Intrusion Detection System Using PCA and Deep Neural Network in Cloud

20BCY10211

July 2022

1 Introduction

2 Related works

Chiba et al. [1] discussed an intrusion detection system based on neural networks and machine learning algorithms for cloud systems. This paper discussed the use of an improved genetic algorithm (IGA) and simulated annealing algorithm (SAA). IGA and SAA are used to find the most optimal parameters like feature selection, data normalization, learning rate etc which are needed in the construction of the deep neural network (DNN). The simulation of the model was performed on CloudSim 4.0 simulator and the three benchmark IDS datasets namely CICIDS2017, NSL-KDD version 2015 and CIDDS-001 were put to use. Yin et al. [10] discussed the recurrent neural network based deep learning solution for intrusion detection (RNN-IDS) following the evaluation of model's performance in binary classification and multiclass classsification. The model performed better in comparison to other IDS's in both binary and multiclass classification.

Wen et al. [9] studied the internal and external attacks in a network and discussed a model on intrusion detection system based on a neural network algorithm called backpropagation neural network (BPNN) alongwith an optimized artificial bee colony algorithm. BP-NN was observed to be very adaptable with the surrounding alongwith its autonomous learning capability to issues even with unclear background. Lu et al. [3]research on developing an IDS for cloud using backpropagation neural network (BPNN) and adaptive clonal genetic algorithm (ACGA). The model is called (BPNN-ACGA). The model was very adaptive and continuously optimized the weights of the features leading to increased accuracy and security. This model outperformed other models like SA-BPNN and GA-BPNN in terms of accuracy and F-score.

Liu et al. [2] discussed fuzzy rough set based IDS. The model was based on fuzzy rough set feature selection and GA-GOGMM-based pattern learning. GA-GOGMM is a greedy algorithm-based global optimum Gaussian mixture model (GMM) clustering approach that extracts the underlying structure of network instances to provide highly discernible and stable intrusion pattern libraries for further network intrusion detection (NID). Samriya et al. [5] used fuzzy based artificial neural networks for IDS's in cloud computing. The model was optimized using a spider-monkey optimization algorithm which helped in effective clustering of anomalies in a network intrusion.

Selvakumar et al. [6] discussed an intrusion detection model with an aim to improve the processing speed of the IDS. The experimental results of the paper revealed that only ten features are enough to detect intrusions. The proposed work deployed a filter and wrapper method with a firefly algorithm in the wrapper to select the ten features. These were then trained and tested with C.5 and Naive Bayes algorithm with KDD CUP 99 dataset. Waskle et al. [8] discussed the principal component analysis (PCA) and random forest classifier for IDS. The PCA was used to reduce dimensionality of the dataset and random forest was used for classification which improved the accuracy of the model.

3 Methodology

3.1 Background

3.1.1 Intrusion Detection System

The term "intrusion" refers to entering a system without authorization and tampering with the data that is already there . Any system's hardware may also be harmed by this intrusion. The intrusion has evolved into a crucial phrase to safeguard the system against. With the help of the IDS, this infiltration inside of any system can be managed or monitored. Although several intrusion detection systems have been employed in the past, accuracy issues have ultimately been found with every technique. For the purpose of assessing the system's accuracy, two terms—detection rate and false alarm rate—are analysed. These two clauses should be written in a way that reduces the likelihood of false alarms and increases the system's detection rate.

The two different types of IDS that it can be used for are as follows:

- **Network Intrusion Detection Systems (NIDS):**This system analyses network traffic and any intrusions that may have occurred over it.
- **Host-based Intrusion Detection Systems (HIDS):** In this case, the system monitors network access to system files.

An IDS subset is also present. The most popular variations rely on anomaly and signature detection.

- **Signature-based:** In this, the system discovered several distinct patterns that malware employs. Signatures are the names given to these found patterns. This works well in identifying existing assaults, but it is ineffective in identifying brand-new attacks.

- **Anomaly-based:** This method was created specifically for the detection of unidentified attacks.

