project-source-code

September 5, 2020

1 Project Overview

In this project, we will create a deep convolutional neural network model based on residual blocks to predict facial expression of a person.

[1]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
import os, PIL, random, cv2, pickle
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, optimizers
from tensorflow.keras.applications import DenseNet121, ResNet50
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.utils import plot_model
from IPython.display import display
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping,␣
↪ModelCheckpoint, LearningRateScheduler
from tensorflow.keras.optimizers import SGD, Adam
```

[2]:
```
df = pd.read_csv("emotion.csv")
df.head()
```

[2]:
```
   emotion                                             pixels
0        0  70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...
1        0  151 150 147 155 148 133 111 140 170 174 182 15...
2        2  24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...
3        2  20 17 19 21 25 38 42 42 46 54 56 62 63 66 82 1...
4        3  77 78 79 79 78 75 60 55 47 48 58 73 77 79 57 5...
```

[3]:
```
df['pixels'] = df[' pixels'] # add space before pixels to make the values␣
↪string.
# df['pixels'][0]
```

[4]:
```
def string2array(x):
    return np.asarray(x.split(" "), dtype=float).reshape(48,48,1)
```

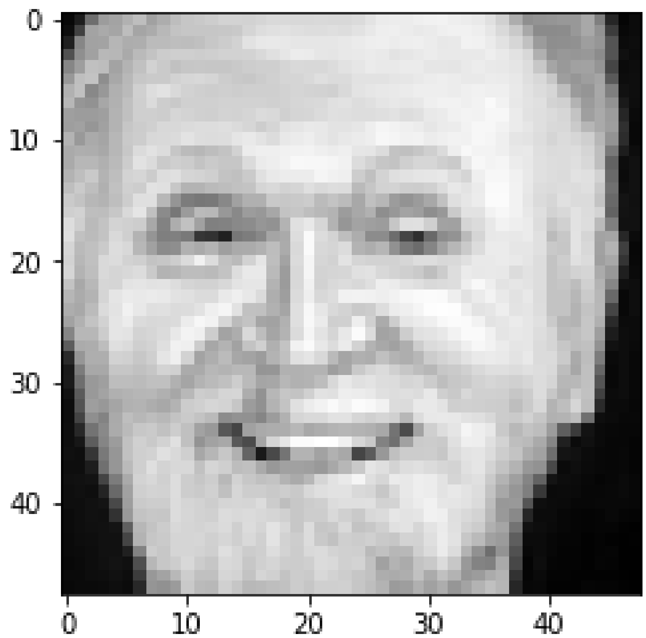
[5]:
```python
df['pixels'] = df['pixels'].apply(lambda x: string2array(x))
df['pixels'][0].shape
```

[5]:
```
(48, 48, 1)
```

[6]:
```python
category_names = {0: "anger", 1: "disgust", 2: "sad", 3: "happiness", 4:␣
↪"surprise"}
```

[7]:
```python
random_index = np.random.randint(0, len(df['pixels']))
plt.imshow(df['pixels'][random_index].squeeze(), cmap='gray')
print(f"This is an image of a person, who shows␣
↪{category_names[df['emotion'][random_index]]}")
```

```
This is an image of a person, who shows happiness
```



[8]:
```python
fig, axes = plt.subplots(1,5,figsize=(10,10))
for index in category_names.keys():
    data = df[df['emotion'] == index][:1]
    img = data['pixels'].item().reshape(48, 48)
    axes[index].imshow(img, cmap='gray')
    axes[index].set_title(category_names[index])
    axes[index].axis('off')
```

We might be also interested in distribution per class of the training examples.

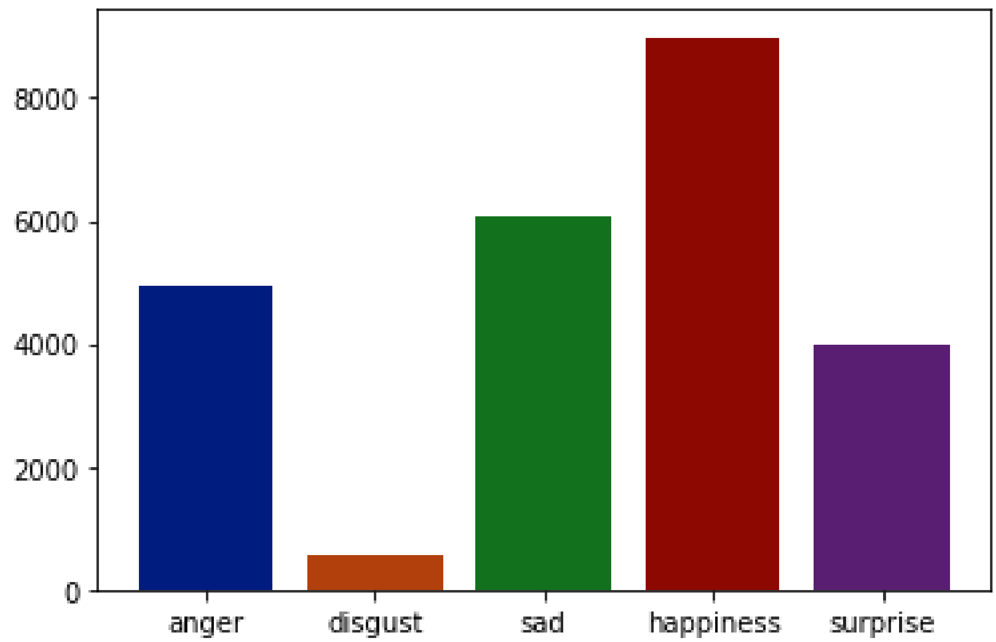
[9]:
```
class_names, class_counts = [], []

for category in category_names.keys():
    class_names.append(category)
    counts = df[df['emotion'] == category].count()
    class_counts.append(counts[0])
for name, count in zip(class_names, class_counts):
    print(f"There are {count} images in the category of {category_names[name]}")
```

