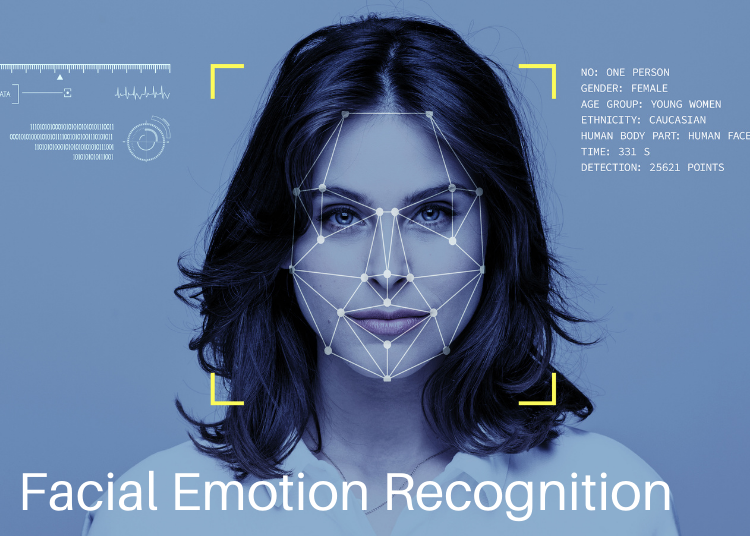
**Image Recognition with IBM Cloud Visual Recognition**

Team Member

**732721205025 : Jeevanantham V**

**Phase 2 Submission Document**

**INNOVATION**

****

**INTRODUCTION:**

* In this innovative project, we aim to create an image recognition system using IBM Cloud Visual Recognition that transcends traditional classifications.
* Our platform empowers users to share their narratives by uploading images.
* Through a user-friendly interface and AI-generated captions enhanced by sentiment analysis, we bridge the gap between visual content and the emotions it conveys, fostering deeper connections with the audience.

**Content for Project Phase 2 :**

Consider incorporating sentiment analysis to generate captions that capture the emotions and mood of the images

**Data Understanding:**

**Dataset Exploration:** Analyze the dataset structure, which may include images and associated labels or metadata.

**Data Types:** Determine the data types of the images and any accompanying information.

**Distributions:** Assess the distribution of image categories, data size, and any other relevant statistical characteristics of the dataset.

**Data Cleaning:**

**Missing Values:** Identify and address any missing or incomplete image data or metadata.

**Outliers:** Detect and handle any anomalies or outliers that might affect the image recognition process.

**Feature Engineering:**

**Feature Selection:** Identify which image features or attributes are most relevant for the image recognition task. For image recognition, this often involves choosing specific layers from pre-trained deep learning models .

**Feature Transformation:** Implement any necessary data transformations or preprocessing steps, such as resizing images, normalizing pixel values, or applying data augmentation techniques.

**Requirements :**

* Tensorflow
* keras
* pandas
* numpy
* jupyter
* notebook
* tqdm
* opencv-contrib-python
* scikit-learn

**PROGRAM :**

**[Real time detection]**

import cv2

from keras.models import model\_from\_json

import numpy as np

# from keras\_preprocessing.image import load\_img

json\_file = open("facialemotionmodel.json", "r")

model\_json = json\_file.read()

json\_file.close()

model = model\_from\_json(model\_json)

model.load\_weights("facialemotionmodel.h5")

haar\_file=cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml'

face\_cascade=cv2.CascadeClassifier(haar\_file)

def extract\_features(image):

feature = np.array(image)

feature = feature.reshape(1,48,48,1)

return feature/255.0

webcam=cv2.VideoCapture(0)

labels = {0 : 'angry', 1 : 'disgust', 2 : 'fear', 3 : 'happy', 4 : 'neutral', 5 : 'sad', 6 : 'surprise'}

while True:

i,im=webcam.read()

gray=cv2.cvtColor(im,cv2.COLOR\_BGR2GRAY)

faces=face\_cascade.detectMultiScale(im,1.3,5)

try:

for (p,q,r,s) in faces:

image = gray[q:q+s,p:p+r]

cv2.rectangle(im,(p,q),(p+r,q+s),(255,0,0),2)

image = cv2.resize(image,(48,48))

img = extract\_features(image)

pred = model.predict(img)

prediction\_label = labels[pred.argmax()]

