CNN-FFA-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 Further Apart *(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/FFA-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-FFA-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: 0.8765 - accuracy: 0.8020
Accuracy: 80.20%
Loss: 87.65%
```

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' [ 589 ] :
[8001, 8002, 8003, 8004, 8006, 8007, 8009, 8011, 8012, 8013, 8014, 8017, 8019,
8020, 8021, 8022, 8023, 8024, 8025, 8028, 8030, 8031, 8032, 8033, 8035, 8037,
8038, 8039, 8040, 8041, 8043, 8044, 8045, 8046, 8047, 8048, 8050, 8051, 8053,
8054, 8055, 8056, 8057, 8058, 8060, 8061, 8062, 8063, 8064, 8065, 8066, 8067,
8069, 8070, 8071, 8072, 8073, 8074, 8075, 8076, 8077, 8079, 8080, 8083, 8084,
8085, 8087, 8088, 8089, 8090, 8091, 8092, 8093, 8094, 8095, 8096, 8097, 8099,
8100, 8101, 8102, 8103, 8104, 8105, 8106, 8109, 8110, 8111, 8113, 8114, 8115,
8116, 8117, 8120, 8121, 8122, 8123, 8124, 8125, 8126, 8127, 8129, 8130, 8131,
8132, 8133, 8135, 8136, 8137, 8138, 8139, 8140, 8141, 8142, 8143, 8144, 8145,
8146, 8148, 8149, 8150, 8151, 8152, 8154, 8155, 8156, 8157, 8158, 8159, 8160,
8161, 8162, 8163, 8164, 8166, 8168, 8169, 8170, 8171, 8172, 8173, 8175, 8177,
8178, 8179, 8180, 8182, 8183, 8184, 8185, 8186, 8190, 8191, 8193, 8194, 8196,
8197, 8198, 8199, 8201, 8202, 8203, 8204, 8206, 8207, 8209, 8210, 8212, 8214,
8216, 8217, 8218, 8219, 8221, 8223, 8224, 8225, 8226, 8227, 8228, 8229, 8230,
8231, 8234, 8235, 8237, 8238, 8241, 8242, 8243, 8244, 8245, 8246, 8248, 8249,
8250, 8252, 8253, 8254, 8255, 8256, 8257, 8258, 8260, 8261, 8262, 8266, 8267,
8268, 8269, 8270, 8271, 8272, 8273, 8274, 8275, 8276, 8277, 8278, 8279, 8283,
8284, 8286, 8287, 8288, 8289, 8290, 8292, 8293, 8294, 8295, 8296, 8297, 8298,
8299, 8300, 8301, 8302, 8303, 8304, 8305, 8306, 8307, 8308, 8309, 8310, 8311,
8313, 8314, 8317, 8318, 8319, 8321, 8322, 8323, 8324, 8325, 8326, 8327, 8329,
8331, 8334, 8337, 8338, 8339, 8341, 8342, 8343, 8344, 8345, 8346, 8347, 8349,
8350, 8351, 8353, 8354, 8355, 8356, 8357, 8358, 8359, 8360, 8361, 8363, 8364,
8365, 8366, 8367, 8369, 8370, 8371, 8372, 8373, 8374, 8375, 8377, 8380, 8382,
8383, 8385, 8387, 8388, 8389, 8390, 8391, 8392, 8393, 8394, 8395, 8396, 8397,
