FCN-FFA-30E-14L-basis-test-01

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-01.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: 0.9878 - accuracy: 0.7877
Accuracy: 78.77%
Loss: 98.78%
```

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
```

```
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' [ 601 ] :
[8001, 8008, 8009, 8010, 8012, 8013, 8014, 8015, 8016, 8017, 8018, 8020, 8021,
8024, 8028, 8029, 8032, 8034, 8035, 8036, 8038, 8040, 8042, 8043, 8044, 8046,
8049, 8050, 8052, 8053, 8054, 8056, 8057, 8059, 8062, 8065, 8067, 8073, 8074,
8075, 8076, 8077, 8078, 8081, 8084, 8087, 8091, 8094, 8095, 8101, 8103, 8107,
8111, 8113, 8118, 8120, 8126, 8132, 8136, 8139, 8144, 8146, 8147, 8151, 8152,
8155, 8157, 8159, 8160, 8162, 8165, 8166, 8168, 8169, 8170, 8171, 8173, 8174,
8177, 8180, 8182, 8185, 8188, 8189, 8190, 8191, 8195, 8198, 8205, 8208, 8213,
8214, 8215, 8217, 8218, 8227, 8232, 8233, 8237, 8240, 8241, 8247, 8248, 8250,
8252, 8256, 8257, 8258, 8259, 8260, 8265, 8267, 8271, 8275, 8279, 8282, 8283,
8290, 8293, 8294, 8296, 8298, 8299, 8305, 8306, 8310, 8312, 8313, 8314, 8315,
8316, 8319, 8320, 8321, 8322, 8324, 8325, 8326, 8327, 8330, 8331, 8333, 8337,
8342, 8343, 8346, 8347, 8355, 8356, 8357, 8361, 8371, 8374, 8377, 8381, 8384,
8387, 8388, 8393, 8397, 8401, 8407, 8410, 8413, 8414, 8415, 8416, 8421, 8425,
8427, 8430, 8431, 8435, 8436, 8438, 8439, 8440, 8442, 8448, 8449, 8452, 8455,
8457, 8458, 8462, 8464, 8465, 8466, 8468, 8472, 8475, 8480, 8481, 8483, 8486,
8487, 8489, 8491, 8495, 8499, 8502, 8503, 8504, 8505, 8508, 8510, 8511, 8512,
8514, 8516, 8517, 8521, 8524, 8525, 8526, 8528, 8533, 8542, 8543, 8544, 8549,
8550, 8551, 8555, 8556, 8557, 8558, 8561, 8563, 8564, 8567, 8571, 8572, 8573,
8575, 8576, 8578, 8579, 8582, 8583, 8584, 8592, 8593, 8594, 8597, 8601, 8603,
8604, 8606, 8607, 8611, 8614, 8615, 8617, 8621, 8625, 8629, 8636, 8638, 8639,
8642, 8644, 8647, 8650, 8654, 8657, 8658, 8661, 8662, 8663, 8664, 8668, 8674,
8675, 8677, 8680, 8682, 8686, 8687, 8688, 8690, 8692, 8693, 8695, 8696, 8697,
8698, 8699, 8706, 8709, 8712, 8713, 8716, 8718, 8719, 8722, 8725, 8732, 8733,
8734, 8735, 8737, 8738, 8740, 8742, 8743, 8744, 8747, 8749, 8750, 8754, 8758,
8760, 8764, 8766, 8767, 8768, 8769, 8770, 8771, 8773, 8776, 8777, 8780, 8781,
8784, 8785, 8786, 8788, 8792, 8793, 8796, 8798, 8799, 8801, 8807, 8808, 8809,
8814, 8815, 8821, 8823, 8827, 8829, 8831, 8833, 8835, 8837, 8839, 8843, 8845,
8850, 8853, 8859, 8860, 8869, 8870, 8876, 8882, 8886, 8887, 8890, 8892, 8897,
8898, 8899, 8902, 8908, 8909, 8912, 8913, 8917, 8920, 8923, 8924, 8927, 8928,
8930, 8931, 8933, 8934, 8935, 8939, 8945, 8948, 8950, 8951, 8955, 8956, 8958,
8961, 8962, 8964, 8966, 8972, 8975, 8977, 8978, 8988, 8989, 8992, 8993, 8995,
9003, 9004, 9006, 9007, 9013, 9014, 9015, 9018, 9021, 9022, 9023, 9024, 9028,
9030, 9032, 9037, 9041, 9052, 9054, 9057, 9058, 9061, 9063, 9067, 9068, 9073,
9074, 9076, 9077, 9078, 9079, 9081, 9084, 9085, 9094, 9095, 9102, 9103, 9105,
9106, 9107, 9109, 9113, 9115, 9116, 9118, 9122, 9124, 9127, 9128, 9131, 9137,
9138, 9141, 9148, 9149, 9150, 9151, 9154, 9162, 9163, 9165, 9166, 9169, 9170,
9171, 9172, 9173, 9174, 9175, 9179, 9183, 9184, 9185, 9186, 9188, 9189, 9194,
9195, 9202, 9203, 9206, 9207, 9210, 9216, 9221, 9225, 9228, 9229, 9234, 9237,
9239, 9243, 9246, 9250, 9251, 9255, 9256, 9259, 9260, 9262, 9273, 9275, 9277,
9280, 9281, 9283, 9284, 9285, 9287, 9291, 9292, 9294, 9295, 9298, 9300, 9302,
9303, 9305, 9307, 9309, 9312, 9316, 9321, 9323, 9325, 9329, 9334, 9335, 9337,
9338, 9344, 9345, 9346, 9352, 9359, 9360, 9368, 9371, 9375, 9376, 9378, 9379,
9381, 9382, 9383, 9386, 9387, 9389, 9392, 9393, 9394, 9397, 9400, 9401, 9402,
9403, 9407, 9413, 9414, 9415, 9416, 9419, 9423, 9427, 9428, 9431, 9432, 9435,
9436, 9438, 9442, 9443, 9444, 9445, 9446, 9449, 9450, 9451, 9452, 9454, 9459,
9460, 9462, 9463, 9464, 9465, 9466, 9471, 9475, 9479, 9483, 9484, 9487, 9493,
9495, 9497, 9498]
--------------------------------------------------------------------------------
--------------
Data from class 'no-face', that was wrongly predicted as 'face' [ 36 ] :
[8094, 8106, 8113, 8126, 8221, 8248, 8276, 8348, 8407, 8501, 8600, 8621, 8650,
8698, 8749, 8840, 8909, 8910, 8925, 8931, 8934, 8935, 8958, 8991, 9025, 9037,
9074, 9085, 9188, 9216, 9229, 9234, 9265, 9322, 9382, 9456]
```

