**NETWORK INTRUSION DETECTION SYSTEM**

**A Report submitted in partial fulfilment of the requirements for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**

**MALINA MEHER SATYA SWAROOP, 1210316426**

**V.R. KAUSHIK, 1210316450**

**J. RAHUL VARMA, 1210316419**

**B. ROHIT, 1210316405**

Under the esteemed guidance of

**P.V. NAGESWARA RAO, Professor, GIT**



**DEPARTMENT OFCOMPUTER SCIENCE & ENGINEERING**

**GITAM**

**(Deemed to be University)**

**VISAKHAPATNAM**

**JUNE, 2019**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM INSTITUTE OF TECHNOLOGY**

**GITAM**

**(Deemed to be University)**



We, hereby declare that the project entitled “NETWORK INTRUSION DETECTION SYSTEM” is an original work done by M.SWAROOP(1210316426),V.R.KAUSHIK(1210316458),B.ROHIT(1210316405), J.RAHUL VARMA(1210316419) in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering.

The work has not been submitted to any other college or University for the award of any degree or diploma.

**Project Guide: Prof. P.V. Nageswara Rao, GIT**

**Signature:**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| ABSTRACT | 4 |
| PROBLEM STATEMENT | 5 |
| INTRODUCTION | 6 |
| DATASET | 8 |
| METHODOLOGY   * RANDOM FORESTS * NEURAL NETWORKS * XGBOOST | 10  13  14  15 |
| FLOWCHART | 16 |
| TOOLS | 17 |
| CONCLUSION | 22 |

**ABSTRACT**

Today, intrusion detection is one of the major concerns in the task of network administration and security. There is a need to safeguard the networks from known vulnerabilities and at the same time take steps to detect new and unseen, but possible, system abuses by developing more reliable and efficient intrusion detection systems.

An intrusion detection system (IDS) is a system that monitors network traffic for doubtful activity and matters alert when such activity is exposed. While anomaly discovery and reportage are the primary functions, some interruption detection systems are capable of taking actions when malicious activity or irregular traffic is detected, including blocking traffic sent from suspicious IP addresses. An IDS is a combination of hardware and software that detects intrusions in the network. It is used to track, identify and detect the intruders.

NETWORK-BASED IDS: Its data is mainly collected from the network generic stream going through network segments, such as: Internet packets.

MISUSE DETECTION: is the ability to identify intrusions based on a known pattern for the malicious activity.

The IDS we are going to develop is a **network-based IDS** and it is going to detect any **misuse** of the network resources (misuse detection) i.e. it detects malicious packets in a network.

An Artificial Neural Network based NIDS and Data Mining based NIDS will be developed so that the accuracy at which the intrusions are detected increases. In this network intrusion detection system, by using the concept of ensemble binary classification and multi-boosting simultaneously it efficiently detects the attack with the low false alarm rate and even at high network traffic. With the use of the Dynamic multi-boosting and the database storage the time taken to detect the attacks has been decreased efficiently. By combining the concepts of the **Artificial Neural network and the Data mining technique of classification** the drawbacks of the later is overcome.

The network intrusion detection system involves the capturing of the packet in real time and those packets that have been received are classified using our proposed model.

**PROBLEM STATEMENT**

Today, intrusion detection is one of the major concerns in the task of network administration and security. There is a need to safeguard the networks from known vulnerabilities and at the same time take steps to detect new and unseen, but possible, system abuses by developing more reliable and efficient intrusion detection systems. The system must be accurate in detecting attacks with the minimum number of false alarms (wrong detections).

