EDA FGSM

November 16, 2020

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import keras
```

```
Using TensorFlow backend.
```

[2]:
```
x_test = np.load('./ATTACKS/FGSM/X_TEST_FGSM.npy')
y_test = np.load('./ATTACKS/FGSM/Y_TEST_FGSM.npy')
x_n_test = np.load('./ATTACKS/FGSM/X_TEST_ATTACKED_FGSM.npy')
```

1 MAL AND BEN ROWS

[3]:
```
MAL = []
BEN = []
for i in range(y_test.shape[0]):
    if y_test[i]==1:
        MAL.append(i)
    else:
        BEN.append(i)
print(len(MAL),len(BEN))
```

```
1667 3333
```

2 Classifier

[4]:
```
from keras.utils import to_categorical
test_labels = to_categorical(y_test)
```

[5]:
```
network = keras.models.load_model('./ATTACKS/FGSM/FGSM_CLASSIFIER_USED.h5py')
network.summary()
```

```
WARNING:tensorflow:From C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\tensorflow_core\python\ops\resource_variable_ops.py:1630: calling
BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops)
with constraint is deprecated and will be removed in a future version.
Instructions for updating:
```

```
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape_1 (Reshape)          (None, 2304)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1180160
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026
=================================================================
Total params: 1,181,186
Trainable params: 1,181,186
Non-trainable params: 0
_________________________________________________________________
```

3 Find basic overall drop in accuracy

[6]:
```
network.evaluate(x_test,test_labels)
```

```
5000/5000 [==============================] - 1s 239us/step
```

[6]:
```
[0.19247934875786304, 0.9476000070571899]
```

[7]:
```
network.evaluate(x_n_test,test_labels)
```

```
5000/5000 [==============================] - 0s 62us/step
```

[7]:
```
[8.37555297266245, 0.6764000058174133]
```

[8]:
```
print("DROP")
print((network.evaluate(x_test,test_labels)[1]-network.
 ↪evaluate(x_n_test,test_labels)[1])*100)
```

```
DROP
5000/5000 [==============================] - 0s 57us/step
5000/5000 [==============================] - 0s 56us/step
27.12000012397766
```

4 But we just attacked Malware. So lets find that particular drop

[9]:
```
network.evaluate(x_test[MAL],test_labels[MAL])
```

```
1667/1667 [==============================] - 0s 77us/step
```

[9]:
```
[0.3696391213490853, 0.9064187407493591]
```

[10]:
```
network.evaluate(x_n_test[MAL],test_labels[MAL])
```

```
1667/1667 [==============================] - 0s 64us/step
```

[10]:
```
[24.91395094371264, 0.09298140555620193]
```

[11]:
```
print("DROP")
print((network.evaluate(x_test[MAL],test_labels[MAL])[1]-network.
 ↪evaluate(x_n_test[MAL],test_labels[MAL])[1])*100)
```

```
DROP
1667/1667 [==============================] - 0s 60us/step
1667/1667 [==============================] - 0s 65us/step
81.34373351931572
```

4.0.1 Among the MAL Samples, we saw a drop of 81.3437% which is great

5 First, We are gonna construct a change matrix, the changes from x_test and x_n_test.

5.1 Sanity Check

[12]:
```
same = 0
one2zero = 0
zero2one = 0
nosense = 0
for i in range(x_test.shape[0]):
    for j in range(x_test.shape[1]):
        for k in range(x_test.shape[2]):
#             print(x_test[i][j][k],x_n_test[i][j][k])
            if x_test[i][j][k] == x_n_test[i][j][k]//0.5:
                same+=1
            elif x_test[i][j][k] == 1 and x_n_test[i][j][k]//0.5 == 0:
                one2zero+=1
            elif x_test[i][j][k] == 0 and x_n_test[i][j][k]//0.5 == 1:
                zero2one+=1
            else:
                nosense+=1
print(same,one2zero,zero2one,nosense)
```

```
11037269 0 0 482731
```

[33]:
```
same = 0
one2zero = 0
zero2one = 0
nosense = 0
changed = []
for i in range(x_test.shape[0]):
    for j in range(x_test.shape[1]):
        for k in range(x_test.shape[2]):
            if x_test[i][j][k] == x_n_test[i][j][k]:
                same+=1
            else:
                nosense+=1
print(same,nosense)
```