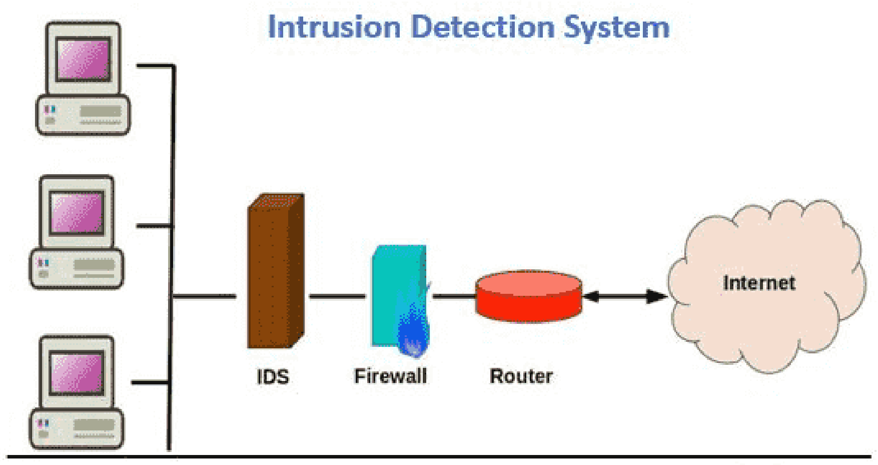


Figure 1: Working of IDS

3.1.2 Principal Component Analysis

PCA is a technique that is used for redusing the dimensionality of the dataset containing large number of dimenions. The principal component analysis is one of the most efficient and accurate methods for reducing data dimensions, and it produces the desired results. This method condenses the attributes of a given dataset into a desired number of attributes known as principal components. This method uses all of the input as the dataset, which has a large number of attributes and a large dimension. It shrinks the dataset by putting all of the data points on the same axis. The data points are shifted along a single axis, and the principal components are performed.
The PCA is performed in following steps:

1. Input all the dimensions 'd' of a dataset.
2. Calculate mean factor for each of the dimension.
3. Calculate the covariance matrix for the dataset.
4. Calculate the eigen vectors and eigen values.
5. Sort the eigenvalue in decreasing order and select n eigenvector with the highest eigenvalues to get a matrix of d*n= M.
6. Form a new sample space suing this M.
7. The obtained sample spaces are the principal components.

3.1.3 Deep Neural Network

A deep neural network (DNN) is an ANN with multiple hidden layers between the input and output layers. Similar to shallow ANNs, DNNs can model complex non-linear relationships.

The main purpose of a neural network is to receive a set of inputs, perform progressively complex calculations on them, and give output to solve real world problems like classification. We restrict ourselves to feed forward neural networks.

We have an input, an output, and a flow of sequential data in a deep network.[**?**]

3.2 Problem statement

An IDS should be able to identify all abnormal patterns and traffic using monitoring, detecting and responding to unauthorized activities within the system. However, regarding its huge and unbalanced datasets, IDS encounters total data processing problem.[7] The dataset used for IDS are huge and poses huge class imbalance. The accuracy of the IDS is compromised when the dataset is huge and there are alot of features. The experimental results of a research showed that only 10 features are enough to detect intrusion. [6]. Most of the IDS research work revolves around using machine learning models for detection of intrusions without proper feature extractions. But they are not highly accurate and produce false alarm rates.

Many researchers have used artificial intelligence but only a few have applied PCA to it for dimension reduction [6, 8, 4].

Using all of the features to detect intrusions poses an increase in time in detection , slow processing speeds and increasing the chances of decreased accuracy because of the use of alot of features, confusing the model. To overcome these limitations, a novel approach to detect intrusions using PCA and BPNN algorithm has been presented in the paper. PCA reduces the dimensionality of the dataset making the BPNN algorithm faster and accurate.

3.3 Proposed solution

This section describes in detail the proposed IDS, and its model.

3.3.1 Approach of the Proposed System

This section discusses the approach for the proposed intrusion detection system for cloud. The model implies the use of dimension reduction technique, principal component analysis(PCA) and building of the model using Deep Neural Networks(DNN).

PCA is used for the reduction of dimension of the dataset. This technique filters the important features from the less important. 10 features have been selected out of the 42 features as only 10 features are enough to detect intrusion [6]. This helps in improving the quality of the data, decreased execution time, increased execution speed, and accuracy. Table 1 below contains the list of all the features of the dataset used.

