# CNN-FFA-30E-13L-shift-test-02

March 24, 2021

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

### 1.1 Testing CNN 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/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-30E-13L-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: 1.0606 - accuracy: 0.7637
Accuracy: 76.37%
Loss: 106.06%
```

### 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' [ 609 ] :
[8002, 8003, 8006, 8007, 8009, 8011, 8014, 8019, 8020, 8021, 8023, 8024, 8030,
8031, 8032, 8035, 8036, 8038, 8039, 8040, 8041, 8045, 8046, 8048, 8051, 8053,
8054, 8055, 8056, 8057, 8058, 8060, 8062, 8063, 8066, 8067, 8069, 8071, 8072,
8073, 8074, 8075, 8076, 8077, 8079, 8083, 8084, 8087, 8088, 8091, 8092, 8094,
8097, 8100, 8101, 8102, 8104, 8109, 8111, 8112, 8113, 8114, 8115, 8116, 8117,
8119, 8120, 8121, 8122, 8123, 8125, 8127, 8129, 8131, 8132, 8133, 8135, 8136,
8137, 8144, 8145, 8148, 8149, 8150, 8151, 8152, 8154, 8155, 8157, 8158, 8160,
8161, 8162, 8163, 8170, 8171, 8172, 8173, 8175, 8176, 8178, 8181, 8182, 8185,
8186, 8187, 8188, 8190, 8191, 8196, 8197, 8198, 8199, 8202, 8204, 8207, 8209,
8210, 8212, 8214, 8218, 8219, 8220, 8224, 8225, 8227, 8230, 8235, 8237, 8238,
8242, 8243, 8245, 8246, 8248, 8249, 8250, 8252, 8254, 8257, 8258, 8259, 8262,
8263, 8267, 8270, 8271, 8272, 8273, 8274, 8275, 8276, 8277, 8279, 8285, 8286,
8287, 8288, 8289, 8290, 8291, 8293, 8295, 8297, 8298, 8299, 8300, 8301, 8303,
8304, 8306, 8307, 8308, 8312, 8317, 8324, 8325, 8326, 8327, 8331, 8334, 8336,
8337, 8338, 8339, 8341, 8342, 8344, 8345, 8346, 8347, 8348, 8349, 8351, 8353,
8355, 8358, 8359, 8360, 8361, 8364, 8365, 8369, 8370, 8371, 8373, 8374, 8375,
8379, 8380, 8381, 8382, 8383, 8385, 8387, 8390, 8391, 8392, 8395, 8399, 8401,
8402, 8407, 8408, 8409, 8411, 8415, 8417, 8418, 8420, 8422, 8428, 8429, 8430,
8431, 8433, 8434, 8436, 8437, 8439, 8440, 8443, 8445, 8446, 8450, 8455, 8459,
8461, 8464, 8466, 8467, 8468, 8469, 8477, 8479, 8482, 8483, 8484, 8485, 8487,
8488, 8491, 8494, 8495, 8497, 8498, 8500, 9001, 9002, 9003, 9004, 9005, 9006,
9007, 9008, 9010, 9011, 9014, 9015, 9016, 9018, 9019, 9020, 9022, 9023, 9024,
9025, 9026, 9027, 9028, 9029, 9030, 9033, 9034, 9035, 9040, 9041, 9042, 9043,
9045, 9046, 9050, 9051, 9052, 9053, 9054, 9055, 9057, 9058, 9059, 9062, 9063,
9064, 9065, 9068, 9069, 9071, 9073, 9074, 9076, 9078, 9085, 9086, 9087, 9088,
9090, 9091, 9092, 9093, 9095, 9096, 9097, 9099, 9100, 9102, 9105, 9106, 9107,
9108, 9110, 9111, 9113, 9114, 9115, 9116, 9118, 9120, 9121, 9122, 9123, 9126,
9127, 9129, 9130, 9131, 9133, 9134, 9136, 9137, 9138, 9141, 9145, 9146, 9147,
9148, 9150, 9151, 9153, 9154, 9155, 9157, 9158, 9159, 9163, 9164, 9168, 9170,
9171, 9172, 9175, 9176, 9177, 9178, 9179, 9180, 9182, 9183, 9185, 9187, 9188,
9189, 9190, 9191, 9192, 9193, 9194, 9195, 9196, 9197, 9198, 9199, 9201, 9202,
9203, 9204, 9206, 9207, 9208, 9211, 9214, 9216, 9217, 9218, 9220, 9222, 9223,
9226, 9227, 9228, 9229, 9231, 9232, 9236, 9237, 9241, 9242, 9245, 9246, 9247,
9249, 9251, 9252, 9254, 9255, 9256, 9258, 9259, 9260, 9262, 9263, 9264, 9265,
9266, 9267, 9270, 9271, 9272, 9273, 9274, 9277, 9278, 9279, 9280, 9282, 9284,
9286, 9288, 9290, 9291, 9292, 9294, 9295, 9296, 9297, 9300, 9301, 9302, 9305,
9306, 9307, 9308, 9311, 9315, 9316, 9319, 9320, 9322, 9324, 9325, 9328, 9329,
9330, 9331, 9332, 9333, 9335, 9336, 9337, 9338, 9339, 9340, 9341, 9342, 9344,
9346, 9347, 9349, 9350, 9351, 9352, 9353, 9354, 9355, 9359, 9361, 9363, 9364,
9365, 9366, 9367, 9371, 9373, 9374, 9375, 9376, 9377, 9378, 9379, 9381, 9383,
9385, 9386, 9387, 9388, 9389, 9391, 9392, 9396, 9397, 9398, 9399, 9400, 9401,
9403, 9406, 9407, 9408, 9410, 9411, 9412, 9413, 9414, 9415, 9416, 9417, 9418,
9419, 9421, 9422, 9423, 9424, 9426, 9427, 9430, 9431, 9432, 9433, 9434, 9436,
9437, 9438, 9440, 9441, 9442, 9443, 9445, 9448, 9449, 9450, 9453, 9454, 9455,
9456, 9457, 9458, 9459, 9462, 9465, 9466, 9467, 9468, 9470, 9471, 9472, 9473,
9474, 9476, 9477, 9478, 9480, 9481, 9482, 9483, 9484, 9485, 9486, 9487, 9488,
9489, 9490, 9491, 9492, 9493, 9494, 9495, 9496, 9497, 9498, 9500]
--------------------------------------------------------------------------------
--------------
Data from class 'no-face', that was wrongly predicted as 'face' [ 100 ] :
[8001, 8026, 8077, 8078, 8141, 8164, 8182, 8216, 8217, 8224, 8249, 8305, 8338,
8418, 8421, 8427, 8445, 8463, 8473, 8475, 8490, 9001, 9002, 9012, 9016, 9017,
9018, 9020, 9041, 9049, 9055, 9061, 9063, 9069, 9081, 9084, 9096, 9100, 9108,
9118, 9121, 9125, 9127, 9135, 9139, 9143, 9161, 9163, 9166, 9171, 9176, 9180,
9193, 9199, 9209, 9222, 9227, 9231, 9250, 9252, 9257, 9258, 9266, 9268, 9277,
9278, 9293, 9297, 9300, 9308, 9309, 9311, 9312, 9315, 9337, 9354, 9355, 9359,
9374, 9383, 9392, 9410, 9413, 9415, 9421, 9426, 9428, 9433, 9435, 9438, 9439,
9450, 9455, 9457, 9462, 9484, 9485, 9489, 9491, 9497]
```

