

CNN-FCT-30E-13L-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 (*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/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-30E-13L-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: 0.8782 - accuracy: 0.8173
Accuracy: 81.73%
Loss: 87.82%

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 ] :
[8001, 8005, 8010, 8012, 8017, 8019, 8020, 8022, 8023, 8026, 8028, 8032, 8034,
8035, 8036, 8039, 8040, 8047, 8049, 8053, 8054, 8056, 8061, 8065, 8068, 8075,
8082, 8090, 8093, 8095, 8096, 8108, 8112, 8113, 8122, 8123, 8133, 8135, 8136,
8137, 8141, 8142, 8144, 8146, 8147, 8156, 8160, 8162, 8169, 8172, 8174, 8177,
8179, 8182, 8184, 8186, 8188, 8193, 8194, 8197, 8200, 8201, 8202, 8207, 8208,
8213, 8217, 8219, 8224, 8225, 8231, 8234, 8237, 8239, 8240, 8242, 8248, 8249,
8250, 8251, 8254, 8256, 8259, 8260, 8262, 8266, 8267, 8269, 8275, 8278, 8280,
8291, 8293, 8298, 8299, 8302, 8304, 8305, 8308, 8310, 8312, 8319, 8326, 8327,
8331, 8336, 8343, 8345, 8348, 8349, 8354, 8356, 8359, 8362, 8366, 8368, 8370,
8371, 8372, 8375, 8376, 8378, 8380, 8382, 8384, 8385, 8391, 8396, 8397, 8402,
8404, 8406, 8408, 8409, 8411, 8412, 8413, 8420, 8426, 8428, 8434, 8435, 8437,
8439, 8440, 8442, 8447, 8448, 8449, 8450, 8453, 8461, 8462, 8465, 8466, 8468,
8471, 8473, 8475, 8476, 8478, 8479, 8480, 8483, 8484, 8486, 8498, 9002, 9006,
9007, 9008, 9009, 9010, 9019, 9020, 9022, 9023, 9024, 9025, 9027, 9029, 9032,
9034, 9038, 9039, 9040, 9041, 9043, 9044, 9045, 9046, 9047, 9048, 9049, 9051,
9052, 9053, 9058, 9060, 9064, 9065, 9066, 9067, 9071, 9072, 9073, 9074, 9076,
9077, 9078, 9080, 9083, 9085, 9087, 9088, 9095, 9097, 9102, 9103, 9104, 9105,
9107, 9108, 9113, 9115, 9116, 9118, 9120, 9121, 9122, 9123, 9124, 9126, 9128,
9131, 9132, 9133, 9134, 9135, 9136, 9137, 9139, 9140, 9141, 9142, 9143, 9145,
9147, 9148, 9149, 9152, 9155, 9156, 9158, 9161, 9165, 9166, 9167, 9168, 9169,
9170, 9172, 9173, 9174, 9175, 9176, 9177, 9178, 9179, 9181, 9182, 9183, 9184,
9186, 9189, 9190, 9191, 9192, 9195, 9196, 9198, 9199, 9200, 9202, 9203, 9206,
9209, 9210, 9211, 9213, 9215, 9216, 9218, 9220, 9223, 9224, 9227, 9228, 9230,
9231, 9232, 9233, 9234, 9235, 9236, 9237, 9239, 9240, 9243, 9244, 9246, 9247,
9248, 9250, 9251, 9252, 9253, 9254, 9256, 9257, 9259, 9261, 9262, 9263, 9264,
9265, 9266, 9267, 9268, 9269, 9270, 9271, 9272, 9275, 9276, 9277, 9279, 9280,
9281, 9282, 9283, 9285, 9286, 9287, 9290, 9291, 9293, 9295, 9297, 9299, 9300,
9301, 9302, 9304, 9305, 9308, 9309, 9310, 9311, 9312, 9313, 9315, 9319, 9320,
9321, 9325, 9326, 9328, 9329, 9331, 9332, 9336, 9338, 9341, 9342, 9344, 9345,
9347, 9348, 9349, 9352, 9354, 9356, 9360, 9362, 9364, 9366, 9368, 9369, 9370,
9372, 9373, 9376, 9377, 9378, 9379, 9380, 9381, 9382, 9384, 9385, 9386, 9388,
9389, 9390, 9391, 9394, 9396, 9397, 9398, 9399, 9402, 9403, 9404, 9405, 9406,
9407, 9412, 9413, 9415, 9419, 9422, 9424, 9425, 9427, 9429, 9430, 9431, 9432,
9433, 9435, 9436, 9437, 9439, 9440, 9442, 9444, 9445, 9446, 9447, 9449, 9450,
9451, 9453, 9454, 9456, 9458, 9459, 9460, 9461, 9462, 9463, 9464, 9467, 9468,
9469, 9471, 9472, 9473, 9476, 9477, 9478, 9479, 9482, 9483, 9484, 9487, 9489,
9491, 9493, 9496, 9497, 9498, 9499]
-----

```

Data from class 'no-face', that was wrongly predicted as 'face' [74] :
[8007, 8021, 8041, 8051, 8062, 8077, 8091, 8101, 8102, 8134, 8145, 8154, 8182,
8185, 8195, 8199, 8202, 8211, 8220, 8266, 8276, 8281, 8289, 8295, 8297, 8301,
8328, 8329, 8358, 8360, 8364, 8380, 8389, 8399, 8403, 8414, 8423, 8454, 8459,
8487, 9034, 9039, 9062, 9066, 9078, 9110, 9116, 9128, 9133, 9138, 9139, 9148,
9153, 9177, 9194, 9197, 9227, 9262, 9268, 9272, 9316, 9321, 9334, 9349, 9350,
9437, 9443, 9447, 9457, 9461, 9466, 9472, 9477, 9491]

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
[[1426 74]
[474 1026]]

