DATA SELECTION

November 14, 2020

1 We decided to select specific data from the already selected data

We went ahead with 2 selection

- data that was classified malware in xtest, was really malware and was successfully missclassified
- data that was true pos or true neg by classifier and missclassified if it was malware

2 Pure Malware Selection

[1]:
```
import numpy as np
import keras
from keras import models
from keras import layers
from keras.utils import to_categorical
```

```
Using TensorFlow backend.
```

[3]:
```
x_clean = np.load('./DATA/FGSM/X_TEST_ORG.npy')
x_adv = np.load('./DATA/FGSM/X_TEST_NOISED.npy')
x_label = np.load('./DATA/FGSM/Y_TEST_ORG.npy').astype('int')
```

[11]:
```
# x_clean.shape
x_label.shape
```

[11]:
```
(15621, 2)
```

3 Load Model

[6]:
```
network = models.load_model('./modelClassifierFGSM.h5')
network.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 2948)              8693652
```

```
_________________________________________________________________
dense_8 (Dense)              (None, 128)               377472
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 129
=================================================================
Total params: 9,071,253
Trainable params: 9,071,253
Non-trainable params: 0
_________________________________________________________________
```

[15]:
```
test_labels = x_label[:,1].copy()
```

[16]:
```
network.evaluate(x_clean,test_labels)
```

```
15621/15621 [==============================] - 4s 226us/step
```

[16]:
```
[0.13288401066285124, 0.9652391076087952]
```

[17]:
```
network.evaluate(x_adv,test_labels)
```

```
15621/15621 [==============================] - 1s 74us/step
```

[17]:
```
[74.8945468614184, 0.7323474884033203]
```

4 Find all Malware from x_test_org/clean using x_label and just take those rows

[18]:
```
MAL = []
for i in range(len(test_labels)):
    if test_labels[i]==1:
        MAL.append(i)
```

[19]:
```
network.evaluate(x_clean[MAL],test_labels[MAL])
```

```
5207/5207 [==============================] - 0s 69us/step
```

[19]:
```
[0.2275015620319078, 0.9514115452766418]
```

[20]:
```
network.evaluate(x_adv[MAL],test_labels[MAL])
```

```
5207/5207 [==============================] - 0s 73us/step
```

[20]:
```
[224.51248885144284, 0.2527366876602173]
```

4.0.1 As we can see here. the classifier classified Malware correctly 95% of the time originally and that dropped to 25% after the FGSM attack

4.1 Now take a copy of just the malware rows. Just real malware

[21]:
```
x_clean_mal1 = x_clean[MAL].copy()
x_adv_mal1 = x_adv[MAL].copy()
test_labels_mal1 = test_labels[MAL].copy()
```

[22]:
```
network.evaluate(x_clean_mal1,test_labels_mal1)
```

```
5207/5207 [==============================] - 0s 73us/step
```

[22]:
```
[0.2275015620319078, 0.9514115452766418]
```

[23]:
```
network.evaluate(x_adv_mal1,test_labels_mal1)
```

```
5207/5207 [==============================] - 0s 74us/step
```

[23]:
```
[224.51248885144284, 0.2527366876602173]
```

5 Now Take only those that were classified corectly in CLEAN

[33]:
```
np.unique(test_labels_mal1)
```

[33]:
```
array([1])
```

[39]:
```
preds = network.predict(x_clean_mal1)

c = 0
cc = 0

correctPred = []
for i in range(len(preds)):
    if preds[i]>=test_labels_mal1[i]/2:
        c+=1
        correctPred.append(i)
    else:
        cc+=1
#     print(preds[i],test_labels_mal1[i])
print(c,cc)
```

```
4954 253
```

5.0.1 Above, I've found out the classifications of all the malware. 253 were classified wrongly. We are gonna drop them now

[40]:
```
x_clean_mal2 = x_clean_mal1[correctPred].copy()
x_adv_mal2 = x_adv_mal1[correctPred].copy()
test_labels_mal2 = test_labels_mal1[correctPred].copy()
```