8399, 8400, 8401, 8402, 8405, 8406, 8407, 8408, 8409, 8410, 8411, 8413, 8415,
8417, 8420, 8421, 8422, 8424, 8425, 8426, 8428, 8429, 8430, 8431, 8432, 8433,
8435, 8436, 8437, 8438, 8439, 8440, 8441, 8443, 8444, 8445, 8446, 8448, 8449,
8450, 8452, 8453, 8454, 8455, 8458, 8459, 8461, 8462, 8463, 8465, 8466, 8467,
8468, 8469, 8472, 8473, 8474, 8475, 8476, 8477, 8479, 8480, 8481, 8482, 8483,
8484, 8485, 8486, 8487, 8488, 8489, 8490, 8491, 8492, 8494, 8495, 8498, 8499,
9004, 9005, 9006, 9010, 9011, 9012, 9015, 9018, 9019, 9020, 9023, 9024, 9025,
9026, 9027, 9030, 9031, 9033, 9034, 9040, 9041, 9043, 9045, 9046, 9051, 9052,
9055, 9056, 9058, 9062, 9063, 9068, 9069, 9076, 9078, 9085, 9088, 9090, 9091,
9092, 9093, 9095, 9096, 9097, 9102, 9105, 9106, 9107, 9110, 9113, 9114, 9116,
9117, 9118, 9125, 9126, 9133, 9136, 9138, 9140, 9141, 9147, 9148, 9153, 9163,
9164, 9167, 9168, 9170, 9174, 9175, 9177, 9178, 9182, 9183, 9185, 9187, 9189,
9190, 9191, 9193, 9195, 9196, 9203, 9207, 9208, 9214, 9217, 9218, 9226, 9228,
9230, 9233, 9234, 9237, 9246, 9248, 9249, 9250, 9252, 9255, 9256, 9257, 9260,
9263, 9265, 9267, 9268, 9270, 9273, 9277, 9278, 9282, 9283, 9286, 9287, 9290,
9294, 9300, 9301, 9303, 9305, 9306, 9307, 9309, 9311, 9312, 9315, 9319, 9320,
9322, 9325, 9331, 9335, 9337, 9340, 9344, 9346, 9347, 9349, 9350, 9352, 9353,
9356, 9358, 9370, 9379, 9381, 9387, 9392, 9399, 9400, 9401, 9404, 9405, 9406,
9407, 9408, 9411, 9412, 9413, 9415, 9418, 9419, 9422, 9423, 9427, 9432, 9434,
9437, 9438, 9440, 9442, 9444, 9447, 9449, 9450, 9457, 9458, 9459, 9462, 9466,
9467, 9469, 9471, 9472, 9473, 9475, 9476, 9480, 9482, 9483, 9484, 9485, 9486,
9489, 9490, 9497, 9500]
--------------------------------------------------------------------------------
--------------
Data from class 'no-face', that was wrongly predicted as 'face' [ 5 ] :
[8475, 9287, 9359, 9421, 9426]
```

1.1.7 Confusion matrix

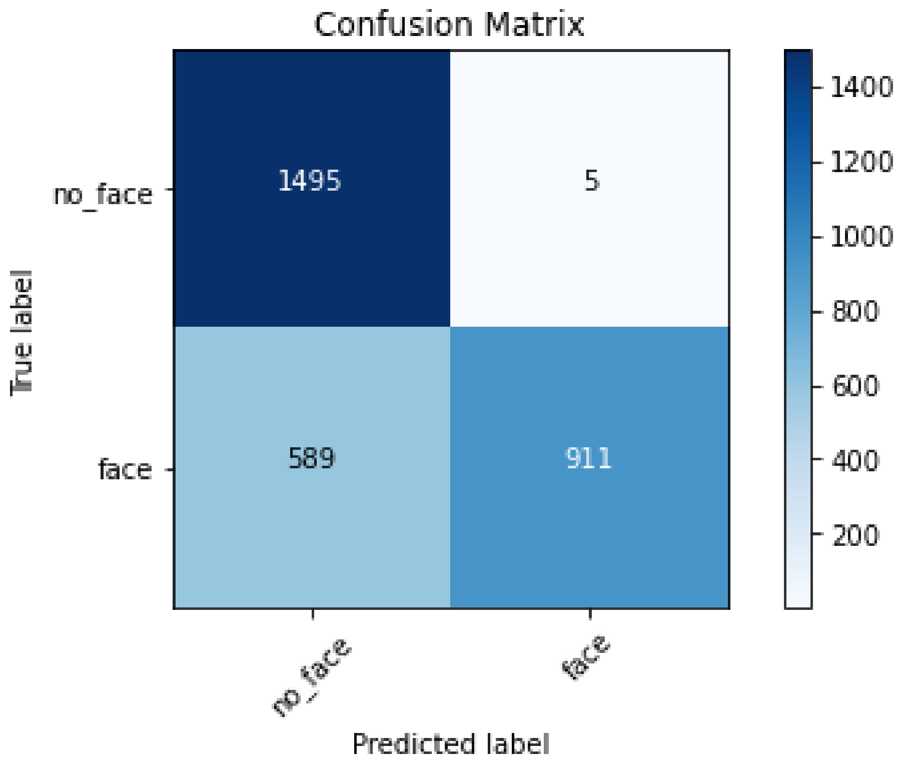
[13]:
```python
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    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]:
```python
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
[[1495    5]
 [ 589  911]]
```


Confusion Matrix
True label
no_face
face
no_face
face
Predicted label