```
There are 4953 images in the category of anger
There are 547 images in the category of disgust
There are 6077 images in the category of sad
There are 8989 images in the category of happiness
There are 4002 images in the category of surprise
```

[10]:
```
plt.bar(class_names, class_counts, color=sns.color_palette('dark',␣
 ↪len(class_counts)))
plt.xticks(np.arange(5), labels=category_names.values())
```

[10]:
```
([<matplotlib.axis.XTick at 0x1cb0875cac8>,
  <matplotlib.axis.XTick at 0x1cb0875c438>,
  <matplotlib.axis.XTick at 0x1cb0875c2e8>,
  <matplotlib.axis.XTick at 0x1cb0878fb00>,
  <matplotlib.axis.XTick at 0x1cb0878fe80>],
 <a list of 5 Text xticklabel objects>)
```



As can be seen the training examples classes of the dataset are not balanced.

2 Creating training, validation, and test sets

[11]:
```
X = np.stack(df['pixels'], axis=0).reshape(-1, 48, 48, 1)
y = np.array(pd.get_dummies(df['emotion'])) # one-hot encoding
print(X.shape, type(X))
print(y.shape, type(y))
```

```
(24568, 48, 48, 1) <class 'numpy.ndarray'>
(24568, 5) <class 'numpy.ndarray'>
```

[12]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,␣
↪shuffle=True, random_state=2020)
X_val, X_test, y_val, y_test = train_test_split(X_test, y_test, test_size=0.5,␣
↪shuffle=True, random_state=2020)
print(f"Train set samples: {X_train.shape[0]}")
print(f"Validation set samples: {X_val.shape[0]}")
print(f"Test set samples: {X_test.shape[0]}")
```

```
Train set samples: 22111
Validation set samples: 1228
Test set samples: 1229
```

3 Data preprocessing

First, we want to standardize data by making it follow standard normal distribution (mean of 0, std of 1).

[13]:
```
def stats(data):
    print(f"Mean of the data: {np.mean(data)}\nStd of the data: {np.std(data)}")
def standardization(data, mean, std):
    return (data - mean) / std
train_mean, train_std = np.mean(X_train), np.std(X_train)
stats(X_train)
X_train = standardization(X_train, train_mean, train_std)
stats(X_train)
```

```
Mean of the data: 129.549699861086
Std of the data: 64.79804876593721
Mean of the data: -1.9663654911194739e-16
Std of the data: 0.9999999999999992
```

[14]:
```
stats(X_val)
X_val = standardization(X_val, train_mean, train_std)
stats(X_val)
```

```
Mean of the data: 129.0658124660695
Std of the data: 65.0084759616264
Mean of the data: -0.007467622933591721
Std of the data: 1.0032474310522732
```

[15]:
```
stats(X_test)
X_test = standardization(X_test, train_mean, train_std)
stats(X_test)
```

```
Mean of the data: 129.30914714424554
Std of the data: 65.7877855902587
Mean of the data: -0.0037123450693611446
Std of the data: 1.0152741763551647
```

After standardization, we perform image augmentation to create more training examples in order to improve generalizability of the model.

[16]:
```
train_datagen = ImageDataGenerator(zoom_range=0.1, shear_range=0.1,
                                   height_shift_range=0.1, width_shift_range=0.1,
                                   rotation_range=15, horizontal_flip=True)
training_data = train_datagen.flow(X_train, y_train, batch_size=128)
```

4 Formulating ResNet model

[17]:
```
def res_net_block(X, filter_sizes, stage):

    X_shortcut, f1, f2, f3 = X, *filter_sizes

    # Main path
    X = Conv2D(filters=f1, kernel_size=(1, 1), padding='same',␣
 ↪kernel_initializer='he_normal', name='res_'+str(stage)+'_conv_a')(X)
    X = MaxPool2D(pool_size=2, strides=2,␣
 ↪name='max_pool_'+str(stage)+'_conv_a')(X)
    X = BatchNormalization(axis=3, name='bn_'+str(stage)+'_conv_a')(X)
    X = Activation('relu', name='activation_'+str(stage)+'_conv_a')(X)

    X = Conv2D(filters=f2, kernel_size=(3, 3), padding='same',␣
 ↪kernel_initializer='he_normal', name='res_'+str(stage)+'_conv_b')(X)
    X = BatchNormalization(axis=3, name='bn_'+str(stage)+'_conv_b')(X)
    X = Activation('relu', name='activation_'+str(stage)+'_conv_b')(X)

    X = Conv2D(filters=f3, kernel_size=(1, 1), padding='same',␣
 ↪kernel_initializer='he_normal', name='res_'+str(stage)+'_conv_c')(X)
    X = BatchNormalization(axis=3, name='bn_'+str(stage)+'_conv_c')(X)

    # Short path
    X_shortcut = Conv2D(filters=f3, kernel_size=(1, 1), padding='same',␣
 ↪kernel_initializer='he_normal',␣
 ↪name='res_'+str(stage)+'_conv_shortcut')(X_shortcut)
    X_shortcut = MaxPool2D(pool_size=2, strides=2,␣
 ↪name='max_pool_'+str(stage)+'_conv_shortcut')(X_shortcut)
    X_shortcut = BatchNormalization(axis=3,␣
 ↪name='bn_'+str(stage)+'_conv_shortcut')(X_shortcut)

    X = Add()([X, X_shortcut])
    X = Activation('relu', name='activation_'+str(stage)+'_add_1')(X)

    # Identity block #1
    X_shortcut = X

    # Main path
    X = Conv2D(filters=f1, kernel_size=(1, 1), padding='same',␣
 ↪kernel_initializer='he_normal', name='res_'+str(stage)+'_identity_1_a')(X)
    X = BatchNormalization(axis=3, name='bn_'+str(stage)+'_identity_1_a')(X)
    X = Activation('relu', name='activation_'+str(stage)+'_identity_1_a')(X)

    X = Conv2D(filters=f2, kernel_size=(3, 3), padding='same',␣
 ↪kernel_initializer='he_normal', name='res_'+str(stage)+'_identity_1_b')(X)
```

```
    X = BatchNormalization(axis=3, name='bn_'+str(stage)+'_identity_1_b')(X)
    X = Activation('relu', name='activation_'+str(stage)+'_identity_1_b')(X)

    X = Conv2D(filters=f3, kernel_size=(1, 1), padding='same', kernel_initializer='he_normal', name='res_'+str(stage)+'_identity_1_c')(X)
    X = BatchNormalization(axis=3, name='bn_'+str(stage)+'_identity_1_c')(X)

    X = Add()([X, X_shortcut])
    X = Activation('relu', name='activation_'+str(stage)+'_add_2')(X)

    # Identity block #2
    X_shortcut = X

    # Main path
    X = Conv2D(filters=f1, kernel_size=(1, 1), padding='same', kernel_initializer='he_normal', name='res_'+str(stage)+'_identity_2_a')(X)
    X = BatchNormalization(axis=3, name='bn_'+str(stage)+'_identity_2_a')(X)
    X = Activation('relu', name='activation_'+str(stage)+'_identity_2_a')(X)

    X = Conv2D(filters=f2, kernel_size=(3, 3), padding='same', kernel_initializer='he_normal', name='res_'+str(stage)+'_identity_2_b')(X)
    X = BatchNormalization(axis=3, name='bn_'+str(stage)+'_identity_2_b')(X)
    X = Activation('relu', name='activation_'+str(stage)+'_identity_2_b')(X)

    X = Conv2D(filters=f3, kernel_size=(1, 1), padding='same', kernel_initializer='he_normal', name='res_'+str(stage)+'_identity_2_c')(X)
    X = BatchNormalization(axis=3, name='bn_'+str(stage)+'_identity_2_c')(X)

    X = Add()([X, X_shortcut])
    X = Activation('relu', name='activation_'+str(stage)+'_add_3')(X)

    return X
```

