FCN-Baseline-30E-14L-FFA-test-02

March 24, 2021

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

1.1 Testing Baseline FCN *(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/FFA-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-B-30E-14L-02.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: 0.4056 - accuracy: 0.8780
Accuracy: 87.80%
Loss: 40.56%
```

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' [ 276 ] :
[8004, 8007, 8008, 8009, 8011, 8015, 8023, 8025, 8030, 8040, 8053, 8056, 8060,
8069, 8077, 8082, 8094, 8096, 8106, 8113, 8115, 8128, 8144, 8151, 8154, 8157,
8162, 8169, 8171, 8172, 8174, 8175, 8182, 8189, 8197, 8202, 8208, 8209, 8215,
8225, 8227, 8233, 8237, 8244, 8246, 8248, 8272, 8274, 8291, 8301, 8307, 8312,
8314, 8321, 8322, 8323, 8325, 8327, 8338, 8339, 8341, 8345, 8347, 8351, 8354,
8365, 8368, 8373, 8378, 8379, 8380, 8397, 8399, 8402, 8407, 8419, 8427, 8431,
8435, 8439, 8442, 8448, 8451, 8452, 8457, 8464, 8468, 8478, 8484, 8491, 8497,
8502, 8503, 8513, 8520, 8522, 8533, 8535, 8536, 8541, 8545, 8551, 8558, 8560,
8561, 8563, 8571, 8572, 8575, 8578, 8580, 8583, 8588, 8594, 8598, 8602, 8608,
8619, 8622, 8623, 8633, 8644, 8645, 8654, 8655, 8674, 8694, 8695, 8696, 8700,
8705, 8707, 8715, 8722, 8744, 8746, 8747, 8748, 8751, 8752, 8759, 8766, 8772,
8781, 8783, 8793, 8800, 8804, 8808, 8816, 8827, 8842, 8845, 8846, 8853, 8854,
8870, 8893, 8894, 8895, 8899, 8909, 8924, 8928, 8930, 8942, 8948, 8949, 8952,
8954, 8962, 8963, 8967, 8968, 8971, 8972, 8978, 8985, 8996, 9000, 9001, 9003,
9006, 9008, 9011, 9025, 9026, 9032, 9035, 9037, 9038, 9039, 9045, 9049, 9051,
9052, 9057, 9061, 9069, 9070, 9071, 9086, 9091, 9112, 9114, 9118, 9125, 9126,
9129, 9132, 9135, 9144, 9145, 9146, 9155, 9159, 9162, 9167, 9170, 9174, 9183,
9187, 9188, 9189, 9208, 9216, 9222, 9248, 9258, 9265, 9266, 9267, 9270, 9280,
9284, 9291, 9298, 9304, 9305, 9307, 9312, 9333, 9339, 9340, 9342, 9343, 9349,
9354, 9359, 9369, 9372, 9375, 9377, 9378, 9382, 9385, 9388, 9390, 9391, 9392,
9398, 9422, 9423, 9426, 9427, 9438, 9440, 9446, 9450, 9451, 9454, 9475, 9476,
9482, 9493, 9496]
--------------------------------------------------------------------------------
--------------
Data from class 'no-face', that was wrongly predicted as 'face' [ 90 ] :
[8007, 8013, 8015, 8029, 8063, 8068, 8083, 8144, 8162, 8174, 8193, 8199, 8244,
8256, 8263, 8287, 8299, 8306, 8329, 8334, 8336, 8338, 8354, 8355, 8372, 8386,
8415, 8420, 8428, 8431, 8432, 8443, 8448, 8451, 8458, 8463, 8466, 8476, 8488,
8497, 8511, 8512, 8530, 8535, 8539, 8541, 8549, 8602, 8604, 8618, 8643, 8652,
8657, 8681, 8697, 8726, 8727, 8774, 8802, 8814, 8820, 8850, 8851, 8883, 8885,
8894, 8958, 8979, 8994, 9058, 9061, 9082, 9090, 9097, 9109, 9137, 9146, 9178,
9200, 9234, 9236, 9241, 9246, 9272, 9338, 9340, 9388, 9395, 9414, 9445]
```

1.1.7 Confusion matrix

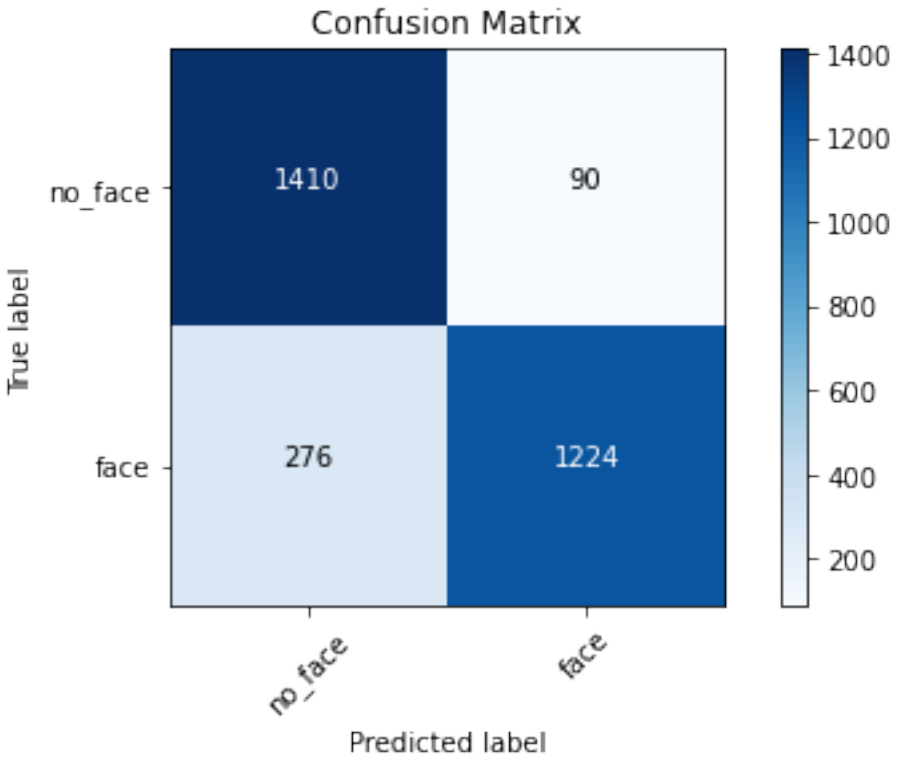
[13]:
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    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
[[1410   90]
 [ 276 1224]]
```


Confusion Matrix
1410
90
276
1224
no_face
face
True label
no_face
face
Predicted label
1400
1200
1000
800
600
400
200