```
9468940 2051060
```

The attacked added 2051060 bits in the whole dataset

[35]:
```
same = 0
c1 = 0
c2 = 0
changedRows = []
for i in range(x_test.shape[0]):
    for j in range(x_test.shape[1]):
        for k in range(x_test.shape[2]):
            if x_test[i][j][k] == x_n_test[i][j][k]:
                same+=1
            elif x_test[i][j][k] == 0:
                c1+=1
#                 x_n_test_copy[i][j][k] = 1
            elif x_test[i][j][k] == 1:
                c2+=1
    if x_test[i].tolist() != x_n_test[i].tolist():
        changedRows.append(i)

print(same,c1,c2,len(changedRows))
```

```
9468940 2051060 0 1667
```

- No features were removed
- 2051060 many features were added
- 1667 many rows were changed, same as all MAL

Find diff matrix to find which features were added

[36]:
```
changedMat = np.zeros(x_test.shape)
```

[40]:
```
for i in range(x_test.shape[0]):
    for j in range(x_test.shape[1]):
        for k in range(x_test.shape[2]):
            if x_test[i][j][k] != x_n_test[i][j][k]:
                changedMat[i][j][k] = 1
print(same,nosense)
```

```
18937880 4102120
```

[41]:
```
np.unique(changedMat,return_counts=True)
```

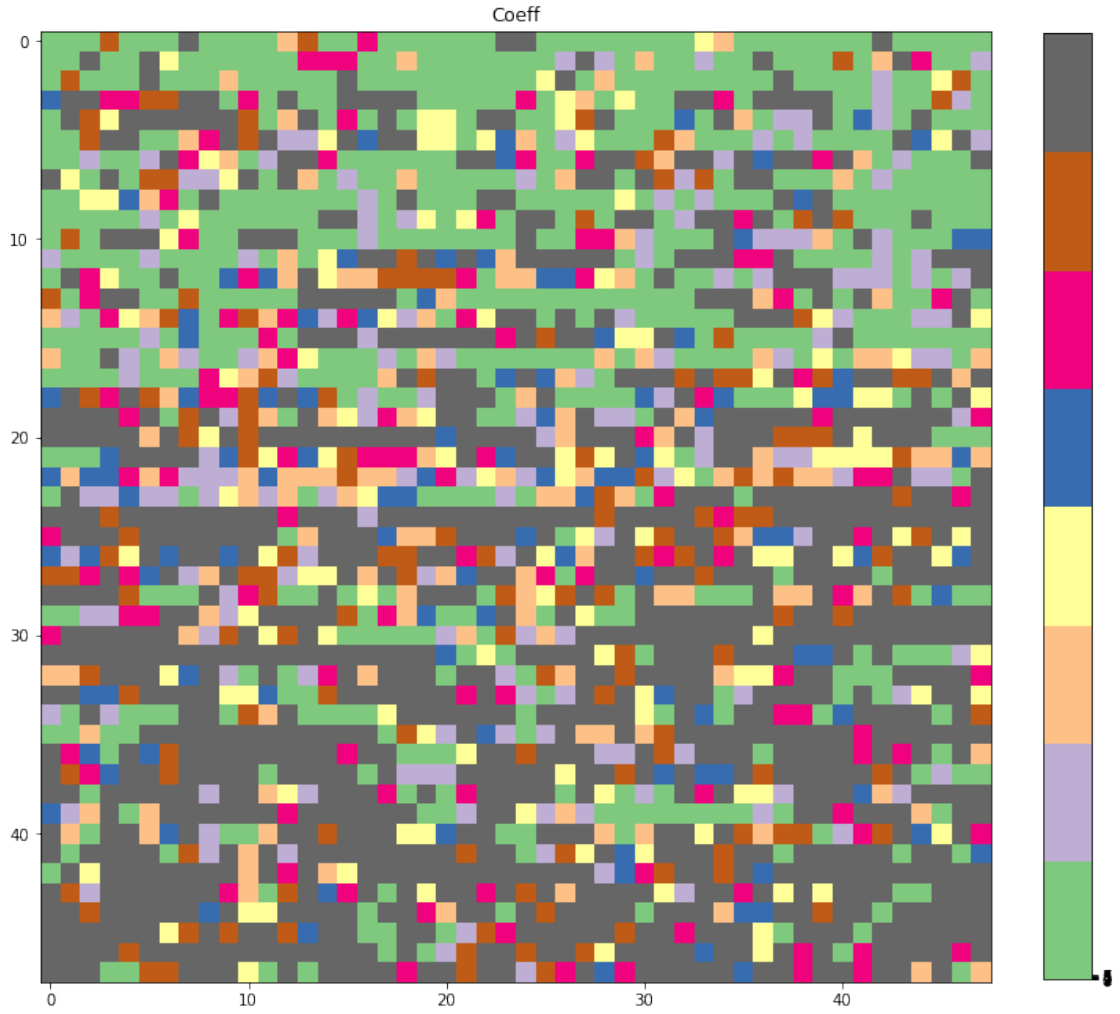
[41]:
```
(array([0., 1.]), array([9468940, 2051060], dtype=int64))
```

[42]:
```
fig, ax = plt.subplots(figsize = (12,12))
ax.set_title('Coeff')
cax = ax.imshow(np.sum(changedMat,axis = 0), cmap = plt.cm.Accent)

