Intrusion Detection In Computer Network Using Neural Network With Keras

Ritu Ganeshe, Manish Kumar Ahirwar, Rajeev Pandey

Abstract: In modern society to protect the confidential information flowing over the networks,there is a need of network security. Intrusion detection in computer networks is very important for preventing from the attackers, So for this an efficient detection is needed. Intrusion Detection model which is based on a feature selection and classification is presented by implementing feature selection, then by the use of machine learning techniques, building of the Intrusion detection model to find attacks on system is done and improvement of the intrusion detection is done using the captured data. In this we perform experimental result by using NSL-KDD cup 99 Dataset to train our model based on Neural Network to classify attacks by using Keras on the top of TensorFlow and calculated the accuracy of the classifier model.

Index Terms: Intrusion Detection system, Anomaly detection, keras, tensor flow, neural network.

1 INTRODUCTION

The worldwide internet is growing, network assaultive becomes additional and additional serious,it's not solely a threat to web users, however conjointly a giant challenge to national securities. Intrusion Detection System [1] is really a classification that determines whether or not an information set traditional or abnormal. Securing computer systems has become a frightening work currently with the ascent of the web and therefore the increasing quality of communication protocols. The raging war between the safety offender and data security professionals has become a lot of sturdy than ever. New sophisticated attack ways are being developed by attackers at associate degree horrible rate by taking advantage of the complicated behavior of today's networks. Certainty has reportable new 8064 vulnerabilities within the year 2006 and this figure has been considerably increasing over the past few years. Though proactive suggests that for achieving security have existed for a protracted time, these approaches have goal to forestall and/or observe of solely famous attacks**.** Novel attacks are raising a long-standing downside within the field of data security have received appreciable attention within the recent past

2 LITERATURE REVIEW

Rahul Vigneswaran K, 2018[1]
In this paper,To predict the attacks on Network Intrusion Detection System,the use of DNNs is done.The 0.1 rate of learning with DNNs is applied and is last one000 variety of epochs and for coaching and benchmarking the network KDDCup-'99' dataset has been used.For comparison functions, the coaching is finished on a similar dataset with many alternative classical machine learning algorithms and DNN of layers starting from one to five. The results were compared and terminated that a DNN of three layers has superior performance over all the opposite classical machine learning algorithms.

Yaping Chang, 2017 [2]
In this paper, they developed a new machine learning approach for predicting. Since there were many potential features for network intrusion classification, random forest were used for feature selection based on variable importance score. The performance of the support vector machine which used the 14 selected features on KDD 99 dataset has been evaluated by comparing it with the total(41) features and popular classifiers.

David Ahmad Effendy, Kusrini Kusrini, Sudarmawan Sudarmawan , 2017
Semi supervised machine learning technique will be utilized in intrusion detection, for each tagged and unlabelled information. Within the planned technique they have a tendency to take a little quantity of tagged information to form model and victimisation this model,they have a tendency to show a way to predict the unlabelled traffic.

3 PROBLEM STATEMENT

The rise of the internet services along with the continued growth of access around the world, network traffic security is becoming a major issue in computer network system. Every day the number of attacks are increasing in computer network. For the reason that Intrusion detection in network is very important to detect and prevent intrusions and analyze huge number of network data and classify all of these network data into anomaly and normal data. With the speedy development of information technology inside the past twenty years. Computer networks unit of measurement wide utilized by business, business and various fields of the human life. Therefore, building reliable networks is also a necessary task for IT administrators. On the alternative hand, the speedy development of information technology created several challenges to form reliable networks which will be a very robust task.Several varieties of attacks threatening the availability, integrity and confidentiality of computer networks.

- *Ritu Ganeshe, Dept. of Computer Science & Engineering, UIT RGPV, Bhopal, India. Email: ganesheritu@gmail.com*
- *Manish Kumar Ahirwar, Dept. of Computer Science & Engineering, UIT RGPV, Bhopal, India.Email: ahirwarmanish@gmail.com*
- *Rajeev Pandey, Dept. of Computer Science & Engineering, UIT RGPV, Bhopal, India. Email: rajeev98iet@gmail.com*

