FCN-Baseline-30E-14L-FFA-test-03

March 24, 2021

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

1.1 Testing Baseline FCN *(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/middle/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 FCN

[8]:
```
filename='../models/FCN-B-30E-14L-03.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: 0.3190 - accuracy: 0.9100
Accuracy: 91.00%
Loss: 31.90%
```

1.1.5 Testing the FCN

[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' [ 34 ] :
[8077, 8080, 8082, 8104, 8113, 8115, 8120, 8128, 8345, 8354, 8380, 8388, 8447,
8468, 8498, 8536, 8563, 8617, 8696, 8759, 8857, 8870, 8962, 9047, 9049, 9065,
9111, 9191, 9268, 9333, 9369, 9372, 9493, 9494]
--------------------------------------------------------------------------------
--------------
Data from class 'no-face', that was wrongly predicted as 'face' [ 236 ] :
[8007, 8013, 8014, 8015, 8025, 8029, 8043, 8053, 8056, 8058, 8059, 8060, 8065,
8068, 8082, 8083, 8084, 8095, 8111, 8114, 8134, 8140, 8144, 8147, 8148, 8160,
8162, 8174, 8185, 8193, 8199, 8200, 8209, 8213, 8220, 8222, 8244, 8249, 8251,
8268, 8281, 8282, 8287, 8290, 8293, 8303, 8306, 8309, 8322, 8323, 8329, 8331,
8334, 8336, 8338, 8346, 8354, 8359, 8360, 8372, 8373, 8374, 8383, 8386, 8403,
8404, 8420, 8428, 8429, 8430, 8431, 8432, 8434, 8436, 8443, 8451, 8458, 8463,
8474, 8476, 8479, 8488, 8497, 8507, 8509, 8511, 8512, 8515, 8530, 8535, 8539,
8541, 8549, 8554, 8556, 8557, 8558, 8560, 8561, 8562, 8563, 8564, 8569, 8602,
8604, 8607, 8613, 8643, 8644, 8646, 8649, 8652, 8670, 8679, 8681, 8688, 8694,
8701, 8709, 8723, 8726, 8727, 8734, 8757, 8764, 8774, 8778, 8784, 8785, 8795,
8796, 8802, 8814, 8819, 8820, 8827, 8834, 8837, 8840, 8846, 8847, 8849, 8850,
8851, 8853, 8879, 8883, 8894, 8898, 8904, 8914, 8915, 8920, 8923, 8950, 8956,
8958, 8965, 8976, 8979, 8980, 8991, 8998, 9003, 9012, 9057, 9058, 9061, 9062,
9069, 9082, 9083, 9084, 9086, 9090, 9093, 9097, 9109, 9112, 9119, 9125, 9126,
9128, 9137, 9139, 9140, 9146, 9174, 9181, 9193, 9200, 9202, 9208, 9210, 9220,
9227, 9234, 9241, 9242, 9243, 9246, 9249, 9252, 9261, 9262, 9265, 9272, 9273,
9279, 9302, 9307, 9338, 9340, 9342, 9351, 9369, 9376, 9386, 9388, 9393, 9395,
9400, 9404, 9410, 9414, 9445, 9448, 9450, 9458, 9462, 9471, 9477, 9492, 9493,
9494, 9498]
```

1.1.7 Confusion matrix

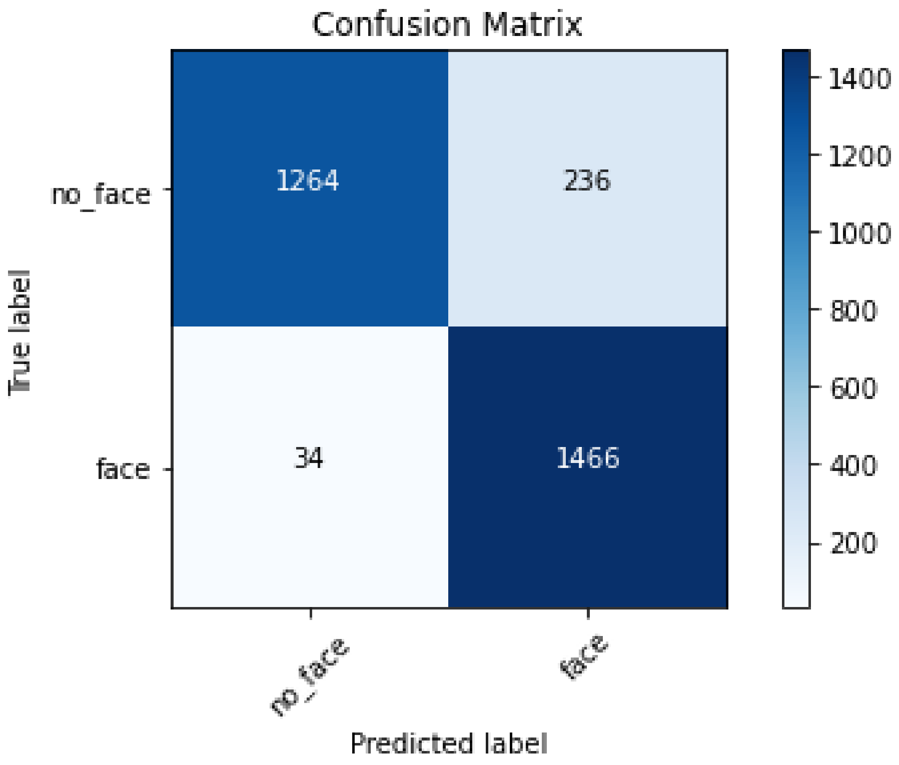
[13]:
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    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
[[1264  236]
 [  34 1466]]
```


Confusion Matrix
True label
no_face
face
1264
236
34
1466
Predicted label
no_face
face
1400
1200
1000
800
600
400
200