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

March 24, 2021

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

1.1 Testing Baseline CNN *(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/basis-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-B-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: 0.6734 - accuracy: 0.8307
Accuracy: 83.07%
Loss: 67.34%
```

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' [ 464 ] :
[8003, 8004, 8005, 8007, 8009, 8011, 8012, 8013, 8014, 8022, 8024, 8025, 8028,
8029, 8031, 8032, 8034, 8035, 8038, 8040, 8042, 8043, 8045, 8047, 8048, 8051,
8053, 8054, 8056, 8060, 8061, 8062, 8063, 8064, 8066, 8069, 8071, 8072, 8073,
8076, 8078, 8079, 8080, 8081, 8083, 8084, 8086, 8092, 8096, 8099, 8102, 8103,
8104, 8107, 8108, 8109, 8110, 8112, 8113, 8114, 8117, 8118, 8119, 8121, 8122,
8123, 8124, 8125, 8126, 8128, 8129, 8133, 8134, 8136, 8137, 8138, 8139, 8142,
8143, 8144, 8145, 8146, 8148, 8149, 8152, 8155, 8156, 8157, 8158, 8159, 8160,
8163, 8165, 8166, 8167, 8169, 8170, 8174, 8175, 8177, 8178, 8179, 8184, 8185,
8187, 8188, 8190, 8191, 8192, 8193, 8194, 8197, 8203, 8204, 8206, 8207, 8210,
8211, 8212, 8215, 8216, 8217, 8218, 8219, 8220, 8221, 8223, 8228, 8229, 8231,
8234, 8235, 8237, 8240, 8241, 8242, 8243, 8244, 8246, 8247, 8248, 8249, 8250,
8251, 8252, 8253, 8254, 8255, 8256, 8257, 8260, 8262, 8263, 8265, 8269, 8270,
8272, 8274, 8276, 8277, 8278, 8280, 8281, 8284, 8285, 8287, 8288, 8289, 8293,
8294, 8297, 8299, 8300, 8303, 8304, 8306, 8307, 8308, 8309, 8310, 8311, 8312,
8313, 8314, 8316, 8317, 8318, 8320, 8324, 8325, 8326, 8327, 8330, 8331, 8334,
8336, 8337, 8338, 8339, 8340, 8343, 8346, 8348, 8349, 8350, 8355, 8358, 8359,
8362, 8363, 8367, 8369, 8371, 8372, 8374, 8375, 8376, 8377, 8378, 8380, 8381,
8382, 8383, 8385, 8386, 8391, 8396, 8397, 8398, 8399, 8400, 8401, 8402, 8403,
8406, 8407, 8409, 8410, 8411, 8413, 8416, 8417, 8418, 8419, 8420, 8421, 8422,
8424, 8427, 8431, 8432, 8436, 8438, 8439, 8440, 8446, 8447, 8449, 8451, 8452,
8453, 8454, 8455, 8456, 8458, 8459, 8460, 8466, 8467, 8468, 8469, 8470, 8473,
8476, 8477, 8479, 8480, 8481, 8482, 8483, 8486, 8487, 8488, 8490, 8492, 8496,
8497, 8498, 9002, 9004, 9005, 9010, 9015, 9016, 9018, 9019, 9020, 9021, 9023,
9024, 9029, 9037, 9038, 9041, 9042, 9046, 9049, 9052, 9055, 9059, 9060, 9061,
9062, 9066, 9068, 9069, 9070, 9072, 9076, 9082, 9083, 9084, 9085, 9088, 9090,
9091, 9094, 9095, 9096, 9100, 9104, 9105, 9109, 9111, 9115, 9120, 9121, 9125,
9127, 9132, 9139, 9140, 9144, 9146, 9148, 9149, 9152, 9158, 9161, 9162, 9164,
9169, 9174, 9179, 9180, 9182, 9183, 9184, 9185, 9186, 9191, 9192, 9198, 9201,
9207, 9209, 9214, 9215, 9216, 9219, 9220, 9224, 9226, 9229, 9230, 9244, 9246,
9248, 9252, 9253, 9257, 9262, 9273, 9278, 9282, 9284, 9285, 9289, 9290, 9293,
9294, 9301, 9302, 9306, 9310, 9318, 9323, 9324, 9327, 9329, 9330, 9332, 9333,
9335, 9338, 9339, 9348, 9357, 9360, 9363, 9368, 9369, 9371, 9372, 9375, 9377,
9380, 9382, 9385, 9386, 9388, 9396, 9398, 9400, 9409, 9410, 9412, 9414, 9417,
9420, 9423, 9427, 9431, 9432, 9438, 9440, 9441, 9442, 9445, 9449, 9452, 9453,
9454, 9458, 9459, 9461, 9462, 9467, 9470, 9472, 9473, 9474, 9476, 9477, 9479,
9482, 9489, 9490, 9491, 9492, 9494, 9498, 9499, 9500]
-------------------------------------------------------------------------------
--------------
```

```
Data from class 'no-face', that was wrongly predicted as 'face' [ 44 ] :
[8007, 8124, 8148, 8180, 8215, 8217, 8237, 8264, 8288, 8292, 8299, 8312, 8401,
8487, 9009, 9014, 9031, 9039, 9061, 9067, 9083, 9088, 9106, 9187, 9219, 9223,
9250, 9270, 9272, 9295, 9303, 9314, 9319, 9329, 9345, 9350, 9355, 9370, 9372,
9401, 9402, 9481, 9483, 9491]
```