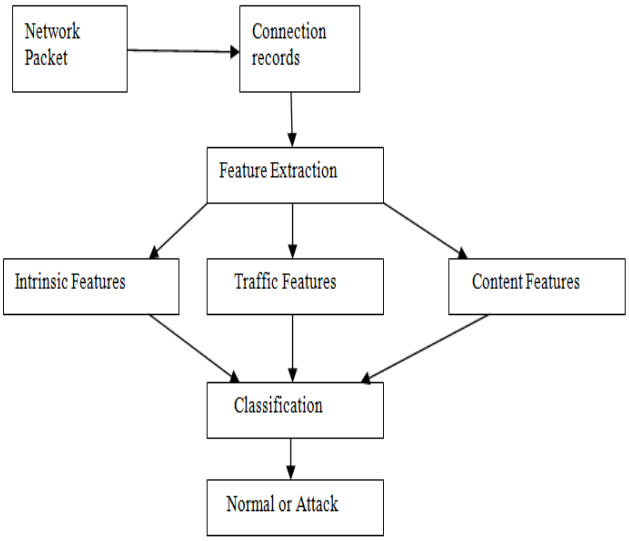

Fig.1. Intrusion Detection System

4 PROPOSED WORK

Network IDS has been proposed which is based on deep learning with keras. Deep learning enhances the detection performance of intrusion from the NSL-KDD dataset. We classify the intrusion dataset into various classes (multi-dimensional classification).We can map packer information inputs to the respected labelled targets, which is happened by various learning rules and decision rules. We can use deep neural network which has a different layers for mapping inputs to outputs, we can use four layer neural network two layer for the input layer and remaining two layers is for the output layers. Each input layers are stored weights which is known as input weights which is updates when there is a loss in the actual or desired outcomes.

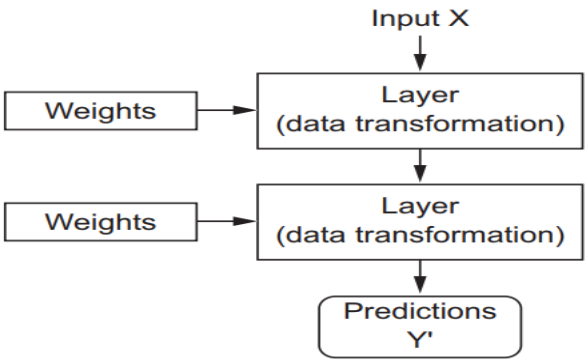

Fig.2 A neural network is parameterized by its weights

To find the output of a neural network we need to processed the inputs from various layers of the neural networks and then we have to calculate the difference between actual output and the desired output and if there is any loss present in the network , based on these error we have to updates the weight of the input hidden layers of the neural networks, the loss function of the network, also called the objective function.

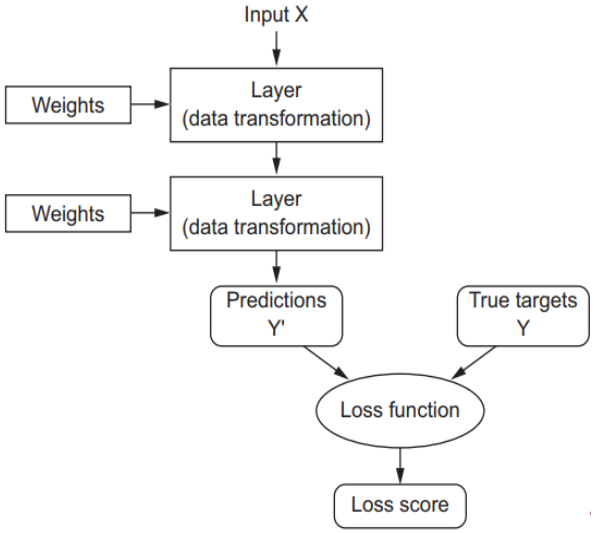

Fig.3. A loss function measures the quality of the network's output.

we have to calculate the difference between actual output and the desired output and if there is any loss present in the network , based on these error we have to updates the weight of the input hidden layers of the neural networks.

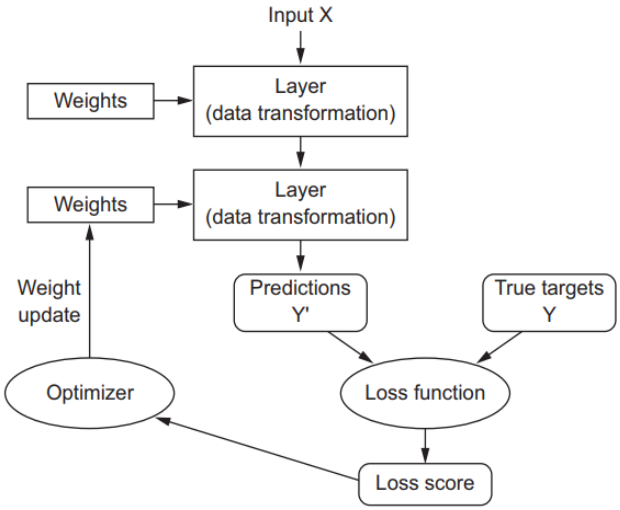

Fig.4. The loss score is used as a feedback signal to adjust the weights.