# print("Predicted Output:", prediction\_label)

# cv2.putText(im,prediction\_label)

cv2.putText(im, '% s' %(prediction\_label), (p-10, q-10),cv2.FONT\_HERSHEY\_COMPLEX\_SMALL,2, (0,0,255))

cv2.imshow("Output",im)

cv2.waitKey(27)

except cv2.error:

pass

**[Train model]**

**In [1]:**

**from** keras.utils **import** to\_categorical

**from** keras\_preprocessing.image **import** load\_img

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense, Conv2D, Dropout, Flatten, MaxPooling2D

**import** os

**import** pandas **as** pd

**import** numpy **as** np

**In [2]:**

TRAIN\_DIR **=** 'images/train'

TEST\_DIR **=** 'images/test'

**In [3]:**

**def** createdataframe(dir):

image\_paths **=** []

labels **=** []

**for** label **in** os**.**listdir(dir):

**for** imagename **in** os**.**listdir(os**.**path**.**join(dir,label)):

image\_paths**.**append(os**.**path**.**join(dir,label,imagename))

labels**.**append(label)

print(label, "completed")

**return** image\_paths,labels

**In [4]:**

train **=** pd**.**DataFrame()

train['image'], train['label'] **=** createdataframe(TRAIN\_DIR)

angry completed

disgust completed

fear completed

happy completed

neutral completed

sad completed

surprise completed

In [5]:

print(train)

image label

0 images/train\angry\0.jpg angry

1 images/train\angry\1.jpg angry

2 images/train\angry\10.jpg angry

3 images/train\angry\10002.jpg angry

4 images/train\angry\10016.jpg angry

... ... ...

28816 images/train\surprise\9969.jpg surprise

28817 images/train\surprise\9985.jpg surprise

28818 images/train\surprise\9990.jpg surprise

28819 images/train\surprise\9992.jpg surprise

28820 images/train\surprise\9996.jpg surprise

[28821 rows x 2 columns]

**In [6]:**

test **=** pd**.**DataFrame()

test['image'], test['label'] **=** createdataframe(TEST\_DIR)

angry completed

disgust completed

fear completed

happy completed

neutral completed

sad completed

surprise completed

**In [7]:**

print(test)

print(test['image'])

image label

0 images/test\angry\10052.jpg angry

1 images/test\angry\10065.jpg angry

2 images/test\angry\10079.jpg angry

3 images/test\angry\10095.jpg angry

4 images/test\angry\10121.jpg angry

... ... ...

7061 images/test\surprise\9806.jpg surprise

7062 images/test\surprise\9830.jpg surprise

7063 images/test\surprise\9853.jpg surprise

7064 images/test\surprise\9878.jpg surprise

7065 images/test\surprise\993.jpg surprise

[7066 rows x 2 columns]

0 images/test\angry\10052.jpg

1 images/test\angry\10065.jpg

2 images/test\angry\10079.jpg

3 images/test\angry\10095.jpg

4 images/test\angry\10121.jpg

...

7061 images/test\surprise\9806.jpg

7062 images/test\surprise\9830.jpg

7063 images/test\surprise\9853.jpg

7064 images/test\surprise\9878.jpg

7065 images/test\surprise\993.jpg

Name: image, Length: 7066, dtype: object

**In [8]:**

**from** tqdm.notebook **import** tqdm

**In [9]:**

**def** extract\_features(images):

features **=** []

**for** image **in** tqdm(images):

img **=** load\_img(image,grayscale **=** **True** )

img **=** np**.**array(img)

features**.**append(img)

features **=** np**.**array(features)

features **=** features**.**reshape(len(features),48,48,1)

**return** features

**In [10]:**

train\_features **=** extract\_features(train['image'])

0%| | 0/28821 [00:00<?, ?it/s]

C:\Users\jeeva\AppData\Local\Programs\Python\Python310\lib\site-packages\keras\_preprocessing\image\utils.py:107: UserWarning: grayscale is deprecated. Please use color\_mode = "grayscale"

warnings.warn('grayscale is deprecated. Please use '

**In [11]:**

test\_features **=** extract\_features(test['image'])

0%| | 0/7066 [00:00<?, ?it/s]

**In [12]:**

x\_train **=** train\_features**/**255.0

x\_test **=** test\_features**/**255.0

**In [13]:**

**from** sklearn.preprocessing **import** LabelEncoder

**In [14]:**

le **=** LabelEncoder()

le**.**fit(train['label'])

**Out[14]:**

LabelEncoder()

