Content-based Image retrieval

Chirag Dhoka Jain, Keval Visaria
Project 2 Submission
course - ***EECE5330***
{dhokajain.c, visaria.k}@northeastern.edu
February 10, 2023
Northeastern University

I. Introduction

This project takes an interesting trip into the world of picture analysis and pattern recognition, combining traditional image processing techniques with sophisticated deep learning technologies. At its heart, the objective is to create a reliable system capable of detecting and matching photos from a database to a target image based on their content. Participants will receive practical experience altering and analysing pictures at the pixel level by combining standard image attributes like as colour, texture, and spatial layout with deep network embeddings generated from a ResNet18 model. This comprehensive method not only improves knowledge of various colour spaces and histograms, but it also introduces the practical use of spatial and texture aspects, in addition to deep learning embeddings.

With its intuitive graphical user interface, the project—which was created as a command-line application—allows users to enter the filename of a target picture and choose a directory that has an image database. Users may learn about picture matching and pattern recognition by selecting the sort of features to analyse and the matching technique using a simple numpad-based interface. The programme then uses this input to find and present the top N photographs in the database that most closely resemble the target image.

A. *Task 1: Baseline Matching*

The baselineMatching function zeroes in on a 7x7 pixel region at the heart of a given image, transforms this snippet into a flat array, and then hands it back. This technique is all about honing in on a distinct central slice of the image to serve as a snapshot of its essence. To gauge the similarity between images, it employs the sum-of-squared-difference method. Additionally, the program enhances user interaction by responding to the press of the numpad key '1', linking this specific action to a predefined operation within the application. This setup not only simplifies the process of comparing images but also makes it more engaging for the user.

The input image serves as the target file for the program, and the output image demonstrates the successful display in the application window.

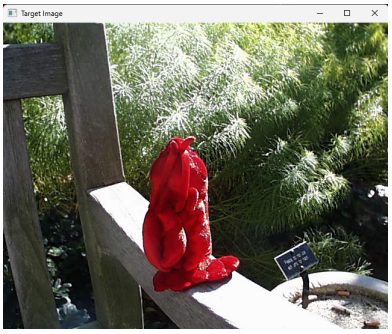

Fig. 1. Target Image

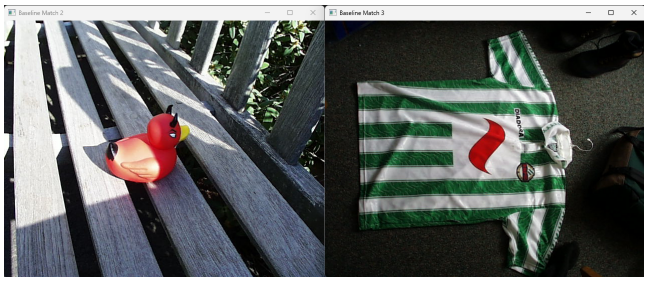

Fig. 2. Output of top images

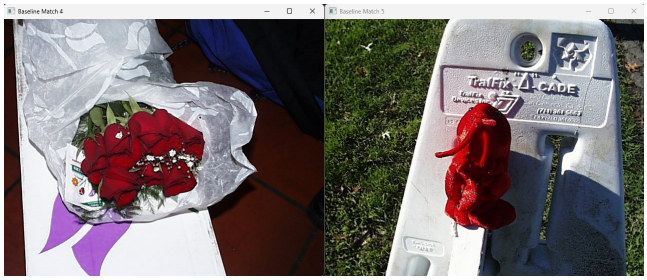

Fig. 3. Output of top images

```
Baseline Matching in Progres...
Top 5 baseline matches for the target image C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus/pic.1016.jpg:
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.1016.jpg (Distance: 0)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0986.jpg (Distance: 118.528)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0641.jpg (Distance: 147.499)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0547.jpg (Distance: 222.942)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.1013.jpg (Distance: 227.022)
```

