Computer vision

CSL7360

Ajit Kumar

Instructor - Prof. Anand Mishra

Question1:

Introduction:

Eigenface recognition is a facial recognition technology that uses mathematical algorithms to analyze and compare facial features. It is a type of biometric authentication that is widely used in security systems, such as access control systems, surveillance systems, and border control.

The Eigenface recognition algorithm works by analyzing a set of facial images and identifying the most distinctive features of each face. These distinctive features are then used to create a mathematical

model or template for each face, which is stored in a database. When a new face is presented to the system, it is compared to the templates in the database to find the best match.

Eigenface recognition is based on the concept of eigenvalues and eigenvectors, which are mathematical terms used to describe the properties of matrices. In this technology, a matrix of facial images is used to extract the most significant features of the face, which are then used to create the eigenvectors. These eigenvectors are then used to create a low-dimensional space that represents the unique characteristics of each face.

Algorithm:

1. **Image Acquisition:** The first step is to acquire a set of facial images. These images should be in the same format and size to ensure consistency in the analysis.
2. **Preprocessing:** The images are preprocessed to normalize the lighting and remove any variations in the background or noise. This can be done using techniques such as histogram equalization, contrast enhancement, and noise reduction.
3. **Face Detection:** The images are analyzed to detect the location of the face. This can be done using techniques such as Viola-Jones algorithm, which uses Haar-like features to detect faces.
4. **Feature Extraction:** The next step is to extract the most significant features of each face. This is done using the Principal Component Analysis (PCA) algorithm, which is a mathematical technique used to reduce the dimensionality of the data. In this

step, the algorithm calculates the mean face and the eigenvectors of the facial images. The eigenvectors represent the most significant features of the faces, such as the eyes, nose, and mouth.

5. **Face Representation:** The eigenvectors are used to create a low-dimensional representation of each face, which is called the eigenface. The eigenface is a mathematical model that represents the unique characteristics of each face.
6. **Face Recognition:** When a new face is presented to the system, it is compared to the eigenfaces in the database to find the best match. This is done by calculating the distance between the new face and the eigenfaces. The face with the closest distance is considered to be the best match.
7. **Decision:** Based on the distance calculations, the system decides whether the new face is a match or not. If the distance is below a certain threshold, the system considers it to be a match.

Advantage:

1. **High Recognition Accuracy**: The Eigenface technique is known for its high recognition accuracy, especially when the input images are of good quality and the dataset is well balanced.
2. **Low Computational Complexity**: The Eigenface technique is computationally efficient and requires fewer computations compared to other face recognition techniques. This makes it well-suited for applications where real-time performance is critical.

3. **Simplicity**: The Eigenface technique is relatively simple and easy to implement compared to other face recognition techniques. It involves straightforward linear algebra computations, which can be easily implemented in most programming languages.
4. **Robustness**: Eigenfaces can be robust to certain types of image variations, such as changes in lighting conditions or facial expressions. This makes them effective for recognizing faces across different environments and situations.
5. **Fewer Samples Required**: Eigenfaces can be trained using a relatively small number of samples, making them well-suited for applications where data collection is difficult or expensive.

Limitation:

1. **Sensitivity to Image Quality**: The Eigenface technique relies heavily on the quality of the input images used for training and testing. If the images are of low quality or contain artifacts, the accuracy of the algorithm may be reduced.

2. **Overfitting**: Eigenfaces are sensitive to overfitting, which occurs when the algorithm becomes too specialized to the training data and cannot generalize well to new data. This can lead to poor performance on new images.

3. **Limited Variation Capture**: Eigenfaces can only capture a limited amount of variation in facial features. This means that the technique may not be able to distinguish between individuals who have similar facial features, such as twins.

4. **Pose and Expression Variability**: Eigenfaces are also sensitive to changes in pose and expression. If the input images used for training and testing contain significant variation in pose or expression, the accuracy of the algorithm may be reduced.

5. **Scalability**: Eigenfaces can be computationally expensive, particularly when dealing with large datasets or high-resolution images. This can limit the scalability of the algorithm in certain applications.