### 1.1.7 Confusion matrix

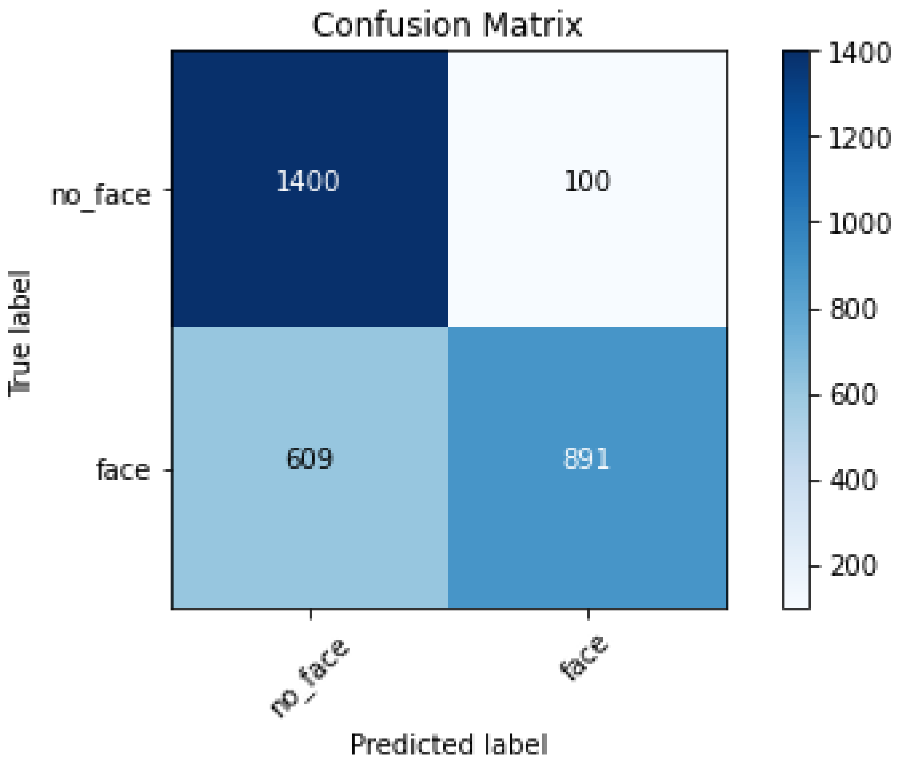
[13]:
```python
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    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
[[1400  100]
 [ 609  891]]
```


Confusion Matrix
no_face
face
True label
no_face
face
Predicted label