VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

ANOMALY DETECTION SYSTEM USING CONVOLUTIONAL NEURAL NETWORK

The Project report as part of Information Security And Audit Analysis
(CSE 3501)

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology
In
COMPUTER SCIENCE AND ENGINEERING
SCOPE

Prepared By:

K. Harish Gokul-19BCT0049
Naga Durga Amarnadh Reddy Dodda-19BCE2003
Sreekar Samineni-19BCE2348

Under the guidance of
Prof. Umadevi K S

School of Computer Science and Engineering
VIT, Vellore.

1. ABSTRACT

Security is the most important aspect in any organisation. A system to detect and prevent these attacks is necessary to ensure that data is preserved from any malicious activity. Attack vectors targeting user data are plenty and new attack vectors are generated every day. There are certain attacks which are known to us and there are plenty unknown attacks which we are not aware of. Although we can define these attacks through signatures of known attacks, it is important to detect them These attacks are an anomaly, and anomaly-based detection techniques are a helpful tool for identifying both known and unknown/new attacks, allowing them to deal with the wide range of attacks and the ever-changing nature of network intrusions. An anomaly detection system is implemented using a deep learning approach.. This method of learning demonstrates the power of generative models with strong categorization, as well as their capacity to deduce a portion of their knowledge from incomplete data and flexibility.

2. INTRODUCTION TO DOMAIN

The Internet and its corresponding enterprises are growing at a fast pace. Data and network traffic are at all-time highs. The methods and procedures used to gather sensitive information are also changing. These attacks are becoming more diverse and are constantly in a state of advancement and evolution. As a result, a versatile security system capable of analysing massive amounts of network traffic to precisely detect a wide range of threats is necessary. Anomaly detection provides a means to detect probable threats by continuously observing and modelling normal network behaviour.

Anomaly detection system helps calculate deviation from the normal behaviour. Thus factors like unusual volume of network traffic, or a large number of requests made in a particular window of contention or unusual changes in the packet information. Anomalies are divided into three types: point anomalies, contextual anomalies, and collective anomalies.These categories of anomaly include 4 main types of attacks namely DoS (Denial of Service), Probe, U2R (User to root) and R2L (Remote to Local). Here is a brief description about these classes:

1. **Denial of Service Attack (DoS):** A DoS attack occurs when an attacker makes a computer or memory resource too busy or full to perform valid requests, or denies legitimate users access to a system.
2. **Probing Attack:** A probing attack is an attempt to collect information about a network of computers with the apparent purpose of circumventing its security defences.
3. **User to Root Attack (U2R):** This is a vulnerability in which an attacker gains access to a system's normal user account and then exploits it to gain root access. This is accomplished using techniques such as password sniffing, dictionary attacks, and social engineering.
4. **Remote to Local Attack(R2L):** A remote to local attack occurs when an attacker who has the capacity to transport packets across a network but does not have an account on that machine leverages a vulnerability to gain local access as a user of that system (R2L).

Network intrusion detection systems must identify all of these anomalies, which must then be investigated and categorised into network attack types mentioned above. Aside from known attacks, anomalies in network traffic are used to detect various unknown assaults. However, in many cases, abnormal behaviours are simply standard system behaviours that are just outdated. As a result, these anomaly-based Network Intrusion Detection System should change their behaviours and adapt to changing network environments with new network protocols.

This is the role of an anomaly detection system. The second half of the project statement enumerates the method of employment. Deep learning is a part of machine learning techniques that employs numerous layers of information processing stages for feature learning and classification, both supervised and unsupervised. Deep learning models, such as the CNN utilised here, have been widely used in image identification applications. Thus a deep learning model can be developed to exploit the advantages of models like CNN in Anomaly based Network Intrusion detection systems.

Thus the system developed is trained on "normal" network data to develop a Model in anomaly-based network intrusion detection. Whenever the system's Model is ready, it's employed to determine if new events, objects, or traffic are abnormal or not. An expected quality of a successful model is its ability to adapt in order to generalise its behaviour and cope with dynamic network environments. This necessitates the use of a self-learning system. Thus for the detection of novel and unknown attack vectors based on unusual behaviour or a deviation from the normal behaviour, an Anomaly detection system is used employing a deep learning model.

3. LITERATURE SURVEY

1. A Convolutional Neural Network for Improved Anomaly-Based Network Intrusion Detection-2021

Al-Turaiki, I., & Altwaijry, N. (2021). A convolutional neural network for improved anomaly-based network intrusion detection. Big Data, 9(3), 233–252. https://doi.org/10.1089/big.2020.0263

The paper provided us the metrics and the problem statement of the project. It details the use of CNN on the 1999 KDD dataset to detect anomalies. The basic principle used here is to convert the Intrusion detection dataset into images which can be classified using a convolutional neural network in 2D. The following are the contributions to the literature made by this study:
(1) A hybrid two-step preprocessing approach using deep feature synthesis.
(2) A novel binary classification ADNIDS based on DNNs,
(3) A novel multiclass classification ADNIDS based on DNNs for network intrusion detection,

The papers also compares the findings to those of other deep learning methodologies and cutting-edge classification models. The proposed models outperformed existing models in the literature in terms of accuracy and precision, and are geared to cope with binary and multiclass classification problems.

2. Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks-2021

Ullah, I., & Mahmoud, Q. H. (2021). Design and development of a deep learning-based model for anomaly detection in IOT networks. IEEE Access, 9, 103906–103926. https://doi.org/10.1109/access.2021.3094024

This paper provided us the motivation to use Convolutional Neural Network 1D and its specifications pertaining to the implementation of the model on a dataset. This paper provides

an array of options of Convolutional Neural Network models that can be applied to a variety of datasets pertaining to IoT networks. It details how CNN 1D, CNN 2D, CNN 3D models can be used to build an anomaly detection system in datasets like the BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, and IoT-23 intrusion detection datasets. The convolutional neural network proposed here was first designed for a particular dataset. Then the method of Transfer learning was used to implement binary and multiclass classification using a convolutional neural network multiclass pre-trained model [1]. The paper details the number of layers of convolution, used for each and every type of model and its specifications of implementation. Since the paper details the both binary and multiclass classification , we implemented a binary classifier and applied the concept of transfer learning by applying this on a different dataset.

3. Anomaly based Intrusion Detection System using Machine Learning-2021

Runwal, A. (2021). Anomaly based Intrusion Detection System using machine learning. International Journal for Research in Applied Science and Engineering Technology, 9(9), 255–260. https://doi.org/10.22214/ijraset.2021.37955

This paper provided a comparative analysis of the use of machine learning techniques in building an anomaly detection system. The main advantage in choosing a deep learning technique is the ability of these semi supervised and unsupervised methods to detect patterns from an existing dataset already. In this paper numerous experiments were conducted and tested to evaluate the efficiency and effectiveness of the following machine learning classifiers: Logistic Regression, Gaussian Naive Bayes, Decision Tree, Random Forest, and Support Vector Machine. The tests were conducted using the NSL-KDD Train Dataset, with the Random Forest Classifier achieving the greatest accuracy rate. We may also rule out some of the other models that did not perform as well as the others, as evidenced by the findings. The paper also presented how hyper tuning of these parameters involved could help improve the performance of Machine Learning models. However the dataset contained redundancies and hence a more advanced version of this dataset had to be considered. This point was taken in by the team and we proceeded onto choosing a better version of the same dataset.

4. A Detailed Analysis of the KDD CUP 99 Data Set-2009

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD Cup 99 data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. https://doi.org/10.1109/cisda.2009.5356528

This paper provided us the details of dataset employed in this project. This paper deals with the features of dataset, its means of synthesis from the DARPA 98 project and the advancements made in this dataset since it was first adopted in the year 1999. Thus the dataset is also referred to as the 1999 KDD dataset, its latest version named as the KDD Cup Dataset. It provides details on the 3 main categories of features namely the basic, traffic and the content features. The explanation of these categories are elucidated in the following sections of the report in detail. This paper provides this new dataset called the KDD cup dataset or NSL- KDD datatset as mentioned above rectifying the problems of redundancy and data labelling encountered earlier in the first dataset.

5. Benchmarking datasets for anomaly-based network intrusion detection: KDD Cup 99 alternatives

This paper gave us an insight into the various datasets which could be used to satisfy the project statement and this details the various benchmarking standards which are used to classify data sets like KDD cup, UNSW-NB15, NSL KDD etc. It provides details about the disadvantages of moving from the KDD cup 99 dataset due to its non-stationarity of training and test datasets. The papers goes on to explain that the NSL KDD dataset is the most widely used dataset to design anomaly detection system due to the lack of other good datasets of this magnitude available,listing out the other options and their advantages and potential downfall

6. Deep Learning Approach for Intelligent Intrusion Detection System

The main takeaways of this paper was the advanatages of using DNN's or Deep neural network models over traditional machine learning classifiers on designing an intrusion detection system.The DNN model proposed in this paper is implemented on a variety of datasets including the NSL KDD dataset which was of prime interest of the project team. They offer scale-hybrid-IDS-AlertNet, a highly scalable and hybrid DNNs framework that can be utilised in real-time to successfully monitor network traffic and host-level events in order to preemptively notify probable threats.

7. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study

This paper is again a comparative study on the deep learning systems which are applied on various datasets in designing an intrusion detection system. This paper provides 7 primary approaches like RNN, restricted Boltzmann machine, main focus of interest convolutional neural networks, deep Boltzmann machines and deep autoencoders.The datasets involved in this range from ones used in network intrusion detection systems and IoT based datasets.Thus the efficacy and method of implementation are different for different datasets.

8. Online-semi supervised neural anomaly detector to identify MQTT-based attacks in real time

This paper provided us the base statement and the primary motivation to choose to build an anomaly detection system. It details a real time anomaly monitoring system which uses a single hidden semi feedforward neural layer network. It is a semi supervised deep learning approach which can detect MQTT based attacks on real time and we extrapolated a few ideas, methods and models which we could implement onto a different dataset with the same principles in place.

9. Applying convolutional neural network for network intrusion detection

This paper gave us the definitive approach to use CNN and variations of CNN in network intrusion detection systems. This is due to the fact owing to CNN's capacity to extract high-level feature representations, which are the abstract form of low-level feature sets of network traffic links. This study models network traffic as a time-series, namely transmission control protocol/internet protocol (TCP/IP) packets in a predetermined time range. They assess the effectiveness of various methods using the most major synthetic ID data set, such as KDD Cup 99.

4. PROPOSED WORK

This work is an exercise in how deep learning can be used to develop a model to classify normal and abnormal behaviour. We use a CNN architecture to create binary classification models for an Anomaly Detection System, Here we have used a 1D Convolutional Neural Network. The motivation behind choosing this model are as follows:

1. The ability of CNNs to harness spatial or temporal correlation in data is its main advantage.
2. In comparison to other deep learning models, a CNN requires less parameters. As a result, the complexity of the model is reduced, and the learning process is improved.
3. CNNs have also been utilised for feature extraction and classification in the context of intrusion detection.

