FCN-FFA-30E-14L-basis-test-03

March 24, 2021

1 Are Relations Relevant in CNNs? *A Study Based on a Facial Dataset*

1.1 Testing FCN with Features Further Apart *(30 Epochs - 13 Layers)*

1.1.1 Imports, Seed, GPU integration

[1]:
```
import numpy as np
import random
import tensorflow as tf
```

[2]:
```
# Seeds for better reproducibility
seed = 42
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)
```

[3]:
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
%matplotlib inline
```

[4]:
```
physical_devices = tf.config.experimental.list_physical_devices('GPU')
print("Num GPUs Available: ", len(physical_devices))
tf.config.experimental.set_memory_growth(physical_devices[0], True)
```

```
Num GPUs Available:  1
```

1.1.2 Data preparation

[5]:
```
test_path = '../../../picasso_dataset/basis-data/middle/test'
```

[6]:
```
test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.
↪vgg16.preprocess_input) \
    .flow_from_directory(directory=test_path, target_size=(224,224),␣
↪classes=['no_face', 'face'], batch_size=10, shuffle=False)
```

```
Found 3000 images belonging to 2 classes.
```

[7]:
```
assert test_batches.n == 3000
assert test_batches.num_classes == 2
```

1.1.3 Loading the trained FCN

[8]:
```
filename='../models/FCN-FFA-30E-14L-03.h5'
loaded_model = load_model(filename)
```

1.1.4 Accuracy and loss of the trained model

[9]:
```
scores = loaded_model.evaluate(test_batches, verbose=2)
print("Accuracy: %.2f%%" % (scores[1]*100))
print("Loss: %.2f%%" % (scores[0]*100))
```

```
300/300 - 7s - loss: 0.7165 - accuracy: 0.8553
Accuracy: 85.53%
Loss: 71.65%
```

1.1.5 Testing the FCN

[10]:
```
predictions = loaded_model.predict(x=test_batches, steps=len(test_batches),␣
↪verbose=0)
```

1.1.6 Index of wrongly predicted pictures

[11]:
```
y_true=test_batches.classes
y_pred=np.argmax(predictions, axis=-1)
cm = confusion_matrix(y_true = y_true, y_pred = y_pred)
```

[12]:
```
face_but_predicted_no_face=[]
no_face_but_predicted_face=[]

for i in range(len(predictions)):
        if y_true[i] != y_pred[i]:
            if y_true[i] == 1:
                face_but_predicted_no_face.append(i+8001-1500) #Index of file␣
↪on disk
            else:
                no_face_but_predicted_face.append(i+8001) #Index of file on disk
```

```python
print("Data from class 'face', that was wrongly predicted as 'no-face' [",␣
 ↪len(face_but_predicted_no_face), "] :")
print(face_but_predicted_no_face)
print("-----------------------------------------------------------------------------------------
print("Data from class 'no-face', that was wrongly predicted as 'face' [",␣
 ↪len(no_face_but_predicted_face), "] :")
print(no_face_but_predicted_face)
```