Fig. 4. Distance metric of o/p

B. *Task 2: Histogram Matching*

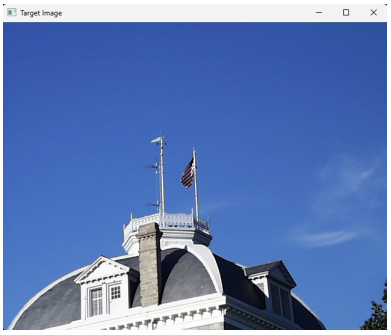

Fig. 5. Target Image

The task begins with obtaining colour histogram information from photos to serve as unique fingerprints. It generates a single, normalised colour histogram for each image, focusing on at least two dimensions (hue and saturation) of the HSV colour space. This approach transforms each image to HSV, divides it into channels, and generates a two-dimensional histogram by binning hue and saturation. The histograms are then normalised to guarantee uniformity between photos. This normalisation method scales the histogram values to a conventional range, allowing for a fair comparison of photographs. The resultant histogram, essentially a flattened array, serves as a feature vector, encapsulating the image's colour distribution in a compact form.

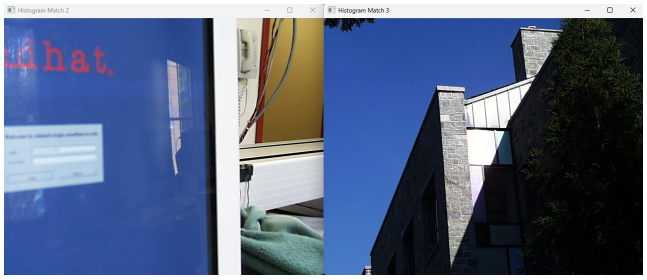

Fig. 6. Output of top images

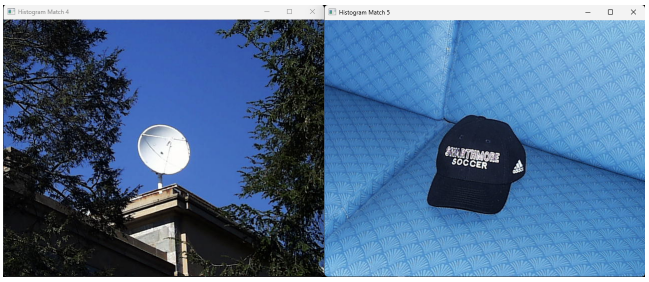

Fig. 7. Output of top images

Following feature extraction, the program uses a customised distance metric—histogram intersection—to determine the similarity between the target image's histogram and those in the database. It computes the distance between each image and the target image by iterating over the database, omitting the target image, and comparing its histogram characteristics. Images are then graded based on their distance, with lower numbers suggesting greater resemblance. The user interacts with this system via a command-line interface, with the numpad key '2' used to initiate the feature matching process. The program shows the top N matching photographs, demonstrating histogram matching's efficacy in locating images with comparable colour distributions.

```
Histogram Matching in Progres...
Top 5 histogram matches for the target image C:/Users/visar/Desktop/OneDrive - Northeastern University/PRC
V/olympus/pic.0164.jpg:
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0164.jpg (Distance: 0)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0080.jpg (Distance: 0.386541)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.1032.jpg (Distance: 0.418312)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0110.jpg (Distance: 0.487347)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0599.jpg (Distance: 0.739174)
```

Fig. 8. Distance metric of o/p

The input image serves as the target file for the program, and the output image demonstrates the successful display in the application window.

C. *Task 3: Multi-histogram Matching*

In this task, the system adopts a complex image matching method that involves using not just one, but two separate colour histograms as feature vectors. The first histogram depicts the colour distribution throughout the image, giving a thorough insight into its colour landscape. This is accomplished by first converting the image to the HSV colour space and then computing and normalising a histogram based on hue and saturation, with specific bins for each. The second histogram focuses on the image's core region, extracting a 7x7 pixel area to provide a more detailed and localised feature collection. This centre histogram follows a similar procedure of conversion, computation, and normalisation, guaranteeing

that both histograms are on an equal footing for comparison. Following feature extraction, the program cleverly mixes these histograms to perform a multi-faceted analysis. It calculates the distance between the target picture's histograms and those of the database images using a custom distance metric for each image in the database, omitting the target image. This measure might use techniques like histogram intersection and weighted averaging to balance the contributions of the whole-image and central-region histograms. The program calculates overall similarity by adding the distances obtained from comparing both sets of histograms. This multi-histogram matching procedure is initiated by pressing Key '3' on the numpad, leading users on a complex pattern recognition trip.