The DNN contains one input layer, three hidden layers, and one output layer. The number of nodes in the input layer is equal to the number of features selected after applying PCA to the dataset i.e. 10. The hidden layers have 144, 72 and 36 nodes respectively. The output layer corresponds to the number of unique classes, in this case 18.

The model has been trained with KDD CUP1999 dataset. The dataset has 41 independent variables and 1 dependent variable. The dataset poses huge class imbalance that has been reduced through sampling. It initially contained 38 unique classes which was reduced to 18 during sampling. Table 2 below contains the list of all the classes of the KDD Cup1999 dataset.

No.	**Feature**	**No.**	**Feature**
1	Duration	22	Count
2	Prototype	23	Srv count
3	Service	24	Serror rate
4	Flag	25	Srv serror rate
5	Source bytes	26	Rerror rate
6	Destination bytes	27	Srv error rate
7	Land	28	Same src rate
8	Wrong fragment	29	Diff src rate
9	Urgent	30	Srv diff host rate
10	Hot	31	Dst host count
11	No. failed logins	32	Dst host srv count
12	Logged in	33	Dst host same src rate
13	No. compromised	34	Dst host diff src rate
14	Root shell	35	Dst host same src port rate
15	Lsu attempted	36	Dst host srv diff host rate
16	Num root	37	Dst host serror rate
17	Num file creations	38	Dst host srv rerror rate
18	Num shells	39	Dst host rerror rate
19	Num access files	40	Dst host srv rerror rate
20	Num outbound cmds	41	Is guest login
21	Is host login	42	Class label

Table 1: List of features of KDD Cup1999 dataset

No.	Feature	No.	Feature
1	smurf	20	normal
2	neptune	21	smpgetattack
3	mailbomb	22	guess$_passwd$
4	snmpgguess	23	satan
5	warezmaster	24	back
6	mscan	25	apache2
7	processtable	26	saint
8	portsweep	27	ipsweep
9	httptunnel	28	pod
10	nmap	29	buffer$_overflow$
11	multihop	30	named
12	sendmail	31	ps
13	rootkit	32	xterm
14	teardrop	33	xlock
15	land	34	xsnoop
16	ftp$_write$	35	load$_module$
17	perl	36	udpstorm
18	worm	37	phf
19	sqlattack	38	imap

Table 2: List of classes of KDD Cup1999 dataset

3.4 Architecture of the proposed system

This section deals with the explanation of various modules of the system. The proposed PCA_DNN contains three modules which are as follows :

1. Data Preprocessing Module
2. Feature Selection Module
3. DNN Detection Module

The architecture of the model has been illustrated in figure 2 below.

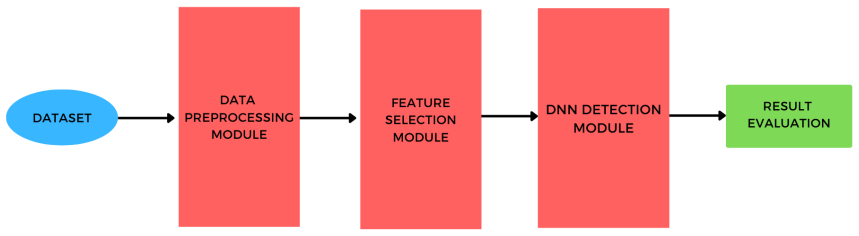


Figure 2: Architecture of PCA_DNN

- **Data preprocessing module:** The data preprocessing modules consists of 3 operations:

  1. Data sampling
  2. Categorical encoding
  3. Data normalization

  In data sampling, a certain number of sample from each class is taken. The objective is to estimate a population parameter. With the use pf data sampling, class imbalance is removed. Categorical encoding is a technique to convert all the textual data to numeric data of the features. It makes handling of the data easier. Data normalization translates the data into a range of -1 to 1. This process is necessary to improve the accuracy and integrity of the data.

- **Feature selection module:** In this module, the dependent labels are shrinked using principal component analysis technique. For the proposed model only 10 out of the 41 components have been selected. This modules helps in reducing the dimension of the data and improves execution speed, time and accuracy.