[18]:
```
input_shape = (48, 48, 1)

X_inp = Input(input_shape)

X = ZeroPadding2D((3, 3))(X_inp)

# First stage
X = Conv2D(filters=64, kernel_size=(7, 7), strides=(2, 2), name='conv1', kernel_initializer='he_normal')(X)
X = BatchNormalization(axis=3, name='bn_conv1')(X)
X = Activation('relu')(X)
X = MaxPooling2D(pool_size=2, strides=2)(X)

# Second stage
```

```python
X = res_net_block(X, filter_sizes=[64, 64, 256], stage=2)

# Third stage
X = res_net_block(X, filter_sizes=[128, 128, 512], stage=3)

# Average Pooling
X = AveragePooling2D((2,2), name='avg_pooling')(X)

# Output layer
X = Flatten()(X)
X = Dense(units=5, activation='softmax', kernel_initializer='he_normal',␣
 ↪name='dense_final')(X)

model = Model(inputs=X_inp, outputs=X, name='ResNet18')
model.summary()
```

```
Model: "ResNet18"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_1 (InputLayer)            [(None, 48, 48, 1)]  0

________________________________________________________________________________
__________________
zero_padding2d (ZeroPadding2D)  (None, 54, 54, 1)    0           input_1[0][0]

________________________________________________________________________________
__________________
conv1 (Conv2D)                  (None, 24, 24, 64)   3200
zero_padding2d[0][0]
________________________________________________________________________________
__________________
bn_conv1 (BatchNormalization)   (None, 24, 24, 64)   256         conv1[0][0]

________________________________________________________________________________
__________________
activation (Activation)         (None, 24, 24, 64)   0           bn_conv1[0][0]

________________________________________________________________________________
__________________
max_pooling2d (MaxPooling2D)    (None, 12, 12, 64)   0
activation[0][0]
________________________________________________________________________________
__________________
res_2_conv_a (Conv2D)           (None, 12, 12, 64)   4160
max_pooling2d[0][0]
________________________________________________________________________________
__________________
max_pool_2_conv_a (MaxPooling2D (None, 6, 6, 64)     0
```

```
res_2_conv_a[0][0]
__________________________________________________________________________________________________
bn_2_conv_a (BatchNormalization (None, 6, 6, 64)     256
max_pool_2_conv_a[0][0]
__________________________________________________________________________________________________
activation_2_conv_a (Activation (None, 6, 6, 64)     0
bn_2_conv_a[0][0]
__________________________________________________________________________________________________
res_2_conv_b (Conv2D)           (None, 6, 6, 64)     36928
activation_2_conv_a[0][0]
__________________________________________________________________________________________________
bn_2_conv_b (BatchNormalization (None, 6, 6, 64)     256
res_2_conv_b[0][0]
__________________________________________________________________________________________________
activation_2_conv_b (Activation (None, 6, 6, 64)     0
bn_2_conv_b[0][0]
__________________________________________________________________________________________________
res_2_conv_shortcut (Conv2D)    (None, 12, 12, 256)  16640
max_pooling2d[0][0]
__________________________________________________________________________________________________
res_2_conv_c (Conv2D)           (None, 6, 6, 256)    16640
activation_2_conv_b[0][0]
__________________________________________________________________________________________________
max_pool_2_conv_shortcut (MaxPo (None, 6, 6, 256)    0
res_2_conv_shortcut[0][0]
__________________________________________________________________________________________________
bn_2_conv_c (BatchNormalization (None, 6, 6, 256)    1024
res_2_conv_c[0][0]
__________________________________________________________________________________________________
bn_2_conv_shortcut (BatchNormal (None, 6, 6, 256)    1024
max_pool_2_conv_shortcut[0][0]
__________________________________________________________________________________________________
add (Add)                       (None, 6, 6, 256)    0
bn_2_conv_c[0][0]
bn_2_conv_shortcut[0][0]
__________________________________________________________________________________________________
```

```
activation_2_add_1 (Activation) (None, 6, 6, 256)    0           add[0][0]
________________________________________________________________________________
__________________
res_2_identity_1_a (Conv2D)     (None, 6, 6, 64)     16448
activation_2_add_1[0][0]
________________________________________________________________________________
__________________
bn_2_identity_1_a (BatchNormali (None, 6, 6, 64)     256
res_2_identity_1_a[0][0]
________________________________________________________________________________
__________________
activation_2_identity_1_a (Acti (None, 6, 6, 64)     0
bn_2_identity_1_a[0][0]
________________________________________________________________________________
__________________
res_2_identity_1_b (Conv2D)     (None, 6, 6, 64)     36928
activation_2_identity_1_a[0][0]
________________________________________________________________________________
__________________
bn_2_identity_1_b (BatchNormali (None, 6, 6, 64)     256
res_2_identity_1_b[0][0]
________________________________________________________________________________
__________________
activation_2_identity_1_b (Acti (None, 6, 6, 64)     0
bn_2_identity_1_b[0][0]
________________________________________________________________________________
__________________
res_2_identity_1_c (Conv2D)     (None, 6, 6, 256)    16640
activation_2_identity_1_b[0][0]
________________________________________________________________________________
__________________
bn_2_identity_1_c (BatchNormali (None, 6, 6, 256)    1024
res_2_identity_1_c[0][0]
________________________________________________________________________________
__________________
add_1 (Add)                     (None, 6, 6, 256)    0
bn_2_identity_1_c[0][0]
activation_2_add_1[0][0]
________________________________________________________________________________
__________________
activation_2_add_2 (Activation) (None, 6, 6, 256)    0           add_1[0][0]
________________________________________________________________________________
__________________
res_2_identity_2_a (Conv2D)     (None, 6, 6, 64)     16448
activation_2_add_2[0][0]
________________________________________________________________________________
__________________
bn_2_identity_2_a (BatchNormali (None, 6, 6, 64)     256
```

```
res_2_identity_2_a[0][0]
__________________________________________________________________________________________________
activation_2_identity_2_a (Acti (None, 6, 6, 64)     0
bn_2_identity_2_a[0][0]
__________________________________________________________________________________________________
res_2_identity_2_b (Conv2D)     (None, 6, 6, 64)     36928
activation_2_identity_2_a[0][0]
__________________________________________________________________________________________________
bn_2_identity_2_b (BatchNormali (None, 6, 6, 64)     256
res_2_identity_2_b[0][0]
__________________________________________________________________________________________________
activation_2_identity_2_b (Acti (None, 6, 6, 64)     0
bn_2_identity_2_b[0][0]
__________________________________________________________________________________________________
res_2_identity_2_c (Conv2D)     (None, 6, 6, 256)    16640
activation_2_identity_2_b[0][0]
__________________________________________________________________________________________________
bn_2_identity_2_c (BatchNormali (None, 6, 6, 256)    1024
res_2_identity_2_c[0][0]
__________________________________________________________________________________________________
add_2 (Add)                     (None, 6, 6, 256)    0
bn_2_identity_2_c[0][0]
activation_2_add_2[0][0]
__________________________________________________________________________________________________
activation_2_add_3 (Activation) (None, 6, 6, 256)    0           add_2[0][0]
__________________________________________________________________________________________________
res_3_conv_a (Conv2D)           (None, 6, 6, 128)    32896
activation_2_add_3[0][0]
__________________________________________________________________________________________________
max_pool_3_conv_a (MaxPooling2D (None, 3, 3, 128)    0
res_3_conv_a[0][0]
__________________________________________________________________________________________________
bn_3_conv_a (BatchNormalization (None, 3, 3, 128)    512
max_pool_3_conv_a[0][0]
__________________________________________________________________________________________________
activation_3_conv_a (Activation (None, 3, 3, 128)    0
```

```
bn_3_conv_a[0][0]
__________________________________________________________________________________________________
res_3_conv_b (Conv2D)           (None, 3, 3, 128)    147584
activation_3_conv_a[0][0]
__________________________________________________________________________________________________
bn_3_conv_b (BatchNormalization (None, 3, 3, 128)    512
res_3_conv_b[0][0]
__________________________________________________________________________________________________
activation_3_conv_b (Activation (None, 3, 3, 128)    0
bn_3_conv_b[0][0]
__________________________________________________________________________________________________
res_3_conv_shortcut (Conv2D)    (None, 6, 6, 512)    131584
activation_2_add_3[0][0]
__________________________________________________________________________________________________
res_3_conv_c (Conv2D)           (None, 3, 3, 512)    66048
activation_3_conv_b[0][0]
__________________________________________________________________________________________________
max_pool_3_conv_shortcut (MaxPo (None, 3, 3, 512)    0
res_3_conv_shortcut[0][0]
__________________________________________________________________________________________________
bn_3_conv_c (BatchNormalization (None, 3, 3, 512)    2048
res_3_conv_c[0][0]
__________________________________________________________________________________________________
bn_3_conv_shortcut (BatchNormal (None, 3, 3, 512)    2048
max_pool_3_conv_shortcut[0][0]
__________________________________________________________________________________________________
add_3 (Add)                     (None, 3, 3, 512)    0
bn_3_conv_c[0][0]
bn_3_conv_shortcut[0][0]
__________________________________________________________________________________________________
activation_3_add_1 (Activation) (None, 3, 3, 512)    0           add_3[0][0]
__________________________________________________________________________________________________
res_3_identity_1_a (Conv2D)     (None, 3, 3, 128)    65664
activation_3_add_1[0][0]
__________________________________________________________________________________________________
bn_3_identity_1_a (BatchNormali (None, 3, 3, 128)    512
```

```
res_3_identity_1_a[0][0]
__________________________________________________________________________________________________
activation_3_identity_1_a (Acti (None, 3, 3, 128)    0
bn_3_identity_1_a[0][0]
__________________________________________________________________________________________________
res_3_identity_1_b (Conv2D)     (None, 3, 3, 128)    147584
activation_3_identity_1_a[0][0]
__________________________________________________________________________________________________
bn_3_identity_1_b (BatchNormali (None, 3, 3, 128)    512
res_3_identity_1_b[0][0]
__________________________________________________________________________________________________
activation_3_identity_1_b (Acti (None, 3, 3, 128)    0
bn_3_identity_1_b[0][0]
__________________________________________________________________________________________________
res_3_identity_1_c (Conv2D)     (None, 3, 3, 512)    66048
activation_3_identity_1_b[0][0]
__________________________________________________________________________________________________
bn_3_identity_1_c (BatchNormali (None, 3, 3, 512)    2048
res_3_identity_1_c[0][0]
__________________________________________________________________________________________________
add_4 (Add)                     (None, 3, 3, 512)    0
bn_3_identity_1_c[0][0]
activation_3_add_1[0][0]
__________________________________________________________________________________________________
activation_3_add_2 (Activation) (None, 3, 3, 512)    0           add_4[0][0]
__________________________________________________________________________________________________
res_3_identity_2_a (Conv2D)     (None, 3, 3, 128)    65664
activation_3_add_2[0][0]
__________________________________________________________________________________________________
bn_3_identity_2_a (BatchNormali (None, 3, 3, 128)    512
res_3_identity_2_a[0][0]
__________________________________________________________________________________________________
activation_3_identity_2_a (Acti (None, 3, 3, 128)    0
bn_3_identity_2_a[0][0]
__________________________________________________________________________________________________
res_3_identity_2_b (Conv2D)     (None, 3, 3, 128)    147584
```

```
activation_3_identity_2_a[0][0]

--------------------------------------------------------------------------------
------------------
bn_3_identity_2_b (BatchNormali (None, 3, 3, 128)    512
res_3_identity_2_b[0][0]

--------------------------------------------------------------------------------
------------------
activation_3_identity_2_b (Acti (None, 3, 3, 128)    0
bn_3_identity_2_b[0][0]

--------------------------------------------------------------------------------
------------------
res_3_identity_2_c (Conv2D)     (None, 3, 3, 512)    66048
activation_3_identity_2_b[0][0]

--------------------------------------------------------------------------------
------------------
bn_3_identity_2_c (BatchNormali (None, 3, 3, 512)    2048
res_3_identity_2_c[0][0]

--------------------------------------------------------------------------------
------------------
add_5 (Add)                     (None, 3, 3, 512)    0
bn_3_identity_2_c[0][0]
activation_3_add_2[0][0]

--------------------------------------------------------------------------------
------------------
activation_3_add_3 (Activation) (None, 3, 3, 512)    0           add_5[0][0]

--------------------------------------------------------------------------------
------------------
avg_pooling (AveragePooling2D)  (None, 1, 1, 512)    0
activation_3_add_3[0][0]

--------------------------------------------------------------------------------
------------------
flatten (Flatten)               (None, 512)          0
avg_pooling[0][0]

--------------------------------------------------------------------------------
------------------
dense_final (Dense)             (None, 5)            2565        flatten[0][0]
================================================================================
==================
Total params: 1,174,021
Trainable params: 1,165,445
Non-trainable params: 8,576

--------------------------------------------------------------------------------
------------------
```