Initially, all the weight are randomly assigned with the random numbers values , and based on these values we maps the input information to the respected outcomes and calculate the error loss through loss function. Loss function give a loss score values and based on the loss score we have to adjust the input weight on the input layer, the loss score will provide a positive or negative value based on that we add the score to the input weights.

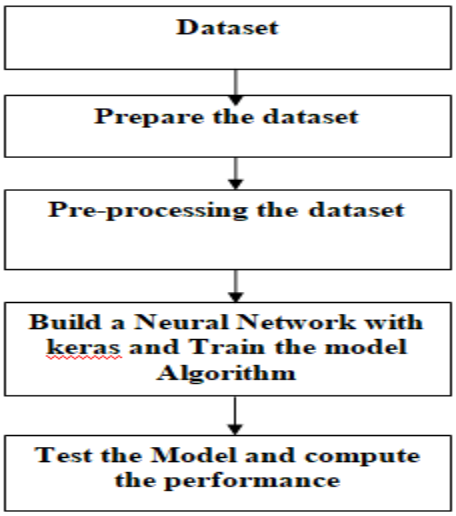

Fig.5. Proposed Flow Diagram

Our Steps or Algorithm Steps will follow:

1. Dataset:: A NSL-KDD dataset is taken for intrusion detection.
2. Data Preparation: The dataset is directly not applied to the intrusion detection models, so we can naming each attributes.
3. Data Pre-processing: Most important phase in detection models is data pre-processing as the data consists of ambiguities we have to remove these errors which needs to be cleaned beforehand.
4. Data Extraction: Through these phase we extract the important attributes for classification.
5. Decision: Based on data extraction and classification models we can take a decision whether the packet is attack or not.

5 EXPERIMENTAL & RESULT ANALYSIS

All the experiment are performed on ubuntu 16 machine on which python-3 in installed, For experiment we can take NSL-KDD dataset which contains millions of records and for creating machine learning model we can use jupyter notebook which is an IDE for python. For classify the attacks we follow the steps

Step-1. Loading a dataset

Step-.2 After loading the dataset, pre-processing is perform over the dataset

Step-3. After preprocessing, split the data into training and testing data

Step-4. Build a neural network model and train the model on training dataset

Step-5. Test the model on testing dataset and compute the performance of the model.

We can load the dataset with the help of pandas library and the dataset comes in a data frames. In these we can load two dataset one for training and another one is for testing which is shown in fig.6

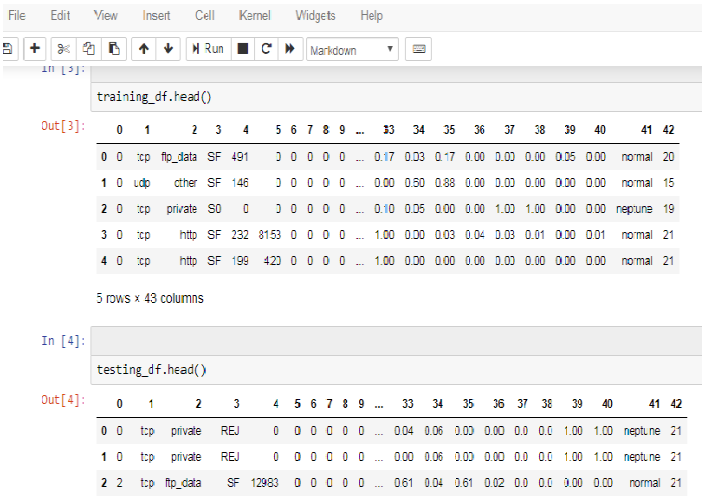

Fig.6. Dataset loaded.

Now we can prepare the features for continuous features we use the MinMaxScaler provided by the scikit-learn library, we only allow the scaler to fit the training set values and then we use it to scale both the training and testing sets. The minmax_scale_values helper function does this task. fig.7 shows the feature preparation.

```
#Helper function to label samples to 5 classes
def label_attack (row):
    if row["outcome"] in dos_attacks:
        return classes[1]
    if row["outcome"] in r2l_attacks:
        return classes[2]
    if row["outcome"] in u2r_attacks:
        return classes[3]
    if row["outcome"] in probe_attacks:
        return classes[4]
    return classes[0]


#We combine the datasets temporarily to do the labeling
test_samples_length = len(testing_df)
df=pd.concat([training_df,testing_df])
df["Class"]=df.apply(label_attack,axis=1)


# The old outcome field is dropped since it was replaced with the Class field
df=df.drop("outcome",axis=1)
df=df.drop("difficulty",axis=1)

# we again split the data into training and test sets.
```