Result:

```python
# Match each test image to the closest training image
for i, test_feature in enumerate(test_features):
    max_score = -np.inf
    max_index = -1    "mean_image" is not defined
    for j, train_feature in enumerate(train_features):
        score = get_similarity_score(test_feature, train_feature)
        if score > max_score:
            max_score = score
            max_index = j
    #print(f'Test image {i+1} matches with training image {max_index+1} with a similarity score of {max_score:.2f}')
    # test_image = cv2.imread(os.path.join(test_folder, test_images[i]))
    # train_image = cv2.imread(os.path.join(train_folder, train_images[max_index]))
    test_image=test_images[i]
    train_image=train_images[max_index]
    #print(test_image.shape)
    #print(train_image.shape)

    plt.imshow( test_image)
    plt.show()
    plt.imshow(train_image)
    plt.show()
```


0
20
40
60
80
0
20
40
60
80
0
20
40
60
80
0
20
40
60
80

Question2:

Introduction:

Visual BoW (Bag-of-Words) is a popular feature extraction and representation technique used in computer vision for tasks such as image classification, object recognition, and image retrieval.

The basic idea behind the Visual BoW model is to represent an image as a histogram of visual "words" or "features" that are extracted from the image. The term "word" or "feature" is used here to refer to a local visual descriptor such as SIFT or SURF that captures information about the appearance and texture of a region of an image.

Algorithm:

1. Extract local features from a set of training images using a feature detector and descriptor such as SIFT or SURF.
2. Cluster the extracted features using a clustering algorithm such as k-means to form a set of visual words or codewords.
3. Assign each local feature in each image to the nearest visual word, creating a histogram of visual words for each image.
4. Normalize the histograms to account for differences in image sizes and numbers of features.
5. Train a classifier such as an SVM using the normalized histograms as features.

Advantage:

1. **Robustness**: The BoW model is robust to small changes in object pose, lighting conditions, and other variations in image appearance, making it a popular choice for object recognition and image retrieval tasks.
2. **Efficiency**: The BoW model can efficiently represent an image as a histogram of visual words, which can be used as input to a classifier. This allows for efficient processing of large datasets, which is important for real-time applications.
3. **Flexibility**: The BoW model can be easily adapted to work with different types of visual features and descriptors, allowing it to be used in a wide range of applications. Additionally, the model can be easily extended to handle more complex representations, such as using spatial information or multiple visual descriptors.
4. **Interpretability:** The BoW model produces a histogram of visual words that can be easily interpreted and visualized, making it useful for understanding the features that are important for different types of image classification tasks.

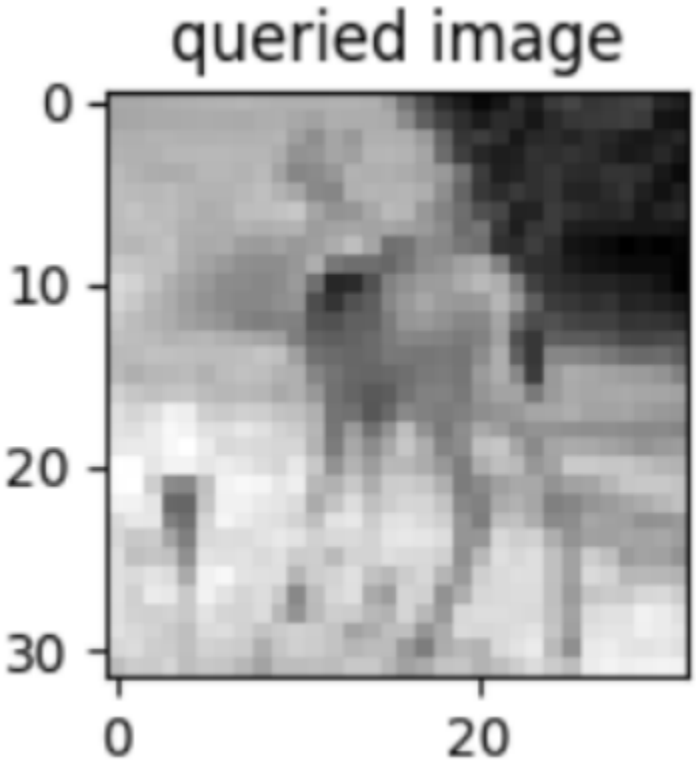
5. **Scalability:** The BoW model can be easily scaled to handle large datasets, making it suitable for applications such as image retrieval in web search engines or video surveillance systems.