[19]:

```
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

[20]:
```
early_stopping = EarlyStopping(monitor='val_loss', verbose=1, patience=10)
checkpointer = ModelCheckpoint(filepath='weights.hdf5', save_best_only=True,␣
 ↪verbose=1)
```

[21]:
```
results = model.fit(training_data, validation_data=(X_val, y_val),
                    steps_per_epoch=len(X_train) // 128,␣
 ↪validation_steps=len(X_val)//128, epochs=50,
                    callbacks=[early_stopping, checkpointer])
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 172 steps, validate on 1228 samples
Epoch 1/50
170/172 [============================>.] - ETA: 0s - loss: 1.4485 - accuracy:
0.3917
Epoch 00001: val_loss improved from inf to 1.47331, saving model to weights.hdf5
172/172 [==============================] - 13s 75ms/step - loss: 1.4467 -
accuracy: 0.3927 - val_loss: 1.4733 - val_accuracy: 0.3672
Epoch 2/50
171/172 [============================>.] - ETA: 0s - loss: 1.2161 - accuracy:
0.4961
Epoch 00002: val_loss improved from 1.47331 to 1.17872, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 1.2166 -
accuracy: 0.4958 - val_loss: 1.1787 - val_accuracy: 0.4731
Epoch 3/50
171/172 [============================>.] - ETA: 0s - loss: 1.1133 - accuracy:
0.5426
Epoch 00003: val_loss did not improve from 1.17872
172/172 [==============================] - 6s 36ms/step - loss: 1.1128 -
accuracy: 0.5430 - val_loss: 1.2302 - val_accuracy: 0.5035
Epoch 4/50
171/172 [============================>.] - ETA: 0s - loss: 1.0420 - accuracy:
0.5769
Epoch 00004: val_loss improved from 1.17872 to 1.04004, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 1.0419 -
accuracy: 0.5766 - val_loss: 1.0400 - val_accuracy: 0.5634
Epoch 5/50
171/172 [============================>.] - ETA: 0s - loss: 0.9709 - accuracy:
0.6086
Epoch 00005: val_loss improved from 1.04004 to 0.91454, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 0.9707 -
accuracy: 0.6086 - val_loss: 0.9145 - val_accuracy: 0.6137
Epoch 6/50
```

