National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics, MSCDAD_JAN21A_I

Sachin Pralhad Langute
Student ID: 19234635

School of Computing
National College of Ireland

Supervisor: Prof. Jorge Basilio

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Sachin Pralhad Langute

Student ID: ……19234635……

Programme: MSC in Data Analytics MSCDAD_JAN21A_I **Year:** …2021-22……..

Module: ……Research Project……

Supervisor: ……Prof. Jorge Basilio……

Submission Due Date: ……31st January 2022……

Project Title: …Facial Emotion Recognition using Deep Convolutional Neural Network…

Word Count: ……1041…… **Page Count**……12……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Sachin Pralhad Langute……

Date: ……31st January 2022……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sachin Pralhad Langute
Student ID: 19234635

1 Introduction

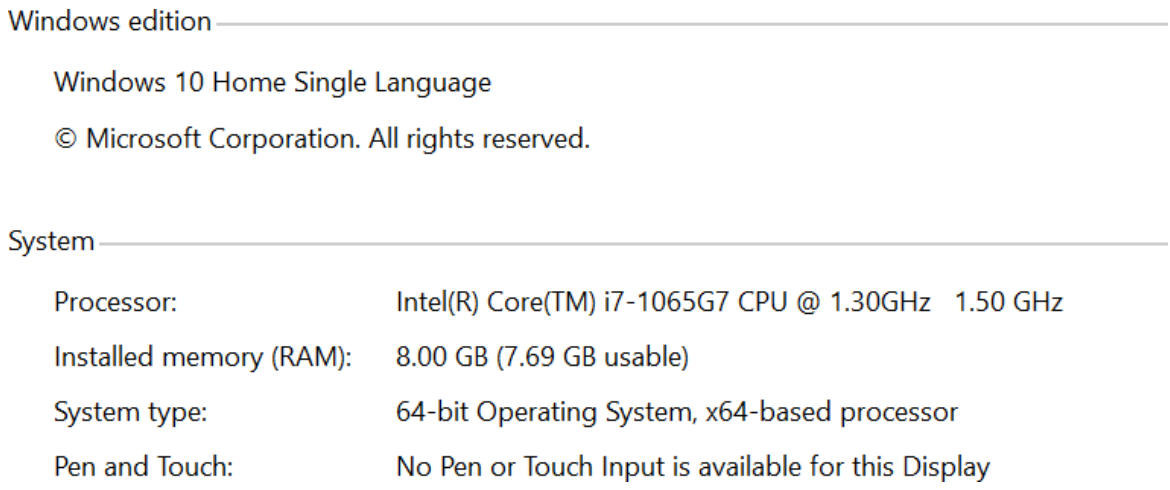
In this configuration manual, the author has given end-to-end details of all the process that was equipped in the study for the facial expression recognition system. Based on these aspects, the results were predicted. This manual also highlights the technical study which shows all the Python libraries which were used to study the working of the CNN model which was used for predictive analysis. The aim of this configuration manual is to make it easy for the reader to understand how things were processed from start to end.

2 Design Specification

It is a documented artifact explaining every single detail of the product, the use or the requirement of the product, the elements that make the product more important, and the technicalities associated with the product. This document must incorporate all the précised work equipped in this study including the challenges faced during the research.

2.1 Machine Hardware Configuration

Following hardware combinations were used to finish the project.