[41]:
```
network.evaluate(x_clean_mal2,test_labels_mal2)
```

```
4954/4954 [==============================] - 0s 74us/step
```

[41]:
```
[0.05139051474853418, 1.0]
```

[42]:
```
network.evaluate(x_adv_mal2,test_labels_mal2)
```

```
4954/4954 [==============================] - 0s 76us/step
```

[42]:
```
[229.96789775012144, 0.23314493894577026]
```

5.0.2 As you can see, clean classification is now 100% but we still get 23% as malware even after attacks.

6 Find and drop all the ones that weren't successfull attacks

[46]:
```
predsClass1 = network.predict(x_adv_mal2)

c = 0
cc = 0

correctPred1 = []
for i in range(len(predsClass1)):
    if predsClass1[i]<test_labels_mal2[i]/2:
        c+=1
        correctPred1.append(i)
    else:
        cc+=1
print(c,cc)
```

```
3799 1155
```

6.0.1 Here, we can see that 1155 rows are still malware in the classifiers eyes. Remove that and evaluate again

[47]:
```
x_clean_mal = x_clean_mal2[correctPred1].copy()
x_adv_mal = x_adv_mal2[correctPred1].copy()
test_labels_mal = test_labels_mal2[correctPred1].copy()
```

[48]:
```
network.evaluate(x_clean_mal,test_labels_mal)
```

```
3799/3799 [==============================] - 0s 85us/step
```

[48]:
```
[0.04282340362950449, 1.0]
```

[49]:
```
network.evaluate(x_adv_mal,test_labels_mal)
```

```
3799/3799 [==============================] - 0s 81us/step
```

[49]:
```
[299.8835335229691, 0.0]
```

6.0.2 Now we have an ideal dataset where all were malware and was then attacked into benign. Save this now

[51]:
```
np.save('./DATA/X_CLEAN_ONLY_MAL.npy',x_clean_mal)
np.save('./DATA/X_ADV_ONLY_MAL.npy',x_adv_mal)
np.save('./DATA/X_LABEL_ONLY_MAL.npy',test_labels_mal)
np.save('./DATA/X_LABEL_1D_ONLY_MAL.npy',test_labels_mal)
```

---

7 Now gonna take onlt True Pos and True Neg as selected

[52]:
```
import numpy as np
import keras
from keras import models
from keras import layers
from keras.utils import to_categorical
```

[53]:
```
x_clean = np.load('./DATA/FGSM/X_TEST_ORG.npy')
x_adv = np.load('./DATA/FGSM/X_TEST_NOISED.npy')
x_label = np.load('./DATA/FGSM/Y_TEST_ORG.npy').astype('int')

# Load Model

network = models.load_model('./modelClassifierFGSM.h5')
network.summary()

test_labels = x_label[:,1].copy()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 2948)              8693652
_________________________________________________________________
```

```
dense_8 (Dense)              (None, 128)               377472
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 129
=================================================================
Total params: 9,071,253
Trainable params: 9,071,253
Non-trainable params: 0
_________________________________________________________________
```

[54]:
```
network.evaluate(x_clean,test_labels)
```

```
15621/15621 [==============================] - 1s 78us/step
```

[54]:
```
[0.13288401066285124, 0.9652391076087952]
```

[55]:
```
network.evaluate(x_adv,test_labels)
```

```
15621/15621 [==============================] - 1s 79us/step
```

[55]:
```
[74.8945468614184, 0.7323474884033203]
```

8 Do not look at variable names. They are very bad.

[56]:
```
x_clean_mal1 = x_clean.copy()
x_adv_mal1 = x_adv.copy()
test_labels_mal1 = test_labels.copy()
```

[57]:
```
network.evaluate(x_clean_mal1,test_labels_mal1)
```

```
15621/15621 [==============================] - 1s 74us/step
```

[57]:
```
[0.13288401066285124, 0.9652391076087952]
```

[58]:
```
network.evaluate(x_adv_mal1,test_labels_mal1)
```

```
15621/15621 [==============================] - 1s 75us/step
```

