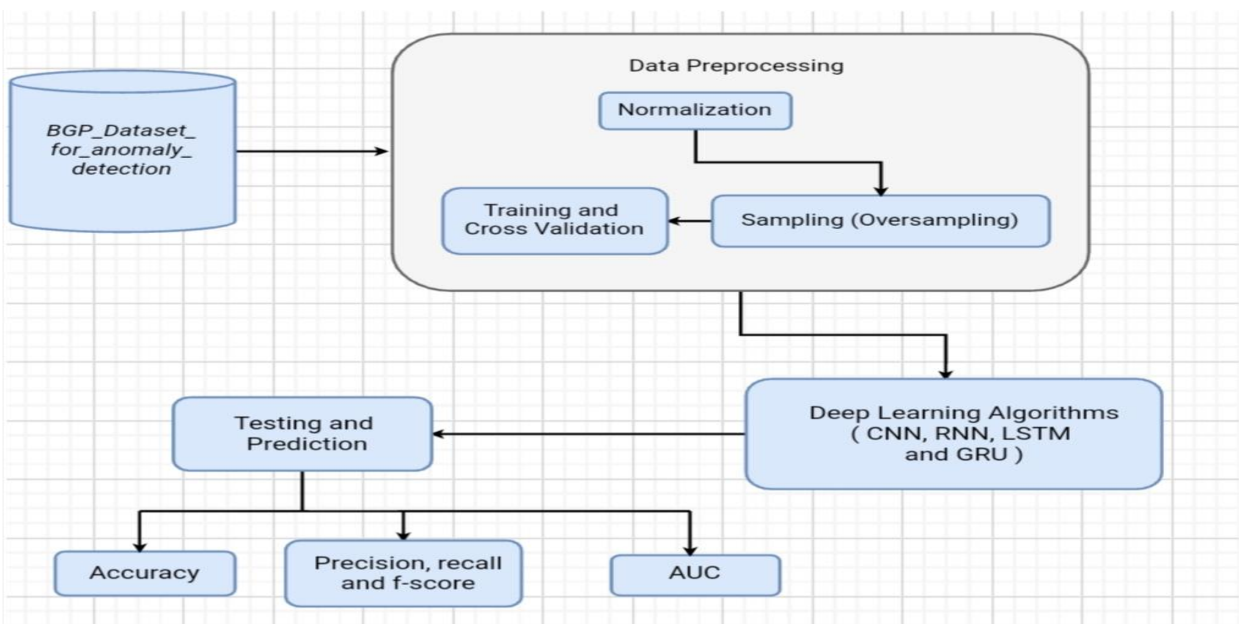
SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

SLOT: G1

INTRUSION DETECTION SYSTEM USING DEEP LEARNING ALGORITHMS: GRU (Gatted recurrent Unit)

BIBHU BHUSHAN 18BIT0134

Design and description of system:



BGP datasets for anomaly detection:

BGP stands for Border Gateway Protocol. BGP is the protocol that makes the Internet work. It does this by enabling data routing on the Internet. when someone submits data across the Internet, BGP is responsible for looking at all of the available paths that data could travel and picking the best route, which usually means hopping between autonomous systems.

Three well-known Border Gateway Protocol (BGP) anomalies Slammer, Nimda, and Code Red I occurred in January 2003, September 2001, and July 2001, respectively.

We are using Border Gateway Protocol anomalies for training and testing our algorithm. We are using deep learning algorithm like CNN for anomaly detection.

Data Preprocessing:

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. Whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

For achieving better results from the applied model in Deep learning projects the format of the data has to be in a proper manner. Data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in one data set and best out of them is chosen.

For data preprocessing we have used various techniques like Normalization, Random Oversampling and Hyper parameter tuning of individual algorithm parameters.

Deep learning Algorithm like CNN is used for detecting anomalous data. BGP datasets for anomaly detection is used for training and testing of the algorithm that we have used in this project.

Testing and Prediction is done using the deep learning algorithm used and various performance metrics like precision, recall, f-score is calculated for the algorithm.