The input image serves as the target file for the program, and the output image demonstrates the successful display in the application window.

Fig. 9. Target Image

Fig. 10. Output of top images

D. Task 4: Texture and Color

In this task, The $textureMatching$ function is responsible for extracting texture descriptors from an input image using gradient information. Here's a step-by-step breakdown of what happens inside this function: **Grayscale Conversion**: The input image img is first converted from the BGR color space to grayscale using the $cvtColor$ function from the OpenCV library. This step simplifies subsequent processing by reducing the image to a single channel representing intensity values. **Gradient Calculation**: $Sobel$ filters are applied to the grayscale image to compute the gradients in both the horizontal (x) and vertical (y) directions. This is achieved using the Sobel function from OpenCV, which calculates the derivatives of the image along the x and y axes. The resulting gradient images $gradientX$ and $gradientY$ represent the intensity changes in the horizontal and vertical directions, respectively. **Magnitude and Angle Calculation**: The magnitudes and angles of the gradients are computed using the $cartToPolar$ function from OpenCV. This step converts the gradient vectors from Cartesian coordinates (x, y) to polar coordinates (magnitude, angle), where the magnitude represents the strength of the gradient, and the angle represents the direction of the gradient. **Histogram Construction**: A histogram of gradient magnitudes (histogram) is constructed to summarize the distribution of gradient strengths in the image. This histogram $bins$ the gradient magnitudes into discrete intervals and counts the number of pixels falling into each bin. The histogram provides a compact representation of the texture characteristics of the image, capturing information about the frequency and distribution of gradient strengths across the image. **Histogram Normalization**: To ensure that the texture descriptor is robust to variations in image intensity and contrast, the histogram is normalized using the normalize function from OpenCV. Normalization scales the histogram values to a predefined range (typically between 0 and 1), ensuring that the descriptor is invariant to changes in image brightness and contrast. **Feature Vector Reshaping**: Finally, the normalized histogram is reshaped into a one-dimensional matrix Mat to create a feature vector representing the texture characteristics of the input image. This feature vector encapsulates the essential texture information extracted from the image and serves as a compact representation for texture matching and comparison tasks.

Overall, the $textureMatching$ function computes texture descriptors from an input image by analyzing the distribution of gradient magnitudes, providing a robust and concise representation of the image's texture characteristics for subsequent texture matching or retrieval tasks. The user interacts with this system via a command-line interface within $task4$, with the numpad key '1' used to initiate the feature matching process.

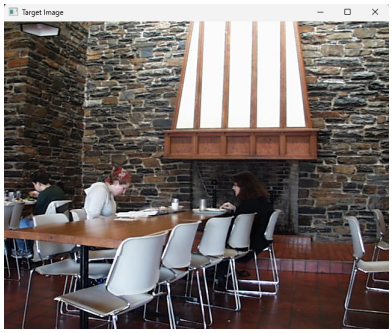

Fig. 11. Target Image

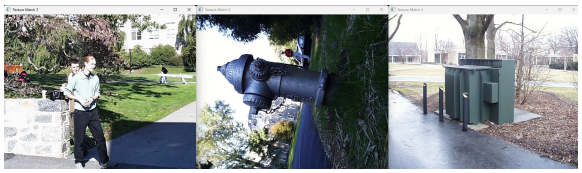

Fig. 12. Output of top three images

```
Performing Sobel Filter
Top 4 texture matches for the target image C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olymp
us/pic.0535.jpg:
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0535.jpg (Distance: 0)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.1033.jpg (Distance: 0.0258154)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0161.jpg (Distance: 0.0420694)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0555.jpg (Distance: 0.0461969)
```

Fig. 13. Distance metric of o/p

1) Extension: Laws filter

In our approach to texture analysis, we start by converting the input image into grayscale, a common first step across all three methods to ensure the focus remains on intensity variations rather than color. Within the Laws' Texture Energy Measures framework, we utilize five distinct masks designed to highlight various texture patterns including level, edge, spot, wave, and ripple. By convolving these masks with the grayscale image, we generate several feature maps, each illuminating a specific texture characteristic of the image. We then calculate the energy of each map through the L2 norm, quantifying the texture energy present. These calculations culminate in a composite feature vector, encapsulating the rich texture information we've distilled using Laws' methodology.