```
171/172 [============================>.] - ETA: 0s - loss: 0.9187 - accuracy:
0.6324
Epoch 00006: val_loss improved from 0.91454 to 0.89607, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 0.9198 -
accuracy: 0.6320 - val_loss: 0.8961 - val_accuracy: 0.6207
Epoch 7/50
171/172 [============================>.] - ETA: 0s - loss: 0.8865 - accuracy:
0.6452
Epoch 00007: val_loss improved from 0.89607 to 0.83200, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 0.8871 -
accuracy: 0.6447 - val_loss: 0.8320 - val_accuracy: 0.6441
Epoch 8/50
171/172 [============================>.] - ETA: 0s - loss: 0.8521 - accuracy:
0.6589
Epoch 00008: val_loss improved from 0.83200 to 0.80249, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 0.8513 -
accuracy: 0.6591 - val_loss: 0.8025 - val_accuracy: 0.6719
Epoch 9/50
171/172 [============================>.] - ETA: 0s - loss: 0.8204 - accuracy:
0.6745
Epoch 00009: val_loss improved from 0.80249 to 0.77769, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 0.8195 -
accuracy: 0.6748 - val_loss: 0.7777 - val_accuracy: 0.6675
Epoch 10/50
171/172 [============================>.] - ETA: 0s - loss: 0.7977 - accuracy:
0.6830
Epoch 00010: val_loss improved from 0.77769 to 0.73781, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 0.7978 -
accuracy: 0.6833 - val_loss: 0.7378 - val_accuracy: 0.6797
Epoch 11/50
171/172 [============================>.] - ETA: 0s - loss: 0.7778 - accuracy:
0.6915
Epoch 00011: val_loss did not improve from 0.73781
172/172 [==============================] - 6s 36ms/step - loss: 0.7773 -
accuracy: 0.6916 - val_loss: 0.7428 - val_accuracy: 0.6927
Epoch 12/50
170/172 [============================>.] - ETA: 0s - loss: 0.7532 - accuracy:
0.7039
Epoch 00012: val_loss did not improve from 0.73781
172/172 [==============================] - 6s 37ms/step - loss: 0.7543 -
accuracy: 0.7036 - val_loss: 0.8204 - val_accuracy: 0.6606
Epoch 13/50
171/172 [============================>.] - ETA: 0s - loss: 0.7407 - accuracy:
```

