|  |  |
| --- | --- |
| Student Name | Sahil Kaundal |
| UID | 21BCS8197 |
| Section & Group | 20BCS-09 & Group-A |
| Department | Computer Science & Engineering |
| Session | Aug-Dec 2023 |
| Course Name | Computer Vision Lab |
| Course Code | 20CSP-422 |
| Semester | 7th |

PRACTICAL FILE

**INDEX**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ex. No** | **List of Experiments** | **Conduct (MM: 12)** | **Viva**  **(MM: 10)** | **Record (MM: 8)** | **Total**  **(MM: 30)** | **Remarks/Signature** |
| 1 | Write a program to implement various feature extraction techniques for image classification. |  |  |  |  |  |
| 2 | Write a program to assess various feature matching algorithms for object recognition. |  |  |  |  |  |
| 3 | Write a program to analyze the impact of refining feature detection for image segmentation. |  |  |  |  |  |
| 4 | Write a program to evaluate the efficacy of human-guided control point selection for image alignment. |  |  |  |  |  |
| 5 | Write a program to compare the performance of different classification models in image recognition. |  |  |  |  |  |
| 6 | Write a program to interpret the effectiveness of Bag of Features in enhancing image classification performance. |  |  |  |  |  |
| 7 | Write a program to analyze various object detection algorithms with machine learning. |  |  |  |  |  |
| 8 | Write a program to determine the effectiveness of incorporating optical flow analysis into object tracking algorithms. |  |  |  |  |  |
| 9 | Write a program to examine the performance of various pretrained deep learning models for real-time object tracking tasks. |  |  |  |  |  |
| 10 | Write a program to interpret the effectiveness of template matching techniques for video stabilization tasks. |  |  |  |  |  |

**Experiment 1.1**

**Aim:** Write a program to implement various feature extraction techniques for image classification.

**Software Required:** Matlab, Google Colab

**Description:** Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set.

There are several feature extraction techniques commonly used in image classification tasks. These techniques aim to capture relevant information from images and transform them into meaningful representations that can be used by machine learning algorithms for classification. Some popular feature extraction techniques are:

Scale-Invariant Feature Transform (SIFT): SIFT is a widely used technique that identifies key points and extracts local invariant descriptors from images. It is robust to changes in scale, rotation, and illumination.

Oriented FAST and Rotated BRIEF(ORB):orb is a feature detection and description algorithm designed for efficiency, often used in real-time applications. It identifies key points using the FAST algorithm, computes rotation-invariant descriptors with binary patterns, and is well-suited for tasks like object tracking and robotics. While ORB is faster, its descriptors may be less distinctive compared to methods like SIFT or SURF.

Histogram of Oriented Gradients (HOG): HOG computes the distribution of gradient orientations in an image. It captures the shape and edge information and has been particularly successful in object detection and pedestrian recognition tasks.

Local Binary Patterns (LBP): LBP encodes the texture information by comparing each pixel's intensity value with its neighboring pixels. It is commonly used in texture analysis tasks and has shown good performance in various image classification applications.

**Code:**

# Step 1: Import necessary libraries

import cv2

from skimage.feature import hog

from skimage import io, color

import matplotlib.pyplot as plt

import numpy as np

# Load an example image

image\_path = '/content/CompCars-1024x567.jpg'

image = io.imread(image\_path)

# Display the original image

plt.figure(figsize=(6, 6))

plt.imshow(image)

plt.title('Original Image')

plt.axis('off')

plt.show()

# Convert the image to grayscale

gray\_image = color.rgb2gray(image)

# Calculate HOG features

hog\_features, hog\_image = hog(gray\_image, visualize=True, block\_norm='L2-Hys')

# Display HOG features

plt.figure(figsize=(6, 6))

plt.imshow(hog\_image, cmap=plt.cm.gray)

plt.title('HOG Features')

plt.axis('off')

plt.show()

# Calculate color histogram features

color\_hist\_features = []

for channel in range(3):

hist, \_ = np.histogram(image[:, :, channel], bins=256, range=(0, 256))

color\_hist\_features.extend(hist)