- **DNN detection module:** In this module the data is splitted into train test splits. The build model consists of 1 input layer, 3 hidden layers and 1 output layer. The input layer contains 10 nodes, hidden layers contain 144, 72 and 36 nodes respectively and the output layer contains 18 nodes. The model is trained and compiled with the fed data further evaluation is done. Table 3 later illustrates the configuration of the PCA_DNN model.

3.5 Implementation

This section focusses on the implementation of the proposed PCA_DNN model.

3.5.1 Hardware / Software requirements

The requirements to build and evaluate the model are as follows:

- **Hardware requirements:** Min. 4GB ram, 128GB ssd, processor for eg. Intel core i3, i5 or apple silicon M1.
- **Software requirements:** Python 3.6+, any IDE for eg. Jupyter Notebook, VS Code etc, windows/macOS operating system etc.
- **Python packages / libraries:** Numpy, sklearn, pandas, keras, tensorflow etc.
- **Dataset:**KDD Cup1999

3.5.2 Algorithm

To build and deploy the model, following steps have been followed:

Step 1: **Data loading and exploration:** The dataset is loaded and explored on various parameters.

Step 2: **Data preprocessing:** In this step the dataset is preprocessed. This step includes 3 additional steps :

1. **Data sampling** To remove the class imbalance that the dataset poses, data sampling is done. In this, 1000 samples or less from each class is filtered. Additionally the classes which have less than 50 samples are also removed. So the dataset contains classes with samples ranging from 50 to 1000. This step ensures data integrity and helps in improving the accuracy for each class.

   After sampling the dataset contains 19 unique classes are present , which was initially 38.

2. **Categorical encoding:** Categorical encoding refers to changing all the non-numeric data to numeric data. This step is carried out as the machine doesn't understand non numeric data. Moreover it makes handling of the data easier and also training the model.

   For eg. if a column contains "yes" / "no" type variables, it can be encoded to 0 and 1 respectively. Same method has been applied on the dataset to convert the datatype of all the feature to 'float' or 'int'.

3. **Data normalization** After categorical encoding, the dataset is normalized. Normalization means to convert the data of the features within a certain range , here -1 to 1. This is a crucial step in preprocessing which ensures general distribution and ratios in the dataset, so that the model treats each feature attribute equally and not rely on just few high value features for detection.

Step 3: **Feature selection:** The KDD Cup1999 dataset contains 41 dependent and 1 dependent variable or features. Using all of these features for training the model leads to overfitting of the model. The model captures noice leading to poor accuracy , detection rate and inefficient model. Feeding alot of features to the model also leads to increase in execution time.

To overcome these problems and reduce the dimensionality of the data, PCA technique has been applied. The PCA helps filter 10 out of 41 features or components. These 10 features are the important features. Including this step ensures little to no noise in the data with increased accuracy and execution speed.

Step 4: **Train-test split** In this step the data is splitted between training and testing set. 70% of the data is used for training purpose and 30% is used for testing purpose.

Step 5: **DNN Model building and compilation:** After train test split, the model is built. It is fitted with the training data and validated with the testing data. The configuration of the model is illustrated in the table 3 later.

Step 6: **Model evaluation:** The build PCA_DNN model is evaluated after bilding and testing. Various metrics like accuracy, precision etc are calculated and compared with other models/algorithms in use.

Layer type	**No. of nodes**	**Activation function**
Input Layer	10	Rectified linear activation unit
Hidden layer 1	144	Rectified linear activation unit
Hidden layer 2	72	Rectified linear activation unit
Hidden layer 3	36	Rectified linear activation unit
Output layer	18	Softmax