2.2 Machine Software Configuration

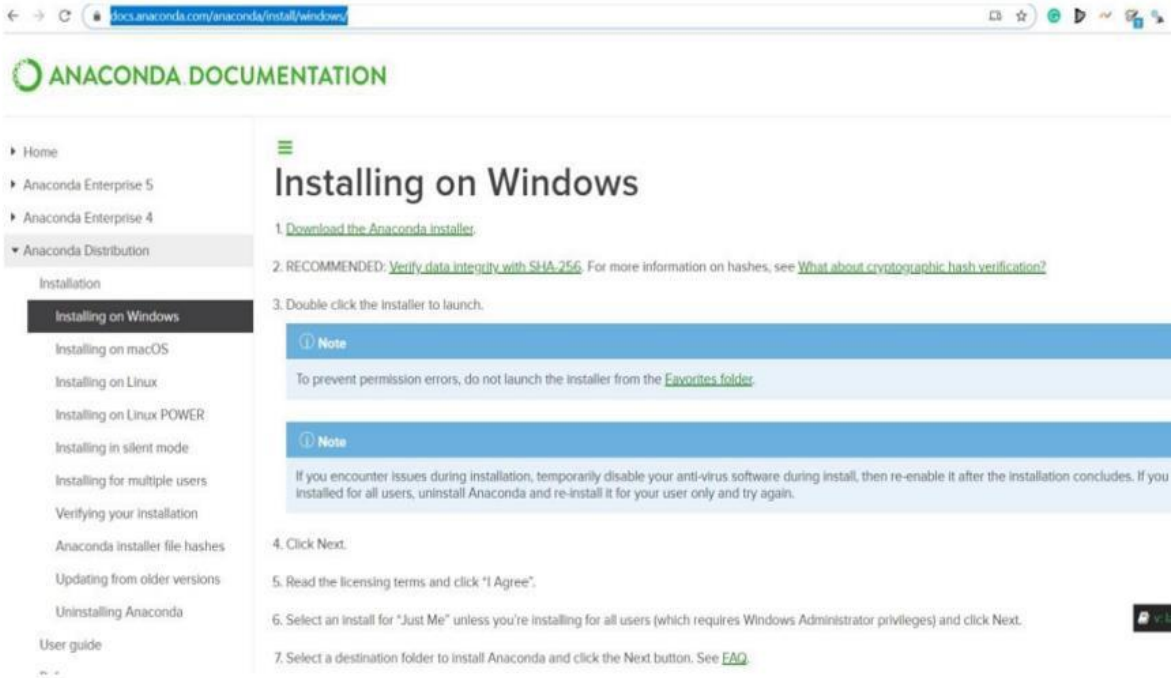
The specifications of the software utilized for the system were as follows:

Software	Configurations
Operating System	Windows 10 (64-bit Operating system)
IDE	Spyder (Anaconda Navigator)
Scripting Language	Python
Scripting Language version	Python 3.7

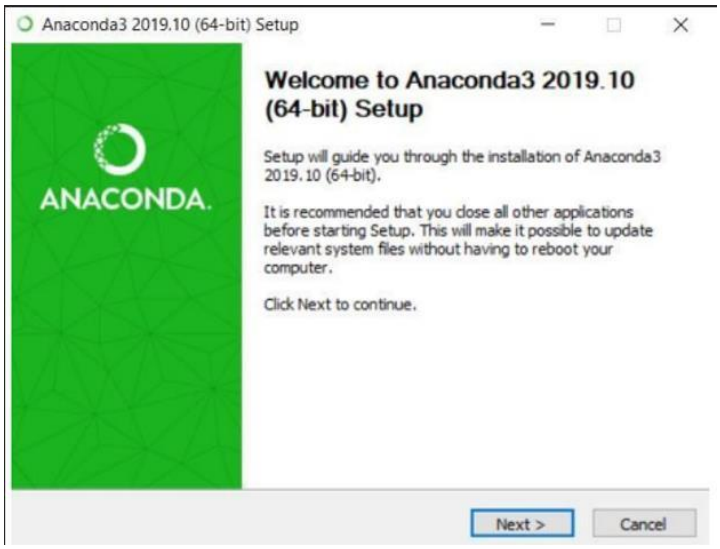
2.2.1 Anaconda Installation

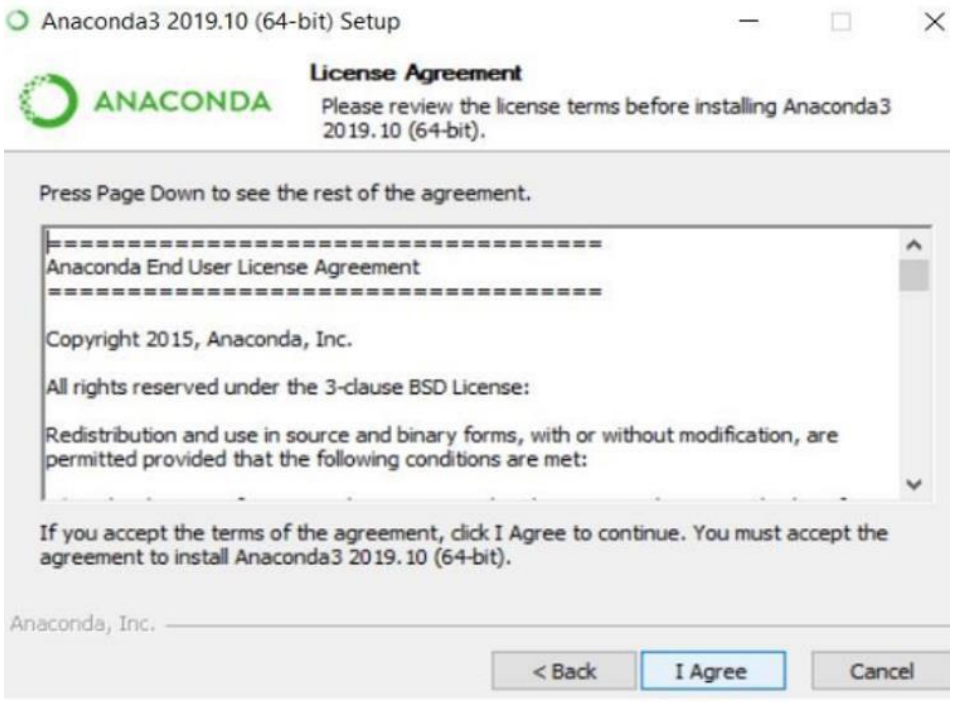
For the machine learning models, the most recent version of anaconda was installed. Its setup procedure involved the below steps: -