# Display color histogram features

plt.figure(figsize=(10, 6))

plt.bar(range(len(color\_hist\_features)), color\_hist\_features)

plt.title('Color Histogram Features')

plt.xlabel('Bin')

plt.ylabel('Frequency')

plt.show()

import cv2

import numpy as np

from google.colab.patches import cv2\_imshow

# Load an image using OpenCV

image\_path = '/content/CompCars-1024x567.jpg'

image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

# Create a SIFT object

sift = cv2.SIFT\_create()

# Detect and compute SIFT keypoints and descriptors

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

# Visualize keypoints on the image

image\_with\_keypoints = cv2.drawKeypoints(image, keypoints, None)

# Display the image with keypoints

cv2\_imshow(image\_with\_keypoints)

import cv2

# Load an image using OpenCV

image\_path = '/content/CompCars-1024x567.jpg'

image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

# Create an ORB object

orb = cv2.ORB\_create()

# Detect and compute ORB keypoints and descriptors

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

# Visualize keypoints on the image

image\_with\_keypoints = cv2.drawKeypoints(image, keypoints, None, flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)

# Display the image with keypoints using cv2\_imshow for Google Colab

# For other environments, you can use cv2.imshow

try:

from google.colab.patches import cv2\_imshow

cv2\_imshow(image\_with\_keypoints)

except ImportError:

cv2.imshow('Image with Keypoints', image\_with\_keypoints)

cv2.waitKey(0)

cv2.destroyAllWindows()

import cv2

import numpy as np

from skimage.feature import local\_binary\_pattern

from skimage import io

import matplotlib.pyplot as plt

# Load an example image

image\_path = '/content/CompCars-1024x567.jpg'

image = io.imread(image\_path, as\_gray=True)

# Define LBP parameters

radius = 1

n\_points = 8 \* radius

# Compute LBP image

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

# Calculate a histogram of the LBP image

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

hist = hist.astype("float")

hist /= (hist.sum() + 1e-8)

# Display original image and LBP image

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(image, cmap="gray")

plt.title("Original Image")

plt.subplot(1, 2, 2)

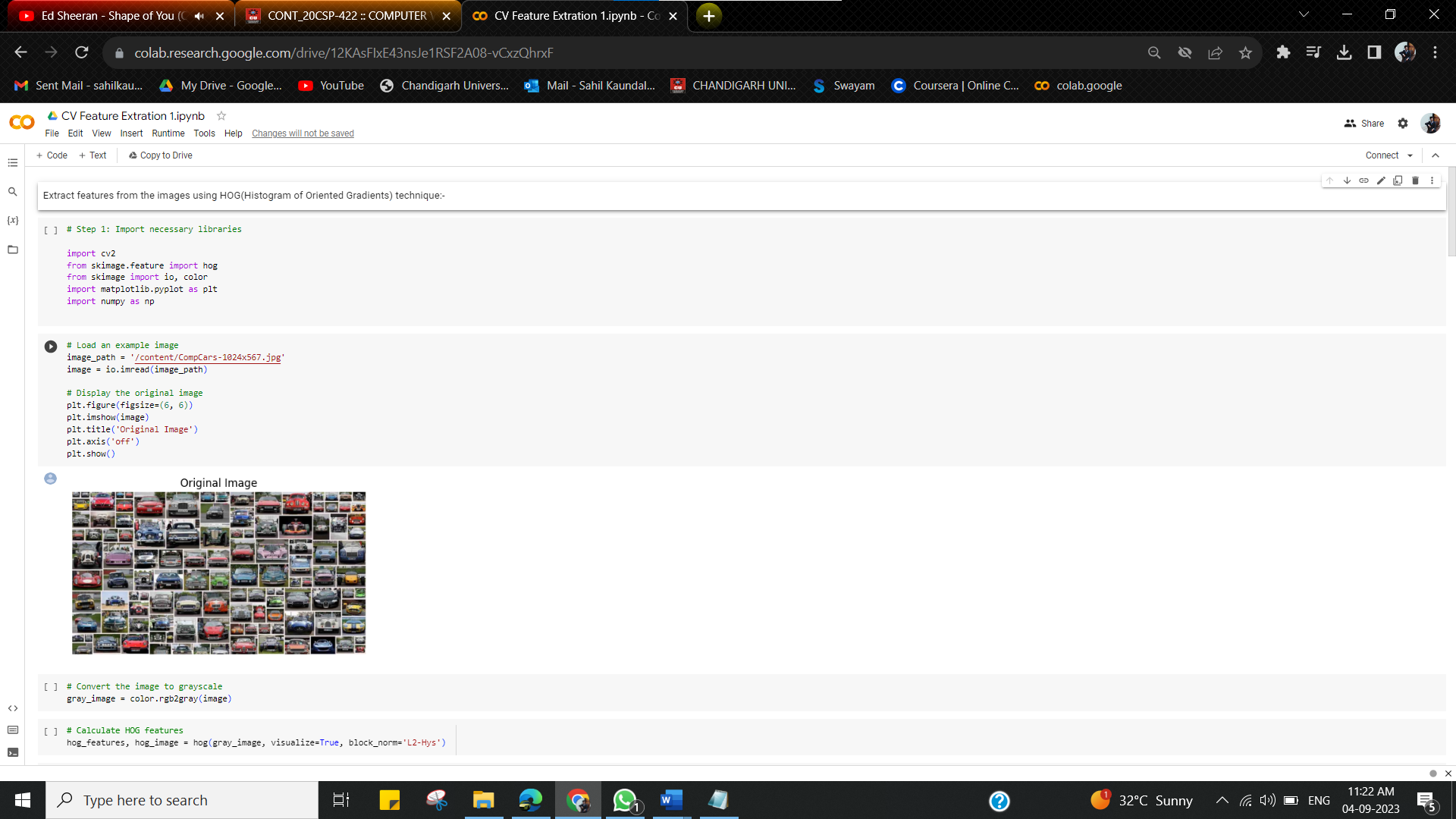
plt.imshow(lbp\_image, cmap="gray")