A convolutional neural network assigns weights and biases to various visual elements and differentiates them. Thus it is widely used in image recognition systems and thus an application of this model in Anomaly Detection systems is the fact that the data is converted into an image and then the problem is processed as an image recognition problem. Thus this novelty in approach provides a method which is faster and accurate than Machine Learning models like Logistic Regression, Support Vector Machines which are used for classification problems like these. Thus the power of CNN can be fully leveraged by translating intrusion detection concerns into image recognition problems.

The model consists of an input layer, three blocks of convolution layers, flatten layer, a fully connected layer, and an output layer. Each block consists of a 1D convolutional layer, and dropout layer. The input layer receives inputs from the reshaping system. The reshaping system transforms network data into a format compatible with CNN1D [2].
A neural network that uses convolution has a possibly overfitting issue and will have to undergo extensive fine-tuning of the test dataset parameters. A dropout layer reduces the chance of overfitting by ignoring some neurons during the training phase. When adjacent frames are interrelated strongly with feature maps, a regular dropdown does not regularize the activations. We used a spatial dropout layer that drops the entire feature maps instead of individual units. The tensor is reshaped to create a flat operation on a tensor with the number of elements in the tensor, excluding the batch size, equal to the number of elements in the tensor[2]. This helps us eliminate this issue and increases the accuracy. In the design of this model we have used 2 activation functions: the *relu* and *softmax* activation function. The relu activation function is used mostly except in the last layer. The sigmoid function is used in many machine learning models notably logistic regression. Here relu provides a better alternative to the sigmoid function as it is computationally more efficient to perform *relu*. The model using relu tend to show better convergence performance than sigmoid. Thus this was one of our main motivation to use it. The second activation function used was softmax in the last dense layer. The main motivation behind using the softmax function is that it helps calculate the relative probability of the output and even though other functions like sigmoid or tanh could be used, this provides the best accuracy in scenarios involving both binary and multi-class classification. The model design is elucidated in the implementation of this project.

Dataset:

The KDD Cup 1999 Dataset is the dataset we used in our project. It is based on the data collected during DARPA'98. MIT Lincoln Labs designed and led the DARPA Intrusion Detection Evaluation Program in 1998. The goal was to survey and assess intrusion detection research. A common set of auditable data was provided, which included a wide variety of simulated intrusions in a military network environment. A variant of this dataset was used in the 1999 KDD intrusion detection competition.The notable features from this dataset are highlighted into 3 categories as follows:

1. **Basic Features:** All the attributes that can be derived from a TCP/IP connection are grouped together in this category. The majority of these characteristics cause an implicit detection delay [4].

2. **Traffic Features:** Features that are computed with regard to a window interval fall into this category, which is separated into two groups:

a. "Same host" features: generate statistics about protocol behaviour, service, and so on based on only the connections that have the same destination host as the current connection in the last 2 seconds.

b. "Same service" features: only look at connections in the last 2 seconds that have the same service as the current connection.

Time-based features are the two sorts of "traffic" characteristics discussed above. There are, however, a number of slow probing attacks that scan hosts (or ports) over a considerably longer time span than 2 seconds, such as once every minute. As a result, no intrusion patterns with a time window of 2 seconds are produced by these attacks. The "same host" and "same service" features are recalculated to tackle this problem, but this time using a connection window of 100 connections rather than a time window of 2 seconds. Connection-based traffic features are the name for these functionalities [4].

3. **Content Features:** The R2L and U2R attacks, unlike the majority of DoS and Probing assaults, do not have any intrusive frequent sequential patterns. This is because DoS and Probing attacks involve a large number of connections to a single host in a short period of time, but R2L and U2R attacks are embedded in the data portions of packets and usually only involve a single connection. We need some functionality to be able to search for strange activity in the data section, such as the amount of failed login attempts, to detect these types of assaults. These are referred to as content features [4]. The important features of the dataset can be used illustrated as follows:

Feature Name	**Description**	**Type**
Duration	Length (number of seconds) of the connection	Continuous
Protocol_type	Type of the protocol, e.g. tcp, udp, etc.	Discrete
Service	Network service on the destination, e.g.,http, telnet, etc.	Discrete

src_bytes	Number of data bytes from source to destination	Continuous
dst_bytes	Number of data bytes from destination to Source.	Continuous
flag	Normal or error status of the connection.	Discrete
land	1 if connection is from/to the same host/port; 0 otherwise	Discrete
wrong_fragment	Number of “wrong” fragments	Continuous
urgent	Number of urgent packets	Continuous

Basic features of TCP Connection[4]

Feature	Name	Type
Hot	Number of “hot” indicators	Continuous
num_failed_logins	Number of failed login attempts	Conitnuous
logged_in	1 if successfully logged in; 0 otherwise	Discrete
num_compromised	Number of “compromised” conditions	Continuous
root_shell	1 if root shell is obtained; 0 otherwise	Discrete
su_attempted	1 if “su root” command attempted; 0 otherwise	Discrete
num_root	Number of “root” accesses	Continuous
num_file_creations	Number of file creation operations	Continuous
num_shells	Number of shell prompts	Continuous
num_access_files	Number of operations on access control files	Continuous
num_outbound_cmds	Number of outbound commands in an ftp session	Continuous
is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise	Discrete
is_guest_login	1 if the login is a “guest” login; 0 otherwise	Discrete

Content features by Domain Knowledge[4]

Dataset Limitations: The proposed work has been a new approach offered in the field of anomaly detections but as every work has its own advantages, there are a few limitations we encountered. These were dataset limitations elucidated as follows:

1. Due to privacy and security concerns, the majority of the datasets that describe contemporary network traffic attacks are private. On the other hand, publicly available datasets are

painstakingly anonymised and contain a variety of flaws. They failed to verify that their datasets frequently display the real-world network traffic profile, in particular. One of the most widely used publicly available datasets is KDD Cup 99. Despite numerous well-known harsh comments, it has been employed as an useful benchmark dataset for many NIDS-related research studies throughout the years.