Go to the official anaconda website and select the link for windows installation.

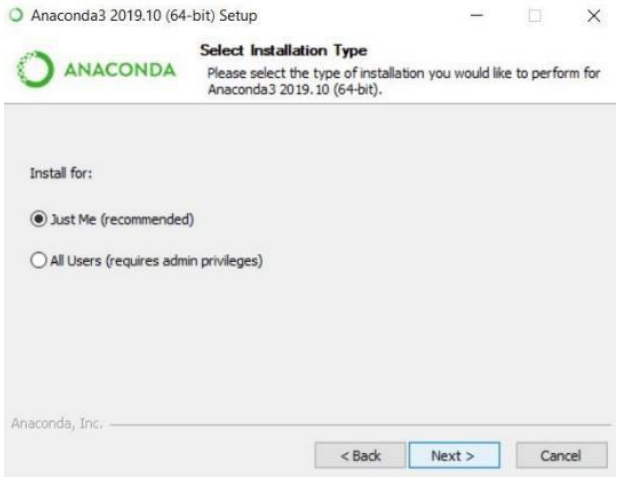


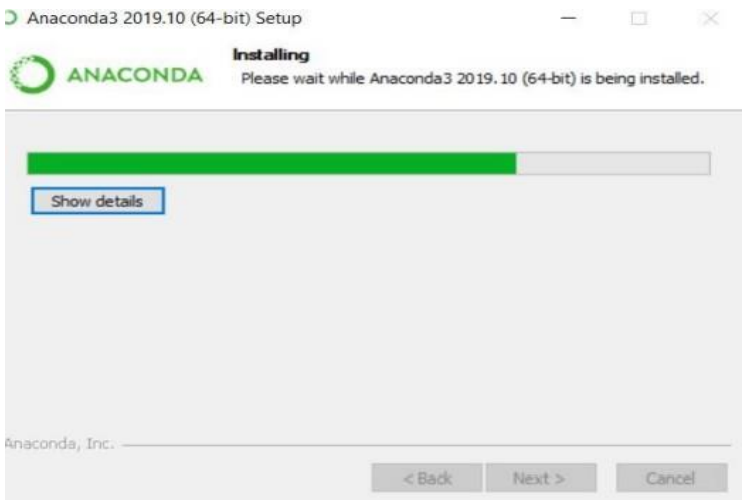
Download the Anaconda installer file and once downloaded, install the software on the machine.