Fig.7. Feature Preparation

Now we can prepare the neural network model in which we implemented a sparse autoencoder with dropout on the inputs, it consists of an input layer of 122 neurons due to the fact that the number of features for each sample is 122 followed by a dropout layer and a hidden layer of 8 neuron units so the hidden representation of the autoencoder has a compression ratio of 122/8 forcing it to learn interesting patterns and relations between the features, finally there is an output layer of 122 units, the activation of both the hidden layer and the output layer is the relu function. The autoencoder was trained to reconstruct its input, in other words it learns the identity function, the model was trained using only the samples labeled “Normal” in the training dataset allowing it to capture

the nature of normal behavior, this was accomplished by training the model to minimize the mean squared error between its output and its input. The regularization constraints enforced over the autoencoder prevent it from simply copying the input to the output and overfitting the data, furthermore the dropout presented on the inputs makes the autoencoder a special case of a denoising autoencoder, this kind of autoencoders is trained to reconstruct the input from a distorted corrupted version of itself, forcing the autoencoder to learn even more properties of the data. The model is trained for 10 epochs using an Adam optimizer with a batch size of 100, furthermore we held out 10% of the normal training samples to validate the model. Fig 8 shows the training steps for neural network.

```
#Buildling and training the model

def getModel():
    inp = Input(shape=(x.shape[1],))
    d1=Dropout(0.5)(inp)
    encoded = Dense(8, activation='relu', activity_regularizer=regularizers.l2(10e-5))(d1)
    decoded = Dense(x.shape[1], activation='relu')(encoded)
    autoencoder = Model(inp, decoded)
    autoencoder.compile(optimizer='adam', loss='mean_squared_error')
    return autoencoder

autoencoder=getModel()
history=autoencoder.fit(x[np.where(y0==0)],x[np.where(y0==0)],
                epochs=10,
                 batch_size=100,
                 shuffle=True,
                 validation_split=0.1
                       )
```

```
Train on 60608 samples, validate on 6735 samples
Epoch 1/10
60608/60608 [==============================] - 1s - loss: 0.0263 - val_loss: 0.0130
Epoch 2/10
60608/60608 [==============================] - 1s - loss: 0.0124 - val_loss: 0.0093
```

Fig. 8. Training of the Model

Prediction

Once the model get trained we can test the model on the testing dataset. The model performs anomaly detection by calculating the reconstruction error of samples, since the model was trained using normal data samples only the reconstruction error of samples that represent attacks should be relatively high compared to the reconstruction error of normal data samples, this intuition allows us to detect attacks by setting a threshold for the reconstruction error, if a data sample has a reconstruction error higher than the preset threshold then the sample is classified as an attack, otherwise it's classified as normal traffic. Fig.9. shows the prediction steps of the model.

[19]:

```
# Helper function that calculates the reconstruction loss of each data sa
def calculate_losses(x,preds):
    losses=np.zeros(len(x))
    for i in range(len(x)):
        losses[i]=((preds[i] - x[i]) ** 2).mean(axis=None)

    return losses

# We set the threshold equal to the training loss of the autoencoder
threshold=history.history["loss"][-1]

testing_set_predictions=autoencoder.predict(x_test)
test_losses=calculate_losses(x_test,testing_set_predictions)
testing_set_predictions=np.zeros(len(test_losses))
testing_set_predictions[np.where(test_losses>threshold)]=1
```

Fig.9. Predicting steps of the model

Performance Measure

We used accuracy, which is derived using confusion matrix.

Table-I: Confusion Matrix

	Classified as Normal	Classified as Attack
Normal	TP	FP
Attack	FN	TN

Where
TN -Instances correctly predicted as non-attacks.
FN - Instances wrongly predicted as non-attacks.
FP -Instances wrongly predicted as attacks.
TP -Instances correctly predicted as attacks.

To evaluate the model we calculate the following performance metrics :

- Accuracy
- Recall
- Precision
- F1

Detection rate for each of the five possible labels which is shown in fig.10.

```
accuracy=accuracy_score(y0_test,testing_set_predictions)
recall=recall_score(y0_test,testing_set_predictions)
precision=precision_score(y0_test,testing_set_predictions)
f1=f1_score(y0_test,testing_set_predictions)
print("Performance over the testing data set \n")
print("Accuracy : {} , Recall : {} , Precision : {} , F1 : {}\n".format(accuracy,recall,pr

#


for class_ in classes:
    print(class_+" Detection Rate : {}".format(len(np.where(np.logical_and(testing_set_pre
```

```
Performance over the testing data set

Accuracy : 0.9032515636783037 , Recall : 0.959167770591444 , Precision : 0.88135471860232

Normal Detection Rate : 0.17064881565396497
Dos Detection Rate : 0.9392352016762703
R2L Detection Rate : 0.9785898855666297
U2R Detection Rate : 0.9701492537313433
Probe Detection Rate : 1.0
```

