pp. 595-600, doi: 10.1109/APCC55198.2022.9943564.

[10] G. Karatas, O. Demir and O. K. Sahingoz, "Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset," in IEEE Access, vol. 8, pp. 32150-32162, 2020, doi: 10.1109/ACCESS.2020.2973219.

[11] N. Abedzadeh and M. Jacobs, "Using Markov Chain Monte Carlo Algorithm for Sampling Imbalance Binary IDS Datasets," 2022 International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 2022, pp. 1-7, doi: 10.1109/ICCCN54977.2022.9868900.

[12] http://205.174.165.80/CICDataset/DoHBrw-2020/Dataset/