Experiment:1.2

**Aim:** Write a program to implement various feature extraction techniques like CNN, LBP and SIFT for image classification.

**Software Required:** Any Python IDE (e.g.: PyCharm, Jupyter Notebook, GoogleCollab)

**Technique used:** Various feature extraction methods are often employed in picture classification problems. These methods seek to extract pertinent data from photos and turn it into meaningful representations that machine learning algorithms may use to classify images. Feature extraction methods include:

**1. SIFT (Scale-Invariant Feature Transform):** SIFT identifies robust keypoints invariant to scale, rotation, and illumination. It computes orientation-embedded descriptors for these keypoints, enabling object recognition, image stitching, and matching tasks.

**2. CNN (Convolutional Neural Network):** CNNs automatically learn hierarchical features from images. They consist of convolutional layers for feature extraction and fully connected layers for classification. CNNs have revolutionized image classification, object detection, and image segmentation.

**3. LBP (Local Binary Pattern):** LBP encodes local texture patterns by comparing pixel intensities in small neighborhoods. It creates histograms of these patterns, making it effective for texture-based image analysis. LBP is used in texture classification and facial recognition.

**SIFT Features:**

**Steps:**

1. Import the necessary libraries for SIFT feature extraction.
2. Load or read the image you want to analyze.
3. Resize the image if needed and print the shape of the resized image.
4. Apply the SIFT algorithm to detect and extract SIFT features.
5. Visualize the detected SIFT keypoints on the image.
6. Print the dimensions of the SIFT feature vectors for further analysis.

**Implementation**:

# SIFT implementation

#import libraries

import cv2

from google.colab.patches import cv2\_imshow

# reading image

image = cv2.imread('/content/drive/MyDrive/Colab Notebooks/Images/Shah Rukh Khan.webp')

# converting image to grayscale

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Create a SIFT feature detector object

sift = cv2.xfeatures2d.SIFT\_create()

# detect features from the images

keypoints, descriptors = sift.detectAndCompute(image,None)

# keypoints: A list of detected keypoints in the image

# descriptors: The descriptors corresponding to the keypoints

# drawing the keypoints on the image to visualize them

image\_with\_keypoints = cv2.drawKeypoints(image, keypoints, image)

# Display the image with keypoints

cv2\_imshow(image\_with\_keypoints)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Output Screenshots:**



**CNN Features:**

**Steps:**

1. Import the necessary libraries, including Keras and OpenCV.
2. Define a function, e.g., extract\_cnn\_features(image), to perform feature extraction.
3. Load a pre-trained CNN model (e.g., VGG16) with weights pre-trained on ImageNet.
4. Preprocess the input image, including converting it to RGB if needed, resizing it to the model's input size (e.g., 224x224 pixels), and applying preprocessing steps like mean subtraction.
5. Pass the preprocessed image through the pre-trained model's feature extraction layers to obtain a feature vector.
6. Return the extracted CNN features as the output of the function

**Implementation**:

import tensorflow as tf

from tensorflow.keras.applications import VGG16

from tensorflow.keras.applications.vgg16 import preprocess\_input

from tensorflow.keras.models import Model

#import libraries

import cv2

from google.colab.patches import cv2\_imshow

# reading image

image = cv2.imread('/content/drive/MyDrive/Colab Notebooks/Images/Shah Rukh Khan.webp')

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Load a pre-trained CNN model (VGG16 in this case)

base\_model = VGG16(weights='imagenet', include\_top=False)

model = Model(inputs=base\_model.input, outputs=base\_model.layers[-1].output)

# Preprocess the image and extract features

image1 = cv2.cvtColor(image, cv2.COLOR\_GRAY2RGB)

img = cv2.resize(image1, (224, 224))

img = preprocess\_input(img)

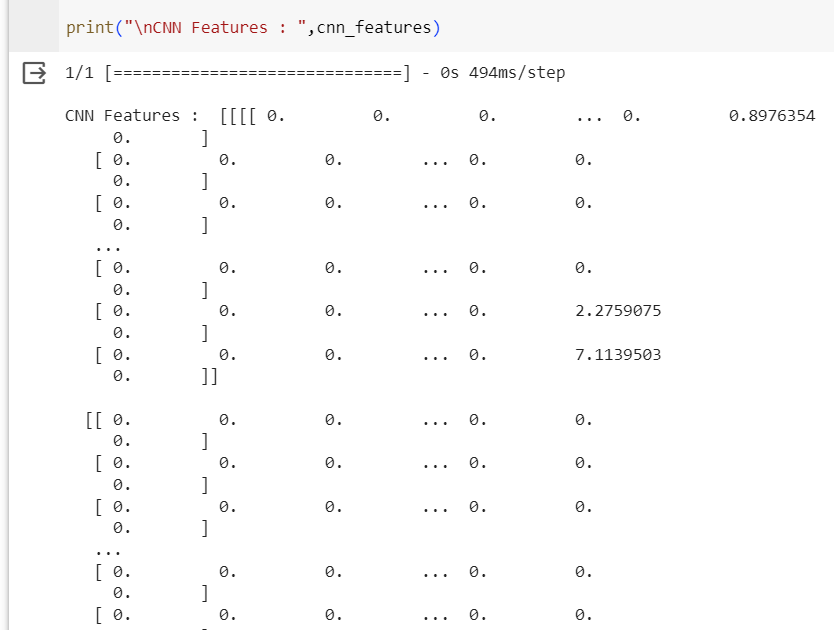
img = np.expand\_dims(img, axis=0)

features = model.predict(img)

cnn\_features = features

print("\nCNN Features : ",cnn\_features)

**Output Screenshots:**



**LBP Features:**

**Steps:**

1. Import the necessary libraries, including OpenCV.

2. Define a function extract\_lbp\_features(image), to perform LBP feature extraction.

3. Load or read the input image.

4. Convert the image to grayscale

5. Apply the LBP operator to the grayscale image, computing LBP values for each pixel by comparing pixel intensities with their neighbors

6. Create a histogram of LBP values to represent the texture pattern of the image region.

7. Return the LBP histogram as the output of the function, representing the LBP features of the image.

**Implementation**:

from google.colab.patches import cv2\_imshow

import cv2

import numpy as np

from skimage.feature import local\_binary\_pattern

# reading image

image = cv2.imread('/content/drive/MyDrive/Colab Notebooks/Images/Shah Rukh Khan.webp')

# converting image to grayscale

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

lbp\_radius = 3

lbp\_points = 24

lbp\_image = local\_binary\_pattern(image, lbp\_points, lbp\_radius, method='uniform')

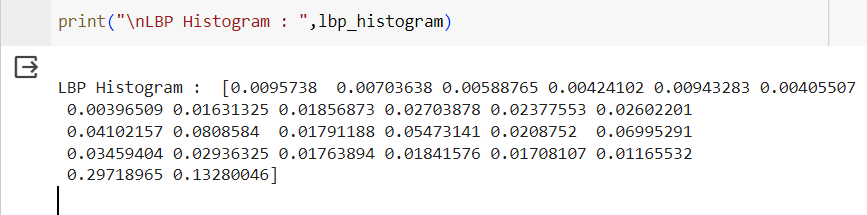
lbp\_histogram, \_ = np.histogram(lbp\_image.ravel(), bins=np.arange(0, lbp\_points + 3), range=(0, lbp\_points + 2))

lbp\_histogram = lbp\_histogram.astype("float")

lbp\_histogram /= (lbp\_histogram.sum() + 1e-6)

print("\nLBP Histogram : ",lbp\_histogram)

**Output Screenshots:**

****