CNN-Baseline-20E-15L-shift-test-03

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-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: 1.7620 - accuracy: 0.7323
Accuracy: 73.23%
Loss: 176.20%
```

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' [ 794 ] :
[8001, 8003, 8004, 8005, 8006, 8007, 8009, 8011, 8012, 8013, 8014, 8016, 8018,
8019, 8022, 8024, 8025, 8027, 8028, 8029, 8030, 8031, 8032, 8034, 8035, 8036,
8037, 8038, 8040, 8042, 8043, 8045, 8046, 8047, 8048, 8051, 8053, 8054, 8055,
8056, 8060, 8061, 8062, 8063, 8064, 8065, 8066, 8067, 8068, 8069, 8070, 8071,
8072, 8073, 8076, 8078, 8079, 8080, 8081, 8082, 8083, 8084, 8086, 8087, 8088,
8090, 8092, 8094, 8096, 8097, 8098, 8099, 8101, 8102, 8103, 8104, 8107, 8108,
8109, 8110, 8111, 8112, 8113, 8114, 8117, 8118, 8119, 8121, 8122, 8123, 8124,
8125, 8126, 8128, 8129, 8131, 8133, 8134, 8135, 8136, 8137, 8138, 8139, 8140,
8141, 8142, 8143, 8144, 8145, 8146, 8147, 8148, 8149, 8151, 8152, 8154, 8155,
8156, 8157, 8158, 8159, 8160, 8162, 8163, 8164, 8165, 8166, 8167, 8168, 8169,
8170, 8171, 8173, 8174, 8175, 8177, 8178, 8179, 8180, 8183, 8184, 8185, 8186,
8187, 8188, 8189, 8190, 8191, 8192, 8193, 8194, 8195, 8196, 8197, 8200, 8201,
8203, 8204, 8206, 8207, 8208, 8209, 8210, 8211, 8212, 8215, 8216, 8217, 8218,
8219, 8220, 8221, 8223, 8227, 8228, 8229, 8231, 8233, 8234, 8235, 8236, 8237,
8238, 8240, 8241, 8242, 8243, 8244, 8246, 8247, 8248, 8249, 8250, 8251, 8252,
8253, 8254, 8255, 8256, 8257, 8258, 8259, 8260, 8261, 8262, 8263, 8264, 8265,
8267, 8268, 8269, 8270, 8271, 8272, 8274, 8275, 8276, 8277, 8278, 8279, 8280,
8281, 8283, 8284, 8285, 8287, 8288, 8289, 8293, 8294, 8297, 8298, 8299, 8300,
8301, 8303, 8304, 8305, 8306, 8307, 8308, 8309, 8310, 8311, 8312, 8313, 8314,
8316, 8317, 8318, 8320, 8321, 8322, 8324, 8325, 8326, 8327, 8329, 8330, 8331,
8332, 8334, 8335, 8336, 8337, 8338, 8339, 8340, 8341, 8342, 8343, 8345, 8346,
8347, 8348, 8349, 8350, 8351, 8352, 8354, 8355, 8358, 8359, 8362, 8363, 8366,
8367, 8369, 8371, 8372, 8373, 8374, 8375, 8376, 8377, 8378, 8379, 8380, 8381,
8382, 8383, 8385, 8386, 8387, 8390, 8391, 8392, 8395, 8396, 8397, 8398, 8399,
8400, 8401, 8402, 8403, 8405, 8406, 8407, 8408, 8409, 8410, 8411, 8413, 8416,
8417, 8418, 8419, 8420, 8421, 8422, 8423, 8424, 8426, 8427, 8428, 8429, 8431,
8432, 8435, 8436, 8438, 8439, 8440, 8441, 8444, 8446, 8447, 8449, 8451, 8452,
8453, 8454, 8455, 8456, 8457, 8458, 8459, 8460, 8461, 8466, 8467, 8468, 8469,
8470, 8473, 8475, 8476, 8477, 8478, 8479, 8480, 8481, 8482, 8483, 8486, 8487,
8488, 8489, 8490, 8492, 8496, 8497, 8498, 9001, 9002, 9004, 9005, 9006, 9007,
9009, 9010, 9011, 9013, 9014, 9015, 9016, 9018, 9019, 9020, 9021, 9023, 9024,