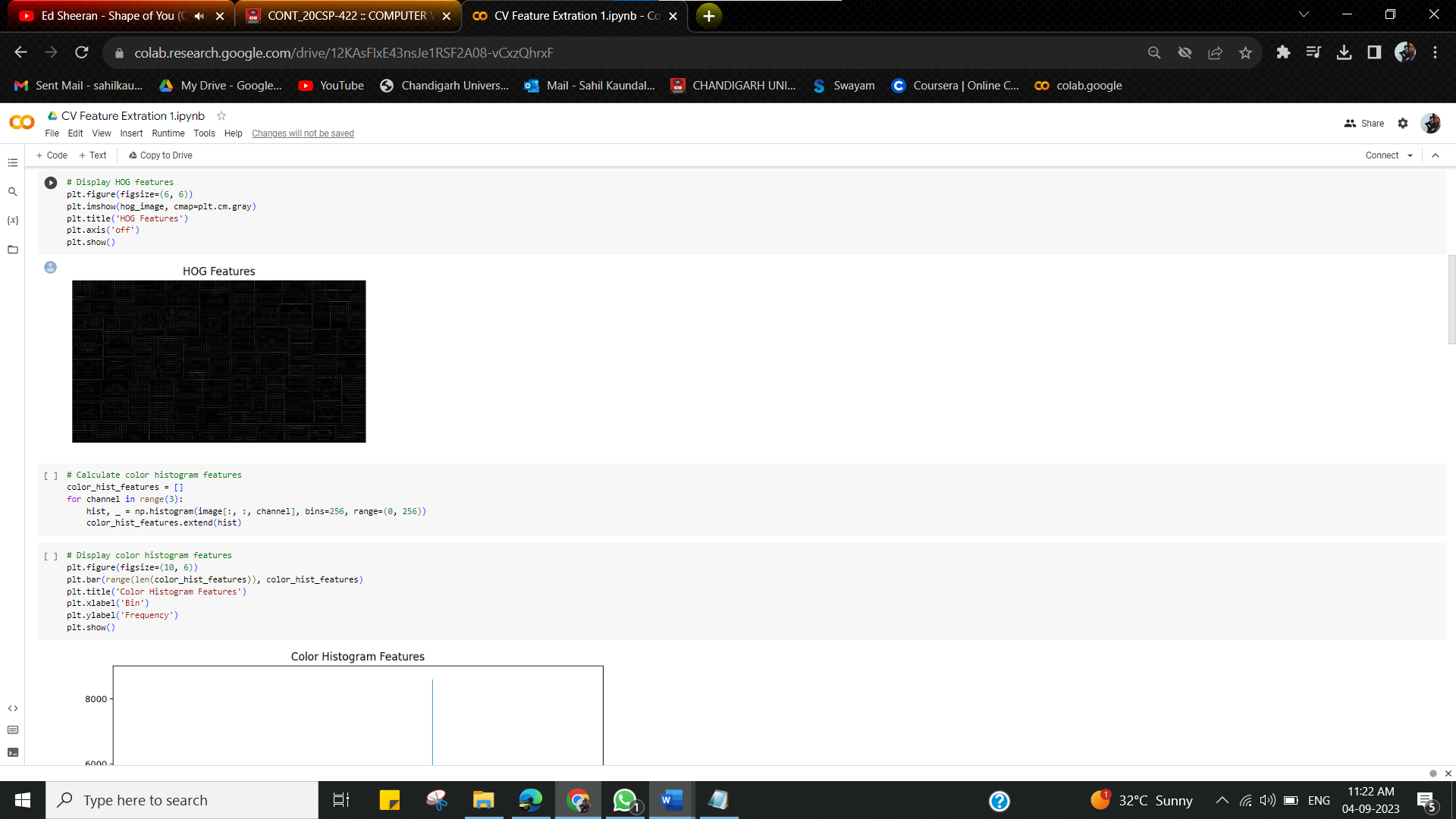
plt.title("LBP Image")

