Importing Libraries

In [1]:

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

# Importing Deep Learning Libraries

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

Displaying Images

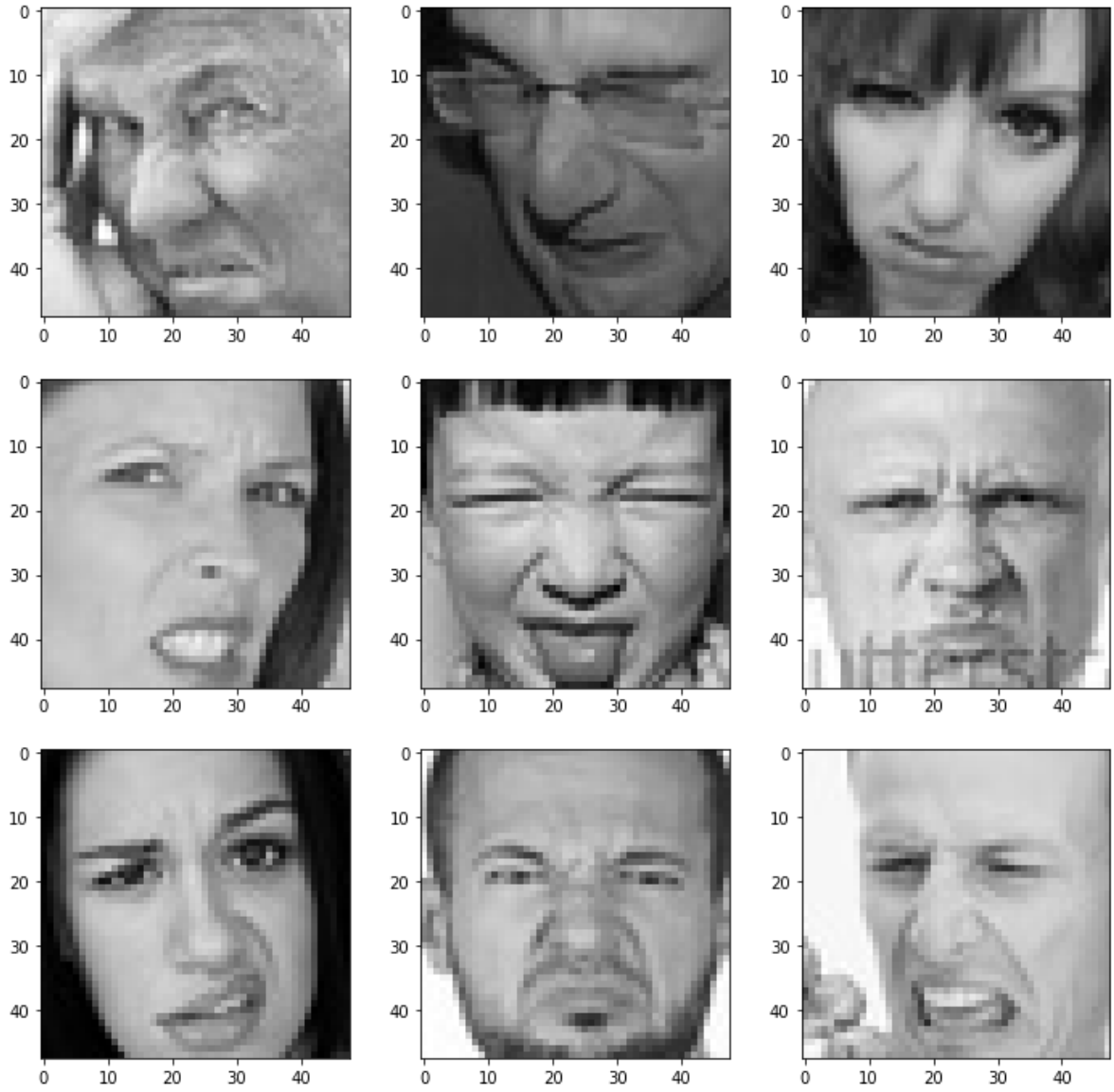
In [2]:

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

In [3]:

```
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_
    plt.imshow(img)
plt.show()
```



Making Training and Validation Data

In [4]:

```
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

In [5]:

```
from keras.optimizers import Adam,SGD,RMSprop


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"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 48, 48, 64)        640
_________________________________________________________________
batch_normalization (BatchNo (None, 48, 48, 64)        256
_________________________________________________________________
activation (Activation)      (None, 48, 48, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 128)       204928
_________________________________________________________________
batch_normalization_1 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
activation_1 (Activation)    (None, 24, 24, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 512)       590336
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 12, 512)       2048
_________________________________________________________________
activation_2 (Activation)    (None, 12, 12, 512)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 512)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 512)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_3 (Batch (None, 6, 6, 512)         2048
_________________________________________________________________
activation_3 (Activation)    (None, 6, 6, 512)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 3, 3, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 256)               1179904
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_5 (Batch (None, 512)               2048
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
```

```
_________________________________________________________________
dense_2 (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

In [8]:

```
from keras.optimizers import RMSprop,SGD,Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint("./model.h5", monitor='val_acc', verbose=1, save_best_only=Tru

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'])
```

In [9]:

```
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
                              )
```

```
Epoch 1/48
225/225 [==============================] - 180s 799ms/step - loss: 1.7711 -
accuracy: 0.3183 - val_loss: 1.7775 - val_accuracy: 0.3482
Epoch 2/48
225/225 [==============================] - 22s 99ms/step - loss: 1.4261 - ac
curacy: 0.4517 - val_loss: 1.3999 - val_accuracy: 0.4814
Epoch 3/48
225/225 [==============================] - 22s 97ms/step - loss: 1.2760 - ac
curacy: 0.5113 - val_loss: 1.2810 - val_accuracy: 0.5163
Epoch 4/48
225/225 [==============================] - 22s 100ms/step - loss: 1.1831 - a
ccuracy: 0.5490 - val_loss: 1.2041 - val_accuracy: 0.5491
Epoch 5/48
225/225 [==============================] - 23s 100ms/step - loss: 1.1257 - a
ccuracy: 0.5724 - val_loss: 1.2334 - val_accuracy: 0.5278
Epoch 6/48
225/225 [==============================] - 22s 98ms/step - loss: 1.0765 - ac
curacy: 0.5909 - val_loss: 1.2720 - val_accuracy: 0.5112
Epoch 7/48
225/225 [==============================] - 23s 101ms/step - loss: 1.0328 - a
ccuracy: 0.6091 - val_loss: 1.1415 - val_accuracy: 0.5705
Epoch 8/48
225/225 [==============================] - 23s 103ms/step - loss: 0.9813 - a
ccuracy: 0.6282 - val_loss: 1.3052 - val_accuracy: 0.5300
Epoch 9/48
225/225 [==============================] - 23s 103ms/step - loss: 0.9526 - a
ccuracy: 0.6418 - val_loss: 1.0722 - val_accuracy: 0.6038
Epoch 10/48
225/225 [==============================] - 27s 120ms/step - loss: 0.9028 - a
ccuracy: 0.6593 - val_loss: 1.0720 - val_accuracy: 0.6024
Epoch 11/48
225/225 [==============================] - 23s 104ms/step - loss: 0.8707 - a
ccuracy: 0.6724 - val_loss: 1.0670 - val_accuracy: 0.6081
Epoch 12/48
225/225 [==============================] - 22s 98ms/step - loss: 0.8188 - ac
curacy: 0.6906 - val_loss: 1.1353 - val_accuracy: 0.5786
Epoch 13/48
225/225 [==============================] - 23s 104ms/step - loss: 0.7788 - a
ccuracy: 0.7076 - val_loss: 1.0726 - val_accuracy: 0.6175
Epoch 14/48
225/225 [==============================] - ETA: 0s - loss: 0.7399 - accurac
y: 0.7234Restoring model weights from the end of the best epoch.

Epoch 00014: ReduceLROnPlateau reducing learning rate to 0.00020000000949949
026.
225/225 [==============================] - 23s 102ms/step - loss: 0.7399 - a
ccuracy: 0.7234 - val_loss: 1.0975 - val_accuracy: 0.6054
Epoch 00014: early stopping
```

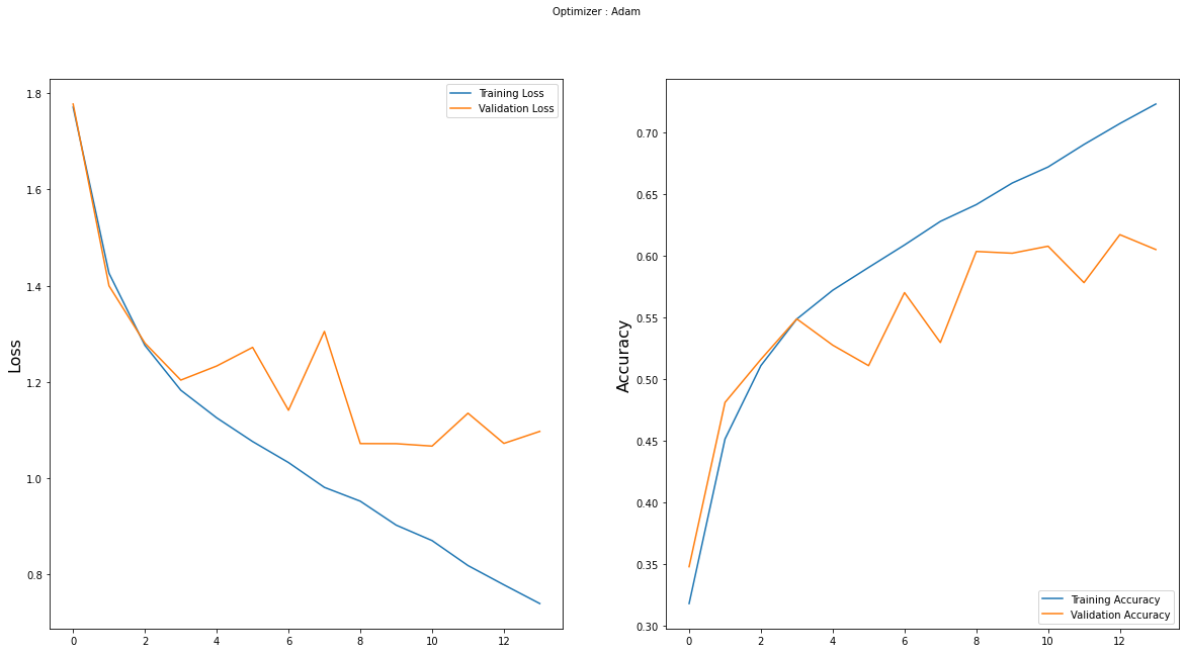
Plotting Accuracy & Loss

In [10]:

```
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()
```



In [ ]:

```

```