**INTRODUCTION**

Intrusion is a set of actions that attempts to compromise the integrity, confidentiality, or availability of any resource on a computing platform. An intrusion detection system (IDS) is a system that monitors network traffic for doubtful activity and matters alert when such activity is exposed. While anomaly discovery and reportage are the primary functions, some interruption detection systems are capable of taking actions when malicious activity or irregular traffic is detected, including blocking traffic sent from suspicious IP addresses. An IDS is a combination of hardware and software that detects intrusions in the network. It is used to track, identify and detect the intruders. It is a combination of both hardware and software that detects intrusions in the network. IDS is used to detect unauthorized intrusions that occur in computer systems and networks. Feature selection for intrusion detection is most important factor for the success of intrusion detection system. The objectives of IDS are confidentiality, integrity, availability and accountability. Intrusion Prevention System is classified into four types:

a**. Network Based Intrusion Prevention System** –Monitors the entire network for suspicious traffic by analysing the protocol activity.

b. **Wireless Intrusion Prevention System** – Monitors the entire network for suspicious traffic by analysing the wireless networking protocols.

c**. Network Behaviour Analysis** – Examines the network to identify the threats such as distributed denial of service.

d**. Host Based Intrusion Prevention System** – An installed software which monitors a host for a suspicious activity by analysing the host.

Intrusions can be defined as the set of actions that attempt to compromise the confidential harmony, integrity or availability of a computer resource. When intruders deliberately gain unauthorized access of the resource, they try to access information, manipulate data, or render information in a system to make unreliable or unusable. An IDS is a union of hardware and software components that detect harmful or malicious attempts in the network. IDS can monitor all the network activities and hence can detect the signs of intrusions. The main aim of IDS is to inform the system administrator that any doubtful activity happened. There are two kinds of intrusion detection techniques:

A) **Anomaly Detection**: Recognizes malicious activities based on deviations from the normal conduct and considers these deviations as attacks.

B) **Misuse Detection**: Recognizes intrusions based on a standard pattern of the malicious activity. It can be very helpful for known attack patterns. Also, the rate of misplaced report is high. One disadvantage of Misuse Detection over Anomaly Detection is that it can only notice intrusions which contain known patterns of attack. An IDS monitors the activities of a given environment and decides whether these activities are malicious or normal based on system integrity, confidentiality and the availability of information resources. When building IDS, one needs to consider many issues, such as data collection, data pre-processing, intrusion recognition, reporting, and response.

Working of Intrusion Detection System: There are 4 steps in the working of IDS. They are Collecting Data, Selecting Features, Analysing the Data, and last the Actions to be Performed.

a. **Collecting Data**: In order to do IDS, we need to collect the information about the network traffic like kinds of traffic, hosts and protocol details.

b. **Selecting Features**: From the large amount of data collected we need to extract the features which we need.

c. **Analysing the Data**: The selected features data is analysed to find whether the data is anomalous or not.

d. **Actions to be Performed**: IDS alarm or alert is made by the system administrator when an attack has occurred, and it tells about the type of the attack. IDS also participates in controlling the attacks by closing the network port and killing the processes.

**DATASET**

Intelligent intrusion detection systems can only be built if there is availability of an effective data set. A data set with a sizable amount of quality data which mimics the real time can only help to train and test an intrusion detection system. The NSL-KDD data set is a refined version of its predecessor KDD‟99 data set.

NSL-KDD data set is used to study the effectiveness of the various classification algorithms in detecting the anomalies in the network traffic patterns

Description:

It contains essential records of the complete KDD data set. There is a collection of downloadable files at the disposal for the researchers.

In each record there are 41 attributes unfolding different features of the flow and a label assigned to each either as an attack type or as normal. The details of the attributes namely the attribute name, their description and sample data. The Table VI contains type information of all the 41 attributes available in the NSL-KDD data set. The 42nd attribute contains data about the various 5 classes of network connection vectors and they are categorized as one normal class and four attack class. The 4 attack classes are further grouped as DoS, Probe, R2L and U2R. The description of the attack classes.

Attribute No. Attribute Name Description Sample Data

1 Duration Length of time duration

of the connection 0

2 Protocol\_type Protocol used in the

Connection Tcp

3 Service Destination network

service used ftp\_data

4 Flag Status of the connection –

Normal or Error SF

5 Src\_bytes Number of data bytes

transferred from src to

dst in single connection 491

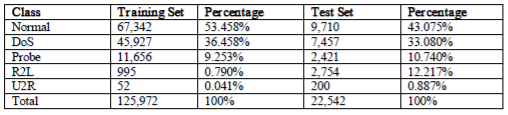
The benefits of using the NSL-KDD dataset are

1) No duplicate records in the test set which has better reduction rates.