The Laws' texture energy measurements are used to extract diverse texture information from photos using convolution with certain kernel filters. Laws' filters are meant to emphasise various texture patterns, including level (L), edge (E), spot (S), wave (W), and ripple (R). Convolution of an image with these filters yields a series of response pictures that highlight various characteristics of the image's texture. To mathematically define the histograms of Laws filter responses:

1. Define Laws' Kernels: Each kernel is a vector that captures specific texture patterns. The kernels are defined as follows: Level (L): $L_5 = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$
Edge (E): $E_5 = \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \end{bmatrix}$
Spot (S): $S_5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \end{bmatrix}$
Wave (W): $W_5 = \begin{bmatrix} -1 & 2 & 0 & -2 & 1 \end{bmatrix}$
Ripple (R): $R_5 = \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \end{bmatrix}$

2. Convolution with Image: Convolution of an image (I) and a Laws' kernel (K) yields the response (R), indicated as: R = I $*K$

3. Compute Texture Energy: To quantify the texture information, the texture energy for each response R is computed. Usually, this is accomplished by computing the norm or by adding up all of the answer values across a window:- representing the equation where E represents the texture energy and denotes the L2 norm operation: E = $\|\mathbf{R}\|$

This histogram captures the core of the texture pattern in the image by offering a succinct depiction of the texture energy distribution caused by the particular Laws' filter. In summary, by measuring the distribution of energy values acquired by convolving the picture with a series of specialised kernels, the histograms of Laws filter responses capture the textural properties of an image.

The user interacts with this system via a command-line interface within $task4$, with the numpad key '2' used to initiate the feature matching process.

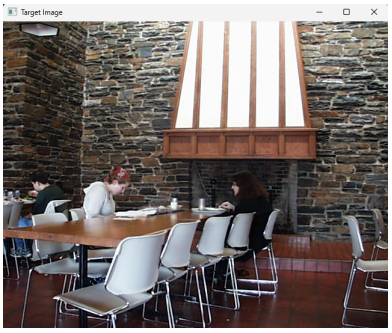

Fig. 14. Target Image

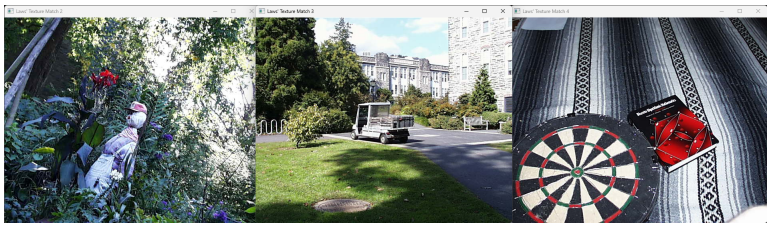

Fig. 15. Output of top three images

```
Top 4 Laws' texture matches for the target image C:/Users/visar/Desktop/OneDrive - Northeastern University
/PRCV/olympus/pic.0535.jpg:
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0535.jpg (Distance: 0)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0431.jpg (Distance: 304.152)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0807.jpg (Distance: 390.659)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0176.jpg (Distance: 472.447)
```

Fig. 16. Distance metric of o/p

2) Extension: Gabor filter

Since the Gabor filter can capture both spatial and frequency information, it is frequently employed in image processing for texture analysis. We used the inbuilt opencv Gabor filter. In essence, a Gabor filter is a sinusoidal wave—a plane wave in two dimensions—that has been modulated by a Gaussian envelope. The following is the mathematical formula for a spatially-domain 2D Gabor filter:

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{2\sigma^2 x'^2 + \gamma^2 y'^2}{\sigma^2}\right) \cos\left(2\pi\lambda x' + \psi\right) \tag{1}$$