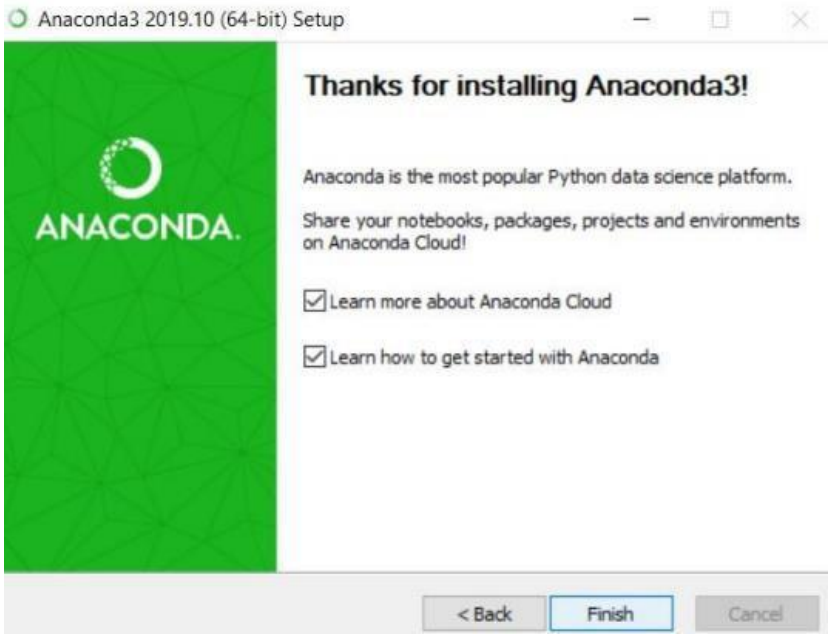


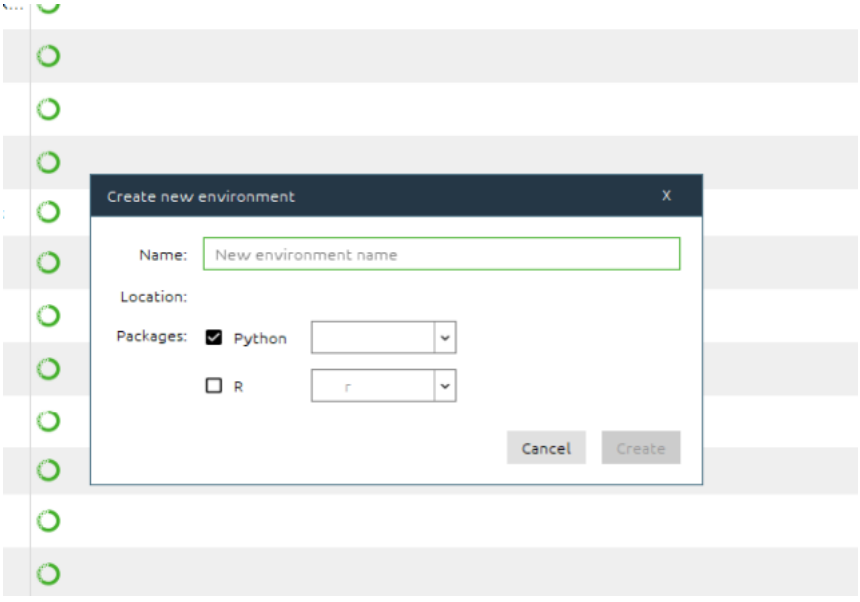
Click on all the positive terms mentioned in the installation setup.





The setup starts installing the application.





Create a new environment and give it a name.

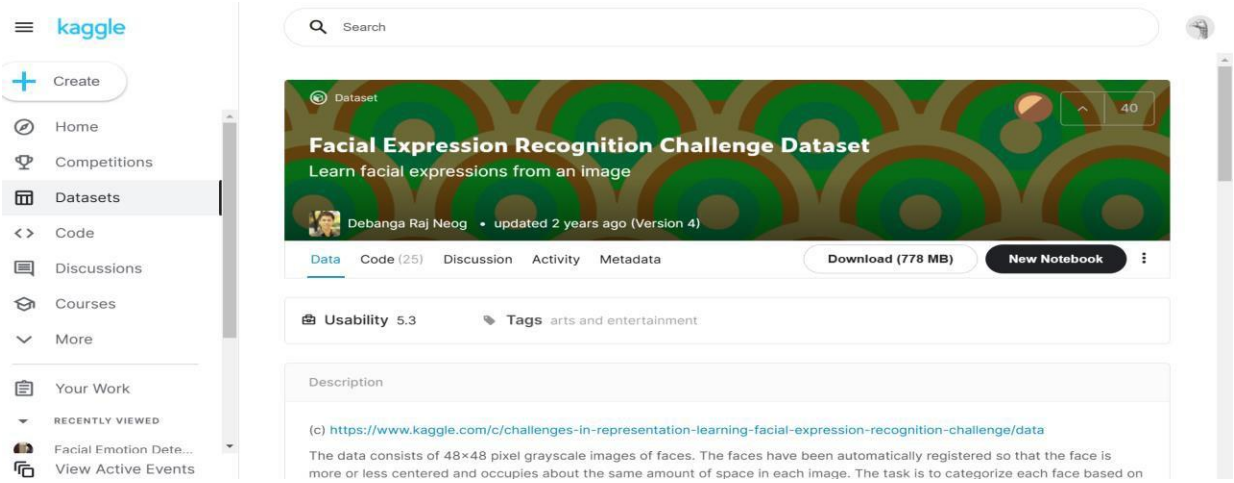
2.3 Libraries used for model implementation