2) The number of selected records from each difficult level group is inversely proportional to the percentage of records in the original KDD dataset.

3) NSL-KDD has fewer data points than KDD-99, all of which are unique. It is thus less computationally expensive to use for training machine learning models.

**NSL-KDD train and test data distribution**



The attack classes present in the NSL-KDD data set are grouped into four categories:

1. **DOS**: Denial of service is an attack category, which depletes the victim’s resources thereby making it unable to handle legitimate requests – e.g. syn flooding. Relevant features: “source bytes” and “percentage of packets with errors”

2. **Probing**: Surveillance and other probing attacks objective is to gain information about the remote victim e.g. port scanning. Relevant features: “duration of connection” and “source bytes”

3. **U2R**: unauthorized access to local super user (root) privileges is an attack type, by which an attacker uses a normal account to login into a victim system and tries to gain root/administrator privileges by exploiting some vulnerability in the victim e.g. buffer overflow attacks. Relevant features: “number of file creations” and “number of shell prompts invoked,”

4. **R2L**: unauthorized access from a remote machine, the attacker intrudes into a remote machine and gains local access of the victim machine. E.g. password guessing. Relevant features: Network level features – “duration of connection” and “service requested” and host level features - “number of failed login attempts”

**METHODOLOGY**

**PROPOSED IDS METHODOLOGY FOR THE ML MODEL:**

The main goal is to build a framework of intrusion detection with minimum number of features in the dataset. The previous researches showed that only a subset of these features is related to ID. So, the aim is to reduce the data set dimensionality to build a better classifier in a reasonable time. The proposed approach consists of four main phases: The first phase is to select the related features for each attack using feature selection method. Then combining the different features to obtain the optimal set of features for all attacks. The final set of features is fed to the classification stage. Finally, the model is tested using a test dataset.

The proposed approach consists of four main phases:

1. Feature selection
2. Combining the optimal features
3. Building a classifier
4. Evaluation

**FEATURE SELECTION**:

While the network intrusion system deals with a large amount of raw data, the feature selection is becoming a basic step in building such system. Feature selection is related to a number of methods and techniques that are used to eliminate the irrelevant and redundant features. The dimensionality of the data set has a big effect in the model complexity that leads to low classification accuracy, and high computational cost and time.

The aim of these methods also is to select the optimal features which will enhance the model’s performance. There are two general categories of methods for feature selection, filter methods and wrapper methods.

In the **Filter algorithms** an independent measure is utilized (such as, information, distance, or consistency) which are used to estimate the relation of a set of features, while **wrapper algorithms** use of one of learning algorithms to make the evaluation of the feature’s value.

In this project, Information Gain (IG) will be used to select the subset of related features. IG is often cost less and faster than the wrapper methods. Information gain is computed for each individual attribute in the training dataset related to one class. If the ranked value is high that means a feature is highly distinctive this class. Otherwise if the value is less than the predetermined threshold, it will be removed from the feature space. To obtain a better threshold value, the distribution of the IG values is examined and tested with different threshold values on the training dataset.

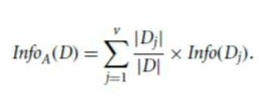
The selection features stage for each attack is divided into three main steps as follows:

Step1: The training dataset is divided into 22 datasets. Each dataset file contains the records of one attack records merged with the normal records. If the whole dataset is used without splitting, then the selection features method will be biased to the most frequent attacks. So, this step is essential to obtain more accurate results.

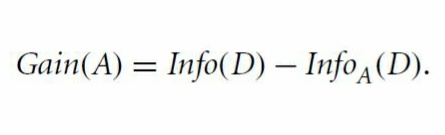
Step2: Each file then is used as an input to IG method to select the most relevant features of that attack.