Limitation:

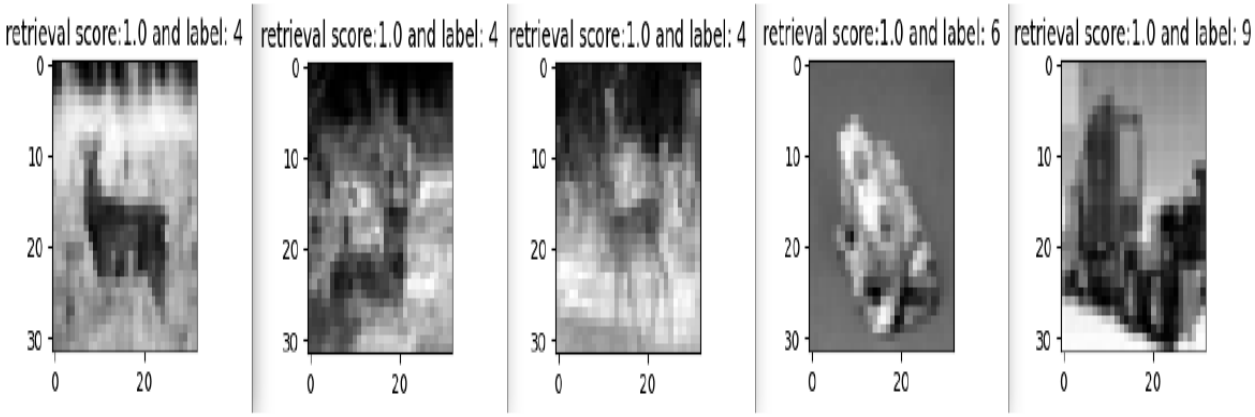
1. **Loss of spatial information**: The BoW model discards the spatial information of local features, which can be important for tasks such as object detection and localization.
2. **Fixed dictionary size**: The BoW model requires a fixed dictionary size, which can limit the model's ability to handle complex or diverse datasets.
3. **Limited representation**: The BoW model is limited to representing images as histograms of visual words, which may not capture all of the important features in an image.
4. **Sensitivity to noise**: The BoW model is sensitive to noise and outliers, which can negatively impact the performance of the model.
5. **Lack of context**: The BoW model treats each visual word as independent, which can lead to a lack of context and the inability to capture spatial relationships between visual words.
6. **Computationally expensive**: The BoW model can be computationally expensive to train and evaluate, especially when dealing with large datasets.

Result:

Query Image:



Top 5 matched Images:

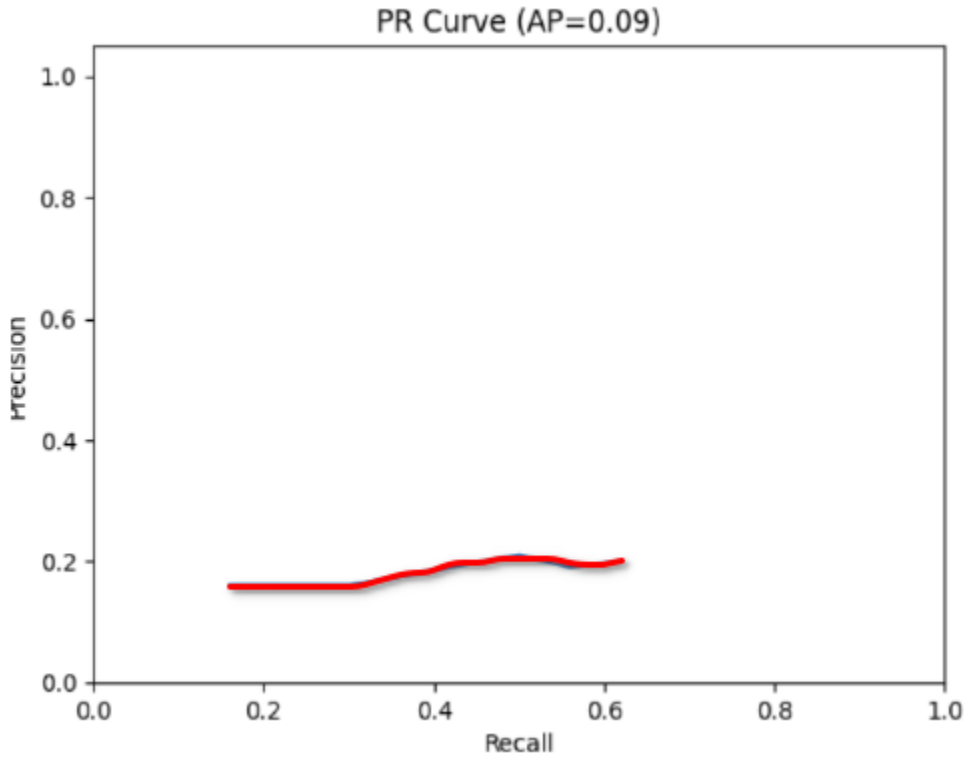


Precision and recall:

```
print(precision)
print(recall)
```
✓ 0.0s

```
0.2029665555
0.47021
```

P-R Curve:



Question3:

Introduction:

Viola-Jones face detection is a popular algorithm for detecting faces in digital images. The algorithm was developed by Paul Viola and Michael Jones in 2001, and it has since become one of the most widely used methods for face detection**.**

The Viola-Jones algorithm uses a cascade of simple classifiers to quickly identify regions of an image that are likely to contain a face. The algorithm first analyzes the image using Haar-like features, which are rectangular features that can be used to detect patterns in the image. These features are chosen based on their ability to distinguish between faces and non-faces.

Once the Haar-like features are selected, the algorithm computes the integral image of the input image. The integral image is a fast way of calculating the sum of pixel values in a rectangular area of an image. This allows the algorithm to quickly evaluate the Haar-like features at any position in the image.

The algorithm then uses a machine learning technique called AdaBoost to select a subset of the most effective features for face detection. The AdaBoost algorithm trains a cascade of simple classifiers that can make decisions based on a subset of the features. The classifiers are trained to detect faces at different scales and orientations.

Finally, the algorithm applies the trained cascade of classifiers to the image to identify regions that are likely to contain a face. If a region is classified as a face, the algorithm outputs the location of the face.

The Viola-Jones algorithm is known for its high accuracy and fast performance, making it well-suited for real-time face detection applications. It is widely used in a variety of computer vision applications, including face recognition, surveillance, and video analysis.

Algorithm:

1. **Haar feature selection**:Haar-like features are rectangular features that can be used to detect patterns in an image. The algorithm first selects a set of features that are most likely to represent facial features like eyes, nose, and mouth. These features are chosen based on their ability to distinguish between faces and non-faces.
2. **Integral image computation:**Integral image is a fast way of calculating the sum of pixels in a rectangular area of an image. It is computed by adding up all the pixels above and to the left of a given pixel. The integral image can be used to quickly calculate the sum of pixels in any rectangular region of the image.
3. **Adaboost training:**The Adaboost algorithm is used to select a subset of the best features from the set of Haar features. Adaboost algorithm selects a set of weak classifiers (i.e., classifiers that can only make decisions based on a subset of the features). The weak classifiers are then combined to form a

strong classifier, which can make accurate face detection decisions.