Where: x' = $x\cos(\theta) + y\sin(\theta) y' = -x\sin(\theta) + y\cos(\theta)$

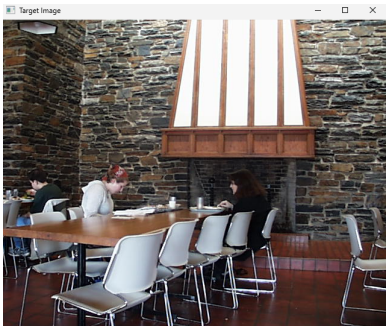

Fig. 17. Target Image

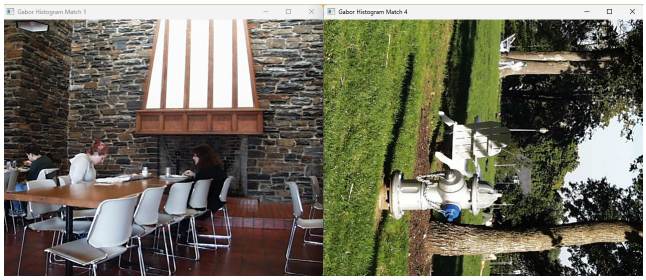

Fig. 18. Output of top images

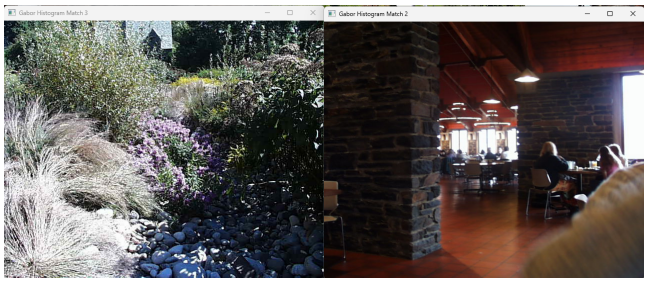

Fig. 19. Output of top images

```
Performing Gabor Filter
Top 4 Gabor histogram matches for the target image C:/Users/visar/Desktop/OneDrive - Northeastern Univers
ity/PRCV/olympus/pic.0535.jpg:
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0535.jpg (Distance: 0)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0551.jpg (Distance: 0.142229)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0124.jpg (Distance: 0.16333)
C:/Users/visar/Desktop/OneDrive - Northeastern University/PRCV/olympus\pic.0165.jpg (Distance: 0.169897)
```

Fig. 20. Distance metric of o/p

By applying Gabor filters with different orientations on a grayscale picture, we may capture a variety of texture features using filter responses. The replies are then merged into a big feature vector. For more detailed texture analysis, we construct and normalise a histogram of these filter responses, resulting in a compact, statistical representation of the textures discovered by the Gabor filters. Finally, this histogram is moulded into a 1D vector that may be used in a variety of computer vision applications, including picture categorization and texture analysis. This simplified approach successfully recovers and condenses texture information from photos, providing a more nuanced view on an image's textural features. The user interacts with this system via a command-line interface within $task4$, with the numpad key '3' used to initiate the feature matching process.

*3) **Extension: Fourier Transform filter***

In this task, The Fourier Transform turns an image's spatial representation into its frequency components, exposing its spectral properties. This transformation is useful for recognising repeated structures, orientations, and the general texture of a picture by studying its frequency content.

The mathematical formula for the 2D Fourier Transform F(u,v) of an image f(x,y) is given by:

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-2\pi i(ux+vy)}, dx, dy \tag{2}$$

Where: f(x,y) is the spatial domain picture; f(u,v) is the frequency domain image; u,v are the horizontal and vertical spatial frequency coordinates; and i is the imaginary unit.

We transform the grayscale image into a 32-bit float repre-

sentation, a prerequisite for applying the Fourier Transform. The Discrete Fourier Transform yields a complex image that encapsulates the frequency domain characteristics of the original image. From this, we calculate the magnitude spectrum, which we then resize and normalize, crafting a standardized feature vector. This vector is particularly insightful as it captures the frequency content of the image, revealing underlying textural patterns that are crucial for various analytical purposes.

The user interacts with this system via a command-line interface within $task4$, with the numpad key '4' used to initiate the feature matching process.