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho et al. The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate. GRU's performance on certain tasks of polyphonic music modeling, speech signal modeling and natural language processing was found to be similar to that of LSTM. GRUs have been shown to exhibit better performance on certain smaller and less frequent datasets

MY CODE

18BIT0134.ipynb

File Edit View Insert Runtime Tools Help Last edited on October 22

+ Code + Text

```
from tensorflow import keras
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

from keras.layers import Dense, LSTM, Dropout, GRU, Bidirectional
```

Double-click (or enter) to edit

```
data = pd.read_csv('combine.csv')
data.head()
```

18BIT0134.ipynb

File Edit View Insert Runtime Tools Help Last edited on October 22

Comment Share

+ Code + Text Connect Editing

```
data.head()
```

	hour+minute	hour	minute	second	Number of announcements	Number of withdrawals	Number of announced NLRI prefixes	Number of withdrawn NLRI prefixes	Average AS-path length	Maximum AS-path length	Average unique AS-path length	Number of duplicate announcements	Number of duplicate withdrawals	Number of implicit withdrawals	Average edit distance	Max dist
0	0	0	0	0	57	8	203	16	6	15	6	206	150	20	6	1
1	1	0	1	0	62	23	361	75	6	16	6	398	355	120	6	
2	2	0	2	0	74	12	398	23	6	12	6	433	323	28	7	
3	3	0	3	0	70	4	543	49	6	27	6	568	210	72	8	
4	4	0	4	0	51	4	347	4	5	8	5	439	263	5	6	

```
data.columns
```

```
Index(['hour+minute', 'hour', 'minute', 'second', 'Number of announcements',
       'Number of withdrawals', 'Number of announced NLRI prefixes',
       'Number of withdrawn NLRI prefixes', 'Average AS-path length',
       'Maximum AS-path length', 'Average unique AS-path length',
       'Number of duplicate announcements', 'Number of duplicate withdrawals',
       'Number of implicit withdrawals', 'Average edit distance',
       'Maximum edit distance', 'Inter-arrival time',
```

```
[ ]        'Maximum edit distance = 9', 'Maximum edit distance = 10',
           'Maximum edit distance = 11', 'Maximum edit distance = 12',
           'Maximum edit distance = 13', 'Maximum edit distance = 14',
           'Maximum edit distance = 15', 'Maximum edit distance = 16',
           'Maximum edit distance = 17', 'Maximum AS-path length = 7',
           'Maximum AS-path length = 8', 'Maximum AS-path length = 9',
           'Maximum AS-path length = 10', 'Maximum AS-path length = 11',
           'Maximum AS-path length = 12', 'Maximum AS-path length = 13',
           'Maximum AS-path length = 14', 'Maximum AS-path length = 15',
           'Number of Interior Gateway Protocol (IGP) packets',
           'Number of Exterior Gateway Protocol (EGP) packets',
           'Number of incomplete packets', 'Packet size (B)', 'Label'],
          dtype='object')
```

```
[ ] print(data.groupby('Label').size())
```

```
Label
-1    28120
 1     4965
dtype: int64
```

```
[ ] data['Label'] = data['Label'].apply(lambda x: 0 if x == -1 else 1)
```

```
[ ] data['Label']
```

```
0        0
1        0
2        0
```

```
[ ] 4        0
         ..
33080    0
33081    0
33082    0
33083    0
33084    0
Name: Label, Length: 33085, dtype: int64
```

```
[ ] data.shape
```

```
(33085, 42)
```

```
[ ] data.dtypes
```

```
hour+minute                                        int64
hour                                               int64
minute                                             int64
second                                             int64
Number of announcements                            int64
Number of withdrawals                              int64
Number of announced NLRI prefixes                  int64
Number of withdrawn NLRI prefixes                  int64
Average AS-path length                             int64
Maximum AS-path length                             int64
Average unique AS-path length                      int64
Number of duplicate announcements                  int64
Number of duplicate withdrawals                    int64
```

```
Average AS-path length                              int64
Maximum AS-path length                              int64
Average unique AS-path length                       int64
Number of duplicate announcements                   int64
Number of duplicate withdrawals                     int64
Number of implicit withdrawals                      int64
Average edit distance                               int64
Maximum edit distance                             float64