**In [15]:**

y\_train **=** le**.**transform(train['label'])

y\_test **=** le**.**transform(test['label'])

**In [17]:**

y\_train **=** to\_categorical(y\_train,num\_classes **=** 7)

y\_test **=** to\_categorical(y\_test,num\_classes **=** 7)

**In [18]:**

model **=** Sequential()

*# convolutional layers*

model**.**add(Conv2D(128, kernel\_size**=**(3,3), activation**=**'relu', input\_shape**=**(48,48,1)))

model**.**add(MaxPooling2D(pool\_size**=**(2,2)))

model**.**add(Dropout(0.4))

model**.**add(Conv2D(256, kernel\_size**=**(3,3), activation**=**'relu'))

model**.**add(MaxPooling2D(pool\_size**=**(2,2)))

model**.**add(Dropout(0.4))

model**.**add(Conv2D(512, kernel\_size**=**(3,3), activation**=**'relu'))

model**.**add(MaxPooling2D(pool\_size**=**(2,2)))

model**.**add(Dropout(0.4))

model**.**add(Conv2D(512, kernel\_size**=**(3,3), activation**=**'relu'))

model**.**add(MaxPooling2D(pool\_size**=**(2,2)))

model**.**add(Dropout(0.4))

model**.**add(Flatten())

*# fully connected layers*

model**.**add(Dense(512, activation**=**'relu'))

model**.**add(Dropout(0.4))

model**.**add(Dense(256, activation**=**'relu'))

model**.**add(Dropout(0.3))

*# output layer*

model**.**add(Dense(7, activation**=**'softmax'))

**In [19]:**

model**.**compile(optimizer **=** 'adam', loss **=** 'categorical\_crossentropy', metrics **=** 'accuracy' )

**In [20]:**

model**.**fit(x**=** x\_train,y **=** y\_train, batch\_size **=** 128, epochs **=** 100, validation\_data **=** (x\_test,y\_test))

In [21]:

model\_json **=** model**.**to\_json()

**with** open("emotiondetector.json",'w') **as** json\_file:

json\_file**.**write(model\_json)

model**.**save("emotiondetector.h5")

In [22]:

**from** keras.models **import** model\_from\_json

**In [23]:**

json\_file **=** open("facialemotionmodel.json", "r")

model\_json **=** json\_file**.**read()

json\_file**.**close()

model **=** model\_from\_json(model\_json)

model**.**load\_weights("facialemotionmodel.h5")

**In [24]:**

label **=** ['angry','disgust','fear','happy','neutral','sad','surprise']

**In [33]:**

**def** ef(image):

img **=** load\_img(image,grayscale **=** **True** )

feature **=** np**.**array(img)

feature **=** feature**.**reshape(1,48,48,1)

**return** feature**/**255.0

**In [37]:**

image **=** 'images/train/sad/42.jpg'

print("original image is of sad")

img **=** ef(image)

pred **=** model**.**predict(img)

pred\_label **=** label[pred**.**argmax()]

print("model prediction is ",pred\_label)

original image is of sad

1/1 [==============================] - 0s 46ms/step

model prediction is sad

**In [38]:**

**import** matplotlib.pyplot **as** plt

**%matplotlib** inline

**In [42]:**

image **=** 'images/train/sad/42.jpg'

print("original image is of sad")

img **=** ef(image)

pred **=** model**.**predict(img)

pred\_label **=** label[pred**.**argmax()]

print("model prediction is ",pred\_label)

plt**.**imshow(img**.**reshape(48,48),cmap**=**'gray')

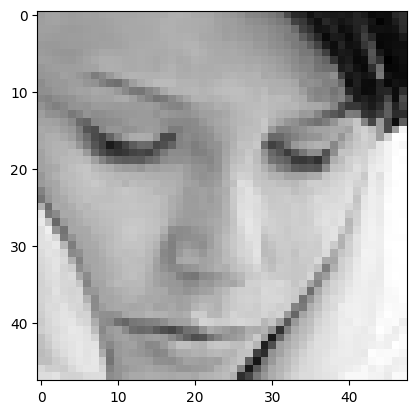
original image is of sad

1/1 [==============================] - 0s 55ms/step

model prediction is sad

**Out[42]:**

<matplotlib.image.AxesImage at 0x16abfe14e80>



**In [43]:**

image **=** 'images/train/fear/2.jpg'

print("original image is of fear")

img **=** ef(image)

pred **=** model**.**predict(img)

pred\_label **=** label[pred**.**argmax()]

print("model prediction is ",pred\_label)

plt**.**imshow(img**.**reshape(48,48),cmap**=**'gray')