2. [10] details why machine learning classifiers in the KDDCup 99 dataset have limited capabilities in detecting assaults in the content ('R2L' and 'U2R') category. None of the machine learning classifiers were able to enhance the attack detection rate using this dataset. They admitted that in most circumstances, the only way to acquire a high assault detection rate is to create a new dataset that combines training and testing datasets. Furthermore, [10] discovered that many' snmpgetattack' attacks are classified as 'R2L' attacks. As a result, machine learning classifiers perform badly with this data in the vast majority of scenarios.

3. Despite the harsh comments, KDDCup 99 has been the most extensively used credible benchmark dataset in most studies linked to the evaluation of ID systems and other security-related tasks [10]. A developed and more improved version of KDDCup 99 dubbed NSL-KDD to address the fundamental flaws in KDDCup 99. They deleted redundant connection records from the full train and test data, as well as the invalid records 136,489 and 136,497 from the test data[10]. As a result, the classifier is protected from being biassed in favour of the more frequent connection records.Thus this is version of the dataset employed in the project.

5. IMPLEMENTATION

TOOLS USED:

The implementation of the project was carried out on Google Colab Notebooks. The programming language used by the team was python and a few notable packages were used in order to build the deep learning model. For the Data preprocessing and preparation of data we included matplotlib, seaborn,numpy and pandas. For building the deep learning model we have used keras which provides an python interface for Artificial Neural Networks. We have also used the sckikit learn library in order to split the data into train datasets and test datasets.

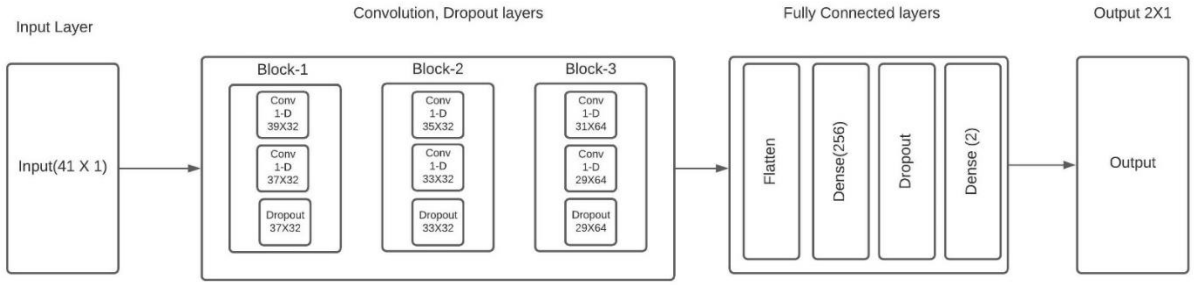
Model Design of CNN 1D:

Convolution is calculated in 1D using time access and the kernel movies in one direction. The input and output data for CNN1D are two-dimensional, and it is most often used for time series data [2]. First, an input vector 41×1 is generated to fit the 41 best features chosen by the feature selection algorithm. Here we have taken into consideration all the features from the dataset in which the value of the correlation between the feature and the output class is greater than 0.97. After the input layer, three convolution layer blocks were added to the model. The three-convolution layer blocks are considered a method of feature learning. The convolution layers extract features from the input image and find image properties from small data samples within the input, retaining the vector relationship [2].

Our first convolution layer uses the *relu* activation function with a filter size of 32 and a kernel size of 5. The output of the first convolution layer is (41, 32). We used a spatial dropout layer to regularize the training data model and reduce overfitting with a drop value of 0.4[2]. Each of the three convolution layers uses identical parameters except the filters, which are the same in

the first 2 layers of convolution but is then doubled to 64 from 32 in the final convolutional layer block. Since the first 2 layers of the convolution use the same filter parameters the need for an average pooling layer to downsample feature maps is not required in this methodology of implementation. We have already performed the process of downsampling earlier in the data preprocessing stage.The classification part consists of fully connected flatten and dense layers. The flatten layer is applied to the model transforming the tensor into a shape equivalent to tensor elements. There is no batch size parameter for the flatten layer [2].The fully connected layers consist of flatten layer followed by a dense layer with 256 neurons with *relu* activation function. Then we use a dropout layer right after this with a drop value of 0.5 to reduce overfitting again. After this we obtain result after applying a dense layer with 2 neurons with the *softmax* activation function. These 2 neurons represent our output label class. Thus the result obtained is either a normal behaviour or an anomaly which is the aim of our problem statement.

