Importing Libraries

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os
```

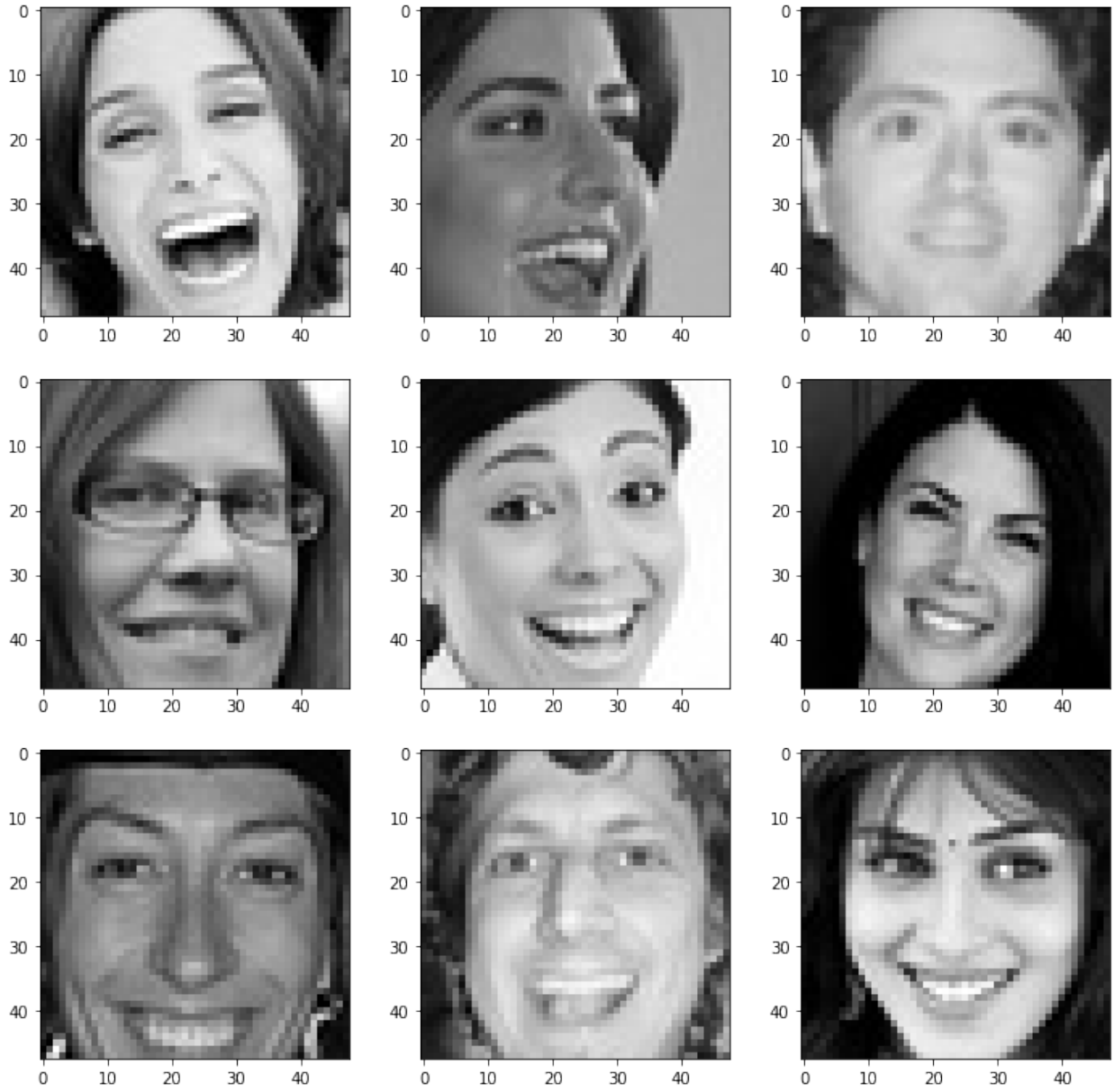
Importing Deep Learning Libraries