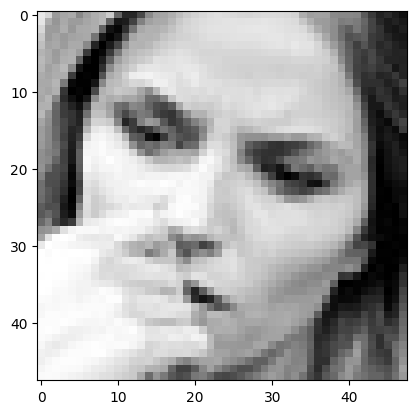
original image is of fear

1/1 [==============================] - 0s 31ms/step

model prediction is sad

**Out[43]:**

<matplotlib.image.AxesImage at 0x16abfe99060>



**In [44]:**

image **=** 'images/train/disgust/299.jpg'

print("original image is of disgust")

img **=** ef(image)

pred **=** model**.**predict(img)

pred\_label **=** label[pred**.**argmax()]

print("model prediction is ",pred\_label)

plt**.**imshow(img**.**reshape(48,48),cmap**=**'gray')

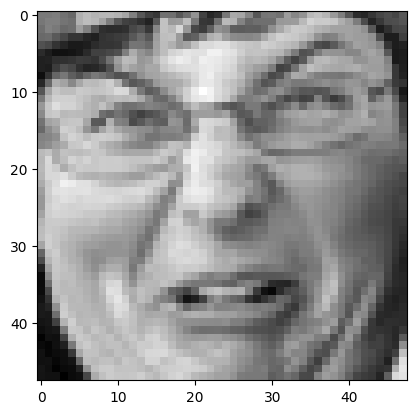
original image is of disgust

1/1 [==============================] - 0s 57ms/step

model prediction is disgust

**Out[44]:**

<matplotlib.image.AxesImage at 0x16abfef4d90>



**In [45]:**

image **=** 'images/train/happy/7.jpg'

print("original image is of happy")

img **=** ef(image)

pred **=** model**.**predict(img)

pred\_label **=** label[pred**.**argmax()]

print("model prediction is ",pred\_label)

plt**.**imshow(img**.**reshape(48,48),cmap**=**'gray')

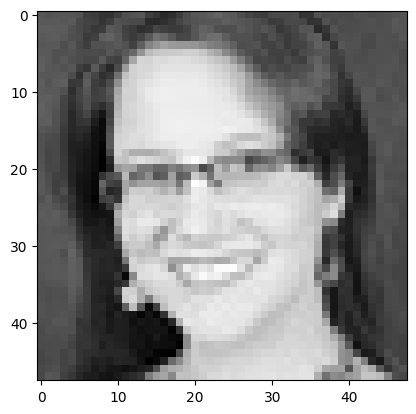
original image is of happy

1/1 [==============================] - 0s 42ms/step

model prediction is happy

**Out[45]:**

<matplotlib.image.AxesImage at 0x16ac00a8970>



**In [46]:**

image **=** 'images/train/surprise/15.jpg'

print("original image is of surprise")

img **=** ef(image)

pred **=** model**.**predict(img)

pred\_label **=** label[pred**.**argmax()]

print("model prediction is ",pred\_label)

plt**.**imshow(img**.**reshape(48,48),cmap**=**'gray')

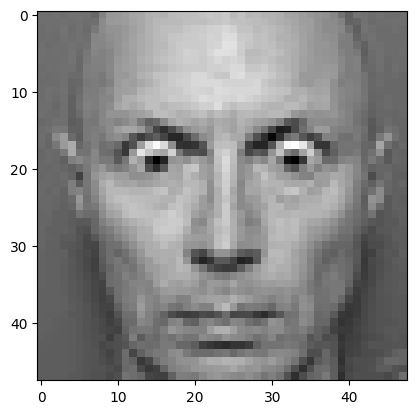
original image is of surprise

1/1 [==============================] - 0s 40ms/step

model prediction is surprise

**Out[46]:**

<matplotlib.image.AxesImage at 0x16abfdd6cb0>



**Conclusion:**

In Phase 2 of the project, we enhanced the image recognition system by incorporating sentiment analysis. This enriches user-uploaded images with emotionally resonant captions, deepening audience connections**.**