Architecture Diagram of the Model designed:

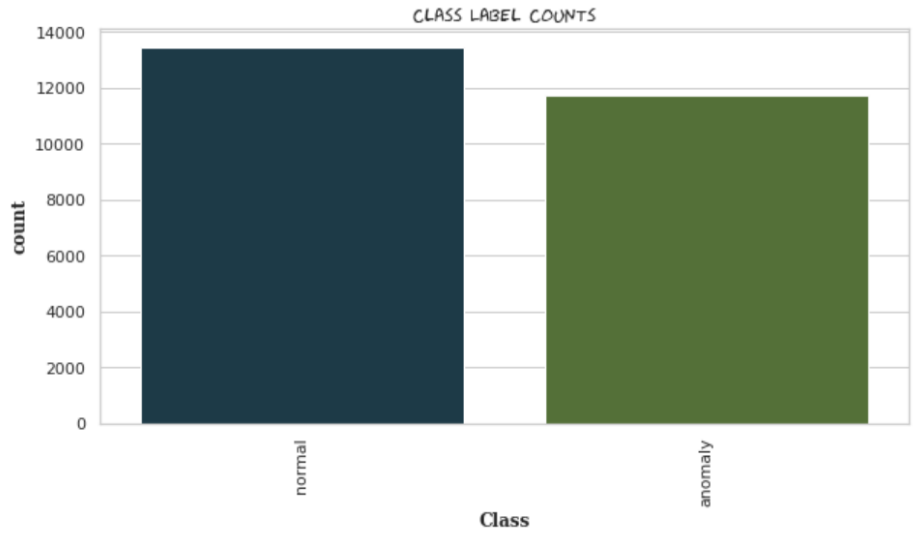


6. RESULTS AND DISCUSSION

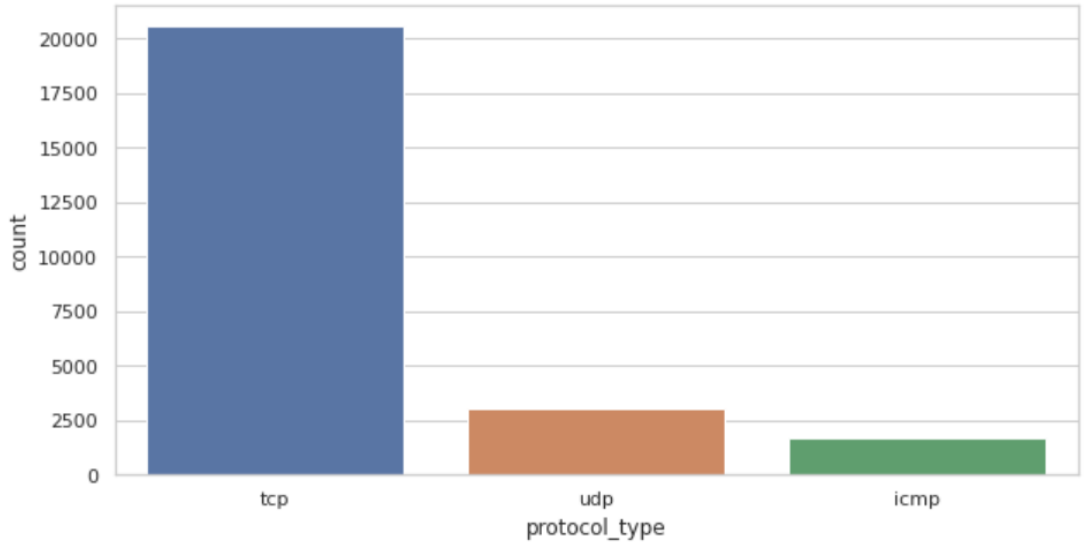
The implementation of Convolutional Neural Network -1D helped the team in designing an Anomaly detection System with a train accuracy of 95.74 % and a test accuracy of 96.17%.The problem statement dictated the team to come with a solution to design a model which had the ability to detect anomalies at a higher accuracy rate compared to traditional machine learning models on the KDD Cup dataset and the team came with the a CNN 1-D model as a means to combat this.

Figures (Data):

1. Class labelling (Anomaly and Normal Class)



2. Protocol types (TCP/ UDP/ ICMP)



Figures(Model):

3. Convolution Model details:

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 39, 32)	128
conv1d_1 (Conv1D)	(None, 37, 32)	3104
dropout (Dropout)	(None, 37, 32)	0
conv1d_2 (Conv1D)	(None, 35, 32)	3104
conv1d_3 (Conv1D)	(None, 33, 32)	3104
dropout_1 (Dropout)	(None, 33, 32)	0
conv1d_4 (Conv1D)	(None, 31, 64)	6208
conv1d_5 (Conv1D)	(None, 29, 64)	12352
dropout_2 (Dropout)	(None, 29, 64)	0
flatten (Flatten)	(None, 1856)	0
dense (Dense)	(None, 256)	475392
dropout_3 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 2)	514

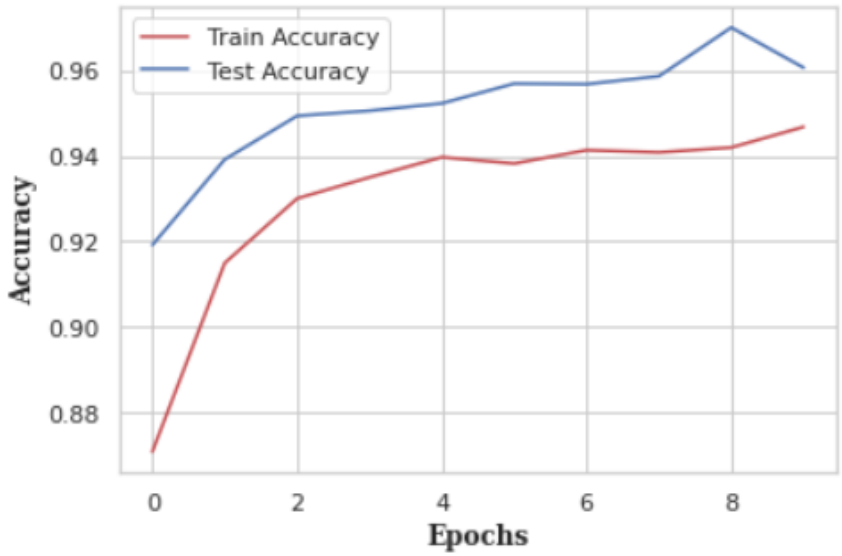
Total params: 503,906
Trainable params: 503,906
Non-trainable params: 0