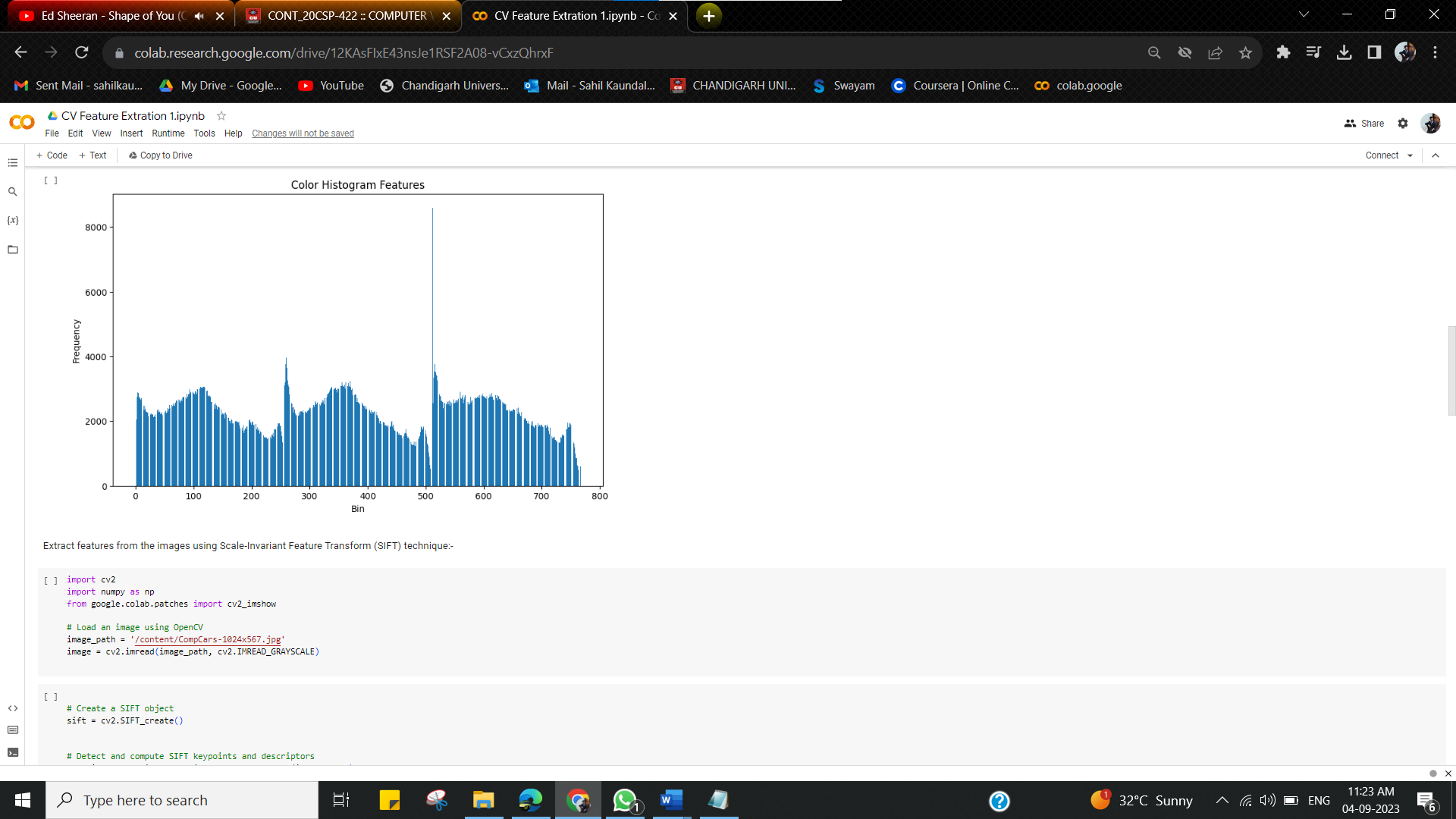
plt.tight\_layout()