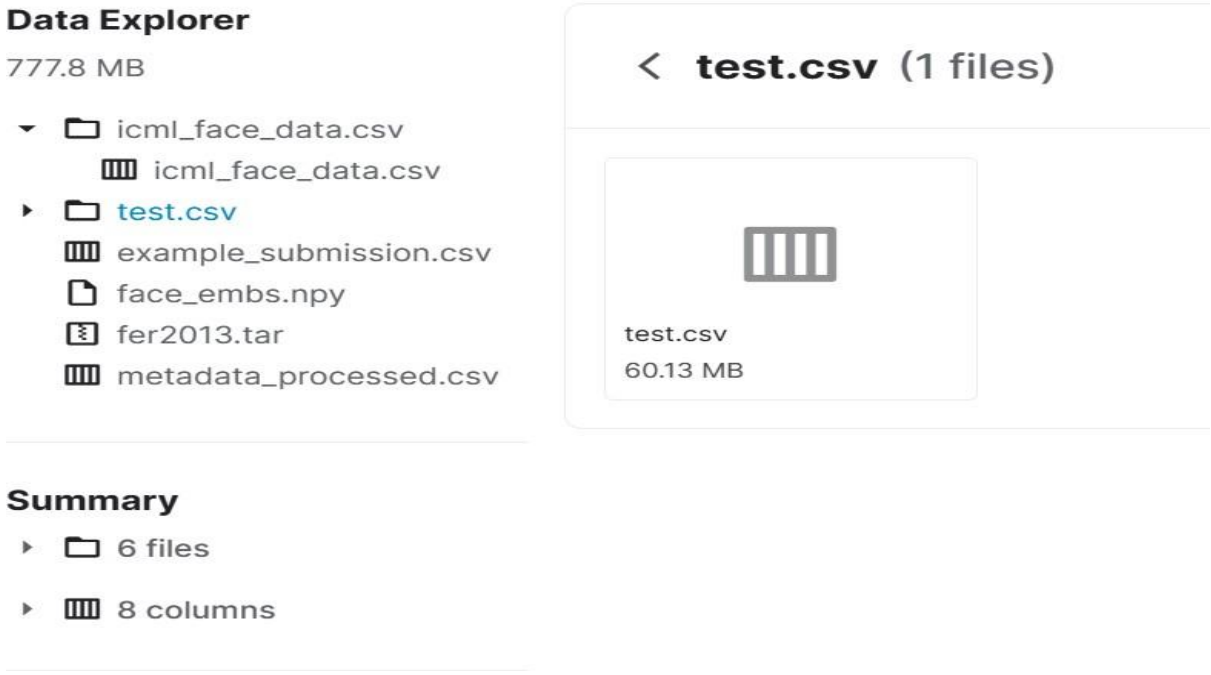
```
import json
import zipfile
import os
import pandas as pd
from matplotlib import pyplot
from math import sqrt
import numpy as np
import scipy.misc
from IPython.display import display
from keras.utils import np_utils
from keras.preprocessing.image import ImageDataGenerator
from keras.utils.vis_utils import plot_model
from keras.models import Model, Sequential
from keras.layers import Input, Dense, Flatten, Dropout, BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import concatenate
from tensorflow.keras.optimizers import Adam, SGD
from keras.regularizers import l1, l2

from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix
import itertools
from pylab import rcParams
```

Model 3: VGG16

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
import skimage.io
import keras.backend as K
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Flatten, Dropout,BatchNormalization ,Activation
from tensorflow.keras.models import Model, Sequential
from keras.applications.nasnet import NASNetLarge
from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint, EarlyStopping
from tensorflow.keras.optimizers import Adam
```

3 Dataset:





The data comprises grayscale pictures of faces 48x48 pixels in size. The faces have been already selected in such a way that highlights the center of the image with equal size for each image. There is a need for feature mapping for each image such that every image is categorized into one of the seven Ekman's emotions classes which include the facial expressions with a labeled numerical class (0=Angry, 1=Disgust, 2=Fear,3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

The dataset used for the FER system has two CSV files namely, train.csv and test.csv. The training dataset includes two-column data having the pixel value of the input image and the emotion associated with the pixel. There is also a column named "feeling" which depicts the mapped emotion class. The emotion column comprises the string value of the emotion which is enclosed in a quotation mark.

There is a single column in the testing dataset file with an emotion column enclosed with a string value for a particular emotion and the job is to predict the emotion depicted in that string value.

The training dataset includes 28,709 instances whereas, the leaderboard's public test set has 3,589 samples.

4 Data Preprocessing:

In Data preprocessing the author has converted the images from grayscale to RGB and also amplified the pixels of each image.

Preprocessing

In [24]:
```
def gray_to_rgb(im):
  '''
  converts images from single channel images to 3 channels
  '''

  w, h = im.shape
  ret = np.empty((w, h, 3), dtype=np.uint8)
  ret[:, :, 2] =  ret[:, :, 1] =  ret[:, :, 0] =  im
  return ret

def convert_to_image(pixels, mode="save", t="gray"):

