```
!wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cool
!unzip Project.zip
```

```
  inflating: Project/train/fear/Training_35319763.jpg
  inflating: Project/train/fear/Training_15157808.jpg
  inflating: Project/train/fear/Training_83387439.jpg
  inflating: Project/train/fear/Training_34531243.jpg
  inflating: Project/train/fear/Training_27946164.jpg
  inflating: Project/train/fear/Training_97995491.jpg
  inflating: Project/train/fear/Training_19743641.jpg
  inflating: Project/train/fear/Training_95343377.jpg
  inflating: Project/train/fear/Training_63558847.jpg
  inflating: Project/train/fear/Training_58420539.jpg
  inflating: Project/train/fear/Training_64070728.jpg
  inflating: Project/train/fear/Training_25554901.jpg
  inflating: Project/train/fear/Training_79381162.jpg
  inflating: Project/train/fear/Training_79677247.jpg
  inflating: Project/train/fear/Training_54477849.jpg
  inflating: Project/train/fear/Training_9927532.jpg
  inflating: Project/train/fear/Training_92733372.jpg
  inflating: Project/train/fear/Training_36613837.jpg
  inflating: Project/train/fear/Training_12920803.jpg
  inflating: Project/train/fear/Training_35896416.jpg
  inflating: Project/train/fear/Training_79433306.jpg
  inflating: Project/train/fear/Training_69526111.jpg
  inflating: Project/train/fear/Training_56012016.jpg
  inflating: Project/train/fear/Training_39107560.jpg
  inflating: Project/train/fear/Training_67477638.jpg
  inflating: Project/train/fear/Training_99694629.jpg
  inflating: Project/train/fear/Training_31859329.jpg
  inflating: Project/train/fear/Training_90213112.jpg
  inflating: Project/train/fear/Training_84064155.jpg
  inflating: Project/train/fear/Training_17695479.jpg
  inflating: Project/train/fear/Training_21589734.jpg
  inflating: Project/train/fear/Training_34784492.jpg
  inflating: Project/train/fear/Training_57271504.jpg
  inflating: Project/train/fear/Training_95166718.jpg
  inflating: Project/train/fear/Training_55588982.jpg
                                    r/Training_91514574.jpg
                                    r/Training_76508064.jpg
                                    r/Training_9826449.jpg
  inflating: Project/train/fear/Training_50831242.jpg
  inflating: Project/train/fear/Training_88630615.jpg
  inflating: Project/train/fear/Training_98636857.jpg
  inflating: Project/train/fear/Training_32230987.jpg
  inflating: Project/train/fear/Training_93097760.jpg
  inflating: Project/train/fear/Training_58373196.jpg
  inflating: Project/train/fear/Training_95888749.jpg
  inflating: Project/train/fear/Training_44816489.jpg
  inflating: Project/train/fear/Training_33039338.jpg
  inflating: Project/train/fear/Training_77685118.jpg
  inflating: Project/train/fear/Training_26116206.jpg
  inflating: Project/train/fear/Training_1508040.jpg
  inflating: Project/Pipfile.lock
  inflating: Project/model.png
  inflating: Project/model.py
  inflating: Project/main.py
  inflating: Project/camera.py
  inflating: Project/model_weights.h5