4. Time Taken for Compilation

```
Epoch 1/10
138/138 [==============================] - 12s 18ms/step - loss: 40.1121 - accuracy: 0.8709 - val_loss: 5.1701 - val_accuracy: 0.9192
Epoch 2/10
138/138 [==============================] - 2s 13ms/step - loss: 27.8782 - accuracy: 0.9149 - val_loss: 0.7697 - val_accuracy: 0.9391
Epoch 3/10
138/138 [==============================] - 2s 14ms/step - loss: 6.6394 - accuracy: 0.9300 - val_loss: 1.2526 - val_accuracy: 0.9493
Epoch 4/10
138/138 [==============================] - 2s 15ms/step - loss: 3.4470 - accuracy: 0.9350 - val_loss: 1.4277 - val_accuracy: 0.9505
Epoch 5/10
138/138 [==============================] - 2s 14ms/step - loss: 9.2912 - accuracy: 0.9397 - val_loss: 0.6922 - val_accuracy: 0.9522
Epoch 6/10
138/138 [==============================] - 2s 15ms/step - loss: 17.0570 - accuracy: 0.9382 - val_loss: 0.8711 - val_accuracy: 0.9569
Epoch 7/10
138/138 [==============================] - 2s 14ms/step - loss: 2.9758 - accuracy: 0.9413 - val_loss: 0.2952 - val_accuracy: 0.9567
Epoch 8/10
138/138 [==============================] - 2s 15ms/step - loss: 0.9477 - accuracy: 0.9408 - val_loss: 0.3793 - val_accuracy: 0.9586
Epoch 9/10
138/138 [==============================] - 2s 15ms/step - loss: 1.8900 - accuracy: 0.9419 - val_loss: 0.3988 - val_accuracy: 0.9700
Epoch 10/10
138/138 [==============================] - 2s 15ms/step - loss: 1.0537 - accuracy: 0.9468 - val_loss: 0.4284 - val_accuracy: 0.9606
```

Observation: From the above runtime history of the model we can say that the total runtime for this model is 2 minutes and 28 seconds. Here we limit the model runtime to only 10 epochs of training. The time taken here is important as most anomaly detection systems need to offer a near real time detection mechanism to combat any malicious activity. Here the use of GPU or a faster core processor helps bring down the time and hence not only the efficacy but the runtime of the model is an advantage here. Commercial systems use epoch rate in the hundreds and thus the runtime with a GPU will be much more faster and efficient.

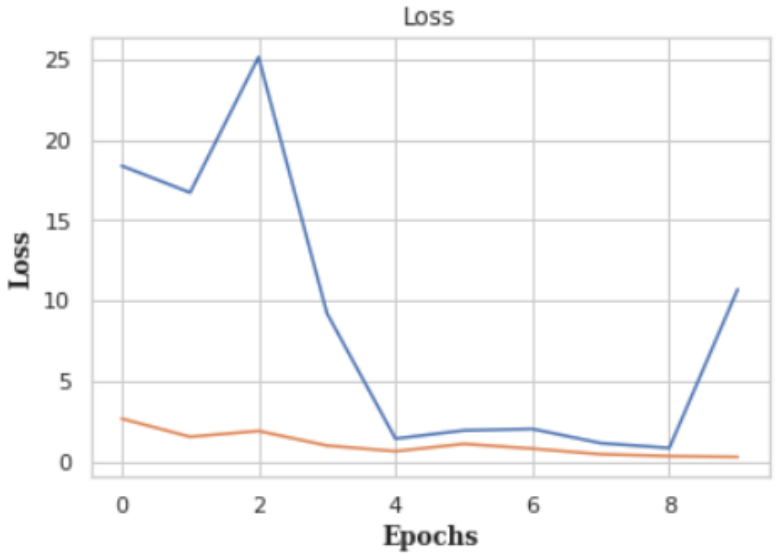
5. Train and Test Accuracy



Observation: We can observe here that there is a higher test accuracy as compared to train accuracy. Now the inference from this fact is the unsupervised nature of Deep learning networks like CNN which work with autoencoders. They provide better accuracy on data which is never seen before as compared to ML models and thus a main characteristic of an anomaly detection system is in agreement. New attack vectors can be identified using the knowledge gained from

the attack signatures of known attacks. Thus identification of an anomaly becomes more accurate.