Step3: A ranked feature list is generated, and according to some thresholds, a number of features are eliminated.

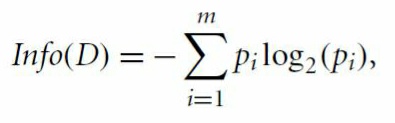
**Entropy of attribute**:



**Information gain**:



**Entropy**:



**COMBINING OPTIMAL FEATURES:**

In this step, a combined list of features for all attacks is generated from the obtained subsets. For some attacks the highest rank of the first three features are selected. But for another set of attacks, like land attack, one feature has been taken, since its rank is equal to 1, while the ranks for other features were very low. That means this feature can fully discriminate this attack.

**BUILDING A CLASSFIER:**

The final combined subset is used as an input to the classification stage. After conducting the experiments, the best two classifiers results are chosen. The next step, is to use the vote ensemble method to enhance the performance of the model.

**PROPOSED MODELS:**

**1. RANDOM FOREST:**

Random Forest algorithm is a supervised classification algorithm. We can see it from its name, which is to create a forest by some way and make it random. There is a direct relationship between the number of trees in the forest and the results it can get i.e., the larger the number of trees, the more accurate the result. But one thing to note is that creating the forest is not the same as constructing the decision with information gain or gain index approach. The difference between Random Forest algorithm and the Decision Tree algorithm is that in Random Forest, the **processes of** **finding the root node and splitting the feature nodes will run randomly**. There are two stages in Random Forest algorithm, one is random forest creation, the other is to make a prediction from the random forest classifier created in the first stage. For applications in classification problems, Random Forest algorithm will avoid the over fitting problem and for both classification and regression tasks, the same random forest algorithm can be used. The Random Forest algorithm can be used for identifying the most important features from the training dataset.

Random Forest Algorithm Creation Method:

1. Randomly select “K” features from total “m” features where k << m

2. Among the “K” features, calculate the node “d” using the best split point

3. Split the node into daughter nodes using the best split

4. Repeat the a to c steps until “l” number of nodes has been reached

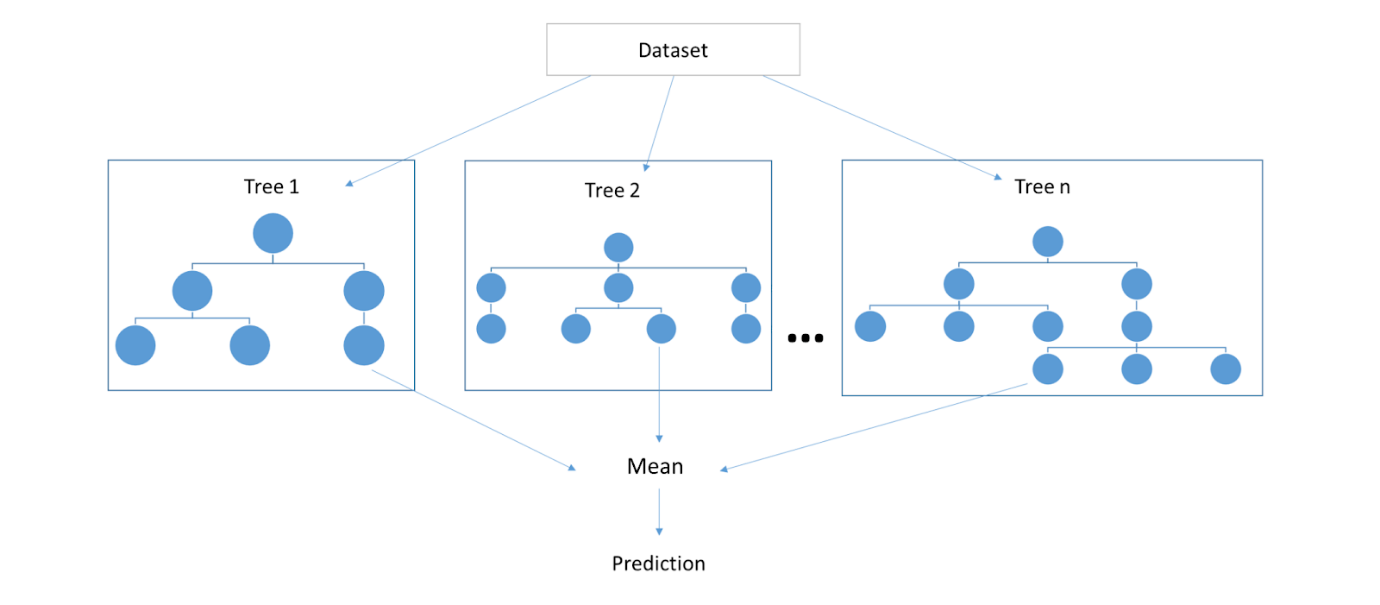
5. Build forest by repeating steps a tod for “n” number times to create “n” number of trees.