Table 3: DNN model configuration

3.5.3 Flowchart of DNN_PCA

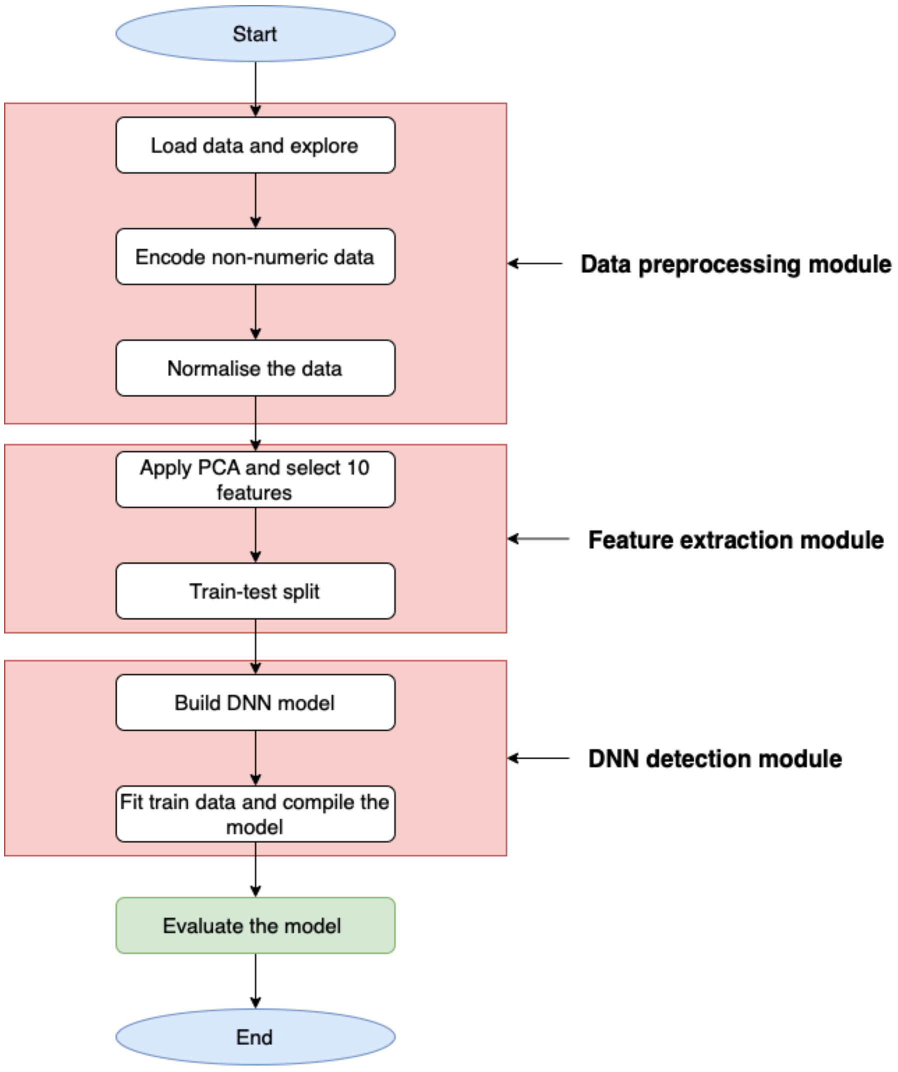


Figure 3: Flowchart of the proposed model DNN_IDS

3.5.4 Code of the model

START
#importing libraries and functions

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import tensorflow as tf
from keras import Sequential
from keras.layers import Dense
```

#loading the dataset
```
df=pd.read_csv(”KDDCUP1999.csv”)
```

data exploration
```
print(”Total number of attack of each type: ”)
print(df['label'].value_counts())
print(”Number of unique labels/attacks: ”,
df['label'].unique().size)
```

#data sampling
```
df=df.groupby('label').filter(lambda x : len(x)¿50)
df['label'].value_counts()
sample 1000 or less samples from dataset
def nsample(x,n):
if len(x) ¡= n:
return x
else:
return x.sample(n=n)
df=df.groupby(df['label'])
n_max = 1000
df=df.apply(lambda x: nsample(x,
n_max)).reset_index(drop=True)
```

# character encoding
```
# label ”label” , ”protocol_type”, and ”service” with numeric values
lab_map=dict(enumerate(df['label'].unique()))
pt_map=dict(enumerate(df['protocol_type'].unique()))
ser_map=dict(enumerate(df['service'].unique()))
flag_map=dict(enumerate(df['flag'].unique()))
#flip key into value for easier access
```

```
lab_map = dict([(value, key) for key, value in lab_map.items()])
pt_map = dict([(value, key) for key, value in pt_map.items()])
ser_map = dict([(value, key) for key, value in ser_map.items()])
flag_map = dict([(value, key) for key, value in flag_map.items()])
#map them to integers
df['label']=df['label'].replace(lab_map)
df['protocol_type']=df['protocol_type'].replace(pt_map)
df['service']=df['service'].replace(ser_map)
df['flag']=df['flag'].replace(flag_map)