6. Loss

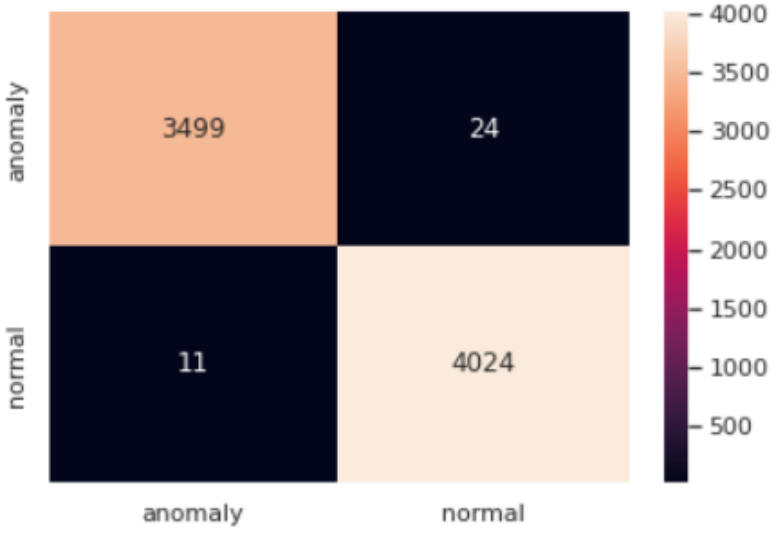


The prediction loss for Binary classification is estimated using Binary cross entropy given by,

$$loss(pd,ed)=-1N\sum i=1N[edilogpdi+(1-edi)log(1-pdi)]$$

Where ed is true probability distribution, pd is predicted probability distribution. We have used adam as an optimizer to minimize the loss of Binary cross entropy[10].

7. Confusion Matrix of the CNN model



Observation: Another metric of classification accuracy is the confusion matrix. Here we can categorise our results into the categories of true positive, true negative, false positive and false negative. The true positive being an anomaly detected correctly, a true negative being a normal behaviour categorised as normal , a false positive being an incorrect labelling of an attack and an false negative being the non-detection of an anomaly. From the results obtained we can see out of the 7558 observation, the number of false positives is 24 and the number of false negatives being the lowest 11.Thus we require a system that allows the lowest possible value of false negative and thus this model accomplishes that goal. From this metric we can calculate the following:

1. Precision:

Precision= True Positive /(True Positive + False Positive)
Precision= (3024)/(3024+24) = 0.992

2. Recall:

Precision= True Positive /(True Positive + False Negative)
Precision= (3024)/(3024+11) = 0.996

3..F1- Score

F1−Score=2× (Precision×Recall/Precision+Recall)
F1- Score=0.993

8.ACCURACY:

Classifier	Train-Accuracy	Test-Accuracy
CNN 1D	95.740000	96.190000

7. CONCLUSION

Deep learning algorithms have shown their ability to correctly classify anomalies in a variety of disciplines of inquiry. Intruders, on the other hand, use cutting-edge methods to launch cyber-attacks. The team was successful in developing an anomaly detection system using Convolutional Neural Network 1-D model on the NSL KDD dataset.The binary classification model designed gave a higher test accuracy than the training accuracy thus determining the efficacy of the model on unknown or untrained data. Our proposed CNN 1-D model gave a train accuracy of 95.74 % and a test accuracy of 96.19% on the test accuracy with a precison of 0..992,a recall value of about 0.996 and a F1 Score of 0.993.

8. REFERENCES

1. Al-Turaiki, I., & Altwaijry, N. (2021). A convolutional neural network for improved anomaly-based network intrusion detection. Big Data, 9(3), 233–252. https://doi.org/10.1089/big.2020.0263
2. Ullah, I., & Mahmoud, Q. H. (2021). Design and development of a deep learning-based model for anomaly detection in IOT networks. IEEE Access, 9, 103906–103926. https://doi.org/10.1109/access.2021.3094024
3. Runwal, A. (2021). Anomaly based Intrusion Detection System using machine learning. International Journal for Research in Applied Science and Engineering Technology, 9(9), 255–260. https://doi.org/10.22214/ijraset.2021.37955
4. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD Cup 99 data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. https://doi.org/10.1109/cisda.2009.5356528
5. Divekar, A., Parekh, M., Savla, V., Mishra, R., & Shirole, M. (2018). Benchmarking datasets for anomaly-based network intrusion detection: KDD Cup 99 alternatives. *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*. https://doi.org/10.1109/cccs.2018.8586840
6. R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," in IEEE Access, vol. 7, pp. 41525-41550, 2019, doi: 10.1109/ACCESS.2019.2895334.
7. Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, *50*, 102419. https://doi.org/10.1016/j.jisa.2019.102419
8. Gao, Z., Cao, J., Wang, W., Zhang, H., & Xu, Z. (2021). Online-semisupervised neural anomaly detector to identify MQTT-based attacks in real time. *Security and Communication Networks*, *2021*, 1–11. https://doi.org/10.1155/2021/4587862
9. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Applying convolutional neural network for network intrusion detection. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. https://doi.org/10.1109/icacci.2017.8126009

Blogs/ Website Links and Tutorials/ Code snippets followed in order to achieve objective:

10. Brownlee, J. (2020, August 26). *How to make predictions with keras*. Machine Learning Mastery. Retrieved December 8, 2021, from https://machinelearningmastery.com/how-to-make-classification-and-regression-predictions-for-deep-learning-models-in-keras/.
11. Ahmad M AzabAhmad M Azab 1111 silver badge22 bronze badges, & user104365user104365 1. (2018, March 1). *Confusion matrix results in CNN keras*. Data Science Stack Exchange. Retrieved December 8, 2021, from https://datascience.stackexchange.com/questions/67424/confusion-matrix-results-in-cnn-keras.
12. *CUP-99 task description*. KDD. (n.d.). Retrieved December 8, 2021, from http://kdd.ics.uci.edu/databases/kddcup99/task.html.