```
0.7105
Epoch 00013: val_loss improved from 0.73781 to 0.73197, saving model to
weights.hdf5
172/172 [==============================] - 7s 38ms/step - loss: 0.7397 -
accuracy: 0.7107 - val_loss: 0.7320 - val_accuracy: 0.7092
Epoch 14/50
171/172 [============================>.] - ETA: 0s - loss: 0.7205 - accuracy:
0.7213
Epoch 00014: val_loss improved from 0.73197 to 0.69733, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 0.7204 -
accuracy: 0.7211 - val_loss: 0.6973 - val_accuracy: 0.6979
Epoch 15/50
171/172 [============================>.] - ETA: 0s - loss: 0.7092 - accuracy:
0.7235
Epoch 00015: val_loss did not improve from 0.69733
172/172 [==============================] - 6s 36ms/step - loss: 0.7092 -
accuracy: 0.7236 - val_loss: 0.7399 - val_accuracy: 0.6979
Epoch 16/50
171/172 [============================>.] - ETA: 0s - loss: 0.6939 - accuracy:
0.7312
Epoch 00016: val_loss did not improve from 0.69733
172/172 [==============================] - 6s 36ms/step - loss: 0.6952 -
accuracy: 0.7306 - val_loss: 0.7814 - val_accuracy: 0.6780
Epoch 17/50
171/172 [============================>.] - ETA: 0s - loss: 0.6813 - accuracy:
0.7335
Epoch 00017: val_loss did not improve from 0.69733
172/172 [==============================] - 6s 36ms/step - loss: 0.6814 -
accuracy: 0.7333 - val_loss: 0.7054 - val_accuracy: 0.7075
Epoch 18/50
171/172 [============================>.] - ETA: 0s - loss: 0.6693 - accuracy:
0.7407
Epoch 00018: val_loss improved from 0.69733 to 0.66087, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 0.6691 -
accuracy: 0.7408 - val_loss: 0.6609 - val_accuracy: 0.7300
Epoch 19/50
171/172 [============================>.] - ETA: 0s - loss: 0.6579 - accuracy:
0.7412
Epoch 00019: val_loss improved from 0.66087 to 0.64958, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 0.6581 -
accuracy: 0.7412 - val_loss: 0.6496 - val_accuracy: 0.7292
Epoch 20/50
171/172 [============================>.] - ETA: 0s - loss: 0.6497 - accuracy:
0.7461
Epoch 00020: val_loss did not improve from 0.64958
```

```
172/172 [==============================] - 6s 37ms/step - loss: 0.6511 -
accuracy: 0.7459 - val_loss: 0.6970 - val_accuracy: 0.7083
Epoch 21/50
171/172 [============================>.] - ETA: 0s - loss: 0.6378 - accuracy:
0.7536
Epoch 00021: val_loss did not improve from 0.64958
172/172 [==============================] - 6s 36ms/step - loss: 0.6380 -
accuracy: 0.7536 - val_loss: 0.6677 - val_accuracy: 0.7144
Epoch 22/50
170/172 [============================>.] - ETA: 0s - loss: 0.6275 - accuracy:
0.7556
Epoch 00022: val_loss did not improve from 0.64958
172/172 [==============================] - 6s 37ms/step - loss: 0.6276 -
accuracy: 0.7554 - val_loss: 0.6509 - val_accuracy: 0.7257
Epoch 23/50
171/172 [============================>.] - ETA: 0s - loss: 0.6110 - accuracy:
0.7655
Epoch 00023: val_loss did not improve from 0.64958
172/172 [==============================] - 6s 36ms/step - loss: 0.6110 -
accuracy: 0.7654 - val_loss: 0.6980 - val_accuracy: 0.7144
Epoch 24/50
171/172 [============================>.] - ETA: 0s - loss: 0.6061 - accuracy:
0.7644
Epoch 00024: val_loss improved from 0.64958 to 0.63840, saving model to
weights.hdf5
172/172 [==============================] - 6s 37ms/step - loss: 0.6065 -
accuracy: 0.7645 - val_loss: 0.6384 - val_accuracy: 0.7361
Epoch 25/50
171/172 [============================>.] - ETA: 0s - loss: 0.6102 - accuracy:
0.7606
Epoch 00025: val_loss did not improve from 0.63840
172/172 [==============================] - 6s 37ms/step - loss: 0.6100 -
accuracy: 0.7610 - val_loss: 0.6562 - val_accuracy: 0.7257
Epoch 26/50
171/172 [============================>.] - ETA: 0s - loss: 0.5883 - accuracy:
0.7688
Epoch 00026: val_loss did not improve from 0.63840
172/172 [==============================] - 6s 36ms/step - loss: 0.5884 -
accuracy: 0.7687 - val_loss: 0.6693 - val_accuracy: 0.7257
Epoch 27/50
171/172 [============================>.] - ETA: 0s - loss: 0.5911 - accuracy:
0.7691
Epoch 00027: val_loss did not improve from 0.63840
172/172 [==============================] - 6s 37ms/step - loss: 0.5911 -
accuracy: 0.7690 - val_loss: 0.7200 - val_accuracy: 0.6970
Epoch 28/50
170/172 [============================>.] - ETA: 0s - loss: 0.5705 - accuracy:
0.7773
```