  '''
  convert the input pixels from the single string row to  48*48 array with real pixel values
  when mode = "save" it keeps the images in flat array shape, otherwise it converts it to 48*48
  when t (for type) = "gray, it keeps the pixels single channel, otherwise it converts it to 3 channels
  '''

  if type(pixels) == str:
      pixels = np.array([int(i) for i in pixels.split()])
  if mode == "show":
    if t == "gray":
      return pixels.reshape(48,48)
    else:
      return gray_to_rgb(pixels.reshape(48,48))
  else:
      return pixels

data["pixels"] = data["pixels"].apply(lambda x : convert_to_image(x, mode="show", t="gray"))
from sklearn.model_selection import train_test_split
#split the data to train, test, and validation
X_train, X_test, y_train, y_test = train_test_split(data["pixels"],  data["emotion"], test_size=0.2, random_state=1)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=1)

X_train = np.array(list(X_train[:]), dtype=np.float)
X_val = np.array(list(X_val[:]), dtype=np.float)
X_test = np.array(list(X_test[:]), dtype=np.float)

y_train = np.array(list(y_train[:]), dtype=np.float)
y_val = np.array(list(y_val[:]), dtype=np.float)
y_test = np.array(list(y_test[:]), dtype=np.float)

X_train = X_train.reshape(X_train.shape[0], 48, 48, 1)
X_val = X_val.reshape(X_val.shape[0], 48, 48, 1)
X_test = X_test.reshape(X_test.shape[0], 48, 48, 1)
```

In [25]:
```
num_train = X_train.shape[0]
num_val = X_val.shape[0]
num_test = X_test.shape[0]
```

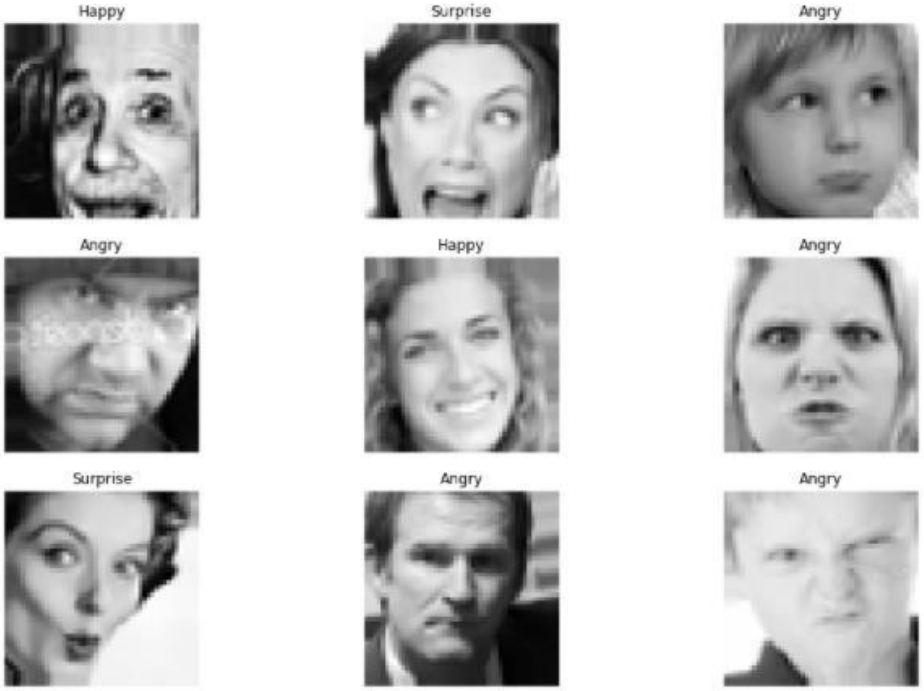
5 Data Augmentation:

The first strategy is data augmentation, which was utilized in all of our models as a preprocessing step. The author used Keras' "ImageDataGenerator" for data augmentation. It generates 32 augmented images from a single image by rotating, flipping, and employing other predefined techniques.

Data Augmentation

In [27]:
```
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range = 10,
    horizontal_flip = True,
    width_shift_range=0.1,
    height_shift_range=0.1,
    fill_mode = 'nearest')