Inter-arrival time                                  int64
Maximum edit distance = 7                           int64
Maximum edit distance = 8                           int64
Maximum edit distance = 9                           int64
Maximum edit distance = 10                          int64
Maximum edit distance = 11                          int64
Maximum edit distance = 12                          int64
Maximum edit distance = 13                          int64
Maximum edit distance = 14                          int64
Maximum edit distance = 15                          int64
Maximum edit distance = 16                          int64
Maximum edit distance = 17                          int64
Maximum AS-path length = 7                          int64
Maximum AS-path length = 8                          int64
Maximum AS-path length = 9                          int64
Maximum AS-path length = 10                         int64
Maximum AS-path length = 11                         int64
Maximum AS-path length = 12                         int64
Maximum AS-path length = 13                         int64
Maximum AS-path length = 14                         int64
Maximum AS-path length = 15                         int64
Number of Interior Gateway Protocol (IGP) packets   int64
```

```
[ ] Y = data['Label'].values
    X = data.drop('Label', axis=1).values
```

```
[ ] Y = np.reshape(Y,(33085,1))
    Y.shape

    (33085, 1)
```

```
[ ] scaler = StandardScaler().fit(X)
    rescaledX = scaler.transform(X)
```

```
[ ] rescaledX = np.reshape(rescaledX,(33085,40,1))
```

```
[ ] rescaledX.shape

    (33085, 40, 1)
```

```
[ ] x_train,x_test,y_train,y_test = train_test_split(rescaledX,Y,test_size=0.20,random_state=21)
```

```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [1],
       [1]])
```

```
regressorGRU = Sequential()
# First GRU layer with Dropout regularisation
regressorGRU.add(GRU(units=50, return_sequences=True, input_shape=(x_train.shape[1],1), activation='sigmoid'))
regressorGRU.add(Dropout(0.2))
# Second GRU layer
regressorGRU.add(GRU(units=50, return_sequences=True, input_shape=(x_train.shape[1],1), activation='sigmoid'))
regressorGRU.add(Dropout(0.2))
# Third GRU layer
regressorGRU.add(GRU(units=50, return_sequences=True, input_shape=(x_train.shape[1],1), activation='sigmoid'))
regressorGRU.add(Dropout(0.2))
# Fourth GRU layer
regressorGRU.add(GRU(units=50, activation='sigmoid'))
regressorGRU.add(Dropout(0.2))
# The output layer
regressorGRU.add(Dense(units=1))
# Compiling the RNN
regressorGRU.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
# Fitting to the training set
regressorGRU.fit(x_train,y_train,epochs=5,batch_size=50,verbose=1,validation_split=0.2)
```


```
# Compiling the RNN
regressorGRU.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
# Fitting to the training set
regressorGRU.fit(x_train,y_train,epochs=5,batch_size=50,verbose=1,validation_split=0.2)
```


```
WARNING:tensorflow:Layer gru will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_1 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_2 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_3 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
Epoch 1/5
424/424 [==============================] - 129s 305ms/step - loss: 1.6936 - accuracy: 0.7899 - val_loss: 0.4187 - val_accuracy: 0.8559
Epoch 2/5
424/424 [==============================] - 127s 300ms/step - loss: 1.2634 - accuracy: 0.7456 - val_loss: 0.4363 - val_accuracy: 0.8559
Epoch 3/5
424/424 [==============================] - 123s 291ms/step - loss: 1.8624 - accuracy: 0.8213 - val_loss: 2.2231 - val_accuracy: 0.8559
Epoch 4/5
424/424 [==============================] - 121s 286ms/step - loss: 1.4570 - accuracy: 0.7908 - val_loss: 0.4125 - val_accuracy: 0.8559
Epoch 5/5
424/424 [==============================] - 121s 285ms/step - loss: 1.2218 - accuracy: 0.7436 - val_loss: 0.4133 - val_accuracy: 0.8559
<tensorflow.python.keras.callbacks.History at 0x7fdfb0142518>
```

```
y_pred=regressorGRU.predict(x_test)
y_pred = np.reshape(y_pred,(6617))
y_pred=list(y_pred)
```

```
for i,val in zip(range(6617),y_pred):
  if val>=0.5:
```

```
  else:
    y_pred[i]=0
```

```
train_acc = regressorGRU.evaluate(x_train, y_train, verbose=0)
test_acc = regressorGRU.evaluate(x_test, y_test, verbose=0)
```

```
print(train_acc)
print(test_acc)
```

```
[0.4249716103076935, 0.8488363027572632]
[0.4158564805984497, 0.8543146252632141]
```

```
matrix = confusion_matrix(y_test, y_pred)
```

```
matrix
```

```
array([[5653,    0],
       [ 964,    0]])
```