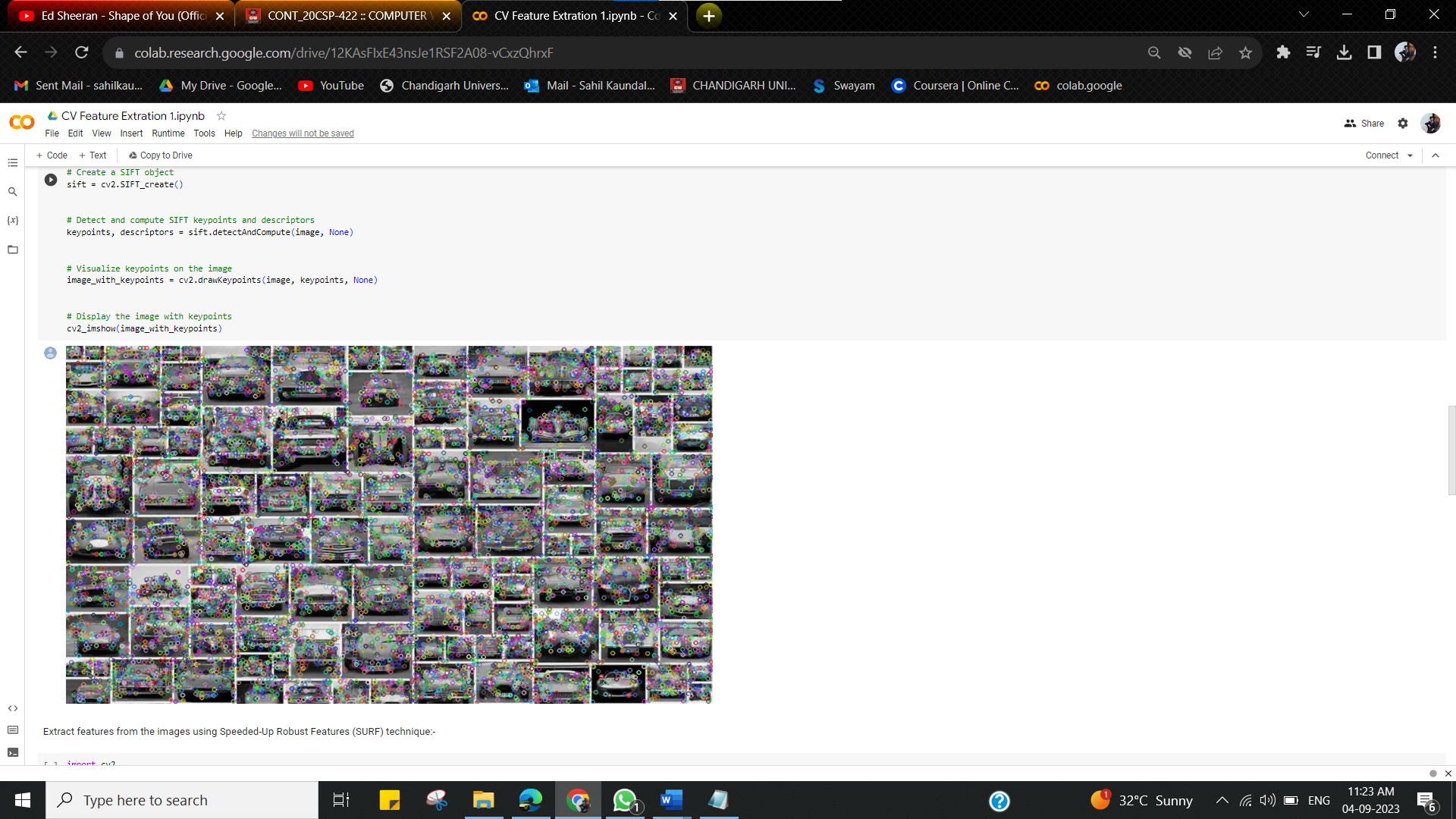
plt.show()