```
from tensorflow.keras.preprocessing.image import load_img,
img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import
Dense,Input,Dropout,GlobalAveragePooling2D,Flatten,Conv2D,BatchNormali
zation,Activation,MaxPooling2D
from tensorflow.keras.models import Model,Sequential
from tensorflow.keras.optimizers import Adam,SGD,RMSprop
```

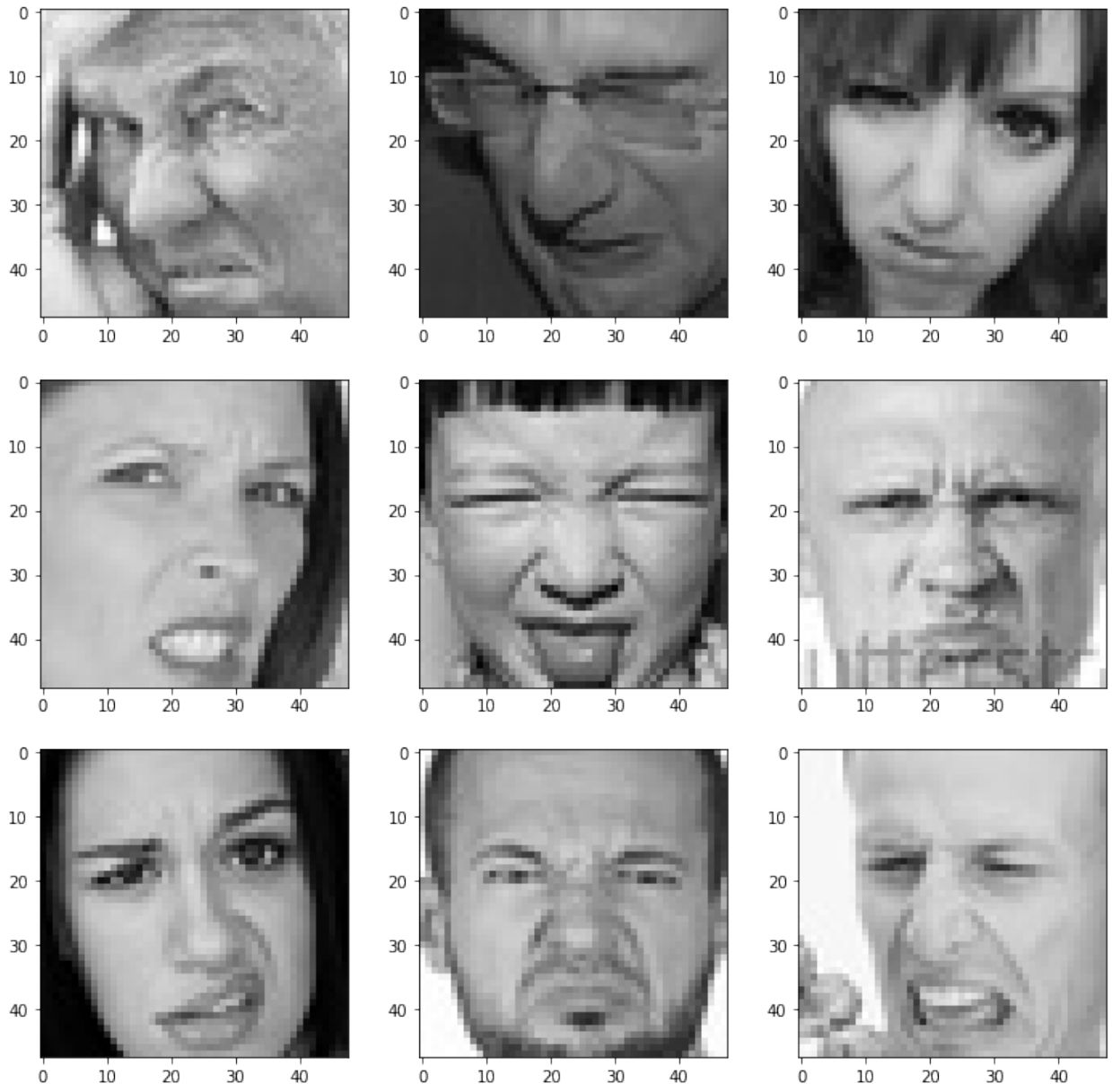
```
2021-09-12 10:12:57.352482: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49]
Successfully opened dynamic library libcudart.so.11.0
```

Displaying Images

