CLASSIFIER - JSMA - FGSM

November 15, 2020

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

session = tf.Session()
keras.backend.set_session(session)
```

```
Using TensorFlow backend.
```

[2]:
```python
x_train = np.load("./DATA/X_train.npy", allow_pickle=True)
x_test = np.load("./DATA/X_test.npy", allow_pickle=True)
y_train = np.load("./DATA/y_train.npy", allow_pickle=True)
y_test = np.load("./DATA/y_test.npy", allow_pickle=True)
```

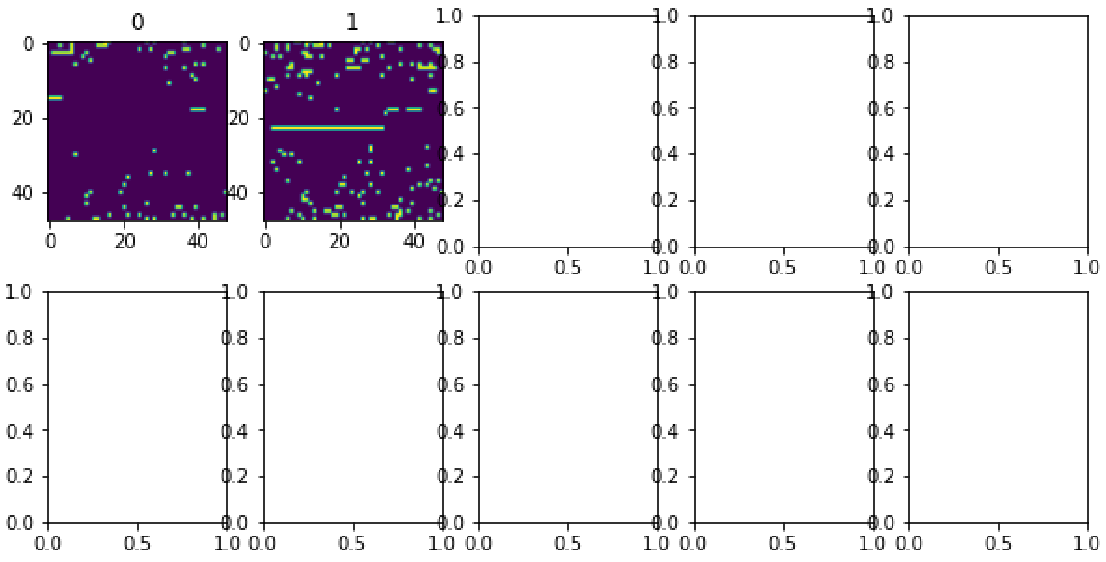
[3]:
```python
y_test.shape
```

[3]:
```
(5000, 1)
```

[4]:
```python
print ("Training Examples: %d" % len(x_train))
print ("Test Examples: %d" % len(x_test))
```

```
Training Examples: 10000
Test Examples: 5000
```

[5]:
```python
n_classes = 2
inds=np.array([y_train==i for i in range(n_classes)])
f,ax=plt.subplots(2,5,figsize=(10,5))
ax=ax.flatten()
for i in range(n_classes):
    ax[i].imshow(x_train[np.argmax(inds[i])].reshape(48,48))
    ax[i].set_title(str(i))
plt.show()
```


0
1

____

____

____

____

____

____

1 Classifier

We build the clssifier that will be used to evaluate the testing and attacks

<Describe the classifier] - BALAJI Specialization. We used convolution - remeber

[6]:
```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Reshape((48*48,),input_shape=(48,48,)))
network.add(layers.Dense(512, activation='relu', input_shape=(48 * 48,)))
network.add(layers.Dense(2, activation='softmax'))

network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])

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.
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

