CNN-FFA-20E-15L-shift-test-02

March 26, 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/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-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 - 9s - loss: 0.2305 - accuracy: 0.9363
Accuracy: 93.63%
Loss: 23.05%
```

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' [ 146 ] :
[8002, 8003, 8007, 8014, 8020, 8021, 8025, 8028, 8030, 8031, 8032, 8038, 8045,
8053, 8055, 8061, 8062, 8065, 8067, 8073, 8076, 8079, 8083, 8090, 8091, 8093,
8096, 8099, 8103, 8105, 8114, 8122, 8131, 8136, 8137, 8149, 8152, 8156, 8157,
8158, 8160, 8161, 8163, 8164, 8166, 8171, 8172, 8179, 8182, 8184, 8186, 8199,
8202, 8210, 8212, 8216, 8217, 8219, 8230, 8234, 8237, 8243, 8245, 8248, 8249,
8254, 8257, 8271, 8275, 8285, 8287, 8289, 8290, 8293, 8295, 8296, 8299, 8301,
8303, 8304, 8305, 8324, 8327, 8334, 8338, 8361, 8365, 8369, 8374, 8382, 8383,
8385, 8388, 8390, 8391, 8392, 8395, 8399, 8401, 8415, 8417, 8430, 8433, 8436,
8437, 8445, 8448, 8455, 8459, 8461, 8467, 8469, 8479, 8480, 8482, 8483, 8484,
8487, 8488, 8491, 8494, 8495, 8499, 9025, 9041, 9046, 9052, 9088, 9141, 9164,
9167, 9168, 9175, 9195, 9228, 9237, 9257, 9283, 9305, 9411, 9412, 9432, 9434,
9466, 9483, 9500]
--------------------------------------------------------------------------------
--------------
Data from class 'no-face', that was wrongly predicted as 'face' [ 45 ] :
[8001, 8021, 8026, 8068, 8077, 8090, 8141, 8186, 8216, 8217, 8224, 8305, 8338,
8343, 8376, 8389, 8418, 8445, 8446, 8454, 8473, 8475, 8482, 8490, 9001, 9012,
9017, 9031, 9101, 9188, 9250, 9257, 9258, 9268, 9278, 9287, 9290, 9355, 9359,
9410, 9421, 9428, 9433, 9450, 9484]
```

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:
```

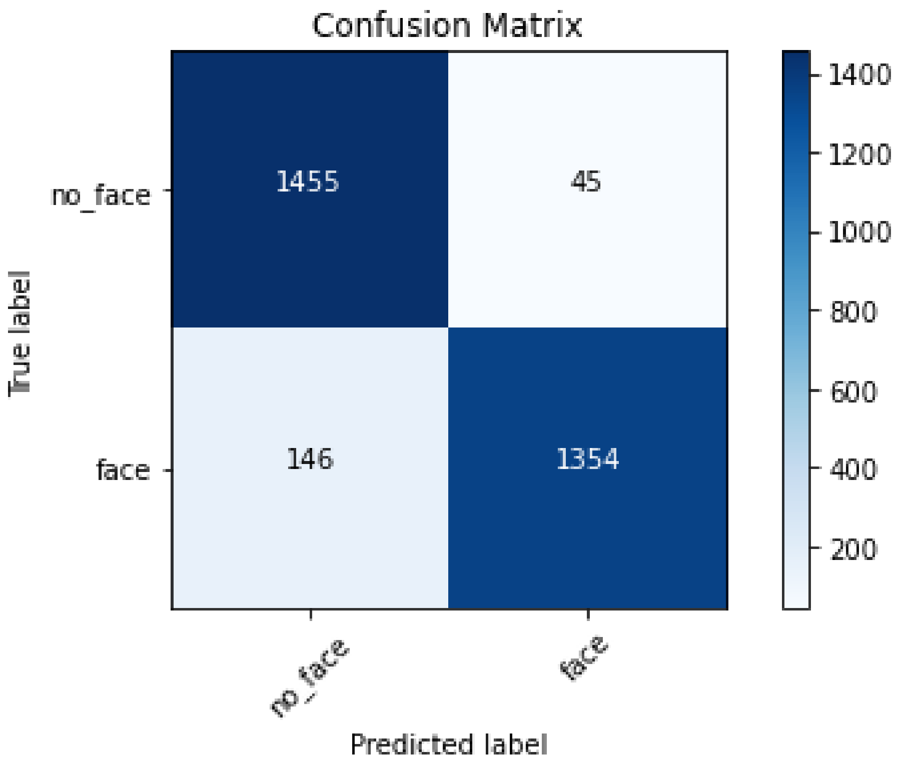
```
        print('Confusion matrix, without normalization')

    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
[[1455   45]
 [ 146 1354]]
```


Confusion Matrix
no_face
face
True label
no_face
face
Predicted label