cbar = plt.colorbar(cax, ticks=[0, 1, 2, 3, 4, 5, 6, 7],
                    orientation='vertical',
                    fraction=0.045, pad=0.05)

print("GREEN IS LOW, BLACK IS HIGH")
print("GREEN MIN : "+ str(int( min( np.sum(changedMat,axis=0).reshape(-1,1) ) )␣
 ↪))
print("BLAC MAX : "+ str(int( max( np.sum(changedMat,axis=0).reshape(-1,1) ) )␣
 ↪))
```

```
GREEN IS LOW, BLACK IS HIGH
GREEN MIN : 0
BLAC MAX : 1544
```



6 Observation

As we can see from the above matrix, when attacked, a lot more BEN features are added than the MAL features, which is clearly seen by the much darker shades of colours in the lower half of the density graph.

A particular feature was even added 1554 times

7 Lets see is any features were changed over 1525 times

[59]:
```
sumCM = np.sum(changedMat,axis = 0)
c1525 = []
```

[60]:
```
for i in range(sumCM.shape[0]):
    for j in range(sumCM.shape[1]):
```

```
        if sumCM[i][j] >= 1525:
            c1525.append([i,j])
```

[61]:
```
len(c1525)
```

[61]: 7

549 features were changed over 1500 times which is huge

7.0.1 Lets see these 7 features

[62]:
```
features = np.load("./DATA/FeatureList.npy", allow_pickle=True)
coeff = np.load("./DATA/coeff_features.npy", allow_pickle=True)
```

[70]:
```
from termcolor import colored
```

[74]:
```
for i,j in c1525:
    print("============================================================")
    print("FREQ of addition")
    print(sumCM[i][j])
    print("----------")
    print("NATURE OF FEATURE")
    if coeff[i][j] >= 0:
        print(colored("MAL","red"))
        print(colored(coeff[i][j],'red'))
    else:
        print(colored("BEN","blue"))
        print(colored(coeff[i][j],'blue'))
    print("----------")
    print("FEATURE NAME")
    print(features[i][j])
    print("============================================================")
```

```
============================================================
FREQ of addition
1544.0
----------
NATURE OF FEATURE
MAL
0.20591928411780536
----------
FEATURE NAME
urls::http://gdata_youtube_com/feeds/api/playlists/
============================================================
============================================================
FREQ of addition
1530.0
----------
```

NATURE OF FEATURE
MAL
0.1304848095502816

FEATURE NAME
api_calls::android/media/AudioManager;->setMicrophoneMute
==========
==========
FREQ of addition
1530.0

NATURE OF FEATURE
MAL
0.005382702835261449

FEATURE NAME
urls::http://ns_adobe_com/xap/1_0/bj/
==========
==========
FREQ of addition
1531.0

NATURE OF FEATURE
BEN
-0.024823092470285343

FEATURE NAME
urls::https://onesignal_com/android_frame_html
==========
==========
FREQ of addition
1526.0

NATURE OF FEATURE
BEN
-0.042945680726528254

FEATURE NAME
urls::https://api_%s/install_data/v3/
==========
==========
FREQ of addition
1530.0

NATURE OF FEATURE
BEN
-0.051427905063169514

```
FEATURE NAME
urls::http://fusion_qq_com
============================================================
============================================================
FREQ of addition
1536.0
----------
NATURE OF FEATURE
BEN
-0.10102430654063958
----------
FEATURE NAME
urls::http://www_youtube_com/playlist?list=
============================================================
```

8 Observations

- 7 features are added super often out of which 3 are MAL and 4 are BEN

9 Conclusion

Going over to apply both APEGAN and Cycle GAN on this dataset to see results

[ ]: