Assignment- 20 Computer Vision

Q.1 Write a program to check human sentimentals from expression.

In [1]:
```
!wget https://www.dropbox.com/s/nilt43hyl1dx82k/dataset.zip?dl=0
```

```
--2024-05-10 07:53:10--  https://www.dropbox.com/s/nilt43hyl1dx82k/dataset.zip?dl=0 (https://www.dropbox.com/s/nilt43hyl1dx82k/dataset.zip?dl=0)
Resolving www.dropbox.com (www.dropbox.com)... 162.125.8.18, 2620:100:6019:18::a27d:412
Connecting to www.dropbox.com (www.dropbox.com)|162.125.8.18|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: /s/raw/nilt43hyl1dx82k/dataset.zip [following]
--2024-05-10 07:53:10--  https://www.dropbox.com/s/raw/nilt43hyl1dx82k/dataset.zip (https://www.dropbox.com/s/raw/nilt43hyl1dx82k/dataset.zip)
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc5287a085141a9bc070991aaa05.dl.dropboxusercontent.com/cd/0/inline/CSluZ_bvxEm2HmhUyzU6l6yD1MR4jFp-7gtTyheixWbeSxQtoYymIRSVeZX5gYUp8-_1jt3260B8qm0Aq1JPACoqtB55dJd95jYJvDQA3WScRA32KJT_OWc3Tq6otLwOPaCFMtMxH4rIlUWNnfEGv2aI/file# (https://uc5287a085141a9bc070991aaa05.dl.dropboxusercontent.com/cd/0/inline/CSluZ_bvxEm2HmhUyzU6l6yD1MR4jFp-7gtTyheixWbeSxQtoYymIRSVeZX5gYUp8-_1jt3260B8qm0Aq1JPACoqtB55dJd95jYJvDQA3WScRA32KJT_OWc3Tq6otLwOPaCFMtMxH4rIlUWNnfEGv2aI/file#) [following]
--2024-05-10 07:53:10--  https://uc5287a085141a9bc070991aaa05.dl.dropboxusercontent.com/cd/0/inline/CSluZ_bvxEm2HmhUyzU6l6yD1MR4jFp-7gtTyheixWbeSxQtoYymIRSVeZX5gYUp8-_1jt3260B8qm0Aq1JPACoqtB55dJd95jYJvDQA3WScRA32KJT_OWc3Tq6otLwOPaCFMtMxH4rIlUWNnfEGv2aI/file (https://uc5287a085141a9bc070991aaa05.dl.dropboxusercontent.com/cd/0/inline/CSluZ_bvxEm2HmhUyzU6l6yD1MR4jFp-7gtTyheixWbeSxQtoYymIRSVeZX5gYUp8-_1jt3260B8qm0Aq1JPACoqtB55dJd95jYJvDQA3WScRA32KJT_OWc3Tq6otLwOPaCFMtMxH4rIlUWNnfEGv2aI/file)
Resolving uc5287a085141a9bc070991aaa05.dl.dropboxusercontent.com (uc5287a085141a9bc070991aaa05.dl.dropboxusercontent.com)... 162.125.8.15, 2620:100:6030:15::a27d:500f
Connecting to uc5287a085141a9bc070991aaa05.dl.dropboxusercontent.com (uc5287a085141a9bc070991aaa05.dl.dropboxusercontent.com)|162.125.8.15|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: /cd/0/inline2/CSlnLLjYGCDozuLVsdRFAunZafbPGeLFbAKuKcY8gHoZuRTxg-0WQHVf0ypEGkhJVutMEJ13S8pQUcS3pLzxXAnvekKfyEXdmMfXROA4CDcsFpCo1F-qD8QN0ntds0woyG07f3zGigq3YgLQ_dd8O303uLZjiaRat8l6aFdHTId0WSmUDoMeLqRnDgfndx6d1WB1AI-QAXnQvLj9-0ZqcCF3nNYs6F5IpgyFoQ3ZIzQuN_o_7Ufa7nk83xAB0r4QbPH5t-cdcmmRZS_hMlSaUEFFnSAEtEFDOSgKvrGsj72V8Wmqb_Njw1y4NgwaDBslzPR4UA7m8GI1dx9_3D1R04ScSgbSMa6aH2rWCdY0x19xJBo8p7-YrU7DSGWWpvnvbwo/file [following]
--2024-05-10 07:53:11--  https://uc5287a085141a9bc070991aaa05.dl.dropboxusercontent.com/cd/0/inline2/CSlnLLjYGCDozuLVsdRFAunZafbPGeLFbAKuKcY8gHoZuRTxg-0WQHVf0ypEGkhJVutMEJ13S8pQUcS3pLzxXAnvekKfyEXdmMfXROA4CDcsFpCo1F-qD8QN0ntds0woyG07f3zGigq3YgLQ_dd8O303uLZjiaRat8l6aFdHTId0WSmUDoMeLqRnDgfndx6d1WB1AI-QAXnQvLj9-0ZqcCF3nNYs6F5IpgyFoQ3ZIzQuN_o_7Ufa7nk83xAB0r4QbPH5t-cdcmmRZS_hMlSaUEFFnSAEtEFDOSgKvrGsj72V8Wmqb_Njw1y4NgwaDBslzPR4UA7m8GI1dx9_3D1R04ScSgbSMa6aH2rWCdY0x19xJBo8p7-YrU7DSGWWpvnvbwo/file (https://uc5287a085141a9bc070991aaa05.dl.dropboxusercontent.com/cd/0/inline2/CSlnLLjYGCDozuLVsdRFAunZafbPGeLFbAKuKcY8gHoZuRTxg-0WQHVf0ypEGkhJVutMEJ13S8pQUcS3pLzxXAnvekKfyEXdmMfXROA4CDcsFpCo1F-qD8QN0ntds0woyG07f3zGigq3YgLQ_dd8O303uLZjiaRat8l6aFdHTId0WSmUDoMeLqRnDgfndx6d1WB1AI-QAXnQvLj9-0ZqcCF3nNYs6F5IpgyFoQ3ZIzQuN_o_7Ufa7nk83xAB0r4QbPH5t-cdcmmRZS_hMlSaUEFFnSAEtEFDOSgKvrGsj72V8Wmqb_Njw1y4NgwaDBslzPR4UA7m8GI1dx9_3D1R04ScSgbSMa6aH2rWCdY0x19xJBo8p7-YrU7DSGWWpvnvbwo/file)
Reusing existing connection to uc5287a085141a9bc070991aaa05.dl.dropboxusercontent.com:443.
HTTP request sent, awaiting response... 200 OK
Length: 63252113 (60M) [application/zip]
Saving to: ‘dataset.zip?dl=0’

dataset.zip?dl=0    100%[===================>]  60.32M   142MB/s    in 0.4s

2024-05-10 07:53:12 (142 MB/s) - ‘dataset.zip?dl=0’ saved [63252113/63252113]
```

In [2]:
```
!unzip dataset.zip?dl=0
```

```
Streaming output truncated to the last 5000 lines.
  inflating: train/sad/Training_65242339.jpg
  inflating: train/sad/Training_65267116.jpg
  inflating: train/sad/Training_65275626.jpg
  inflating: train/sad/Training_6529266.jpg
  inflating: train/sad/Training_65329617.jpg
  inflating: train/sad/Training_65338712.jpg
  inflating: train/sad/Training_65338797.jpg
  inflating: train/sad/Training_65387162.jpg
  inflating: train/sad/Training_65404494.jpg
  inflating: train/sad/Training_65426218.jpg
  inflating: train/sad/Training_65430136.jpg
  inflating: train/sad/Training_65437377.jpg
  inflating: train/sad/Training_6545735.jpg
  inflating: train/sad/Training_65463385.jpg
  inflating: train/sad/Training_65473985.jpg
  inflating: train/sad/Training_65502829.jpg
  inflating: train/sad/Training_65505359.jpg
  inflating: train/sad/Training_65508578.jpg
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from keras.layers import Flatten, Dense
from keras.models import Model
from keras.preprocessing.image import ImageDataGenerator , img_to_array, load_img
from keras.applications.mobilenet import MobileNet, preprocess_input
from keras.losses import categorical_crossentropy
```