```
picture_size = 48
folder_path = "../input/face-expression-recognition-dataset/images/"

expression = 'happy'

plt.figure(figsize= (12,12))
for i in range(1, 10, 1):
    plt.subplot(3,3,i)
    img = load_img(folder_path+"train/"+expression+"/"+
                   os.listdir(folder_path + "train/" + expression)[i],
target_size=(picture_size, picture_size))
    plt.imshow(img)
plt.show()
```



```
expression = 'disgust'

plt.figure(figsize= (12,12))
for i in range(1, 10, 1):
    plt.subplot(3,3,i)
    img = load_img(folder_path+"train/"+expression+"/"+
                   os.listdir(folder_path + "train/" + expression)[i],
target_size=(picture_size, picture_size))
    plt.imshow(img)
plt.show()
```



Making Training and Validation Data

```
batch_size=128
datagen_train=ImageDataGenerator()
datagen_val=ImageDataGenerator()

train_set= datagen_train.flow_from_directory(folder_path+"train",

target_size=(picture_size,picture_size),
                                             color_mode="grayscale",
                                              batch_size=batch_size,
                                             class_mode="categorical",
```

```
                                                shuffle=True)
test_set=datagen_val.flow_from_directory(folder_path+"validation",

target_size=(picture_size,picture_size),
                                         color_mode="grayscale",
                                         batch_size=batch_size,
                                         class_mode="categorical",
                                         shuffle=False)
```

```
Found 28821 images belonging to 7 classes.
Found 7066 images belonging to 7 classes.
```

Model Building

```
no_of_classes=7

model=Sequential()

#1st CNN layer
model.add(Conv2D(64,(3,3),padding="same",input_shape=(48,48,1)))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))


#2nd CNN layer
model.add(Conv2D(128,(5,5),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

#3rd CNN layer
model.add(Conv2D(512,(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

#4th CNN layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
```

```
#Fully connected 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(no_of_classes,activation="softmax"))

opt = Adam(lr = 0.0001)
model.compile(optimizer=opt,loss='categorical_crossentropy',
metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 48, 48, 64)        640
_________________________________________________________________
batch_normalization_9 (Batch (None, 48, 48, 64)        256
_________________________________________________________________
activation_9 (Activation)    (None, 48, 48, 64)        0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 24, 24, 64)        0
_________________________________________________________________
dropout_9 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 24, 24, 128)       204928
_________________________________________________________________
batch_normalization_10 (Batc (None, 24, 24, 128)       512
_________________________________________________________________
activation_10 (Activation)   (None, 24, 24, 128)       0
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_10 (Dropout)         (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 12, 12, 512)       590336
_________________________________________________________________
batch_normalization_11 (Batc (None, 12, 12, 512)       2048
_________________________________________________________________
activation_11 (Activation)   (None, 12, 12, 512)       0
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 6, 6, 512)         0
```

