Intrusion Detection in KDD 99 Dataset

Abhishek Kumar (18111002), Aneet Kumar Dutta (18111401),
Dixit Kumar (18111405), Komal Kalra (18111032),
Nitin Vivek Bharti (18111048), Riya James (18111054)

November 2018

1 Introduction

KDD 99 dataset is used to develop an Intrusion Detection System. This Intrusion Detection System is used for detecting intrusions in network by analyzing the network traffic. The training data consists of 5million connection records where connection is defined as a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol.Each connection is labeled as normal or attack.

2 Dataset

The dataset consists of 41 features which are defined for the connection records. The training data consists of 24 different attacks which falls to one of this 4 categories:
DOS: denial-of-service, e.g. syn flood;
R2L: unauthorized access from a remote machine, e.g. guessing password;
U2R: unauthorized access to local superuser (root) privileges, e.g., various “buffer overflow” attacks;
probing: surveillance and other probing, e.g., port scanning.

The 22 different attacks are:
back dos, buffer_overflow u2r, ftp_write r2l, guess_passwd r2l, imap r2l, ipsweep probe, land dos, loadmodule u2r, multihop r2l, neptune dos, nmap probe, perl u2r, phf r2l, pod dos, portsweep probe, rootkit u2r, satan probe, smurf dos, spy r2l, teardrop dos, warezclient r2l, warezmaster r2l

The test data consists of 14 additional attack which is not present in the training data.

3 Modelling

DNN Model

We have used DNN (Deep Neural Network Model) to make this Classifications. Reason for choosing DNN was that this can be trained in online fashion using adam batch gradient descent and also it can easily learn the underlying features that separates each attack from other and these feature can also represent if a new attack is a combination of some attacks or not (i.e we are assuming that the output of a dense layer at certain depth represents a feature set that defines a attack combination) So to model such a neural network assumption we require a single neural network that work on these new feature set and gives three levels of output y1,y2,y3.

Where, y1 is binary (attack, normal), y2 is 5 categories and y3 is 23 categories. and all of these outputs are based on a final feature set obtain at certain depth of nueral network.

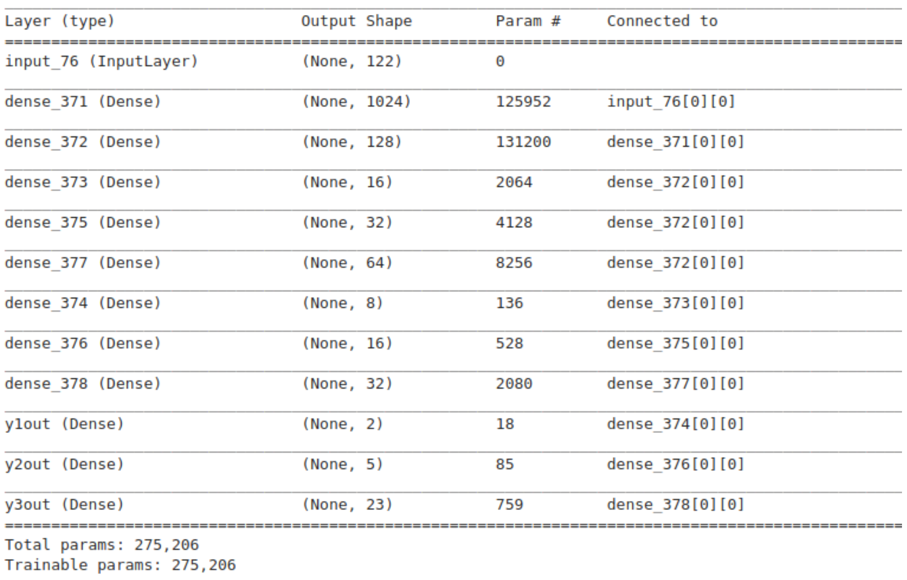
```
____________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
============================================================================================
input_76 (InputLayer)           (None, 122)          0
____________________________________________________________________________________________
dense_371 (Dense)               (None, 1024)         125952      input_76[0][0]
____________________________________________________________________________________________
dense_372 (Dense)               (None, 128)          131200      dense_371[0][0]
____________________________________________________________________________________________
dense_373 (Dense)               (None, 16)           2064        dense_372[0][0]
____________________________________________________________________________________________
dense_375 (Dense)               (None, 32)           4128        dense_372[0][0]
____________________________________________________________________________________________
dense_377 (Dense)               (None, 64)           8256        dense_372[0][0]
____________________________________________________________________________________________
dense_374 (Dense)               (None, 8)            136         dense_373[0][0]
____________________________________________________________________________________________
dense_376 (Dense)               (None, 16)           528         dense_375[0][0]
____________________________________________________________________________________________
dense_378 (Dense)               (None, 32)           2080        dense_377[0][0]
____________________________________________________________________________________________
y1out (Dense)                   (None, 2)            18          dense_374[0][0]
____________________________________________________________________________________________
y2out (Dense)                   (None, 5)            85          dense_376[0][0]
____________________________________________________________________________________________
y3out (Dense)                   (None, 23)           759         dense_378[0][0]
============================================================================================
Total params: 275,206
Trainable params: 275,206
```