   creating: Project/.ipynb_checkpoints/
  inflating: Project/.ipynb_checkpoints/Facial_Expression_Training-checkpoint.ipynb
```



```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt


from tensorflow.keras.preprocessing.image import load_img, img_to_array, ImageDataGenerator


datagen = ImageDataGenerator(rotation_range=45,
                             vertical_flip=True,
                             horizontal_flip=True,
                             width_shift_range=0.2,
                             height_shift_range=0.2,
                             zoom_range=0.5,
                             shear_range=0.2,
                             brightness_range=[0.1,0.25])
```

```
sampleImage = load_img(path='/content/Project/train/angry/Training_10118481.jpg', keep_aspect_ratio=True)
```

```
sampleImage = img_to_array(sampleImage)
```

```
sampleImage = np.expand_dims(sampleImage, axis=0)
```

```
datagen.fit(sampleImage)
```

```
#count = 0
#for Xbatch in datagen_srk.flow(sampleImage_srk, save_to_dir='/content/actors/srk/', save_format='jpeg'):
  #count = count+1
  #if count==10:
   # break
```

```
train_dir = "/content/Project/train"
test_dir = "/content/Project/test"
```

```
len(train_dir)
```

```
22
```

```
len(test_dir)
```

```
21
```

```
import os
Class_name=os.listdir("/content/Project/train")
Class_name
```

```
['disgust', 'fear', 'neutral', 'surprise', 'angry', 'happy', 'sad']
```

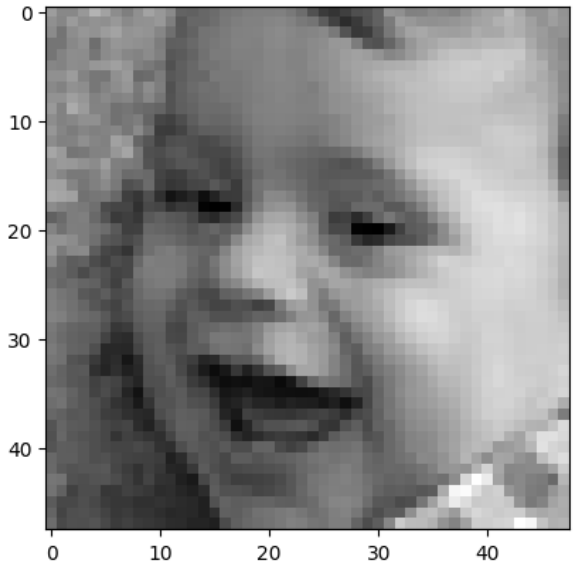
Saved successfully!

```
ear',3:'Happy',4:'Neutral',5:'Sad',6:'Surprise'}
```

```
import keras
from keras.preprocessing import image
```

```
from PIL import Image
image = Image.open('/content/Project/train/happy/Training_10070997.jpg')
plt.imshow(image,cmap='gray')
```

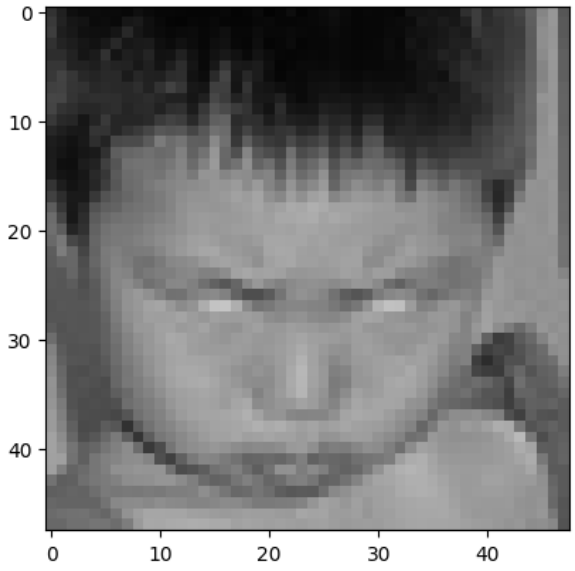
```
<matplotlib.image.AxesImage at 0x7f912673d070>
```




```
import cv2
#read first file name
filepath = '/content/Project/train/angry/Training_10118481.jpg'
#read image
img_read = load_img(filepath, color_mode='rgb', target_size=(48, 48))
```

```
#covert to numpy
img_np = np.array(img_read)
#add dimension
img_np_Angry = np.expand_dims(img_read, axis=0)
#shape
img_np_Angry.shape
#scale
img_np_Angry_scaled = img_np_Angry/255.0


plt.imshow(img_np)
```

```
<matplotlib.image.AxesImage at 0x7f9118d33c70>
```




```
 width_shift_range = 0.1,
 height_shift_range = 0.1,
 horizontal_flip = True,
 rescale = 1./255,
 validation_split = 0.2
)

train_generator = train_datagen.flow_from_directory(directory = train_dir,
                                                    target_size = (img_size,img_size),
                                                    batch_size = 16,
                                                    color_mode = "grayscale",
                                                    class_mode = "categorical",
                                                    subset = "training"
                                                   )
```

```
Found 22968 images belonging to 7 classes.