1.1.7 Confusion matrix

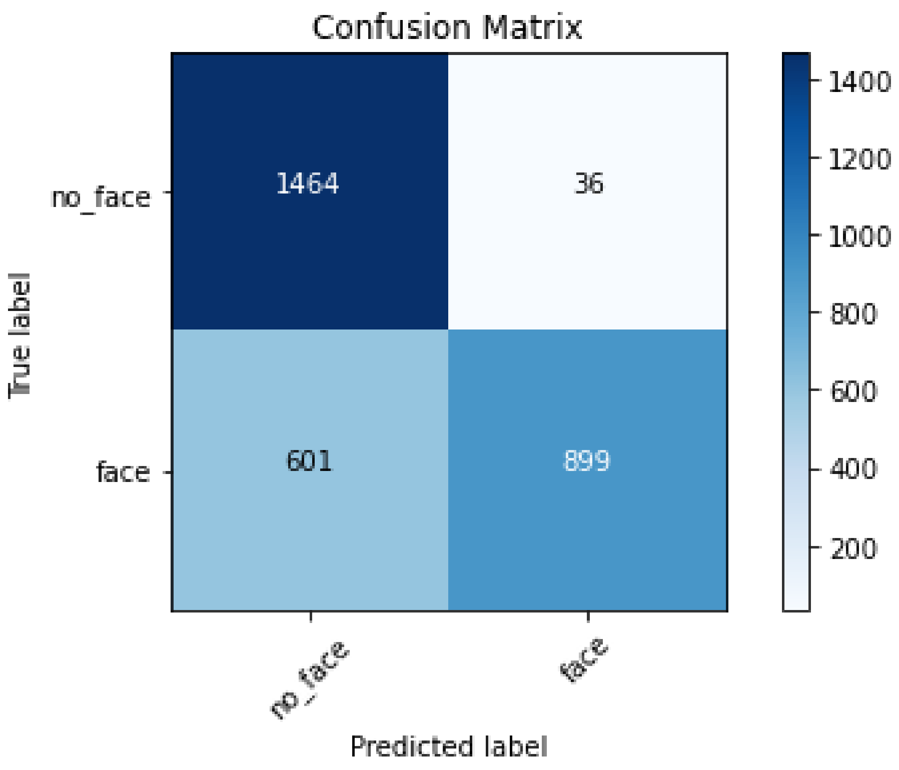
[13]:
```python
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    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
[[1464   36]
 [ 601  899]]
```


Confusion Matrix
True label
no_face
face
Predicted label
no_face
face