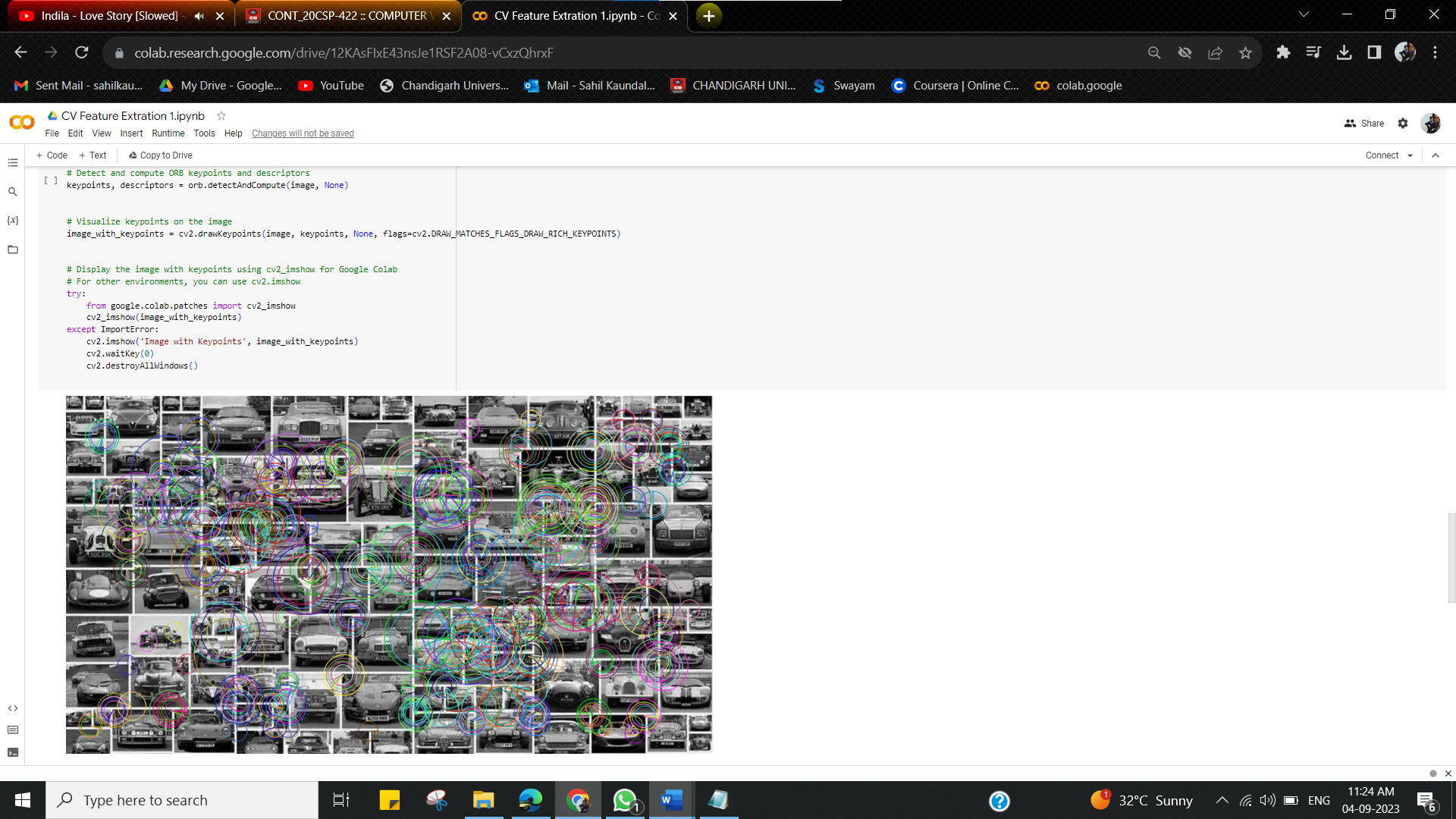
**Implementation:**