9. APPENDIX:CODE

#imports used in the project

```
import pandas as pd
%matplotlib inline
import os

import matplotlib.pyplot as plt
import pandas as pd
import cv2
import numpy as np
from glob import glob
import seaborn as sns
import random
import pickle

from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
```

 # Data Preprocessing

```
data = pd.read_csv('2.Intrusion_detection.csv')
data
data.info()
data.describe()
data.isnull().sum()
data.isnull().any()
data.corr()
data['class'].value_counts()
#Removing duplicates
data = data.drop_duplicates()
data.shape
```

#Encoding Categorical Data

```
from sklearn.preprocessing import LabelEncoder
columns=data.columns
label_encoder=LabelEncoder()
for cols in columns:
    if(isinstance(data[cols].values[0],str)):
        data[cols]=label_encoder.fit_transform(data[cols].values)
```

#Data Visualisation

```
#Protocol Type
fig = plt.figure(figsize =(10, 5))
sns.countplot(data['protocol_type'])
plt.show()
```

 # Service Type

```
fig = plt.figure(figsize =(20, 5))
sns.countplot(data['service'])
plt.show()

sns.set(style="whitegrid")
plt.figure(figsize=(10, 5))
ax = sns.countplot(x="class", data=data, palette=sns.color_palette("cubehelix", 4))
plt.xticks(rotation=90)
plt.title("Class Label Counts", {"fontname":"fantasy", "fontweight":"bold",
"fontsize":"medium"})
plt.ylabel("count", {"fontname": "serif", "fontweight":"bold"})
plt.xlabel("Class", {"fontname": "serif", "fontweight":"bold"})

sns.countplot(df_balanced['class'])
plt.grid()
plt.legend()
plt.title(' 0 :ANOMALY & 1 : NORMAL ')
plt.show()
print(' ')
plt.pie([730,730],labels=['NORMAL','ANOMALY'],autopct='%.2f%%')
plt.legend(loc=(1,0.5))
plt.title(' 0 :ANOMALY & 1 : NORMAL ')
plt.show()
```

 # Resampling Process

```
from sklearn.utils import resample
# Separate majority and minority classes
df_majority = data[data['class']== 1]
df_minority = data[data['class']== 0]
```

 # Downsample majority class and upsample the minority class

```
df_minority_upsampled = resample(df_minority,
replace=True,n_samples=2500,random_state=100)
df_majority_downsampled = resample(df_majority,
replace=False,n_samples=2500,random_state=100)
```

 # Combine minority class with downsampled majority class

```
df_balanced = pd.concat([df_minority_upsampled, df_majority_downsampled])
```

 # Display new class counts

```
df_balanced['class'].value_counts()
```

#Display after encoding and resampling

```
data.head()
x = data.iloc[:,:-1]
y = data.iloc[:,-1]
x.head()
```

```
y.tail()
```

#CNN Implementation

```
from keras.utils.np_utils import to_categorical#convert to one-hot-encoding
from keras.models import Sequential
from keras.layers import
Dense,Dropout,Flatten,Conv1D,MaxPool1D,GlobalAvgPool1D,GlobalMaxPooling1D
from tensorflow.keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.utils.np_utils import to_categorical
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
```

#Split of test and train dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

#Data Display

```
Y_train= to_categorical(Y_train)
Y_test= to_categorical(Y_test)
Y_train.shape
Y_test.shape
Y_train[741]
Y_train.shape
X_train.shape
X_test = X_test.values.reshape((len(X_test),41,1))
X_train = X_train.values.reshape((len(X_train),41,1))
X_test.shape
X_train.shape
```

 # CNN 1-D Model Design

```
model = Sequential()

model.add(Conv1D(filters = 32, kernel_size = 3,activation ='relu', input_shape = (41,1)))
model.add(Conv1D(filters = 32, kernel_size = 3, activation ='relu'))
model.add(Dropout(0.4))

model.add(Conv1D(filters = 32, kernel_size = 3,activation ='relu'))
model.add(Conv1D(filters = 32, kernel_size = 3, activation ='relu'))
model.add(Dropout(0.4))

model.add(Conv1D(filters = 64, kernel_size = 3, activation ='relu'))
model.add(Conv1D(filters = 64, kernel_size = 3, activation ='relu'))
model.add(Dropout(0.4))


model.add(Flatten())
```

```
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(2, activation = "softmax"))
model.summary()
```

#Model Compilation

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

#Model Fitting

```
history = model.fit(X_train,Y_train, batch_size= 128,
                    epochs = 10, validation_data = (X_test,Y_test))
```

#Accuracy Calculation

```
plt.plot(history.history['accuracy'], 'r')
plt.plot(history.history['val_accuracy'], 'b')
plt.legend({'Train Accuracy': 'r', 'Test Accuracy':'b'})
plt.show()
```

#Results

```
score = model.evaluate(X_test, Y_test, verbose=0)
print('Test accuracy:', score[1])
score = model.evaluate(X_train, Y_train, verbose=0)
print('train accuracy:', score[1])
score = model.evaluate(X_train, Y_train, verbose=0)
print('train accuracy:', score[1])
new = ['CNN 1D',95.74, 96.19]
all_model_result.loc[2] = new
all_model_result
```

Google Notebook Link of the Code:

https://colab.research.google.com/drive/1yy8_ejCk4BRdXDXBSOuvCHni6mLnpYez?usp=sharing

---