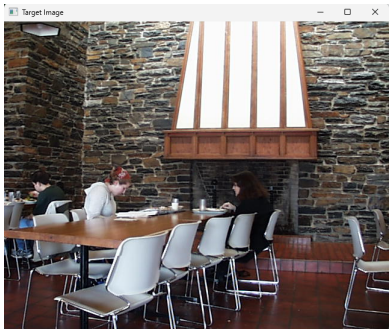

Fig. 21. Target Image

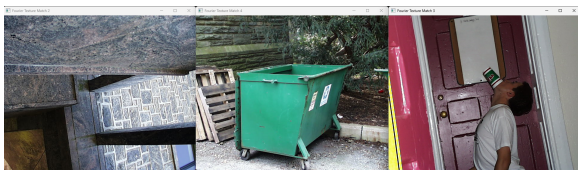

Fig. 22. Output of top three images

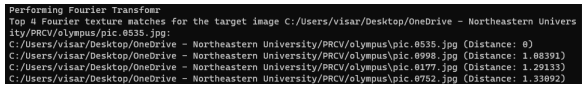

Fig. 23. Distance metric of o/p

*E. **Task 5: Deep Network Embeddings (DNN)***

In this task image retrieval system, deep neural network (DNN) embeddings are critical in detecting photos with comparable content. According to the code provided in the assignment, the method begins with extracting high-dimensional feature vectors from pictures using the final average pooling layer of a ResNet18 model pretrained on the ImageNet database.

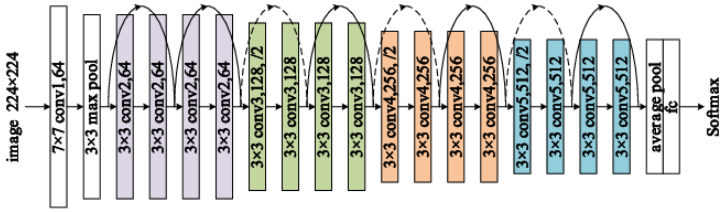

Fig. 24. ResNet18 Model

Each image is represented as a 512-dimensional vector, which captures its visual essence in a format that allows for comparison. These feature vectors are placed in a CSV file, with each row representing an image file and its corresponding 512-value feature vector. To enable picture matching, the system uses Euclidean distance as a metric, computing the spatial distance between the feature vectors of a query image and every other image in the dataset.

To determine the degree of similarity between images, we explore two separate approaches for measuring the closeness between these vectors: sum-square distance and cosine distance.

1. Sum-Square distance: This approach calculates the distance between two feature vectors by summing the squares of the differences between their members. Given two vectors:

$\mathrm{v}_1 = [x_1, x_2, ..., x_n]$ and $\mathbf{v}_2 = [y_1, y_2, ..., y_n]$

sum-square distance is determined by the formula: $\sum_{i=1}^{n}(x_i - y_i)^2$

2. Cosine Distance: Cosine distance, which is preferred in scenarios with high-dimensional data, calculates the cosine of the angle between two vectors to determine their similarity. $\mathrm{d}(\mathrm{v}_1, \mathbf{v}_2) = 1 - \cos(\theta)$

The cosine of the angle between two normalised vectors v1 v1 and v2 v2 may be computed as follows: $\cos(\theta) = \frac{\|\mathbf{v}_1\|\|\mathbf{v}_2\|}{\mathbf{v}_1 \cdot \mathbf{v}_2}$

These distance metrics allow for the assessment of similarity or dissimilarity between feature vectors. These metrics have applications in disciplines such as image retrieval and content-based image search, where they help identify images that are similar to a target image based on their deep feature representations. The closer the distance, the more identical the images appear to be. This assignment employs the pre-trained ResNet18 model to provide relevant and semantically rich representations of images for such comparisons.

When a user selects key 5 for feature matching, the system goes into action, getting the feature vector for the given query picture and comparing it to the vectors of all other images in the database, but not the query image itself. This comparison returns a list of photos ranked by their resemblance to the query image using the computed distances.

The input image serves as the target file for the program, and the output image demonstrates the successful display in the application window.

