CNN-FCT-20E-15L-shift-test-01

March 24, 2021

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

1.1 Testing CNN with Features Closer Together *(20 Epochs - 15 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/FCT-data/shifted/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 CNN

[8]:
```
filename='../models/CNN-FCT-20E-15L-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: 0.6838 - accuracy: 0.8373
Accuracy: 83.73%
Loss: 68.38%
```

1.1.5 Testing the CNN

[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' [ 474 ] :
[8005, 8010, 8012, 8017, 8019, 8020, 8021, 8022, 8023, 8026, 8034, 8035, 8036,
8039, 8049, 8054, 8056, 8065, 8068, 8075, 8090, 8095, 8096, 8108, 8113, 8116,
8120, 8122, 8133, 8135, 8136, 8137, 8142, 8144, 8146, 8149, 8153, 8160, 8161,
8162, 8172, 8179, 8182, 8186, 8188, 8193, 8194, 8197, 8200, 8201, 8207, 8217,
8218, 8219, 8225, 8231, 8240, 8246, 8254, 8259, 8266, 8280, 8291, 8293, 8295,
8296, 8298, 8299, 8302, 8304, 8309, 8312, 8318, 8324, 8326, 8327, 8331, 8343,
8348, 8349, 8354, 8356, 8359, 8362, 8366, 8367, 8370, 8371, 8375, 8378, 8382,
8391, 8402, 8404, 8406, 8409, 8411, 8426, 8434, 8435, 8439, 8440, 8441, 8444,
8448, 8449, 8450, 8456, 8462, 8471, 8473, 8476, 8478, 8480, 8482, 8484, 8487,
8490, 8491, 8494, 8496, 8498, 9002, 9003, 9005, 9006, 9008, 9009, 9010, 9014,
9015, 9016, 9019, 9020, 9022, 9023, 9024, 9025, 9027, 9029, 9030, 9032, 9033,
9034, 9035, 9036, 9038, 9039, 9041, 9042, 9043, 9045, 9046, 9047, 9048, 9049,
9052, 9053, 9058, 9059, 9061, 9062, 9063, 9064, 9065, 9066, 9067, 9069, 9071,
9072, 9073, 9074, 9075, 9077, 9078, 9080, 9083, 9085, 9088, 9092, 9097, 9098,
9100, 9101, 9102, 9103, 9104, 9105, 9106, 9107, 9108, 9109, 9111, 9112, 9113,
9114, 9115, 9118, 9119, 9120, 9121, 9122, 9123, 9125, 9129, 9130, 9131, 9132,
9135, 9136, 9137, 9139, 9140, 9141, 9142, 9143, 9145, 9146, 9147, 9148, 9149,
9151, 9152, 9155, 9156, 9158, 9159, 9166, 9167, 9168, 9170, 9172, 9173, 9174,
9177, 9179, 9180, 9181, 9182, 9183, 9184, 9186, 9189, 9190, 9191, 9192, 9194,
9195, 9196, 9198, 9199, 9200, 9202, 9203, 9208, 9209, 9210, 9211, 9213, 9214,
9215, 9218, 9219, 9220, 9221, 9223, 9224, 9227, 9228, 9230, 9231, 9232, 9233,
9234, 9235, 9236, 9237, 9238, 9239, 9240, 9241, 9243, 9244, 9247, 9248, 9249,
9250, 9251, 9252, 9253, 9254, 9256, 9257, 9258, 9259, 9261, 9262, 9263, 9264,
9265, 9266, 9267, 9268, 9269, 9270, 9272, 9273, 9275, 9276, 9277, 9279, 9280,
9281, 9282, 9283, 9284, 9285, 9287, 9288, 9290, 9291, 9293, 9295, 9297, 9298,
9299, 9300, 9301, 9302, 9303, 9304, 9306, 9307, 9308, 9310, 9311, 9314, 9315,
9316, 9319, 9320, 9321, 9322, 9323, 9324, 9325, 9326, 9328, 9329, 9331, 9332,
9333, 9334, 9335, 9336, 9338, 9341, 9342, 9345, 9346, 9347, 9348, 9349, 9357,
9358, 9360, 9361, 9362, 9363, 9364, 9365, 9366, 9367, 9368, 9369, 9370, 9372,
9373, 9376, 9377, 9378, 9379, 9380, 9381, 9383, 9384, 9385, 9386, 9387, 9388,
9389, 9390, 9391, 9392, 9393, 9394, 9395, 9397, 9399, 9401, 9402, 9403, 9404,
9405, 9406, 9407, 9408, 9409, 9410, 9411, 9412, 9413, 9414, 9415, 9416, 9421,
9422, 9423, 9425, 9426, 9427, 9429, 9430, 9431, 9432, 9433, 9434, 9435, 9436,
9438, 9439, 9440, 9441, 9442, 9443, 9444, 9445, 9446, 9447, 9448, 9449, 9450,
9452, 9453, 9454, 9455, 9456, 9458, 9459, 9460, 9461, 9462, 9463, 9464, 9467,
9472, 9473, 9474, 9476, 9477, 9478, 9479, 9482, 9483, 9484, 9487, 9488, 9490,
9492, 9496, 9497, 9498, 9499, 9500]
-------------------------------------------------------------------------------
```