_________________________________________________________________
```

[7]:
```
from keras.utils import to_categorical #this just converts the labels to␣
↪one-hot class
train_labels = to_categorical(y_train)
test_labels = to_categorical(y_test)
```

[8]:
```
from keras.callbacks import ModelCheckpoint

h=network.fit(x_train,
              train_labels,
              epochs=5,
              batch_size=256,
              shuffle=True,
              callbacks=[ModelCheckpoint('image_space.h5',save_best_only=True)])
```

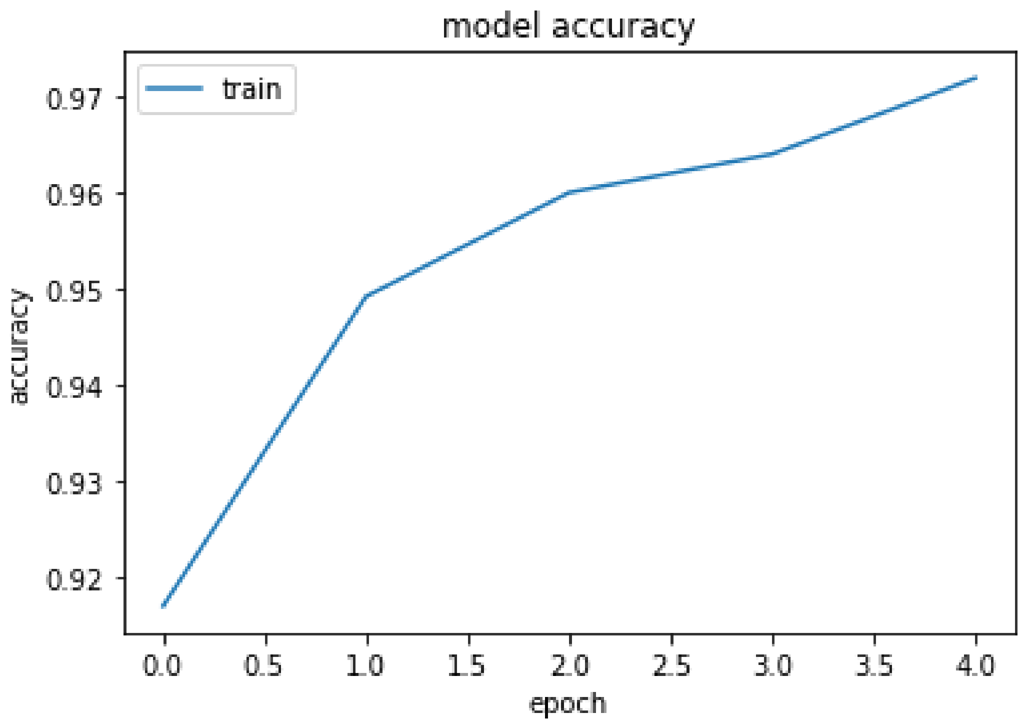
```
WARNING:tensorflow:From C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/5
10000/10000 [==============================] - 1s 72us/step - loss: 0.2328 -
accuracy: 0.9171
Epoch 2/5
 3072/10000 [========>…] - ETA: 0s - loss: 0.1615 -
accuracy: 0.9443
```

```
C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\keras\callbacks\callbacks.py:707: RuntimeWarning: Can save best model
only with val_loss available, skipping.
  'skipping.' % (self.monitor), RuntimeWarning)
```

```
10000/10000 [==============================] - 0s 39us/step - loss: 0.1519 -
accuracy: 0.9493
Epoch 3/5
10000/10000 [==============================] - 0s 41us/step - loss: 0.1161 -
accuracy: 0.9601
Epoch 4/5
10000/10000 [==============================] - 0s 39us/step - loss: 0.0982 -
accuracy: 0.9641
Epoch 5/5
10000/10000 [==============================] - 0s 46us/step - loss: 0.0828 -
accuracy: 0.9720
```

[9]:
```
# summarize history for accuracy
plt.plot(h.history['accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train'], loc='upper left')
plt.show()
```



[10]:
```
network.evaluate(x_train, train_labels)
```

```
10000/10000 [==============================] - 1s 64us/step
```

[10]:
```
[0.08402268831133843, 0.9724000096321106]
```

[11]:
```
score, acc = network.evaluate(x_test,test_labels,)
print ("Test Accuracy: %.5f" % acc)
```

```
5000/5000 [==============================] - 0s 64us/step
Test Accuracy: 0.94760
```

1.1 SAVE CLASSIFIER

[12]:
```
network.save("CLASSIFIER.h5py")
```

1.2 Find all MAL and BEN Rows

[13]:
```python
MAL = []
BEN = []
c = cc = 0
for i in range(len(test_labels[:,1])):
    if test_labels[i][1] == 0:
        BEN.append(i)
    else:
        MAL.append(i)
```

2 FGSM