[58]:
```
[74.8945468614184, 0.7323474884033203]
```

9 Classified correctly in clean

[64]:
```
preds = network.predict(x_clean_mal1)


predsClass = preds.copy()
```

```
c = 0
cc = 0

correctPred = []
for i in range(len(preds)):
    if predsClass[i]//0.5==test_labels_mal1[i]:
        c+=1
        correctPred.append(i)
    else:
        cc+=1
#         print(test_labels_mal1[i][1])
print(c,cc)
```

```
14539 1082
```

9.0.1 We have 14539 correct and 1082 wrong classifications

[69]:
```
x_clean_mal2 = x_clean_mal1[correctPred].copy()
x_adv_mal2 = x_adv_mal1[correctPred].copy()
test_labels_mal2 = test_labels_mal1[correctPred].copy()
```

[70]:
```
network.evaluate(x_clean_mal1,test_labels_mal1)
```

```
15621/15621 [==============================] - 1s 76us/step
```

[70]:
```
[0.13288401066285124, 0.9652391076087952]
```

[71]:
```
network.evaluate(x_clean_mal2,test_labels_mal2)
```

```
14539/14539 [==============================] - 1s 74us/step
```

[71]:
```
[0.024115363377748493, 1.0]
```

9.0.2 Now we have an ideal classifier for the clean

10 Now to find correct and wrong attacks in adv

[73]:
```
preds2Adv = network.predict(x_adv_mal2)
preds2Clean = network.predict(x_clean_mal2)

predsClass2Adv = preds2Adv.copy()
predsClass2Clean = preds2Clean.copy()

c = 0
cc = 0

WrongRows = []
```

```
CorrectRows = []

for i in range(len(preds)):
    if predsClass2Adv[i]>=0.5:
        WrongRows.append(i)
        c+=1
    else:
        cc+=1
        CorrectRows.append(i)

print(c,cc)
```

```
1100 13439
```

[74]:
```
x_clean_mal3 = x_clean_mal2[CorrectRows].copy()
x_adv_mal3 = x_adv_mal2[CorrectRows].copy()
test_labels_mal3 = test_labels_mal2[CorrectRows].copy()
```

[75]:
```
MAL = []
CLEAN = []
for i in range(len(test_labels_mal3)):
    if test_labels_mal3[i]//0.5 == 0:
        CLEAN.append(i)
    else:
        MAL.append(i)
print(len(CLEAN),len(MAL),len(CLEAN)+len(MAL))
```

```
10124 3315 13439
```

10.0.1 We have 3315 MAL and 10124 BEN in our final dataset. A total of 13439 out of the staring 15k

[78]:
```
10124/13439
```

[78]:
```
0.753329860852742
```

10.0.2 Classifier evals

[80]:
```
network.evaluate(x_clean_mal3,test_labels_mal3)
```

```
13439/13439 [==============================] - 1s 79us/step
```

[80]:
```
[0.019250745316190514, 1.0]
```

[81]:
```
network.evaluate(x_adv_mal3,test_labels_mal3)
```

```
13439/13439 [==============================] - 1s 75us/step
```

[81]: [73.85825694427842, 0.753329873085022]

[83]:
```
network.evaluate(x_adv_mal3[CLEAN],test_labels_mal3[CLEAN])
```

```
10124/10124 [==============================] - 1s 76us/step
```

[83]: [0.009848853657497537, 1.0]

[82]:
```
network.evaluate(x_adv_mal3[MAL],test_labels_mal3[MAL])
```

```
3315/3315 [==============================] - 0s 77us/step
```

[82]: [299.392187398735, 0.0]

10.1 We now have a classifier and a dataset that is perfect in clean and the attack only messed up the MAL but did that perfectly

10.2 Save this

[85]:
```
np.save('./DATA/X_CLEAN_IDEAL.npy',x_clean_mal3)
np.save('./DATA/X_ADV_IDEAL.npy',x_adv_mal3)
np.save('./DATA/X_LABEL_IDEAL.npy',test_labels_mal3)
np.save('./DATA/X_LABEL_1D_IDEAL.npy',test_labels_mal3)
```

[ ]:
```

```