testgen = ImageDataGenerator(
    rescale=1./255
    )
datagen.fit(X_train)
batch_size = 64
```

In [28]:
```
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9):
    for i in range(0, 9):
        pyplot.axis('off')
        pyplot.subplot(330 + 1 + i)
        # print(np.where(y_batch[i] == 1)[0][0])
        pyplot.title(emotion_labels[np.where(y_batch[i] == 1)[0][0]])
        pyplot.imshow(X_batch[i].reshape(48, 48), cmap=pyplot.get_cmap('gray'))
    pyplot.axis('off')
    pyplot.show()
    break
```



6 Design specification:

The author has used 3 models namely:

1. Convolutional Neural Network with 5 convolutional layers
2. Convolutional Neural Network with 3 convolutional layers
3. VGG16 model

Out of all the three models, VGG performs the best with an accuracy of 85.75% and a loss of 1.73. The CNN model with 5 convolutional layers has the lowest accuracy of 25.65% and loss of 1.78 and on the other hand, the CNN model with 3 convolutional layers performs the second-best with an accuracy of 57.27% and loss of 1.89.

6.1 Convolutional Neural Network with 5 convolutional layers:

Model 1: 5-Layered CNN ¶

```
conv5_model = Sequential()

conv5_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu',name = 'conv1'))
conv5_model.add(BatchNormalization())
conv5_model.add(MaxPooling2D(pool_size=(2, 2)))
conv5_model.add(Dropout(0.25))

conv5_model.add(Conv2D(128, kernel_size=(3, 3),padding="same", activation='relu', name = 'conv2'))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(128, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv3'))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(128, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv4'))
conv5_model.add(BatchNormalization())
conv5_model.add(MaxPooling2D(pool_size=(2, 2)))
conv5_model.add(Dropout(0.25))

conv5_model.add(Conv2D(128, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv5'))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(128, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv6'))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(128, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv7'))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(128, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv8'))
conv5_model.add(BatchNormalization())
conv5_model.add(MaxPooling2D(pool_size=(2, 2)))
conv5_model.add(Dropout(0.25))

conv5_model.add(Conv2D(128, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv9'))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(128, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv10'))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(128, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv11'))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(128, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv12'))
conv5_model.add(BatchNormalization())
conv5_model.add(MaxPooling2D(pool_size=(2, 2)))
conv5_model.add(Dropout(0.25))

conv5_model.add(Conv2D(256, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv13'))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(256, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv14'))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(256, kernel_size=(3, 3), padding="same", activation='relu',  name = 'conv16'))
conv5_model.add(BatchNormalization())
conv5_model.add(Conv2D(256, kernel_size=(3, 3), padding="same", activation='relu', name = 'conv17'))
conv5_model.add(BatchNormalization())
conv5_model.add(MaxPooling2D(pool_size=(2, 2)))
conv5_model.add(Dropout(0.25))


conv5_model.add(Flatten())
conv5_model.add(Dense(num_classes, activation='softmax'))
print(conv5_model.summary())
```

```
Model: "sequential"
```

Result Evaluation for CNN5:

```
loss = conv5_model.evaluate_generator(test_flow, steps=len(X_test) / batch_size)
print("Test Loss " + str(loss[0]))
print("Test Acc: " + str(loss[1]))
```

```
C:\Users\langu\anaconda3\lib\site-packages\keras\engine\training.py:2006: UserWarning:
and will be removed in a future version. Please use `Model.evaluate`, which supports g
  warnings.warn('`Model.evaluate_generator` is deprecated and '

Test Loss 1.7888416051864624
Test Acc: 0.2565308213233948
```

6.2 Convolutional Neural Network with 3 convolutional layers:

Model 2: CNN with less convulational layers.