```
[14]: from cleverhans.utils_keras import KerasModelWrapper
      wrap = KerasModelWrapper(network)
```

```
WARNING:tensorflow:From C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\cleverhans\utils_tf.py:341: The name tf.GraphKeys is deprecated. Please
use tf.compat.v1.GraphKeys instead.
```

```
[15]: from cleverhans.attacks import FastGradientMethod
      fgsm = FastGradientMethod(wrap, sess=session)
```

```
[16]: x = tf.placeholder(tf.float32, shape=(None, 2304))
      y = tf.placeholder(tf.float32, shape=(None, 2))
```

```
[17]: fgsm_rate = 0.08
      fgsm_params = {'eps': fgsm_rate,'clip_min': 0.,'clip_max': 1.}
```

```
[18]: test_images_mal = x_test[MAL].copy()
```

```
[19]: adv_x = fgsm.generate_np(test_images_mal, **fgsm_params)
```

```
[INFO 2020-11-15 17:08:00,706 cleverhans] Constructing new graph for attack
FastGradientMethod

WARNING:tensorflow:From C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\cleverhans\attacks\__init__.py:283: to_float (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use `tf.cast` instead.
WARNING:tensorflow:From C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\cleverhans\utils_tf.py:624: The name tf.assert_greater_equal is
deprecated. Please use tf.compat.v1.assert_greater_equal instead.

WARNING:tensorflow:From C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\cleverhans\utils_tf.py:615: The name tf.assert_less_equal is
deprecated. Please use tf.compat.v1.assert_less_equal instead.

WARNING:tensorflow:From C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\cleverhans\compat.py:124: calling
softmax_cross_entropy_with_logits_v2_helper (from tensorflow.python.ops.nn_ops)
with dim is deprecated and will be removed in a future version.
Instructions for updating:
dim is deprecated, use axis instead
```

```
[20]: for i in range(test_images_mal.shape[0]):
          for j in range(test_images_mal.shape[1]):
              for k in range(test_images_mal.shape[2]):
                  if test_images_mal[i][j][k]==1:
                      adv_x[i][j][k] = 1
```

```
[21]: network.evaluate(test_images_mal, test_labels[MAL], batch_size=128)
```

```
1667/1667 [==============================] - 0s 76us/step
```

```
[21]: [0.36963911920374715, 0.9064187407493591]
```

```
[22]: network.evaluate(adv_x, test_labels[MAL], batch_size=128)
```

```
1667/1667 [==============================] - 0s 38us/step
```

```
[22]: [24.913951785829372, 0.09298140555620193]
```

```
[23]: x_test_after_attack_FGSM = x_test.copy()
```

```
[24]: x_test_after_attack_FGSM[MAL] = adv_x
```

```
[25]: network.evaluate(x_test, test_labels, batch_size=128)
```

```
5000/5000 [==============================] - 0s 43us/step
```

```
[25]: [0.19247934894561766, 0.9476000070571899]
```

```
[26]: network.evaluate(x_test_after_attack_FGSM, test_labels, batch_size=128)
```

```
5000/5000 [==============================] - 0s 42us/step
```

```
[26]: [8.375553169500828, 0.6764000058174133]
```

2.1 SAVE ALL VARS

- X_TEST
- Y_TEST
- X_TEST_AFTER_ATTACK_FGSM

```
[27]: np.save('./ATTACKS/FGSM/X_TEST_FGSM.npy',x_test)
      np.save('./ATTACKS/FGSM/Y_TEST_FGSM.npy',y_test)
      np.save('./ATTACKS/FGSM/X_TEST_ATTACKED_FGSM.npy',x_test_after_attack_FGSM)
      network.save('./ATTACKS/FGSM/FGSM_CLASSIFIER_USED.h5py')
```

3 LOAD MODEL

[28]:
```python
from keras import models
from keras import layers
```

[29]:
```python
network = keras.models.load_model('CLASSIFIER.h5py')
network.summary()
```