```

```
validation_datagen = ImageDataGenerator(rescale = 1./255,
                                        validation_split = 0.2)

validation_generator = validation_datagen.flow_from_directory( directory = test_dir,
                                                               target_size = (img_size,img_size),
                                                               batch_size = 16,
                                                               color_mode = "grayscale",
                                                               class_mode = "categorical",
                                                               subset = "validation"
                                                              )
```

```
Found 1432 images belonging to 7 classes.
```

```
img_np_Angry.shape
```

```
(1, 48, 48, 3)
```

```
img_np.shape
```

```
(48, 48, 3)
```

```
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout


emotionsANN=  Sequential()


emotionsANN.add(Flatten())


#256*256*3 - input dimensions
emotionsANN.add(Dense(units=128, activation='relu'))
emotionsANN.add(Dropout(rate=0.2))
#[variance] overfit - increase dropout and [bias] underfit we decrease dropout

emotionsANN.add(Dense(units=512, activation='relu'))
#rate-it is ratio of units of previous layer to randomly cancel at each iteration
emotionsANN.add(Dropout(rate=0.2))

#final layer - classification problem with 2 classes
emotionsANN.add(Dense(units=7, activation='softmax'))


from tensorflow.keras.optimizers.schedules import ExponentialDecay
from keras.optimizers import Adam

initial_learning_rate = 0.001
lr=ExponentialDecay(initial_learning_rate,
    decay_steps=100000,
    decay_rate=0.96,
    staircase=True)

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


Saved successfully!
ckpoint, EarlyStopping, ReduceLROnPlateau
uracy", patience=10, verbose=1)
#rd = ReduceLROnPlateau(monitor="val_accuracy",factor=0.1,patience=5,verbose=1)

mc = ModelCheckpoint(filepath='bestmodel.h5', monitor='val_accuracy', mode='max', verbose=1, save_best_only=True)


epochs= 15
batch_szie=60
callbacks=[mc,es]
history = emotionsANN.fit(x = train_generator,epochs = epochs,validation_data = validation_generator)
```

```
Epoch 1/15
1436/1436 [==============================] - 34s 22ms/step - loss: 1.8233 - accuracy: 0.2469 - val_loss: 1.8152 - val_accur
Epoch 2/15
1436/1436 [==============================] - 28s 20ms/step - loss: 1.8135 - accuracy: 0.2511 - val_loss: 1.8133 - val_accur
Epoch 3/15