#normalize the data
#standardizing the data
scaler=StandardScaler()
#select all columns except label
X=df.iloc[:, :42]
y=df.iloc[:,42]
X_scaled=scaler.fit_transform(X)

#applying pca for dimension reduction
pca=PCA(n_components=10)
X_pca=pca.fit_transform(X_scaled)

#train-test split X_train_pca , X_test_pca, y_train_pca, y_test_pca= train_test_split(X_pca,
y, random_state=1, test_size=0.3)

#DNN model building and testing
# define classification model
def classification_model():
create model
model = Sequential()
model.add(Dense(10, activation='relu',
input_shape=(10,)))
model.add(Dense(144, activation='relu'))
model.add(Dense(72, activation='relu'))
model.add(Dense(36, activation='relu'))
model.add(Dense(len(lab_map), activation='softmax'))
# compile model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', met-
rics=['accuracy'])
return model
model = classification_model()
# fit the model
model.fit(X_train_pca, y_train_pca, validation_data=(X_test_pca, y_test_pca), epochs=1000,
verbose=2) evaluate the model scores = model.evaluate(X_test_pca, y_test_pca,
verbose=0)
```

END

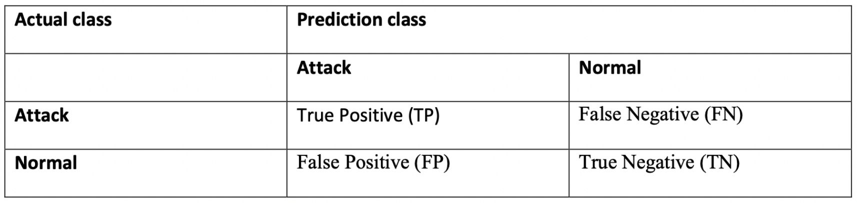
4 Experiment and results

The proposed model's experiment has been done with KDD Cup1999 dataset. The following configurations are used for our performance analysis:

- **Hardware requirements:** *GB ram, 256GB ssd, apple silicon M1 processor.
- **Software requirements:** Python 3.8, VS Code, macOS Monterey operating system
- **Python packages / libraries:** Numpy, sklearn, pandas, keras, tensorflow etc.
- **Dataset:**KDD Cup1999

4.1 Performance metrics

Consider the ability of IDS to make correct predictions as a measure of its effectiveness. There are four prospect outputs that are illustrated in the figure below as the confusion matrix based on comparisons between the results that predict via intrusion detection system and the true nature of the event. True negative (TN) indicates correct prediction of normal behaviour, true positive (TP) indicates correct prediction of attack behaviour, false positive (FP) indicates incorrect