Building our Model to train the data

In [4]:
```
# Working with pre trained model

base_model = MobileNet( input_shape=(224,224,3), include_top= False )

for layer in base_model.layers:
  layer.trainable = False


x = Flatten()(base_model.output)
x = Dense(units=7 , activation='softmax' )(x)

# creating our model.
model = Model(base_model.input, x)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobilenet_1_0_224_tf_no_top.h5 (https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobilenet_1_0_224_tf_no_top.h5)
17225924/17225924 [==============================] - 0s 0us/step
```

In [5]:
```
model.compile(optimizer='adam', loss= categorical_crossentropy , metrics=['accuracy']  )
```

Preparing data using data generator

In [6]:
```
train_datagen = ImageDataGenerator(
     zoom_range = 0.2,
     shear_range = 0.2,
     horizontal_flip=True,
     rescale = 1./255
)

train_data = train_datagen.flow_from_directory(directory= "/content/train",
                                               target_size=(224,224),
                                               batch_size=32,
                                    )


train_data.class_indices
```

```
Found 28709 images belonging to 7 classes.
```

Out[6]:
```
{'angry': 0,
 'disgust': 1,
 'fear': 2,
 'happy': 3,
 'neutral': 4,
 'sad': 5,
 'surprise': 6}
```

In [7]:
```
val_datagen = ImageDataGenerator(rescale = 1./255 )

val_data = val_datagen.flow_from_directory(directory= "/content/test",
                                           target_size=(224,224),
                                           batch_size=32,
                                    )
```

```
Found 7178 images belonging to 7 classes.
```

Visualising the data fed

In [ ]:
```
```

In [8]:
```
# to visualize the images in the traing data denerator

t_img , label = train_data.next()

#-----------------------------------------------------------------------------
# function when called will prot the images
def plotImages(img_arr, label):
  """
  input  :- images array
  output :- plots the images
  """
  count = 0
  for im, l in zip(img_arr,label) :
    plt.imshow(im)
    plt.title(im.shape)
    plt.axis = False
    plt.show()

    count += 1
    if count == 10:
      break

#-----------------------------------------------------------------------------
# function call to plot the images
plotImages(t_img, label)
```

Early Stopping and Check points

In [9]:
```
## having early stopping and model check point

from keras.callbacks import ModelCheckpoint, EarlyStopping

# early stopping
es = EarlyStopping(monitor='val_accuracy', min_delta= 0.01 , patience= 5, verbose= 1, mode='auto')

# model check point
mc = ModelCheckpoint(filepath="best_model.h5", monitor= 'val_accuracy', verbose= 1, save_best_only= True, mode = 'auto')

# puting call back in a list
call_back = [es, mc]
```

In [10]:
```
hist = model.fit_generator(train_data,
                           steps_per_epoch= 10,
                           epochs= 30,
                           validation_data= val_data,
                           validation_steps= 8,
                           callbacks=[es,mc])
```