1436/1436 [==============================] - 29s 20ms/step - loss: 1.8121 - accuracy: 0.2513 - val_loss: 1.8161 - val_accur
Epoch 4/15
1436/1436 [==============================] - 29s 20ms/step - loss: 1.8120 - accuracy: 0.2513 - val_loss: 1.8133 - val_accur
Epoch 5/15
1436/1436 [==============================] - 28s 19ms/step - loss: 1.8126 - accuracy: 0.2513 - val_loss: 1.8143 - val_accur
Epoch 6/15
1436/1436 [==============================] - 28s 20ms/step - loss: 1.8110 - accuracy: 0.2516 - val_loss: 1.8137 - val_accur
Epoch 7/15
1436/1436 [==============================] - 41s 28ms/step - loss: 1.8112 - accuracy: 0.2517 - val_loss: 1.8131 - val_accur
Epoch 8/15
1436/1436 [==============================] - 45s 31ms/step - loss: 1.8115 - accuracy: 0.2515 - val_loss: 1.8144 - val_accur
Epoch 9/15
1436/1436 [==============================] - 31s 22ms/step - loss: 1.8104 - accuracy: 0.2516 - val_loss: 1.8142 - val_accur
Epoch 10/15
1436/1436 [==============================] - 29s 20ms/step - loss: 1.8115 - accuracy: 0.2515 - val_loss: 1.8140 - val_accur
Epoch 11/15
1436/1436 [==============================] - 29s 20ms/step - loss: 1.8111 - accuracy: 0.2517 - val_loss: 1.8161 - val_accur
Epoch 12/15
1436/1436 [==============================] - 28s 20ms/step - loss: 1.8114 - accuracy: 0.2515 - val_loss: 1.8147 - val_accur
Epoch 13/15
1436/1436 [==============================] - 34s 24ms/step - loss: 1.8107 - accuracy: 0.2516 - val_loss: 1.8135 - val_accur
Epoch 14/15
1436/1436 [==============================] - 30s 21ms/step - loss: 1.8111 - accuracy: 0.2516 - val_loss: 1.8148 - val_accur
Epoch 15/15
1436/1436 [==============================] - 29s 20ms/step - loss: 1.8108 - accuracy: 0.2516 - val_loss: 1.8140 - val_accur
```

```
emotionsANN.compile(
    optimizer = 'Adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
  )
```

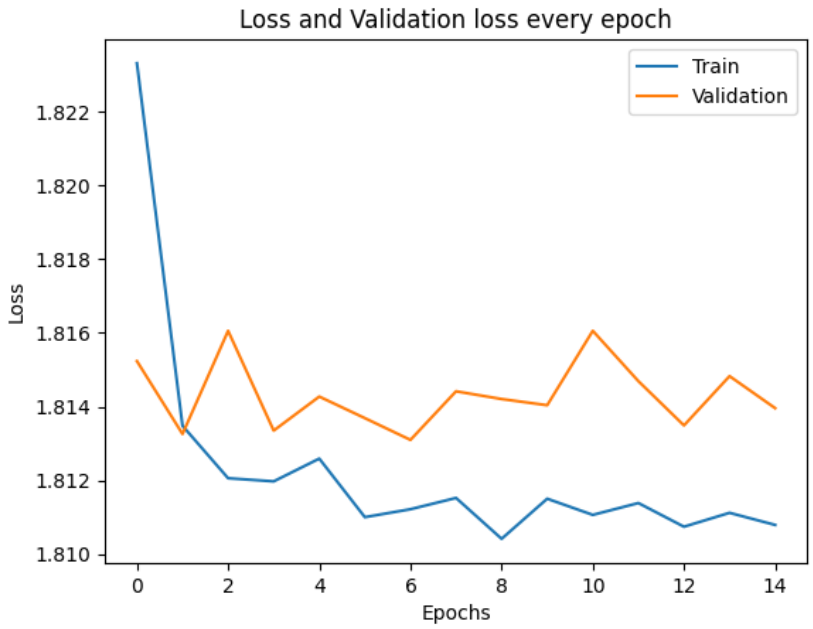
```
emotionsANN.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, None)              0

 dense (Dense)               (None, 128)               295040

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 512)               66048

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 7)                 3591

=================================================================
Total params: 364,679
Trainable params: 364,679
Non-trainable params: 0
_________________________________________________________________
```

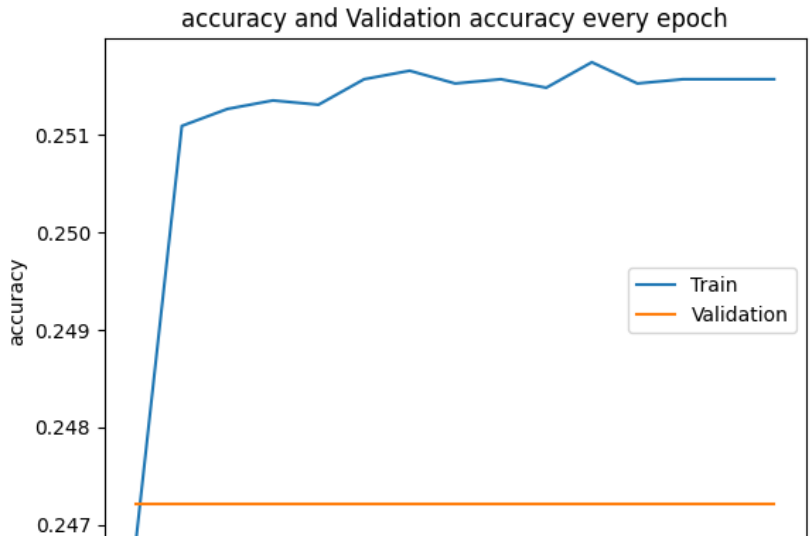
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

Saved successfully!

