

CNN-FFA-30E-13L-basis-test-03

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/basis-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 CNN

```
[8]: filename='../models/CNN-FFA-30E-13L-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 - 8s - loss: 0.2465 - accuracy: 0.9450  
Accuracy: 94.50%  
Loss: 24.65%
```

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' [ 165 ] :
[8006, 8022, 8023, 8057, 8061, 8064, 8075, 8079, 8091, 8093, 8095, 8107, 8128,
8130, 8139, 8161, 8170, 8171, 8184, 8186, 8188, 8196, 8200, 8201, 8209, 8216,
8218, 8220, 8223, 8224, 8225, 8233, 8238, 8247, 8255, 8281, 8295, 8304, 8313,
8320, 8324, 8325, 8329, 8369, 8375, 8405, 8406, 8407, 8417, 8419, 8443, 8446,
8448, 8451, 8459, 8464, 8475, 8493, 8498, 8501, 8510, 8518, 8527, 8537, 8550,
8558, 8559, 8568, 8585, 8588, 8596, 8599, 8604, 8648, 8650, 8652, 8655, 8670,
8684, 8699, 8709, 8736, 8737, 8743, 8757, 8769, 8811, 8818, 8823, 8835, 8836,
8838, 8844, 8846, 8859, 8864, 8868, 8878, 8922, 8925, 8940, 8942, 8948, 8963,
8975, 8987, 8991, 8998, 9002, 9005, 9008, 9028, 9032, 9034, 9035, 9059, 9062,
9090, 9098, 9103, 9117, 9127, 9171, 9177, 9186, 9188, 9190, 9204, 9209, 9215,
9220, 9231, 9247, 9253, 9258, 9264, 9267, 9284, 9308, 9310, 9311, 9313, 9315,
9317, 9324, 9327, 9333, 9340, 9342, 9343, 9346, 9357, 9362, 9375, 9395, 9401,
9425, 9432, 9455, 9457, 9459, 9461, 9472, 9476, 9481]

```

```

-----
Data from class 'no-face', that was wrongly predicted as 'face' [ 0 ] :
[]

```

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

```
[[1500   0]
 [ 165 1335]]
```

