Algorithm 1 Algorithm

```
procedure AUDIO_TO_IMAGE
    for each file in audio_files do
        Data = librosa.load(file)
        Image = librosa.plot_melspectrogram(Data)
        save_image("decoded_audio_file_name")

procedure CREATETRAINING_DATA(Images)
    for each I in Images do
        New_Image_1 = filter(I.mirror)
        New_Image_2 = filter(I.saturate)
    N_Images = New_Image_1 + New_Image_2
    return N_Images

procedure FINDLABEL(Image_Name)
    if 'neutral' in Image_Name then
        return 0
    else if 'calm' in Image_Name then
        return 1
    else if 'happy' in Image_Name then
        return 2
    else if 'sad' in Image_Name then
        return 3
    else if 'angry' in Image_Name then
        return 4
    else if 'fearful' in Image_Name then
        return 5
    else if 'disgust' in Image_Name then
        return 6
    else if 'surprise' in Image_Name then
        return 7

procedure NORMALIZE(N_Images)
    return N_Images.pixelvalues/255

procedure SPLIT_DATA(N_Images,labels)
    Training_data, Testing_data = split(N_Images, labels)
    Training_data, Validation_data = split(Training_data, Training_label)

Optimizer = Adam
Loss = Categorical_crossentropy
Epochs = 100

procedure SELF_CNN(Training_data,Training_label,Validation_data,Testing_data)
    model1.compile(Optimizer, Loss)
    model1.fit(Training_data, Training_label, Validation_data, Epochs)
    predictions = model1.predict(Testing_data)
    return  model1

procedure XCEPTION(Training_data,Training_label,Validation_data,Testing_data)
    model2.compile(Optimizer, Loss = Binary_crossentropy)
    model2.fit(Training_data, Training_label, Validation_data, Epochs)
    predictions = model2.predict(Testing_data)
    return  model2
```

procedure VGG_19(Training_data,Training_label,Validation_data,Testing_data)
    $model3.compile(Optimizer, Loss = Binary\_crossentropy)$
    $model3.fit(Training\_data, Training\_label, Validation\_data, Epochs)$
    $predictions = model3.predict(Testing\_data)$
    $return \quad model3$

procedure VGG_16(Training_data,Training_label,Validation_data,Testing_data)
    $model4.compile(Optimizer, Loss)$
    $model4.fit(Training\_data, Training\_label, Validation\_data, Epochs)$
    $predictions = model4.predict(Testing\_data)$
    $return \quad model4$

procedure INCEPTION_V3(Training_data,Training_label,Validation_data,Testing_data)
    $model5.compile(Optimizer, Loss = Binary\_crossentropy)$
    $model5.fit(Training\_data, Training\_label, Validation\_data, Epochs)$
    $predictions = model5.predict(Testing\_data)$
    $return \quad model5$

$m1 = Self\_CNN(Training\_data, Training\_label, Validation\_data, Testing\_data)$
$m2 = Vgg\_19(Training\_data, Training\_label, Validation\_data, Testing\_data)$
$m3 = Vgg\_16(Training\_data, Training\_label, Validation\_data, Testing\_data)$
$m4 = Inception\_V3(Training\_data, Training\_label, Validation\_data, Testing\_data)$

procedure INTEGRATED_MODEL(Training_data,Training_label,Validation_data,Testing_data)
    $QiCNN = concatenate(m1, m2, m3, m4)$
    $QiCNN.compile(Optimizer, Loss)$
    $QiCNN.fit(Training\_data, Training\_label, Validation\_data, Epochs)$
    $predictions = QiCNN.predict(Testing\_data)$
End