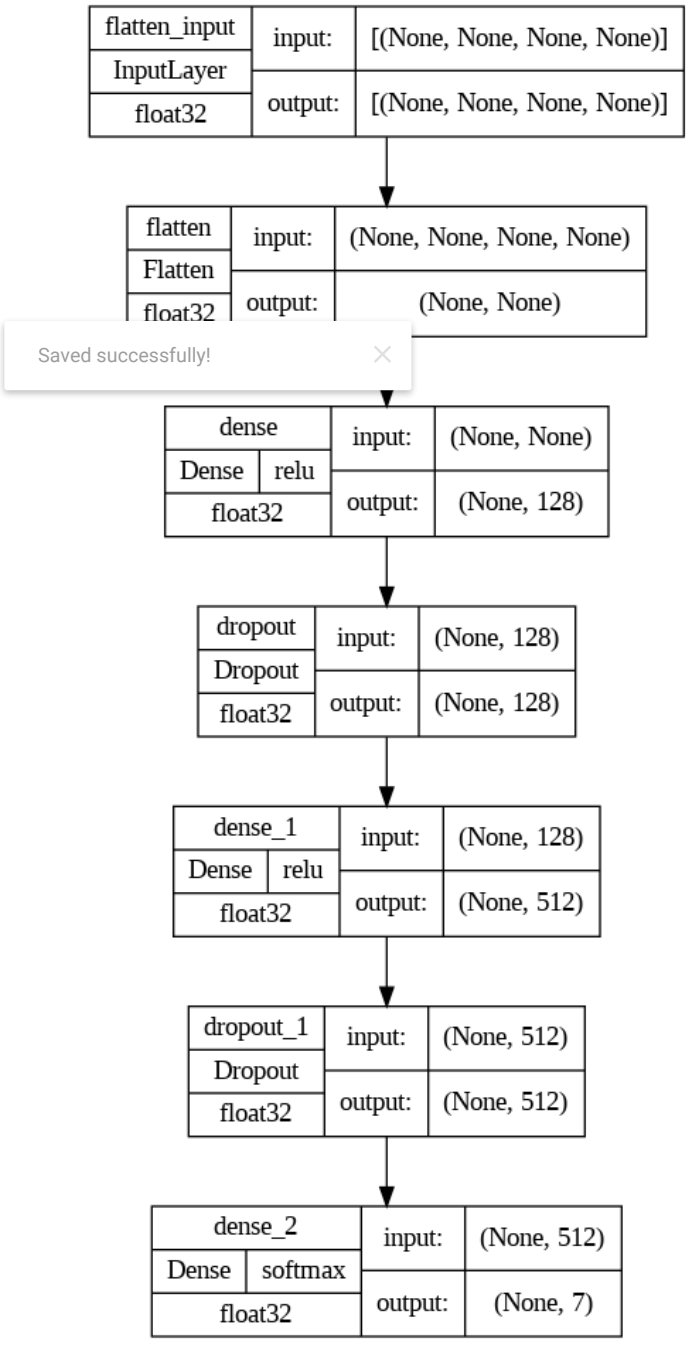
```
every epoch')
```




```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel('Epochs')
plt.ylabel('accuracy')
plt.legend(['Train', 'Validation'])
plt.title('accuracy and Validation accuracy every epoch')
plt.show()
```




```
from tensorflow.keras.utils import plot_model
plot_model(emotionsANN, show_shapes=True, show_dtype=True, show_layer_activations=True, show_layer_names=True)
```



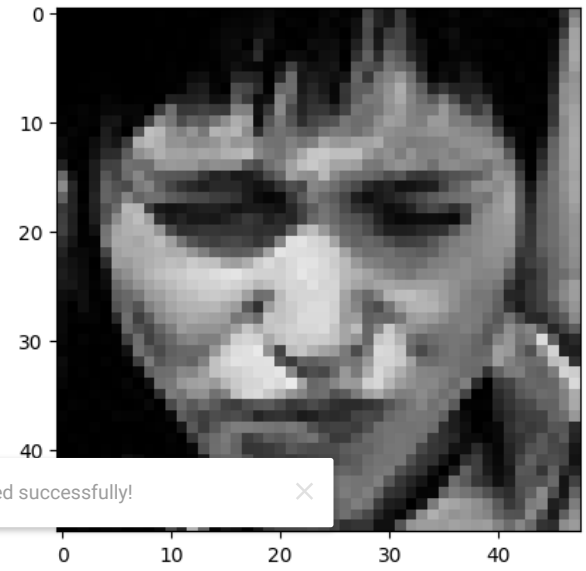
Saved successfully!

