**CSE3011: Network Programming**

**Detecting DDoS Attacks with ML Technique**

**Submitted By**

KOUTHARAPU PAVAN NAGA SAI SREERAM

**20BCN7097**

Under the Guidance of

**Prof. Ganesh Reddy Karri**

A black background with red text

Description automatically generated with low confidence

*A picture containing text, font, white, screenshot

Description automatically generated*

**ABSTRACT:**

* To detect the DDoS attack using hybrid approach for a DDoS dataset
* Hybrid approach is the best way to detect DDoS attack.
* This results into accuracy up to 90%
* This approach first trains the model from the dataset
* Further it validates and tests it accuracy and also save logs for future reference in case of error
* This was done using TensorFlow and Keras

**Distributed denial of service (DDoS):**

Distributed denial of service (DDoS) attacks is a subclass of denial of service (DoS) attacks. A DDoS attack involves multiple connected online devices, collectively known as botnets, which are used to overwhelm a target website with fake traffic.

**TensorFlow**

TensorFlow is an open-source framework developed by Google researchers to run machine learning TensorFlow API to develop and train machine learning models.

TensorFlow provides a powerful platform for building and training various types of machine learning models, including neural networks, deep learning models, and reinforcement learning models.

TensorFlow has become one of the most popular machine learning libraries due to its flexibility, scalability, and ease of use.

**Keras**

Keras is a neural network Application Programming Interface (API) for Python that is tightly integrated with TensorFlow, which is used to build machine learning models.

Keras was developed with a focus on enabling fast experimentation and prototyping, with an emphasis on user-friendliness and modularity. Keras supports both CPU and GPU acceleration and provides a way to train models on multiple GPUs or even across multiple machines.

**Code:**

import arff

import numpy as np

from matplotlib import pyplot as plt

from keras.layers import Dense, Dropout

from keras.models import Sequential

from keras.models import load\_model

from keras.callbacks import TensorBoard

from keras.utils import to\_categorical

file=open("final-dataset.arff", 'r')

def scrape\_data():

    decoder=arff.ArffDecoder()

    data=decoder.decode(file, encode\_nominal=True)

    vals=[val[0: -1] for val in data['data']]

    labels =[label[-1] for label in data['data']]

    for val in labels:

        if labels[val] != 0:

            labels[val]=1

    training\_data=vals[0: int(.9\*len(vals))]

    training\_labels=labels[0: int(.9\*len(vals))]

    validation\_data=vals[int(.9\*len(vals)):]

    validation\_labels=labels [int(.9\*len(vals)):]

    training\_labels=to\_categorical(training\_labels, 5)

    validation\_labels=to\_categorical(validation\_labels, 5)

    return np.asarray(training\_data), np.asarray(training\_labels), np.asarray(validation\_data),np.asarray(validation\_labels)

def generate\_model(shape):

    model=Sequential()

    model.add(Dense (30, input\_dim=shape, kernel\_initializer="uniform", activation='relu'))

    model.add(Dropout (0.4))

    model.add(Dense (10, activation="relu"))

    model.add(Dropout (0.4))

    model.add(Dense (10, activation='relu'))

    model.add(Dropout (0.4))

    model.add(Dense (64, activation='relu'))

    model.add(Dropout (0.4))

    model.add(Dense(5, activation='softmax'))

    print (model.summary())

    return model

data\_train,label\_train,data\_eval, label\_eval =scrape\_data()

model=generate\_model(len(data\_train[0]))

model.compile(loss='categorical\_crossentropy',optimizer='adam', metrics=['accuracy'])

tensorboard=TensorBoard(log\_dir='logs/', histogram\_freq=0, write\_graph=True, write\_images=True)