```
Epoch 00028: val_loss improved from 0.63840 to 0.60358, saving model to
weights.hdf5
172/172 [==============================] - 7s 38ms/step - loss: 0.5711 -
accuracy: 0.7772 - val_loss: 0.6036 - val_accuracy: 0.7387
Epoch 29/50
170/172 [============================>.] - ETA: 0s - loss: 0.5663 - accuracy:
0.7804
Epoch 00029: val_loss did not improve from 0.60358
172/172 [==============================] - 6s 37ms/step - loss: 0.5671 -
accuracy: 0.7801 - val_loss: 0.6286 - val_accuracy: 0.7526
Epoch 30/50
170/172 [============================>.] - ETA: 0s - loss: 0.5561 - accuracy:
0.7871
Epoch 00030: val_loss did not improve from 0.60358
172/172 [==============================] - 6s 36ms/step - loss: 0.5556 -
accuracy: 0.7873 - val_loss: 0.6635 - val_accuracy: 0.7300
Epoch 31/50
171/172 [============================>.] - ETA: 0s - loss: 0.5508 - accuracy:
0.7876
Epoch 00031: val_loss did not improve from 0.60358
172/172 [==============================] - 6s 36ms/step - loss: 0.5498 -
accuracy: 0.7881 - val_loss: 0.7909 - val_accuracy: 0.6745
Epoch 32/50
171/172 [============================>.] - ETA: 0s - loss: 0.5467 - accuracy:
0.7887
Epoch 00032: val_loss did not improve from 0.60358
172/172 [==============================] - 6s 36ms/step - loss: 0.5468 -
accuracy: 0.7888 - val_loss: 0.7079 - val_accuracy: 0.7101
Epoch 33/50
171/172 [============================>.] - ETA: 0s - loss: 0.5307 - accuracy:
0.7973
Epoch 00033: val_loss did not improve from 0.60358
172/172 [==============================] - 6s 36ms/step - loss: 0.5311 -
accuracy: 0.7971 - val_loss: 0.6274 - val_accuracy: 0.7378
Epoch 34/50
171/172 [============================>.] - ETA: 0s - loss: 0.5298 - accuracy:
0.7957
Epoch 00034: val_loss did not improve from 0.60358
172/172 [==============================] - 6s 36ms/step - loss: 0.5297 -
accuracy: 0.7957 - val_loss: 0.6348 - val_accuracy: 0.7335
Epoch 35/50
171/172 [============================>.] - ETA: 0s - loss: 0.5220 - accuracy:
0.7969
Epoch 00035: val_loss did not improve from 0.60358
172/172 [==============================] - 6s 36ms/step - loss: 0.5217 -
accuracy: 0.7971 - val_loss: 0.6562 - val_accuracy: 0.7292
Epoch 36/50
171/172 [============================>.] - ETA: 0s - loss: 0.5146 - accuracy:
```

```
0.8019
Epoch 00036: val_loss did not improve from 0.60358
172/172 [==============================] - 6s 36ms/step - loss: 0.5145 -
accuracy: 0.8020 - val_loss: 0.6493 - val_accuracy: 0.7266
Epoch 37/50
171/172 [============================>.] - ETA: 0s - loss: 0.5041 - accuracy:
0.8082
Epoch 00037: val_loss did not improve from 0.60358
172/172 [==============================] - 6s 36ms/step - loss: 0.5038 -
accuracy: 0.8083 - val_loss: 0.6117 - val_accuracy: 0.7457
Epoch 38/50
171/172 [============================>.] - ETA: 0s - loss: 0.5014 - accuracy:
0.8046
Epoch 00038: val_loss did not improve from 0.60358
172/172 [==============================] - 6s 36ms/step - loss: 0.5013 -
accuracy: 0.8048 - val_loss: 0.6609 - val_accuracy: 0.7326
Epoch 00038: early stopping
```

5 Model Evaluation

[22]:
```
test_acc_score = model.evaluate(X_test, y_test)
print(f"Accuracy score of the model on test data: {test_acc_score[1]:.3f}")
```

```
1229/1229 [==============================] - 1s 720us/sample - loss: 0.7157 -
accuracy: 0.7364
Accuracy score of the model on test data: 0.736
```

Let’s check training and validation loss values as well as accuracy scores.

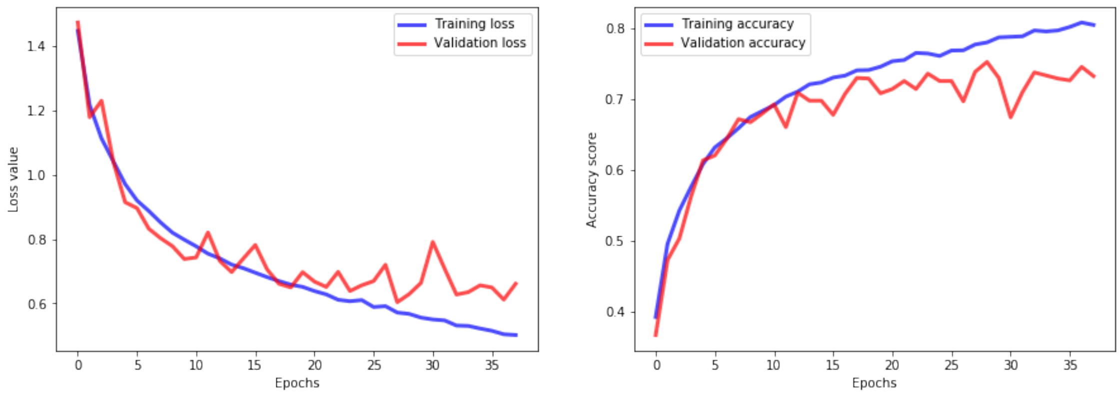
[23]:
```
tr_loss = results.history['loss']
val_loss = results.history['val_loss']
tr_acc = results.history['accuracy']
val_acc = results.history['val_accuracy']
epochs = range(len(tr_loss))

fig, axes = plt.subplots(1,2,figsize=(15, 5))
axes[0].plot(epochs, tr_loss, c='b', alpha=0.7, linewidth=3, label="Training␣
 ↪loss")
axes[0].plot(epochs, val_loss, c='r', alpha=0.7, linewidth=3, label="Validation␣
 ↪loss")
axes[0].set_xlabel("Epochs")
axes[0].set_ylabel("Loss value")
axes[0].legend()

axes[1].plot(epochs, tr_acc, c='b', alpha=0.7, linewidth=3, label='Training␣
 ↪accuracy')
axes[1].plot(epochs, val_acc, c='r', alpha=0.7, linewidth=3, label='Validation␣
 ↪accuracy')
```