```
print("Testing Accuracy",emotionsANN.evaluate(validation_generator ))
```

```
90/90 [==============================] - 1s 9ms/step - loss: 1.8140 - accuracy: 0.2472
Testing Accuracy [1.8139580488204956, 0.2472067028284073]
```

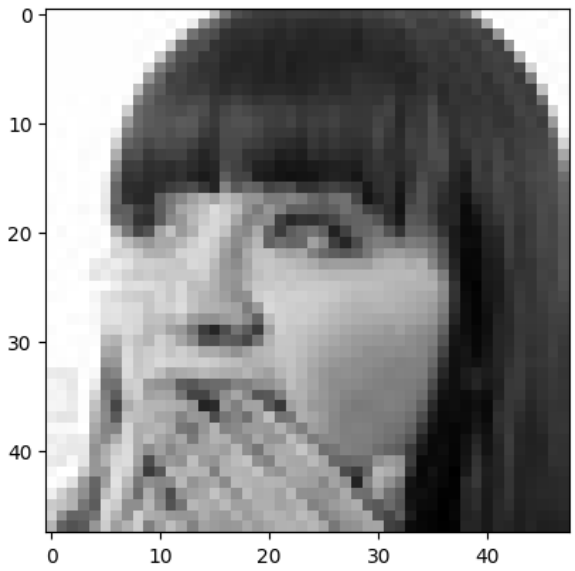
```
from tensorflow.keras.utils import load_img, img_to_array

import keras
import tensorflow as tf

color_mode = "grayscale"
image = keras.utils.load_img('/content/Project/test/sad/PrivateTest_10455506.jpg', target_size=(48,48))



image = np.array(image)
plt.imshow(image)
print(image.shape) #prints (48,48) that is the shape of our image
```

```
(48, 48, 3)
```




```
image2 = keras.utils.load_img('/content/Project/test/surprise/PrivateTest_104142.jpg', target_size=(48,48))



image = np.array(image)
plt.imshow(image)
print(image.shape) #prints (48,48) that is the shape of our image
```

```
(48, 48, 3)
```




```
label_dict = {0:'Angry',1:'Disgust',2:'Fear',3:'Happy',4:'Neutral',5:'Sad',6:'Surprise'}
img = np.expand_dims(image,axis = 0) #makes image shape (1,48,48)
img = img.reshape(-1,48,48,1)
```

```
result = emotionsANN.predict(img)
result = list(result[0])
print(result)
img_index = result.index(max(result))
print(label_dict[img_index])
plt.show()
```

```
1/1 [==============================] - 0s 23ms/step
[0.14795981, 0.01544925, 0.14838952, 0.24726321, 0.16832735, 0.16295227, 0.10965856]
Happy
```

```
label_dict = {0:'Angry',1:'Disgust',2:'Fear',3:'Happy',4:'Neutral',5:'Sad',6:'Surprise'}
image2 = np.expand_dims(image2,axis = 0) #makes image shape (1,48,48)
image2 = image2.reshape(-1,48,48,1)
result2 = emotionsANN.predict(image2)
result2 = list(result2[0])
print(result2)
img_index = result2.index(max(result2))
print(label_dict[img_index])
plt.show()
```

```
1/1 [==============================] - 0s 34ms/step
[0.14795981, 0.01544925, 0.14838952, 0.24726321, 0.16832735, 0.16295227, 0.10965856]
Happy
```

```
img = np.expand_dims(image,axis = 0) #makes image shape (1,48,48)
img = image.reshape(-1,48,48,1)
result = emotionsANN.predict(img)
result = list(result[0])
print(result)
```

```
1/1 [==============================] - 0s 114ms/step
[0.14795981, 0.01544925, 0.14838952, 0.24726321, 0.16832735, 0.16295227, 0.10965856]
```

```
img_index = result.index(max(result))
```

Saved successfully!

```
Happy
```

```
train_loss, train_acc = emotionsANN.evaluate(train_generator)
test_loss, test_acc   = emotionsANN.evaluate(validation_generator)
```

```
1436/1436 [==============================] - 25s 17ms/step - loss: 1.8099 - accuracy: 0.2516
90/90 [==============================] - 1s 12ms/step - loss: 1.8140 - accuracy: 0.2472
```

✓ 0s completed at 3:03 PM