Figure 1: Caption

Decision Tree Model

We have also done decision tree modelling since intrusion detection requires a real time analysis and deep learning models can be a bit slow as compared to normal Machine Learning Models. Decision Tree is one of the fastest models used for classification purpose so we tried to incorporate that here but one disadvantage is that they might not be able to predict new type of attack as a combination like DNNs.

Feature Modelling

There were binary features which were left as it is. some continuous features left as it is, categorical features were converted into one-hot vectors so that they can be given as input to DNN model.

Class Imbalance problem

The data set was highly skewed as the number of some attack categories were really small.

neptune.	107201
normal.	97278
back.	2203
satan.	1589
ipsweep.	1247
portsweep.	1040
warezclient.	1020
teardrop.	979
pod.	264
nmap.	231
guess_passwd.	53
buffer_overflow.	30
land.	21
warezmaster.	20
imap.	12
rootkit.	10
loadmodule.	9
ftp_write.	8
multihop.	7
phf.	4
perl.	3
spy.	2

This posed a greater challenge since the class imbalance could lead any model towards wrong classification. So we used oversampling methods on data set to make number of tuples of each class equal which had a chance of over fitting the network bu since we did not have enough data. We can afford to do this.

Oversampling : It was done by simply repeating the data points up till we reach the desired maximum limit.

Training the DNN

Splitting the Data

The data was split in two parts train and test data sets with a ratio of 0.25 by randomly picking points from whole data set with out replacement and made sure that training data consisted every type of category.

Mini Batch Training

Training of DNN was done in batch-wise operation using Adam Gradient Descent optimization method. Since the size of dataset was so large that it could not be loaded all at a time. So we streamed the data in batches from the files and trained on that batch after oversampling it with minor class data points.

Loss function

Loss function over all three levels of output used is Categorical_crossentropy Loss with equal weights to all three losses plus l2 regularization loss on the training parameters.

4 Results

As observed Random Forest Classifier based on Decision Tree over performed Deep Neural Networks in this case since the data is highly skewed information given by features can be captured much better in decision trees. But DNN have much higher potential like classification of unseen data. If data was not so much skewed it would have been much favourable to do DNN as compared to Decision Tree Modelling.

Model	Train Accuracy	Test Accuracy	Avg Recall	Avg Precision	Avg F_Score
DNN	0.9806	0.9792349	0.050232	0.000089	0.000177
D Tree	0.9999892	0.99978638	0.834030	0.804184	0.806702

Conclusion

We conclude by stating that decision trees can be very power full classifier but may over fit since we don't know how it will work for unknown attacks plus its real time prediction so is preferable. whereas DNN can be very slow and hard to model if data is this skewed and even with oversampling it tends to under fit.

Although DNN has potential (if trained for a long time wiht a huge network) to learn and classify new types of attacks as well.