1.1.7 Confusion matrix

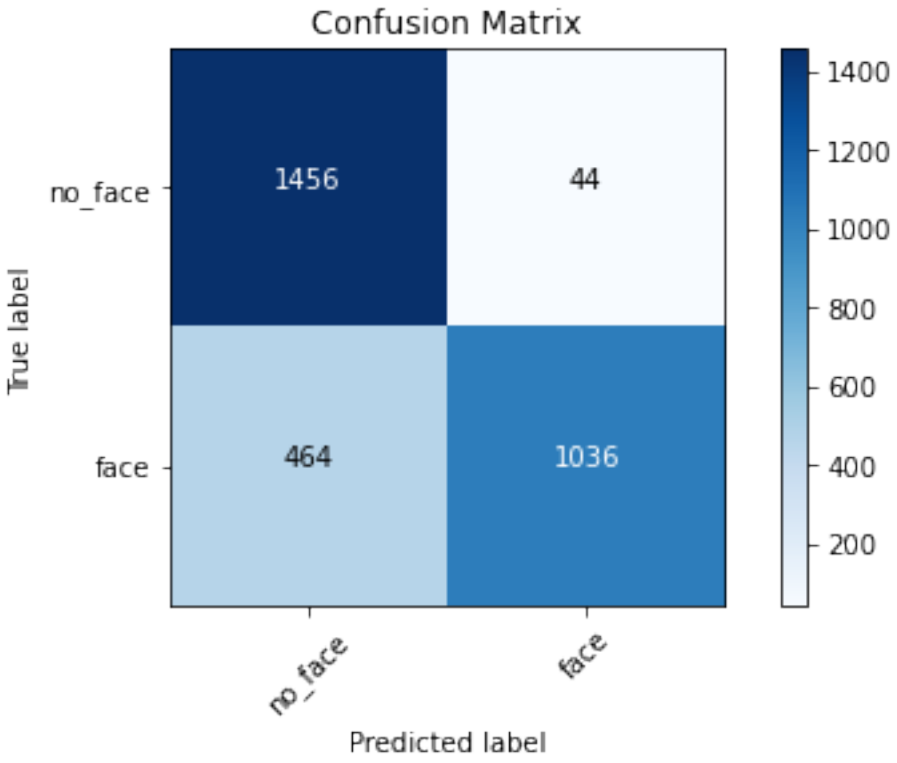
[13]:
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    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
[[1456   44]
 [ 464 1036]]
```


Confusion Matrix
1456
44
464
1036
no_face
face
True label
no_face
face
Predicted label
1400
1200
1000
800
600
400
200