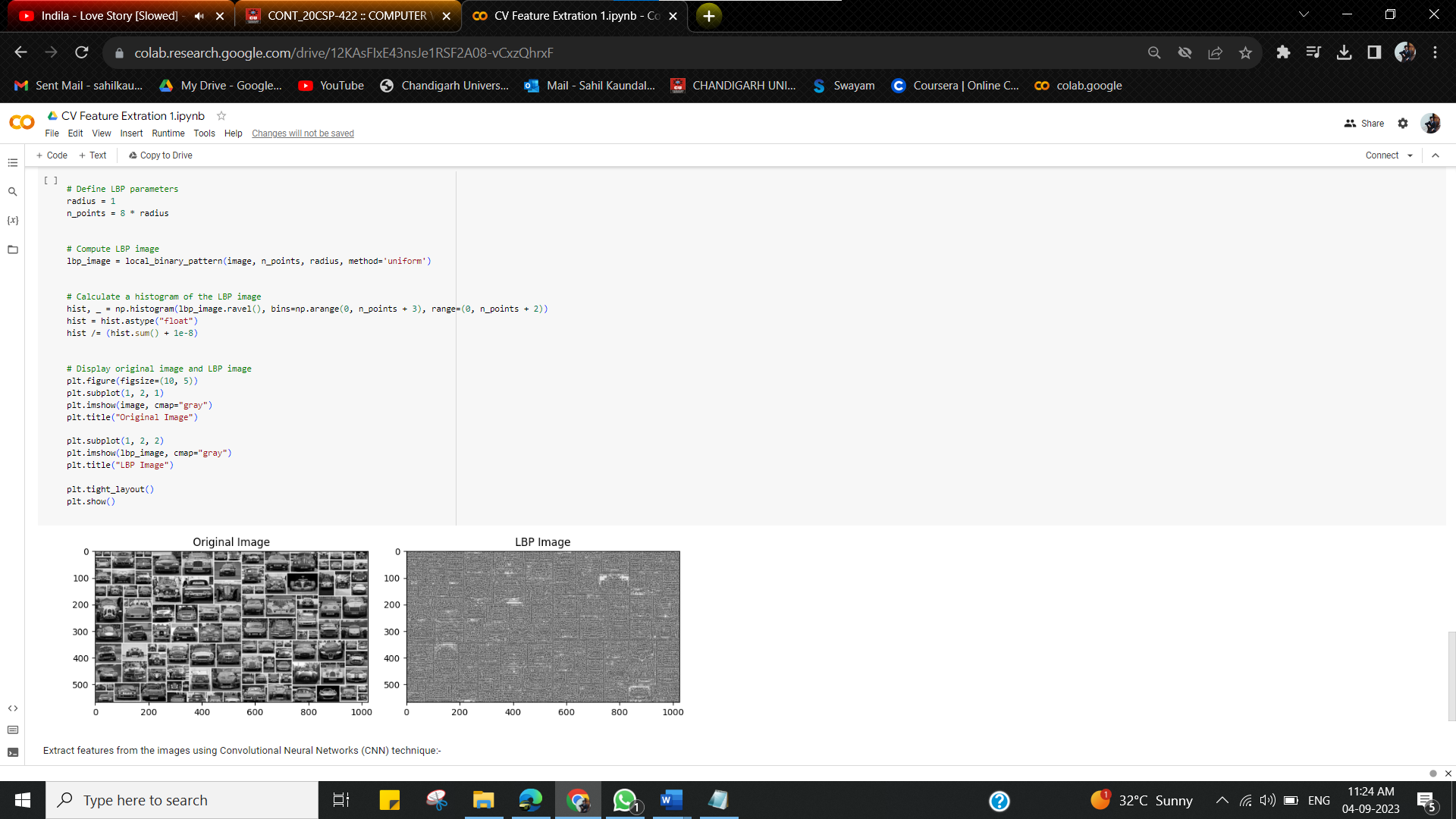
****

****

****

****

****

****