The random forest prediction pseudo code is shown below:

1. Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)

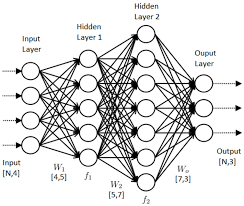
2. Calculate the votes for each predicted target

3. Consider the high voted predicted target as the final prediction from the random forest algorithm



**MULTILAYER PERCEPTRON**:

An MLP can be viewed as a logistic regression classifier where the input is first transformed using a learnt non-linear transformation. An MLP consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called back propagation for training.



**XGBOOST**

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Sample training and prediction using xgboost in python

**import** **xgboost** **as** **xgb**

*# read in data*

dtrain **=** xgb**.**DMatrix('demo/data/agaricus.txt.train')

dtest **=** xgb**.**DMatrix('demo/data/agaricus.txt.test')

*# specify parameters via map*

param **=** {'max\_depth':2, 'eta':1, 'objective':'binary:logistic' }

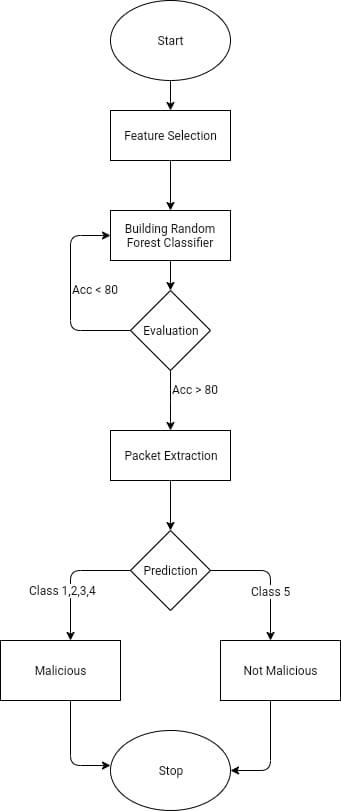
num\_round **=** 2

bst **=** xgb**.**train(param, dtrain, num\_round)

*# make prediction*

preds **=** bst**.**predict(dtest)

**FLOWCHART**



**TOOLS**

**TensorFlow**

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization for the purposes of conducting machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well.

TensorFlow provides stable Python and C++ APIs, as well as non-guaranteed backward-compatible API for other languages.

**TensorFlow Architecture**

Tensorflow architecture works in three parts:

* Pre-processing the data
* Build the model
* Train and estimate the model

It is called Tensorflow because it takes input as a multi-dimensional array, also known as tensors. You can construct a sort of flowchart of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

**Components of TensorFlow**

**Tensor**

Tensorflow's name is directly derived from its core framework: **Tensor**. In Tensorflow, all the computations involve tensors. A tensor is a **vector** or **matrix** of n-dimensions that represents all types of data. All values in a tensor hold identical data type with a known (or partially known) **shape**. The shape of the data is the dimensionality of the matrix or array.

A tensor can be originated from the input data or the result of a computation. In TensorFlow, all the operations are conducted inside a **graph**. The graph is a set of computation that takes place successively. Each operation is called an **op node** and is connected to each other.