```
_________________________________________________________________
dropout_11 (Dropout)         (None, 6, 6, 512)         0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_12 (Batc (None, 6, 6, 512)         2048
_________________________________________________________________
activation_12 (Activation)   (None, 6, 6, 512)         0
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 3, 3, 512)         0
_________________________________________________________________
dropout_12 (Dropout)         (None, 3, 3, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               1179904
_________________________________________________________________
batch_normalization_13 (Batc (None, 256)               1024
_________________________________________________________________
activation_13 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_13 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_14 (Batc (None, 512)               2048
_________________________________________________________________
activation_14 (Activation)   (None, 512)               0
_________________________________________________________________
dropout_14 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 7)                 3591
=================================================================
Total params: 4,478,727
Trainable params: 4,474,759
Non-trainable params: 3,968
_________________________________________________________________
```

Fitting the Model with Training and Validation Data

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau

checkpoint = ModelCheckpoint("./model.h5", monitor='val_acc',
verbose=1, save_best_only=True, mode='max')
```

```
early_stopping = EarlyStopping(monitor='val_loss',
                               min_delta=0,
                               patience=3,
                               verbose=1,
                               restore_best_weights=True
                               )

reduce_learningrate = ReduceLROnPlateau(monitor='val_loss',
                                        factor=0.2,
                                        patience=3,
                                        verbose=1,
                                        min_delta=0.0001)

callbacks_list = [early_stopping,checkpoint,reduce_learningrate]

epochs = 48

model.compile(loss='categorical_crossentropy',
              optimizer = Adam(lr=0.001),
              metrics=['accuracy'])

history = model.fit_generator(generator=train_set,

steps_per_epoch=train_set.n//train_set.batch_size,
                                    epochs=epochs,
                                    validation_data = test_set,
                                    validation_steps =
test_set.n//test_set.batch_size,
                                    callbacks=callbacks_list
                                    )
```

```
/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/
engine/training.py:1844: UserWarning: `Model.fit_generator` is
deprecated and will be removed in a future version. Please use
`Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
2021-09-12 10:29:59.646073: I
tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of
the MLIR optimization passes are enabled (registered 2)
2021-09-12 10:29:59.650537: I
tensorflow/core/platform/profile_utils/cpu_utils.cc:112] CPU
Frequency: 2199995000 Hz

Epoch 1/48

2021-09-12 10:30:01.184226: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49]
Successfully opened dynamic library libcublas.so.11
2021-09-12 10:30:02.080952: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49]
Successfully opened dynamic library libcublasLt.so.11
```

```
2021-09-12 10:30:02.118051: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49]
Successfully opened dynamic library libcudnn.so.8

225/225 [==============================] - 165s 700ms/step - loss:
1.9170 - accuracy: 0.2713 - val_loss: 1.5695 - val_accuracy: 0.4071
Epoch 2/48
225/225 [==============================] - 30s 132ms/step - loss:
1.4542 - accuracy: 0.4382 - val_loss: 1.3962 - val_accuracy: 0.4744
Epoch 3/48
225/225 [==============================] - 30s 131ms/step - loss:
1.2821 - accuracy: 0.5069 - val_loss: 1.2841 - val_accuracy: 0.5138
Epoch 4/48
225/225 [==============================] - 30s 133ms/step - loss:
1.1848 - accuracy: 0.5479 - val_loss: 1.1735 - val_accuracy: 0.5614
Epoch 5/48
225/225 [==============================] - 30s 133ms/step - loss:
1.1237 - accuracy: 0.5754 - val_loss: 1.3103 - val_accuracy: 0.4747
Epoch 6/48
225/225 [==============================] - 29s 129ms/step - loss:
1.0747 - accuracy: 0.5898 - val_loss: 1.3306 - val_accuracy: 0.4913
Epoch 7/48
225/225 [==============================] - 29s 130ms/step - loss:
1.0090 - accuracy: 0.6181 - val_loss: 1.0748 - val_accuracy: 0.5972
Epoch 8/48
225/225 [==============================] - 29s 129ms/step - loss:
0.9811 - accuracy: 0.6287 - val_loss: 1.1196 - val_accuracy: 0.5778
Epoch 9/48
225/225 [==============================] - 29s 127ms/step - loss:
0.9303 - accuracy: 0.6496 - val_loss: 1.0919 - val_accuracy: 0.5876
Epoch 10/48
225/225 [==============================] - 29s 128ms/step - loss:
0.8880 - accuracy: 0.6676 - val_loss: 1.1163 - val_accuracy: 0.5786
Restoring model weights from the end of the best epoch.

Epoch 00010: ReduceLROnPlateau reducing learning rate to
0.00020000000949949026.
Epoch 00010: early stopping
```

```
model.save_weights('face_emotion_model.h5')
```

Plotting Accuracy & Loss

```
plt.style.use('dark_background')

plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
```

```
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```