9025, 9026, 9027, 9028, 9029, 9030, 9031, 9033, 9034, 9035, 9036, 9037, 9038,
9039, 9040, 9041, 9042, 9046, 9049, 9050, 9051, 9052, 9053, 9055, 9056, 9057,
9059, 9060, 9061, 9062, 9064, 9065, 9066, 9067, 9068, 9069, 9070, 9071, 9072,
9073, 9075, 9076, 9077, 9079, 9080, 9081, 9082, 9083, 9084, 9085, 9086, 9088,
9089, 9090, 9091, 9092, 9094, 9095, 9096, 9097, 9099, 9100, 9102, 9104, 9105,
9106, 9109, 9110, 9111, 9112, 9115, 9116, 9119, 9120, 9121, 9122, 9123, 9124,
9125, 9126, 9127, 9128, 9130, 9132, 9133, 9134, 9135, 9136, 9137, 9138, 9139,
```

```
9140, 9141, 9142, 9144, 9145, 9146, 9148, 9149, 9150, 9151, 9152, 9154, 9155,
9156, 9157, 9158, 9159, 9161, 9162, 9164, 9165, 9167, 9168, 9169, 9171, 9172,
9173, 9174, 9177, 9179, 9180, 9181, 9182, 9183, 9184, 9185, 9186, 9188, 9189,
9190, 9191, 9192, 9193, 9194, 9196, 9197, 9198, 9199, 9201, 9202, 9205, 9206,
9207, 9208, 9209, 9210, 9212, 9214, 9215, 9216, 9217, 9218, 9219, 9220, 9221,
9222, 9223, 9224, 9225, 9226, 9227, 9228, 9229, 9230, 9233, 9234, 9235, 9237,
9238, 9239, 9240, 9241, 9242, 9243, 9244, 9245, 9246, 9248, 9249, 9251, 9252,
9253, 9254, 9255, 9257, 9258, 9259, 9260, 9261, 9262, 9263, 9265, 9266, 9267,
9269, 9272, 9273, 9276, 9277, 9278, 9279, 9280, 9281, 9282, 9283, 9284, 9285,
9286, 9288, 9289, 9290, 9291, 9292, 9293, 9294, 9295, 9297, 9298, 9301, 9302,
9305, 9306, 9307, 9308, 9309, 9310, 9311, 9312, 9313, 9314, 9315, 9316, 9317,
9318, 9320, 9321, 9323, 9324, 9325, 9326, 9327, 9328, 9329, 9330, 9332, 9333,
9335, 9336, 9337, 9338, 9339, 9340, 9341, 9342, 9343, 9344, 9345, 9346, 9348,
9350, 9351, 9352, 9353, 9354, 9356, 9357, 9359, 9360, 9361, 9362, 9363, 9364,
9365, 9366, 9367, 9368, 9369, 9370, 9371, 9372, 9373, 9375, 9376, 9377, 9379,
9380, 9381, 9382, 9384, 9385, 9386, 9388, 9389, 9391, 9392, 9394, 9395, 9396,
9397, 9398, 9400, 9403, 9404, 9405, 9407, 9409, 9410, 9411, 9412, 9413, 9414,
9415, 9416, 9417, 9420, 9421, 9422, 9423, 9425, 9426, 9427, 9428, 9429, 9430,
9431, 9432, 9434, 9435, 9436, 9437, 9438, 9439, 9440, 9441, 9442, 9443, 9444,
9445, 9447, 9448, 9449, 9451, 9452, 9453, 9454, 9455, 9456, 9457, 9458, 9459,
9460, 9461, 9462, 9463, 9464, 9465, 9466, 9467, 9468, 9469, 9470, 9471, 9472,
9473, 9474, 9475, 9476, 9477, 9478, 9479, 9480, 9481, 9482, 9483, 9484, 9485,
9486, 9487, 9489, 9490, 9491, 9492, 9493, 9494, 9495, 9496, 9497, 9498, 9499,
9500]
--------------------------------------------------------------------------------
--------------
Data from class 'no-face', that was wrongly predicted as 'face' [ 9 ] :
[8007, 8217, 8288, 8292, 8312, 8401, 8487, 9319, 9370]
```

1.1.7 Confusion matrix

[13]:
```python
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
[[1491    9]
 [ 794  706]]
```