```
axes[1].set_xlabel("Epochs")
axes[1].set_ylabel("Accuracy score")
axes[1].legend()
```

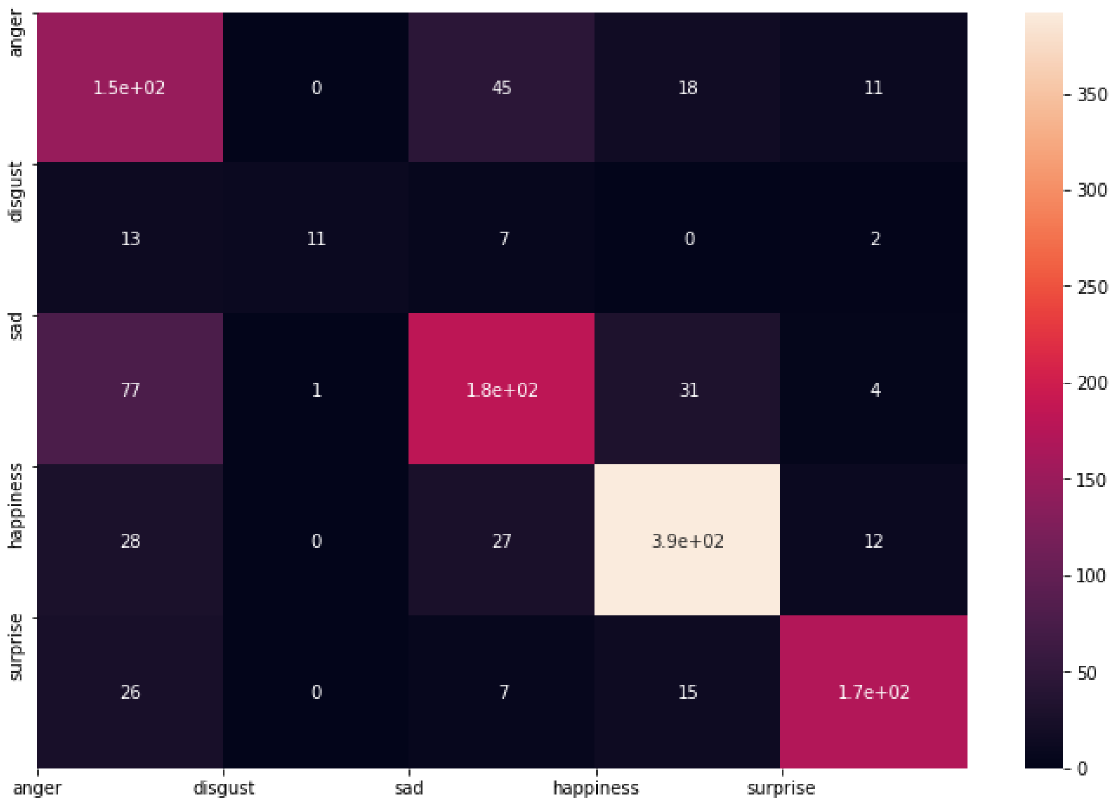
[23]: <matplotlib.legend.Legend at 0x1cc4d613a90>



We can also plot confusion matrix to find out in which classes the model made majority of prediction mistakes.

[24]:
```
from sklearn.metrics import confusion_matrix
test_preds = np.argmax(model.predict(X_test), axis=1)
cm = confusion_matrix(np.argmax(y_test, axis=1), test_preds)
plt.figure(figsize=(12, 8))
sns.heatmap(cm, annot=True, cbar=True)
plt.xticks(np.arange(5), labels=['anger', 'disgust', 'sad', 'happiness',␣
 ↪'surprise'])
plt.yticks(np.arange(5), labels=['anger', 'disgust', 'sad', 'happiness',␣
 ↪'surprise'])
```

[24]:
```
([<matplotlib.axis.YTick at 0x1cc5870c668>,
  <matplotlib.axis.YTick at 0x1cc58703f60>,
  <matplotlib.axis.YTick at 0x1cc58760cc0>,
  <matplotlib.axis.YTick at 0x1cc58778208>,
  <matplotlib.axis.YTick at 0x1cc587786d8>],
 <a list of 5 Text yticklabel objects>)
```



As it can be seen anger was mainly misclassified as sad (45 cases), sad was misclassified as anger (77 cases).

[25]:
```
from sklearn.metrics import classification_report
print(classification_report(np.argmax(y_test, axis=1), test_preds))
```

```
              precision    recall  f1-score   support

           0       0.51      0.67      0.57       221
           1       0.92      0.33      0.49        33
           2       0.68      0.62      0.65       296
           3       0.86      0.85      0.86       459
           4       0.86      0.78      0.82       220

    accuracy                           0.74      1229
   macro avg       0.76      0.65      0.68      1229
weighted avg       0.75      0.74      0.74      1229
```

[26]:
```
n_rows, n_cols = 5, 5

fig, axes = plt.subplots(n_rows, n_cols, figsize=(25, 25))
axes = axes.ravel()
```

```python
for i in range(n_rows * n_cols):
    random_index = np.random.randint(0, len(X_test))
    axes[i].imshow(X_test[random_index].reshape(48, 48), cmap='gray')
    axes[i].set_title(f"Prediction: {category_names[test_preds[random_index]]}\nGround Truth: {category_names[np.argmax(y_test, axis=1)[random_index]]}")
    axes[i].axis('off')
```

This is the end of this project! Thank you for following!