Fig.10. Performance Measure of the model

Table-II: Accuracy of Models

Models	Accuracy	Precision	Recall	F1-Score
NN with keras	90.32	88.13	95.91	91.86

Table –III: Class labels wise performance measure

Labels	Accuracy
Normal Detection rate	**17.06%**
DOS detection rate	**93.92%**
R2L detection rate	**97.85%**
U2R detection rate	**97.01%**
Probe detection rate	**100%**

6 CONCLUSION

With the advancement in the technology, millions of people are now connected with each other through one or other form of network where they share lots of important data. Hence the need of security to safeguard data integrity and confidentiality is increased rapidly. The largest drawback that was discovered throughout the thesis was finding smart tagged datasets that can be accustomed train the machine learning algorithms. If an honest training dataset is employed to coach a machine learning algorithmic rule, it may be accustomed produce an intrusion detection system that offers acceptable performance out-of-the-box. Plenty depends on the standard of the training dataset. If the training dataset doesn't contain enough samples of the various intrusions, the machine learning algorithmic rule can exhibit an outsized quantity of false positives and false negatives. Classification of Intrusion Detection System (IDS) with NSL-KDD99 dataset applies classification technique based on deep learning with keras.

REFERENCES

[1]. Yaping chang ; wei li ; zhongming yang, "Network intrusion detection based on random forest and support vector machine" in IEEE 2017.

[2]. L.Haripriya, M.A.Jabbar, " Role of Machine Learning in Intrusion Detection System: Review" in Proceedings of the 2nd International conference on Electronics, Communication and Aerospace Technology (ICECA 2018) IEEE.

[3]. Chidananda Murthy P, A S Manjunatha, " Predicting Unlabeled Traffic For Intrusion Detection Using Semi-Supervised Machine Learning" in IEEE 2016.

[4]. Mathew G. Schultz and Eleazar Eskin and Erez Zadok, “Data Mining Methods for Detection Of New Malicious Executables”, Department of Computer Science Columbia University.

[5]. Vipin Das 1, Vijaya Pathak2, Sattvik Sharma3, Sreevathsan4, MVVNS.Srikanth5, Gireesh Kumar T 6, "Network Intrusion Detection System Based On Machine Learning" in International Journal of Computer Science & Information Technology (IJCSIT), Vol 2, No 6, December 2010.

[6]. Syam Akhil Repalle1, Venkata Ratnam Kolluru2, "Intrusion Detection System using AI and Machine Learning Algorithm" in International Research Journal of Engineering and Technology (IRJET), Dec 2017.

[7]. Chanakya G*, Kunal P, Sumedh S, Priyanka W, Mahalle PN, "Network Intrusion Prevention System Using Machine Learning Techniques " in International Journal of Innovative Research in Computer and Communication Engineering, july 2017.

[8]. SAEYS, Y., ABEEL, T., AND PEER, Y. “Robust feature selection using ensemble feature selection techniques”. In Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II (2008), ECML PKDD ’08, pp. 313–325.

[9]. MACQUEEN, J. B. “Some methods for classification and analysis of multivariate observations”. In Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability (1967), L. M. L. Cam and J. Neyman, Eds., vol. 1, University of California Press, pp. 281–297

[10]. Seongjun Shin, S. L. (2013). “Advanced probabilistic approach for network intrusion forecasting and detection”. Expert Systems with Applications,ELSEVIER .

[11]. Ravi Ranjan, G. S. (2014). “A New Clustering Approach For Anamoly Intrusion Detection”. International Journal of Data Mining & Knowledge

Management Process (IJDKP) .
[12]. Lee, W., Stolfo, S. J., And Mok, K. W. “A data mining framework for building intrusion detection models”. IEEE Symposium on Security and Privacy 0 (1999).
[13]. HALL, M. A. “Correlation-based feature selection for discrete and numeric class machine learning”. In International Conference on Machine Learning (2000), pp. 359–366
[14]. Prabhjeet Kaur, A. K. (2012). “MADAM ID for Intrusion Detection Using Data Mining”.International Journal of Research in IT & Management,IJRIM
[15]. A.M.Chandrashekhar, K. (2013) “Fortification of hybrid intrusion detection system using variants of neural networks & support vector machines”. International Journal of Network Security & Its Applications (IJNSA)