prediction of normal behaviour as attack, and false negative (FN) indicates incorrect prediction of attack behaviour as normal.Both (TN) and (TP) are regarded as guides for the proper operation of the IDS. Furthermore, FN and FP rates reduce the effectiveness of IDS, where FP reduces the system's detection capability and FN makes the system vulnerable to intrusion [18]. As a result, for an IDS to be effective, FP and FN rates should be minimised while TP and TN rates should be maximised.



Actual class	**Prediction class**	
	Attack	**Normal**
Attack	True Positive (TP)	False Negative (FN)
Normal	False Positive (FP)	True Negative (TN)

Figure 4: Confusion matrix

Based on the value of the confusion matrix we evaluate the following values:

- Accuracy(ACC) =
$$\frac{\text{TP + TN}}{\text{TP + TN + FP + FN}}$$
- Detection rate/ Recall =
$$\frac{\text{TP}}{\text{TP + FN}}$$
- Precision =
$$\frac{\text{TP}}{\text{TP +FP}}$$
- True Negative Rate (TNR) =
$$\frac{\text{TN}}{\text{TN + FP}}$$
- False Negative Rate (TNR) =
$$\frac{\text{FN}}{\text{FN + TN}}$$

4.2 Evaluation

KDD Cup1999 dataset has been used for training the model. The dataset undergoes character encoding and then it is normalized. After this PCA rechnique is applied to select 10 features out of the 41 features and train-test split is done. The DNN model is built and training data is fed to the model. The confusion matrix formed by the model illustrated in figure 5 below.

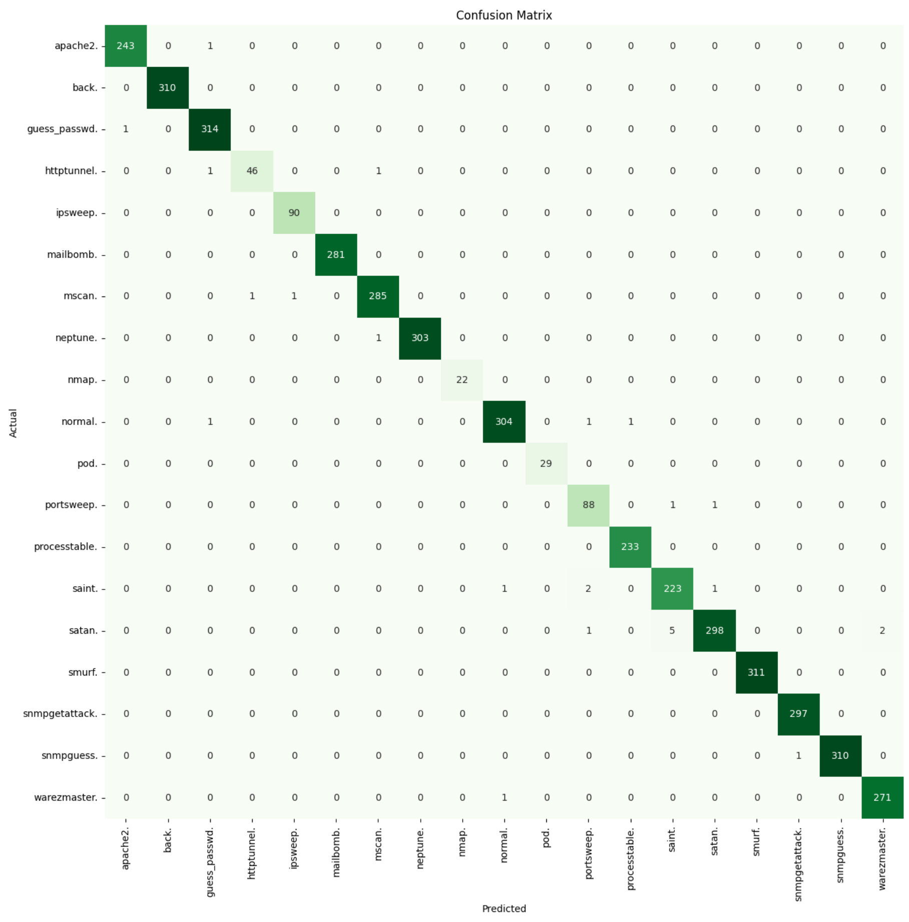


Figure 5: Confusion matrix of PCA_DNN

The performance metrics of the model is given in table 4 below .

Performance metric	**Score**
Accuracy	99.95%
Precision	99.18%
Detection rate	99.56%
Missing rate	0.44%
False alarm rate	0.02%
True negative rate	99.98%
F-score	0.99%

Table 4: Performance of proposed model PCA_DNN

The model's accuracy has been compared with other models / algorithms whose chart is given below in figure 6.

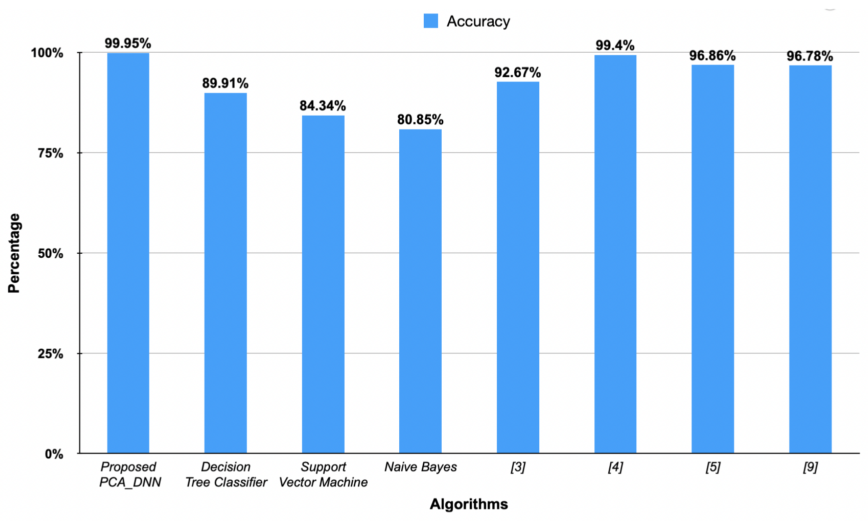


Figure 6: Comparison of "Accuracy" of proposed PCA_DNN and other works