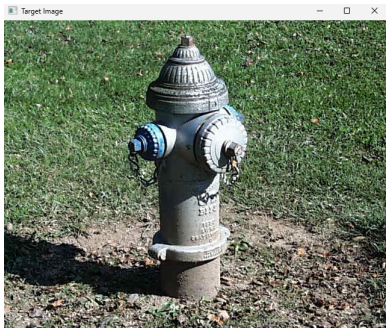

Fig. 25. Target Image

Fig. 26. Output of top images

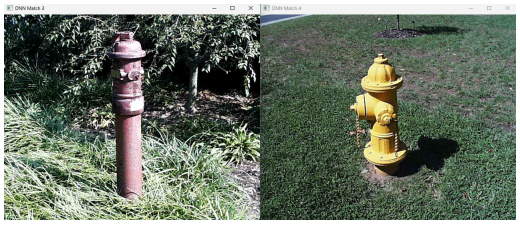

Fig. 27. Output of top images

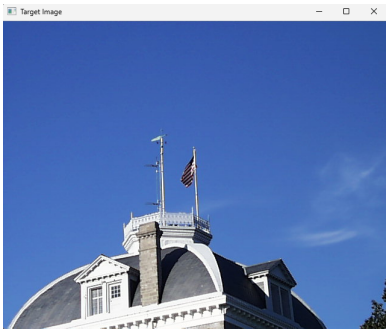

Fig. 28. Target Image

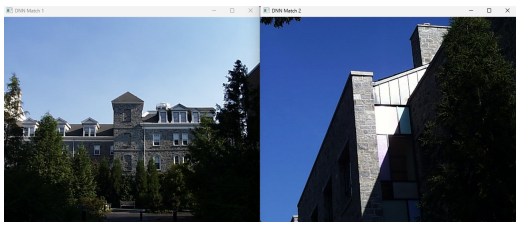

Fig. 29. Output of top three image

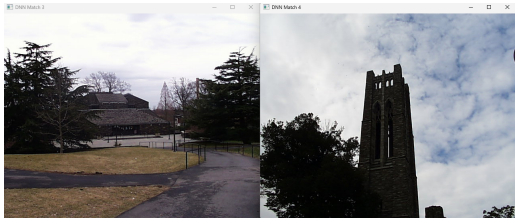

Fig. 30. Output of top image

F. Task 6: Compare DNN Embeddings and Classic Features

DNN embeddings improve feature extraction precision well beyond what classic methods such as histogram matching or basic baseline approaches can provide, owing to their comprehensive comprehension of visual content. These embeddings can capture sophisticated, high-level properties that standard methods, which frequently depend on direct pixel comparisons or simple statistical representations, cannot. This expanded feature set captures not only the visual, but also the semantic and contextual aspects of the picture, resulting in a more comprehensive understanding and representation. When using DNN embeddings to compare pictures, selecting the appropriate distance metric, such as Euclidean or cosine similarity, is critical for properly evaluating how similar they are. Unlike prior approaches that focused on fundamental components such as colour or texture, DNN embeddings delve into the deeper, semantic aspects of pictures. This enables for considerably more precise comparisons because the metrics now represent the real content and significance of the photos. As a result, this strategy significantly increases the efficacy of picture searches and categorization by capturing their underlying essence, representing a considerable advancement over standard feature extraction techniques.