The graph outlines the ops and connections between the nodes. However, it does not display the values. The edge of the nodes in the tensor, i.e., a way to populate the operation with data.

**Graphs**

TensorFlow makes use of a graph framework. The graph gathers and describes all the series computations done during the training. The graph has lots of advantages:

* It was done to run on multiple CPUs or GPUs and even mobile operating system
* The portability of the graph allows to preserve the computations for immediate or later use. The graph can be saved to be executed in the future.
* All the computations in the graph are done by connecting tensors together
  + A tensor has a node and an edge. The node carries the mathematical operation and produces endpoints outputs. The edges explain the input/output relationships between nodes.

**Install**

To install the current release for CPU-only:

$ pip install tensorflow

Use the GPU package for [CUDA-enabled GPU cards](https://www.tensorflow.org/install/gpu):

$ pip install tensorflow-gpu

**scikit-learn**

scikit-learn is a Python module for machine learning built on top of SciPy.

The project was started in 2007 by David Cournapeau as a Google Summer of Code project, and since then many volunteers have contributed.

**Important features of scikit-learn:**

* Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
* Accessible to everybody and reusable in various contexts.
* Built on the top of NumPy, SciPy, and matplotlib.
* Open source, commercially usable – BSD license.

**Installation**

**Dependencies**

scikit-learn requires:

* Python (>= 3.5)
* NumPy (>= 1.11.0)
* SciPy (>= 0.17.0)
* joblib (>= 0.11)

Scikit-learn 0.20 was the last version to support Python 2.7 and Python 3.4. scikit-learn 0.21 and later require Python 3.5 or newer.

Scikit-learn plotting capabilities (i.e., functions start with "[plot\_](https://github.com/scikit-learn/scikit-learn#id2)" and classes end with "Display") require Matplotlib (>= 1.5.1). For running the examples Matplotlib >= 1.5.1 is required. A few examples require scikit-image >= 0.12.3, a few examples require pandas >= 0.18.0.

**User installation**

If you already have a working installation of numpy and scipy, the easiest way to install scikit-learn is using pip

pip install -U scikit-learn

or conda:

conda install scikit-learn

**PyCharm**

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license.

**Installing Python**

* To download and install Python visit the official website of Python http://www.python.org/downloads/ and choose your version. We have chosen Python version 3.6.3
* Once the download is complete, run the exe for install Python. Now click on Install Now.
* You can see Python installing at this point.
* When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

**Installing Pycharm**

* To download PyCharm visit the website https://www.jetbrains.com/pycharm/download/ and click the "DOWNLOAD" link under the Community Section.
* Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.
* On the next screen, change the installation path if required. Click “Next”.
* On the next screen, you can create a desktop shortcut if you want and click on “Next”.
* Choose the start menu folder. Keep selected JetBrains and click on “Install”.
* Wait for the installation to finish.
* Once the installation is finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.

**Keras**

Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer.

Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend. So Keras is a high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano.

Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. At this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine

**Install Keras**

After we install Tensorflow, let's start installing keras. Type this command in the terminal

pip install keras

it will begin installing Keras and also all of its dependencies.

**Verifying**

Before we start using Keras, we should check if Keras is using Tensorflow as its backend by opening the configuration file:

gedit ~/.keras/keras.json

It will show something like this

{

    "floatx": "float32",

    "epsilon": 1e-07,

    "backend": "tensorflow",

    "image\_data\_format": "channels\_last"

}

as we can see, the "backend" uses tensorflow. It means that keras is using Tensorflow as its backend.

**CONCLUSION**

Intrusion Detection System (IDS) has been an effective way to achieve higher security in detecting malicious activities for the past couple of years. Anomaly detection is an intrusion detection system. Current anomaly detection is often associated with high false alarm rates and only moderate accuracy and detection rates because it’s unable to detect all types of attacks correctly. An experiment is carried out to evaluate the performance of the different machine learning algorithms using NSL-KDD datasets. Results show which approach has performed better in term of accuracy, detection rate with reasonable false alarm rate.

.