The model's false alarm rate has been compared with other models / algorithms whose chart is given below in figure 7.

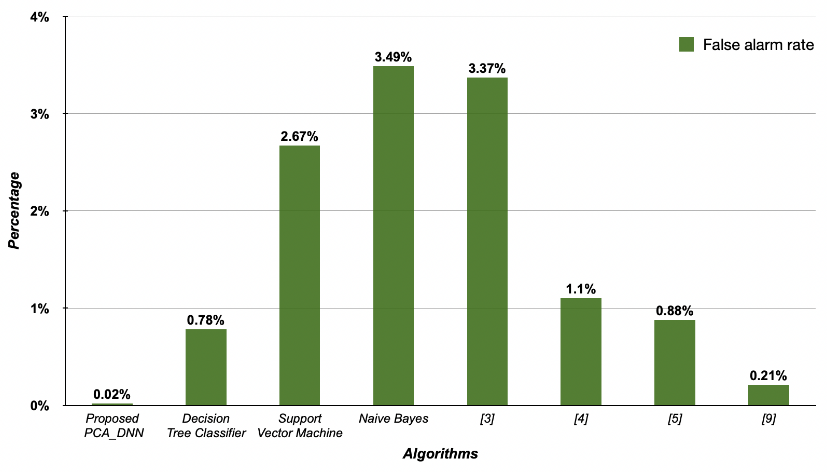


Figure 7: Comparison of "False alarm rate" of proposed PCA_DNN and other works

5 Conclusion and Future work

References

[1] Z. Chiba, N. Abghour, K. Moussaid, M. Rida, et al. Intelligent approach to build a deep neural network based ids for cloud environment using combination of machine learning algorithms. *computers security*, 86:291–317, 2019.

[2] J. Liu, W. Zhang, Z. Tang, Y. Xie, T. Ma, J. Zhang, G. Zhang, and J. P. Niyoyita. Adaptive intrusion detection via ga-gogmm-based pattern learning with fuzzy rough set-based attribute selection. *Expert Systems with Applications*, 139:112845, 2020.

[3] Y. Lu, M. Liu, J. Zhou, and Z. Li. Intrusion detection method based on adaptive clonal genetic algorithm and backpropagation neural network. *Security and Communication Networks*, 2021, 2021.

[4] R. V. Mendonça, A. A. Teodoro, R. L. Rosa, M. Saadi, D. C. Melgarejo, P. H. Nardelli, and D. Z. Rodríguez. Intrusion detection system based on fast hierarchical deep convolutional neural network. *IEEE Access*, 9:61024–61034, 2021.

[5] J. K. Samriya and N. Kumar. A novel intrusion detection system using hybrid clustering-optimization approach in cloud computing. *Materials Today: Proceedings*, 2020.

[6] B. Selvakumar and K. Muneeswaran. Firefly algorithm based feature selection for network intrusion detection. *Computers & Security*, 81:148–155, 2019.

[7] R. Singh, H. Kumar, and R. Singla. An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Systems with Applications*, 42(22):8609–8624, 2015.

[8] S. Waskle, L. Parashar, and U. Singh. Intrusion detection system using pca with random forest approach. In *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 803–808. IEEE, 2020.

[9] L. Wen. Cloud computing intrusion detection technology based on bp-nn. *Wireless Personal Communications*, pages 1–18, 2021.

[10] C. Yin, Y. Zhu, J. Fei, and X. He. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, 5:21954–21961, 2017.