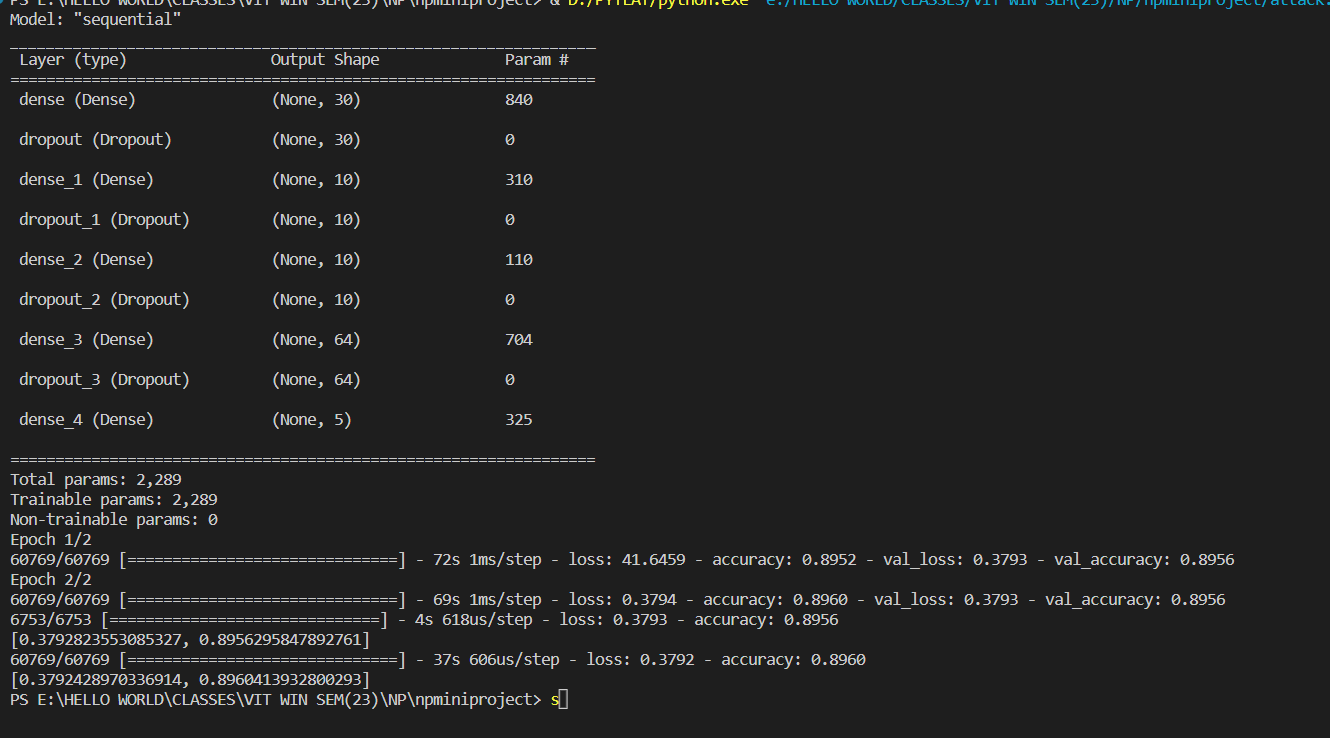
history=model.fit(data\_train, label\_train, validation\_data=( data\_eval, label\_eval), epochs=2, callbacks=[tensorboard])

loss\_history=history.history["loss"]

print(model.evaluate(data\_eval, label\_eval))

print(model.evaluate(data\_train, label\_train))

**Output:**

****

**Conclusion**

We can conclude that we got accuracy of 90% even though after the model runs three times as epoch was given three, we can modify it as per our needs.

In TensorFlow, the first step is to define the computational graph using its API. The graph contains a set of operations and tensors that represent the mathematical operations and data that flow between them.

We also have logs stored in the file.

The accuracy of the machine learning model is 90%

The work done in this project helps in identifying DDoS attacks in a simpler and more efficient manner.

Keras provides a wide range of functions and classes for building and training different types of neural networks and other machine learning models.

Here Real-time datasets are used for efficient analysis.