```
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
_________________________________________________________________
```

4 JSMA

[30]:
```
from cleverhans.attacks import SaliencyMapMethod
from cleverhans.utils_keras import KerasModelWrapper
wrap = KerasModelWrapper(network)
jsma = SaliencyMapMethod(wrap, sess=session)
```

[31]:
```
# x = tf.placeholder(tf.float32, shape=(None, 2304))
# y = tf.placeholder(tf.float32, shape=(None, 2))
```

[32]:
```
jsma_params = {'theta': 1.,
               'gamma': 0.1,
               'clip_min': 0.,
               'clip_max': 1.,
               'y_target': None}
```

[33]:
```
y_test.shape
```

[33]:
```
(5000, 1)
```

[34]:
```
from keras.utils import to_categorical #this just converts the labels to␣
 ↪one-hot class
test_labels = to_categorical(y_test)
```

[35]:
```
x_test_afterattack=np.zeros(x_test.shape)

crafted=[]
```

[36]:
```
nb_classes = 2
```

[37]:
```
x_test_mal_noisy = []
x_test_mal_noisy_idx = []
for i in range(x_test.shape[0]):
    if i in MAL:
        sample = x_test[i: i + 1]
        one_hot_target = np.zeros((1, nb_classes), dtype=np.float32)
        one_hot_target[0, 0] = 1
        jsma_params['y_target'] = one_hot_target
        adv_x = jsma.generate_np(sample, **jsma_params)
        x_test_afterattack[i]=adv_x
    else:
        x_test_afterattack[i] = x_test[i]
    if i%100 == 0:
        print("AT : " + str(i))
```

```
[INFO 2020-11-15 17:08:12,777 cleverhans] Constructing new graph for attack
SaliencyMapMethod
```

```
AT : 0
WARNING:tensorflow:From C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\cleverhans\attacks_tf.py:446: The name tf.mod is deprecated. Please use
tf.math.mod instead.

WARNING:tensorflow:From C:\Users\Pitch\.conda\envs\tf1-gpu\lib\site-
packages\cleverhans\attacks_tf.py:447: The name tf.floordiv is deprecated.
Please use tf.math.floordiv instead.

AT : 100
AT : 200
AT : 300
AT : 400
AT : 500
AT : 600
AT : 700
AT : 800
AT : 900
AT : 1000
AT : 1100
AT : 1200
AT : 1300
AT : 1400
AT : 1500
AT : 1600
AT : 1700
AT : 1800
AT : 1900
AT : 2000
AT : 2100
AT : 2200
AT : 2300
AT : 2400
AT : 2500
AT : 2600
AT : 2700
AT : 2800
AT : 2900
AT : 3000
AT : 3100
AT : 3200
AT : 3300
AT : 3400
AT : 3500
AT : 3600
AT : 3700
AT : 3800
AT : 3900
AT : 4000
AT : 4100
AT : 4200
AT : 4300
AT : 4400
AT : 4500
AT : 4600
AT : 4700
AT : 4800
AT : 4900
```

```
[38]: # adv_x = jsma.generate_np(x_test[MAL], **jsma_params)
```

```
[39]: x_test_afterattack.shape
```

```
[39]: (5000, 48, 48)
```

```
[40]: network.evaluate(x_test, test_labels, batch_size=128)
```

```
5000/5000 [==============================] - 0s 50us/step
```

```
[40]: [0.19247934894561766, 0.9476000070571899]
```

```
[41]: network.evaluate(x_test_afterattack, test_labels, batch_size=128)
```

```
5000/5000 [==============================] - 0s 39us/step
```

```
[41]: [0.8315977962493897, 0.6453999876976013]
```

```
[42]: x_test_after_attack_JSMA = x_test_afterattack.copy()
```

4.1 SAVE ALL VARS

- X_TEST
- Y_TEST
- X_TEST_AFTER_ATTACK_FGSM

[43]:
```
np.save('./ATTACKS/JSMA/X_TEST_JSMA.npy',x_test)
np.save('./ATTACKS/JSMA/Y_TEST_JSMA.npy',y_test)
np.save('./ATTACKS/JSMA/X_TEST_ATTACKED_JSMA.npy',x_test_after_attack_JSMA)
network.save('./ATTACKS/JSMA/JSMA_CLASSIFIER_USED.h5py')
```

5 Pretty straightforward. Talk about Attacks, Drop in acc. Rest dealt in Attack EDA

[ ]:
```

```