```
#Let's try smaller Conv model
model_conv = Sequential()

model_conv.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model_conv.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model_conv.add(MaxPooling2D(pool_size=(2, 2)))
model_conv.add(Dropout(0.25))

model_conv.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model_conv.add(MaxPooling2D(pool_size=(2, 2)))
model_conv.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model_conv.add(MaxPooling2D(pool_size=(2, 2)))
model_conv.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model_conv.add(MaxPooling2D(pool_size=(2, 2)))
model_conv.add(Dropout(0.25))

model_conv.add(Flatten())
model_conv.add(Dense(1024, activation='relu'))
model_conv.add(Dropout(0.5))
model_conv.add(Dense(7, activation='softmax'))
```

Result:

```
loss = model_conv.evaluate_generator(test_flow, steps=len(X_test) / batch_size)
print("Test Loss " + str(loss[0]))
print("Test Acc: " + str(loss[1]))
```

```
Test Loss 1.189395785331726
Test Acc: 0.5727969408035278
```

6.3 VGG model:

```
# Building Model

model=Sequential()
model.add(base_model)
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(4096,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1024,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7,activation='softmax'))
```

```
# Model Summary
for layer in base_model.layers:
    layer.trainable=False
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 1, 1, 512)         14714688
_________________________________________________________________
flatten_2 (Flatten)          (None, 512)               0
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_11 (Dense)             (None, 4096)              2101248
_________________________________________________________________
dropout_9 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_12 (Dense)             (None, 1024)              4195328
_________________________________________________________________
dropout_10 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_13 (Dense)             (None, 7)                 7175
=================================================================
Total params: 21,018,439
Trainable params: 6,303,751
Non-trainable params: 14,714,688
```

Result:

```
history=model.fit(train_dataset,validation_data=valid_dataset,epochs = 5,verbose = 1,callbacks=[lrd,mcp,es])
```

```
Epoch 1/5
359/359 [==============================] - 505s 1s/step - loss: 1.8006 - accuracy: 0.8560 - precision: 0.3931 - recall: 0.0143
- auc: 0.6741 - f1_score: 0.0259 - val_loss: 1.7264 - val_accuracy: 0.8581 - val_precision: 0.6194 - val_recall: 0.0167 - val_a
uc: 0.7160 - val_f1_score: 0.0323
Epoch 2/5
359/359 [==============================] - 671s 2s/step - loss: 1.7517 - accuracy: 0.8573 - precision: 0.5168 - recall: 0.0207
- auc: 0.6938 - f1_score: 0.0390 - val_loss: 1.7014 - val_accuracy: 0.8580 - val_precision: 0.6957 - val_recall: 0.0111 - val_a
uc: 0.7267 - val_f1_score: 0.0219
Epoch 3/5
359/359 [==============================] - 509s 1s/step - loss: 1.7398 - accuracy: 0.8572 - precision: 0.5068 - recall: 0.0226
- auc: 0.7000 - f1_score: 0.0426 - val_loss: 1.6890 - val_accuracy: 0.8591 - val_precision: 0.6840 - val_recall: 0.0253 - val_a
uc: 0.7316 - val_f1_score: 0.0480
Epoch 4/5
359/359 [==============================] - 463s 1s/step - loss: 1.7365 - accuracy: 0.8571 - precision: 0.4942 - recall: 0.0241
- auc: 0.7019 - f1_score: 0.0450 - val_loss: 1.7137 - val_accuracy: 0.8575 - val_precision: 0.8571 - val_recall: 0.0031 - val_a
uc: 0.7304 - val_f1_score: 0.0063
Epoch 5/5
359/359 [==============================] - 404s 1s/step - loss: 1.7344 - accuracy: 0.8575 - precision: 0.5249 - recall: 0.0243
- auc: 0.7036 - f1_score: 0.0456 - val_loss: 1.6914 - val_accuracy: 0.8586 - val_precision: 0.7083 - val_recall: 0.0178 - val_a
uc: 0.7324 - val_f1_score: 0.0343
```

References

Facial Expression Recognition Challenge Dataset | Kaggle

Mishra S., Prasada G.R.B., Kumar R.K., Sanyal G. (2017) Emotion Recognition Through Facial Gestures - A Deep Learning Approach. In: Ghosh A., Pal R., Prasath R. (eds) Mining Intelligence and Knowledge Exploration. MIKE 2017. Lecture Notes in Computer Science, vol 10682. Springer, Cham. https://doi.org/10.1007/978-3-319-71928-3_2

Mehta, D.; Siddiqui, M.F.H.; Javaid, A.Y. Facial Emotion Recognition: A Survey and Real-World User Experiences in Mixed Reality. Sensors 2018, 18, 416. https://doi.org/10.3390/s18020416

Castellano G., Kessous L., Caridakis G. (2008) Emotion Recognition through Multiple Modalities: Face, Body Gesture, Speech. In: Peter C., Beale R. (eds) Affect and Emotion in Human-Computer Interaction. Lecture Notes in Computer Science, vol 4868. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3- 540-85099-1_8