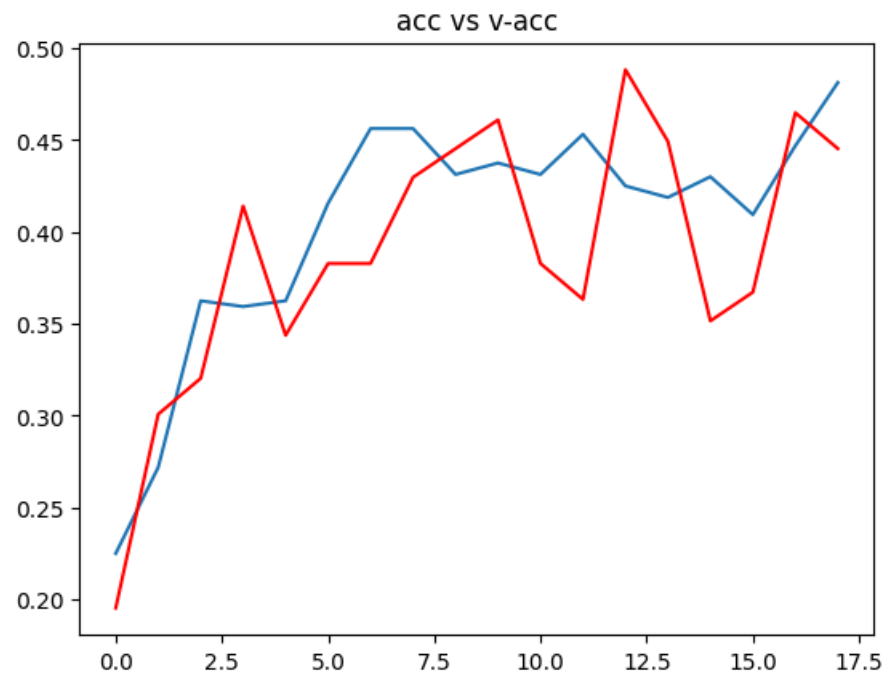
```
<ipython-input-10-7dbb6b2d1ed1>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Mod
el.fit`, which supports generators.
  hist = model.fit_generator(train_data,

Epoch 1/30
10/10 [==============================] - ETA: 0s - loss: 17.3073 - accuracy: 0.2250
Epoch 1: val_accuracy improved from -inf to 0.19531, saving model to best_model.h5

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `mode
l.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(

10/10 [==============================] - 10s 450ms/step - loss: 17.3073 - accuracy: 0.2250 - val_loss: 14.0523 - val_accuracy: 0.1953
Epoch 2/30
10/10 [==============================] - ETA: 0s - loss: 14.6760 - accuracy: 0.2719
Epoch 2: val_accuracy improved from 0.19531 to 0.30078, saving model to best_model.h5
10/10 [==============================] - 8s 896ms/step - loss: 14.6760 - accuracy: 0.2719 - val_loss: 9.5517 - val_accuracy: 0.3008
Epoch 3/30
10/10 [==============================] - ETA: 0s - loss: 9.4273 - accuracy: 0.3625
Epoch 3: val_accuracy improved from 0.30078 to 0.32031, saving model to best_model.h5
10/10 [==============================] - 4s 427ms/step - loss: 9.4273 - accuracy: 0.3625 - val_loss: 9.3336 - val_accuracy: 0.3203
Epoch 4/30
10/10 [==============================] - ETA: 0s - loss: 8.1260 - accuracy: 0.3594
Epoch 4: val_accuracy improved from 0.32031 to 0.41406, saving model to best_model.h5
10/10 [==============================] - 6s 574ms/step - loss: 8.1260 - accuracy: 0.3594 - val_loss: 5.8851 - val_accuracy: 0.4141
Epoch 5/30
10/10 [==============================] - ETA: 0s - loss: 7.4087 - accuracy: 0.3625
Epoch 5: val_accuracy did not improve from 0.41406
10/10 [==============================] - 4s 408ms/step - loss: 7.4087 - accuracy: 0.3625 - val_loss: 6.8454 - val_accuracy: 0.3438
Epoch 6/30
10/10 [==============================] - ETA: 0s - loss: 6.0202 - accuracy: 0.4156
Epoch 6: val_accuracy did not improve from 0.41406
10/10 [==============================] - 5s 430ms/step - loss: 6.0202 - accuracy: 0.4156 - val_loss: 6.7367 - val_accuracy: 0.3828
Epoch 7/30
10/10 [==============================] - ETA: 0s - loss: 5.7904 - accuracy: 0.4563
Epoch 7: val_accuracy did not improve from 0.41406
10/10 [==============================] - 4s 413ms/step - loss: 5.7904 - accuracy: 0.4563 - val_loss: 5.9508 - val_accuracy: 0.3828
Epoch 8/30
10/10 [==============================] - ETA: 0s - loss: 4.9099 - accuracy: 0.4563
Epoch 8: val_accuracy improved from 0.41406 to 0.42969, saving model to best_model.h5
10/10 [==============================] - 5s 566ms/step - loss: 4.9099 - accuracy: 0.4563 - val_loss: 5.0625 - val_accuracy: 0.4297
Epoch 9/30
10/10 [==============================] - ETA: 0s - loss: 5.3911 - accuracy: 0.4313