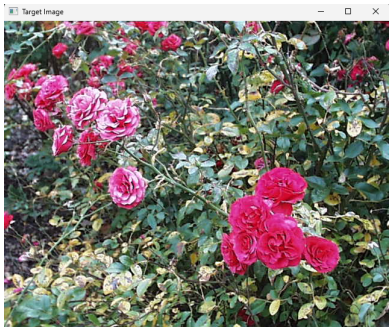

Fig. 31. Target Image one

Fig. 32. Output of top DNN images

Fig. 33. Output of top Hist images

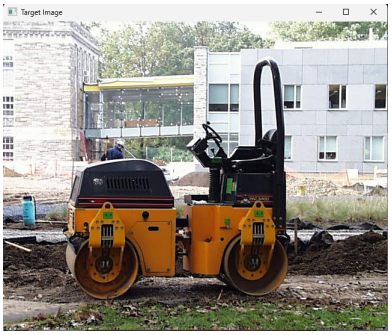

Fig. 34. Target Image two

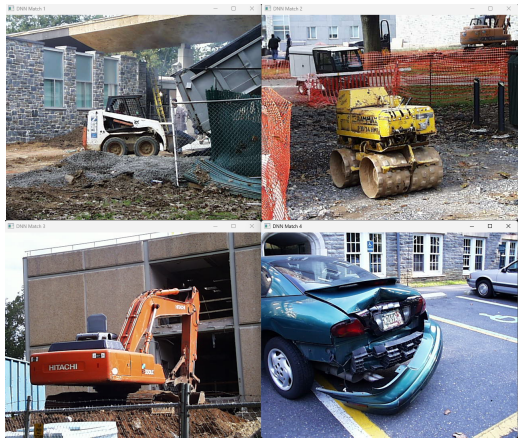

Fig. 35. Output of top DNN images

Fig. 36. Output of top Hist images

G. *Task 7: Custom design*

In this task, We've integrated three main components into our image retrieval system to identify how similar a target image is to photographs in our database. First, we use a pre-trained deep neural network (DNN) to extract high-level characteristics from photos and store them in a CSV file. We next use the cosine function to calculate the cosine distance between the target picture's feature vectors and each image in the database, which measures their similarity.

Furthermore, we use Sobel gradients, which capture edge information, to improve our similarity evaluation. We calculate the gradients for both the target and database pictures and compare their differences, taking into account how closely their edge structures correspond. These Sobel gradients are included in our similarity computation, along with DNN-based cosine similarity, to provide a full evaluation of picture similarity. This combination technique enables us to efficiently obtain photographs from the database that closely resemble the features of our target image.

Fig. 37. Target Image one

Fig. 38. Output of top similar images

Fig. 39. Output of top dissimilar images

The input image serves as the target file for the program, and the output image demonstrates the successful display in the application window.

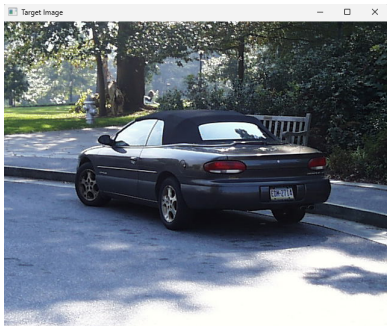

Fig. 40. Target Image two

Fig. 41. Output of top similar images

Fig. 42. Output of top dissimilar images

H. Extension: Face detection

In this task, we explore further into picture retrieval, not just by extracting feature vectors, but also by including color matching and the task 7 function itself. It's like looking through a big library of photographs and selecting the ones that most closely match our target image in the entire directory. However, in the instance of task 7 similarity and dissimilarity, we only provide the top 5 photos based on this. Here, we can take a lot of shots. The cosine similarity metric helps us determine the overall similarity of pictures, but the Sobel gradients provide a higher degree of detail, capturing edge structures for a more detailed comparison. It's similar to recognising patterns not just by their general shape, but also by the subtle complexities.

The input image serves as the target file for the program, and the output image demonstrates the successful display in the application window.

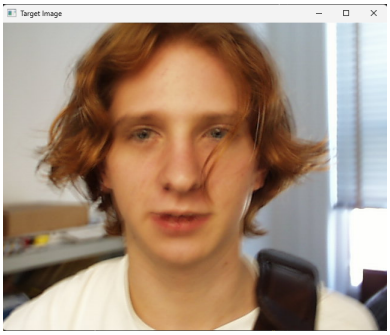

Fig. 43. Target Image

Fig. 44. Output of top similar images with face detected

II. WHAT WE LEARNED!!!

1) We've learned the value of feature extraction in picture analysis. Classic approaches, such as algorithms, are used to recognise aspects like as colour, texture, and form. Histograms are concise representations of pixel intensity or picture attribute distributions. They are important for capturing features like colour dispersion. Histogram-based measures, such as intersection, are frequently used to compare images.

2) We obtained insights on CNNs: Our experiment helped us better comprehend convolutional neural networks (CNNs). These networks handle structured grid data, such as photographs, and automatically learn hierarchical characteristics from raw pixel data. We used pre-trained CNNs like ResNet and VGG to efficiently extract features in a variety of image-related tasks.

III. ACKNOWLEDGEMENT

Acknowledging the wealth of knowledge available online, particularly in the 'OpenCV' documentation and tutorials, played a crucial role. This project's success is a testament to the collective effort of the educational community, emphasizing the shared wealth of expertise. Interactions with ChatGPT proved invaluable for exploring concepts and troubleshooting, enhancing the overall learning experience of the project. This acknowledgment is a tribute to the collaborative and supportive environment that significantly facilitated the learning journey in this project.

For the code related to this project, please refer to: https://github.com/ChiragDhoka/Project2.git[1]

[1]Code for the project.