```
Data from class 'face', that was wrongly predicted as 'no-face' [ 432 ] :
[8001, 8006, 8013, 8014, 8015, 8018, 8019, 8020, 8024, 8027, 8031, 8034, 8035,
8036, 8037, 8038, 8039, 8040, 8046, 8049, 8050, 8054, 8055, 8056, 8057, 8060,
8061, 8062, 8064, 8065, 8067, 8068, 8070, 8074, 8075, 8076, 8078, 8081, 8084,
8085, 8087, 8088, 8091, 8094, 8095, 8099, 8112, 8113, 8120, 8124, 8126, 8127,
8130, 8132, 8139, 8146, 8147, 8148, 8149, 8150, 8152, 8154, 8157, 8162, 8165,
8168, 8170, 8173, 8177, 8183, 8186, 8188, 8189, 8191, 8195, 8200, 8217, 8219,
8228, 8233, 8235, 8240, 8250, 8252, 8256, 8264, 8267, 8273, 8279, 8289, 8290,
8293, 8296, 8297, 8298, 8299, 8302, 8303, 8305, 8307, 8308, 8311, 8315, 8322,
8326, 8327, 8333, 8337, 8341, 8342, 8343, 8350, 8363, 8373, 8374, 8378, 8380,
8381, 8388, 8393, 8394, 8397, 8400, 8401, 8407, 8408, 8413, 8414, 8415, 8421,
8423, 8425, 8431, 8434, 8436, 8443, 8444, 8449, 8455, 8457, 8462, 8465, 8472,
8474, 8476, 8478, 8481, 8487, 8492, 8493, 8494, 8495, 8499, 8501, 8505, 8508,
8512, 8513, 8514, 8516, 8521, 8524, 8525, 8526, 8530, 8531, 8554, 8557, 8565,
8567, 8572, 8573, 8578, 8579, 8582, 8583, 8584, 8602, 8603, 8604, 8608, 8612,
8613, 8616, 8621, 8623, 8625, 8629, 8636, 8642, 8643, 8645, 8646, 8650, 8657,
8661, 8663, 8664, 8668, 8669, 8672, 8674, 8687, 8692, 8695, 8702, 8704, 8716,
8718, 8726, 8734, 8735, 8737, 8738, 8739, 8740, 8743, 8744, 8750, 8760, 8771,
8777, 8784, 8785, 8786, 8788, 8791, 8798, 8809, 8814, 8815, 8819, 8823, 8829,
8831, 8836, 8840, 8847, 8849, 8853, 8855, 8863, 8865, 8867, 8869, 8870, 8873,
8882, 8887, 8888, 8890, 8891, 8897, 8898, 8900, 8902, 8911, 8912, 8913, 8914,
8917, 8920, 8923, 8929, 8932, 8933, 8934, 8939, 8940, 8942, 8945, 8947, 8948,
8952, 8955, 8957, 8959, 8963, 8964, 8966, 8969, 8973, 8978, 8979, 8980, 8985,
8990, 8992, 8995, 9000, 9001, 9009, 9011, 9013, 9018, 9021, 9030, 9032, 9034,
9043, 9049, 9056, 9058, 9064, 9067, 9068, 9071, 9073, 9074, 9077, 9082, 9085,
9086, 9087, 9088, 9094, 9102, 9103, 9105, 9106, 9111, 9113, 9116, 9118, 9123,
9124, 9128, 9131, 9132, 9133, 9138, 9140, 9145, 9146, 9148, 9152, 9154, 9159,
9161, 9162, 9167, 9172, 9175, 9179, 9183, 9184, 9185, 9189, 9192, 9195, 9207,
9211, 9216, 9217, 9218, 9225, 9228, 9233, 9234, 9237, 9239, 9243, 9244, 9249,
9259, 9265, 9266, 9268, 9273, 9277, 9283, 9284, 9285, 9286, 9293, 9295, 9299,
9302, 9303, 9306, 9307, 9325, 9334, 9335, 9337, 9345, 9346, 9350, 9367, 9368,
9371, 9372, 9374, 9375, 9377, 9379, 9381, 9382, 9385, 9387, 9389, 9393, 9399,
9400, 9403, 9405, 9407, 9413, 9415, 9418, 9419, 9435, 9438, 9439, 9441, 9443,
9444, 9446, 9450, 9452, 9457, 9459, 9462, 9463, 9464, 9477, 9485, 9491, 9495,
9496, 9497, 9498]
--------------------------------------------------------------------------------
--------------
Data from class 'no-face', that was wrongly predicted as 'face' [ 2 ] :
[8549, 9467]
```

1.1.7 Confusion matrix

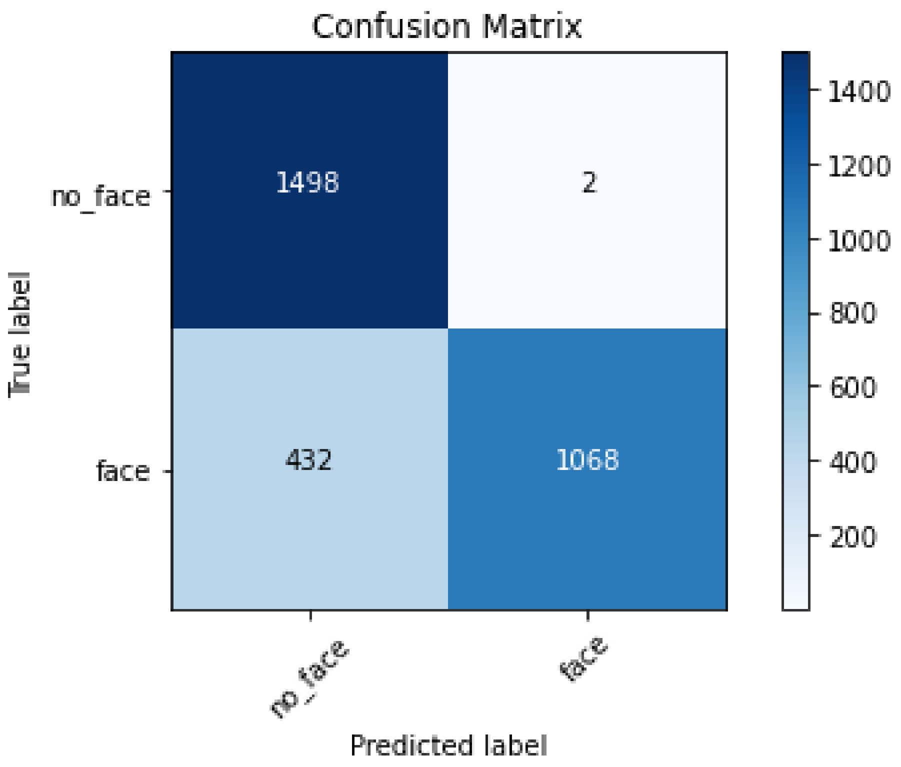
[13]:
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

[14]:
```
test_batches.class_indices
```

[14]:
```
{'no_face': 0, 'face': 1}
```

[15]:
```
cm_plot_labels = ['no_face','face']
plot_confusion_matrix(cm=cm, classes=cm_plot_labels, title='Confusion Matrix')
```

```
Confusion matrix, without normalization
[[1498    2]
 [ 432 1068]]
```


Confusion Matrix
True label
no_face
face
1498
2
432
1068
Predicted label
no_face
face
1400
1200
1000
800
600
400
200