Epoch 9: val_accuracy improved from 0.42969 to 0.44531, saving model to best_model.h5
10/10 [==============================] - 4s 425ms/step - loss: 5.3911 - accuracy: 0.4313 - val_loss: 4.3755 - val_accuracy: 0.4453
Epoch 10/30
10/10 [==============================] - ETA: 0s - loss: 4.8029 - accuracy: 0.4375
Epoch 10: val_accuracy improved from 0.44531 to 0.46094, saving model to best_model.h5
10/10 [==============================] - 5s 528ms/step - loss: 4.8029 - accuracy: 0.4375 - val_loss: 5.0747 - val_accuracy: 0.4609
Epoch 11/30
10/10 [==============================] - ETA: 0s - loss: 5.0435 - accuracy: 0.4313
Epoch 11: val_accuracy did not improve from 0.46094
10/10 [==============================] - 4s 400ms/step - loss: 5.0435 - accuracy: 0.4313 - val_loss: 4.6358 - val_accuracy: 0.3828
Epoch 12/30
10/10 [==============================] - ETA: 0s - loss: 5.1884 - accuracy: 0.4531
Epoch 12: val_accuracy did not improve from 0.46094
10/10 [==============================] - 5s 494ms/step - loss: 5.1884 - accuracy: 0.4531 - val_loss: 5.3386 - val_accuracy: 0.3633
Epoch 13/30
10/10 [==============================] - ETA: 0s - loss: 5.6525 - accuracy: 0.4250
Epoch 13: val_accuracy improved from 0.46094 to 0.48828, saving model to best_model.h5
10/10 [==============================] - 5s 427ms/step - loss: 5.6525 - accuracy: 0.4250 - val_loss: 4.5562 - val_accuracy: 0.4883
Epoch 14/30
10/10 [==============================] - ETA: 0s - loss: 5.4321 - accuracy: 0.4187
Epoch 14: val_accuracy did not improve from 0.48828
10/10 [==============================] - 4s 409ms/step - loss: 5.4321 - accuracy: 0.4187 - val_loss: 6.0207 - val_accuracy: 0.4492
Epoch 15/30
10/10 [==============================] - ETA: 0s - loss: 6.2157 - accuracy: 0.4300
Epoch 15: val_accuracy did not improve from 0.48828
10/10 [==============================] - 5s 553ms/step - loss: 6.2157 - accuracy: 0.4300 - val_loss: 7.7263 - val_accuracy: 0.3516
Epoch 16/30
10/10 [==============================] - ETA: 0s - loss: 7.4774 - accuracy: 0.4094
Epoch 16: val_accuracy did not improve from 0.48828
10/10 [==============================] - 4s 411ms/step - loss: 7.4774 - accuracy: 0.4094 - val_loss: 8.1603 - val_accuracy: 0.3672
Epoch 17/30
10/10 [==============================] - ETA: 0s - loss: 5.7995 - accuracy: 0.4469
Epoch 17: val_accuracy did not improve from 0.48828
10/10 [==============================] - 5s 437ms/step - loss: 5.7995 - accuracy: 0.4469 - val_loss: 6.1236 - val_accuracy: 0.4648
Epoch 18/30
10/10 [==============================] - ETA: 0s - loss: 5.4460 - accuracy: 0.4812
Epoch 18: val_accuracy did not improve from 0.48828
10/10 [==============================] - 4s 400ms/step - loss: 5.4460 - accuracy: 0.4812 - val_loss: 5.6344 - val_accuracy: 0.4453
Epoch 18: early stopping
```