```
--------------
Data from class 'no-face', that was wrongly predicted as 'face' [ 14 ] :
[8018, 8051, 8077, 8154, 8297, 8301, 8364, 8380, 8414, 9190, 9239, 9268, 9280,
9290]
```

1.1.7 Confusion matrix

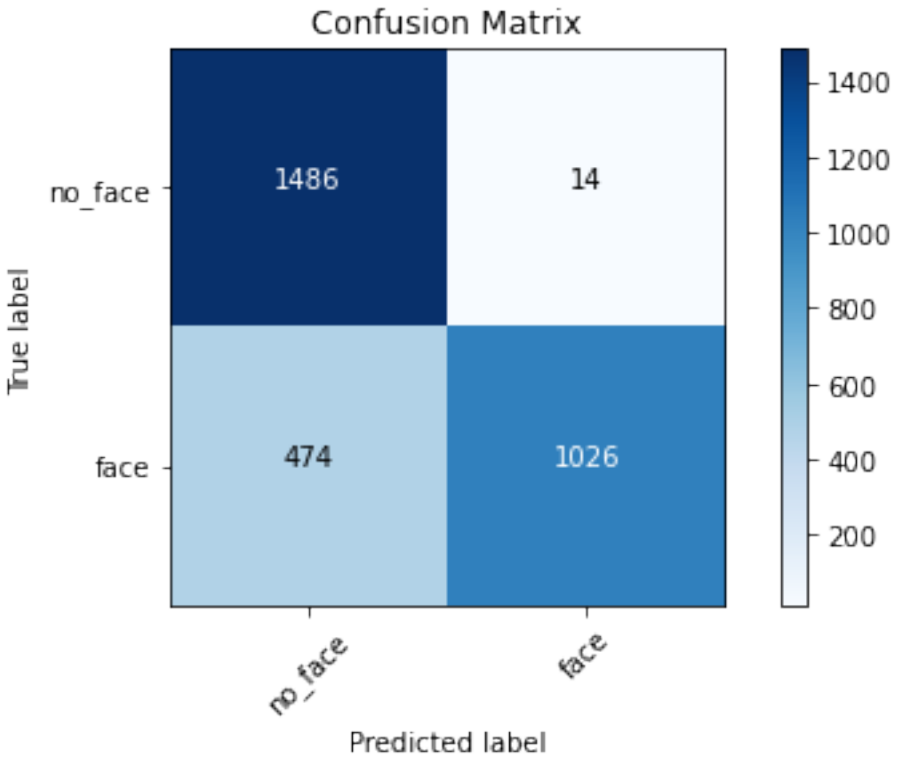
[13]:
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    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
[[1486   14]
 [ 474 1026]]
```


Confusion Matrix
1486
14
474
1026
no_face
face
True label
no_face
face
Predicted label
1400
1200
1000
800
600
400
200