FCN-FFA-20E-16L-basis-test-01

March 24, 2021

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

1.1 Testing FCN with Features Further Apart *(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/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-FFA-20E-16L-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: 0.0247 - accuracy: 0.9910
Accuracy: 99.10%
Loss: 2.47%
```

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' [ 24 ] :
[8200, 8224, 8330, 8371, 8380, 8388, 8409, 8452, 8575, 8698, 8705, 8784, 8823,
8859, 8957, 9034, 9070, 9076, 9080, 9194, 9308, 9337, 9346, 9432]
--------------------------------------------------------------------------------
--------------
Data from class 'no-face', that was wrongly predicted as 'face' [ 3 ] :
[8179, 8698, 9321]
```

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
[[1497    3]
 [  24 1476]]
```