In [11]:
```
# Loading the best fit model
from keras.models import load_model
model = load_model("/content/best_model.h5")
```

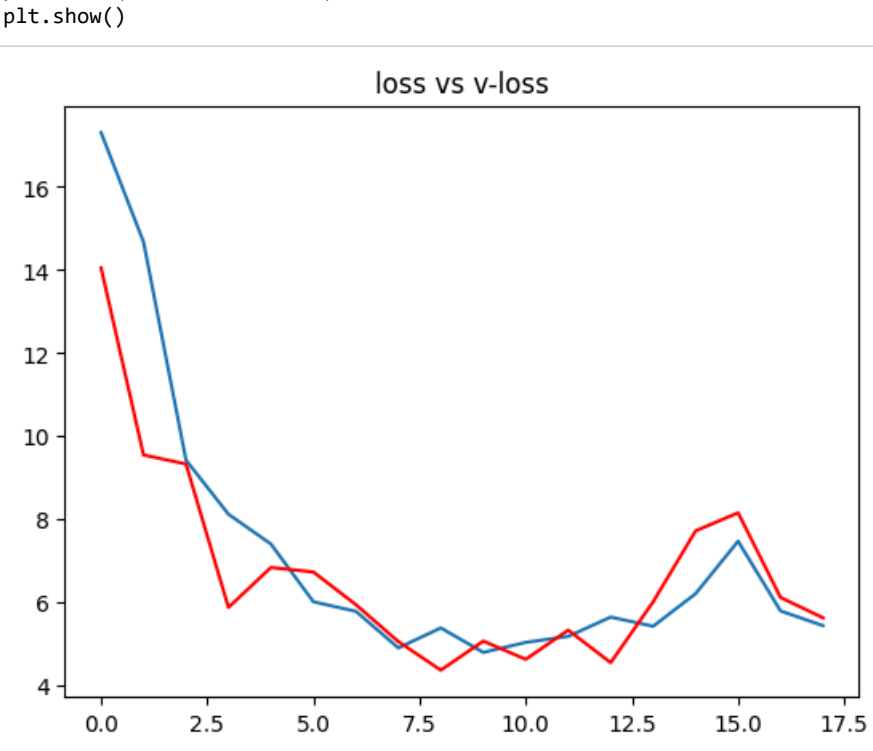
In [12]:
```
h =  hist.history
h.keys()
```

Out[12]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [13]:
```
plt.plot(h['accuracy'])
plt.plot(h['val_accuracy'] , c = "red")
plt.title("acc vs v-acc")
plt.show()
```



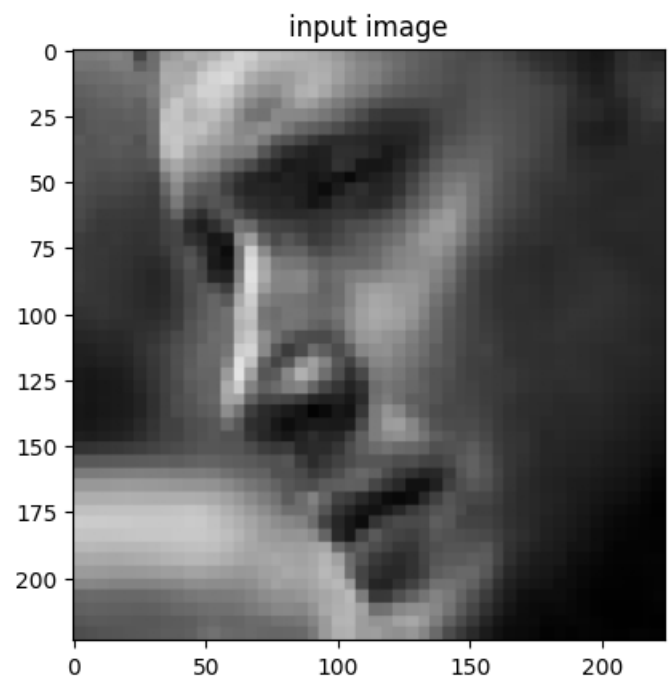
In [14]:
```
plt.plot(h['loss'])
plt.plot(h['val_loss'] , c = "red")
plt.title("loss vs v-loss")
plt.show()
```



In [15]:
```
# just to map o/p values
op = dict(zip( train_data.class_indices.values(), train_data.class_indices.keys()))
```

In [16]:
```
# path for the image to see if it predics correct class

path = "/content/test/angry/PrivateTest_1054527.jpg"
img = load_img(path, target_size=(224,224) )

i = img_to_array(img)/255
input_arr = np.array([i])
input_arr.shape

pred = np.argmax(model.predict(input_arr))

print(f" the image is of {op[pred]}")

# to display the image
plt.imshow(input_arr[0])
plt.title("input image")
plt.show()
```

```
1/1 [==============================] - 1s 1s/step
 the image is of sad
```



After testing extensively, it's clear that our model does a good job predicting emotions in images.

In [ ]:
```
Submitted by- Shweta Kanungo
```

In [ ]:
```

```