4. **Cascading classifiers**:In order to reduce the number of false positives, a cascading classifier is used to sequentially apply the weak classifiers to different parts of the image. This helps to quickly eliminate non-face regions of the image that are unlikely to contain a face.
5. **Face detection**:Once the Haar features are selected, the integral images are computed, and the Adaboost algorithm is trained, the face detection algorithm can be applied to new images. The image is scanned with a sliding window of various sizes and aspect ratios. At each window position, the Haar features are calculated and passed through the Adaboost classifier. If the classifier returns a positive result, the window is classified as containing a face.

Advantage:

1. **High Accuracy**: The Viola-Jones algorithm is known for its high accuracy in detecting faces. The algorithm is able to detect faces in various orientations, lighting conditions, and poses, making it effective in a variety of real-world scenarios.
2. **Fast Performance**: The Viola-Jones algorithm is also known for its fast performance. The algorithm uses a cascade of simple classifiers that can quickly eliminate non-face regions of the image. This results in fast face detection even on low-power devices.

3. **Robustness**: The Viola-Jones algorithm is robust to changes in the background and lighting conditions. The algorithm is able to distinguish between faces and non-faces even in complex backgrounds.
4. **Easy to Implement**: The Viola-Jones algorithm is easy to implement and can be used with minimal computing resources. The algorithm has been widely implemented in various programming languages, making it accessible to developers.
5. **Widely Used**: The Viola-Jones algorithm is widely used in a variety of computer vision applications, including face recognition, video analysis, and surveillance. This has led to the development of various libraries and tools that can be used to implement the algorithm.

Limitation:

1. **False Positives**: The Viola-Jones algorithm can sometimes produce false positive detections, where non-face regions of the image are classified as faces. This can occur when the algorithm encounters regions of the image that are similar to the features used to detect faces.
2. **Limited Detection Range**: The Viola-Jones algorithm is optimized to detect faces at a specific size and orientation. This means that the algorithm may not be able to detect faces that are too small or too large, or those that are tilted or rotated in a certain way.
3. **Sensitivity to Lighting Conditions:** The Viola-Jones algorithm may not perform well under certain lighting conditions, such as

low light or backlight. This is because the algorithm relies on the contrast between the face and the background to detect faces.

4. **Training Data Bias**: The performance of the Viola-Jones algorithm can be affected by the training data used to train the classifiers. If the training data is biased or limited, the algorithm may not be able to accurately detect faces in real-world scenarios.
5. **Computationally Intensive**: The Viola-Jones algorithm requires a significant amount of computation to train the classifiers and detect faces in an image. This can be a limitation for resource-constrained devices, such as smartphones or embedded systems.

Question4:

Introduction

HOG

HOG (Histogram of Oriented Gradients) features are a type of feature descriptor commonly used in computer vision for object detection and image recognition.

HOG features are extracted by dividing an image into small

rectangular cells and computing a histogram of gradient orientations within each cell. The orientation of each gradient is quantized into a discrete set of orientation bins, and the magnitude of each gradient is used to weight the contribution of that gradient to the histogram.

After computing the histograms for each cell, the histograms are then normalized within a larger block of cells to reduce the effects of illumination variations. The resulting HOG feature vector represents the distribution of oriented gradients within the image, and can be used for tasks such as object detection and pedestrian detection.

Sliding Window:

The sliding window technique is a common method used in computer vision and image processing for object detection and recognition.

The technique involves defining a rectangular window of a fixed size and sliding it over an image at different positions and scales. At each position and scale, the contents of the window are analyzed using a feature descriptor such as HOG (Histogram of Oriented Gradients) or SIFT (Scale-Invariant Feature Transform) to determine whether an object of interest is present.

The sliding window technique is useful for detecting objects at

different scales and positions within an image. However, it can be computationally expensive, as it involves analyzing many different regions of an image. To reduce the computational cost, various optimizations can be applied, such as using image pyramids to analyze images at different scales, or using selective search to identify regions of an image that are likely to contain objects of interest.

SVM:

SVM (Support Vector Machine) is a supervised machine learning algorithm used for classification and regression analysis. SVM classifiers are used to classify data into one of two or more classes, based on a set of labeled training data.

The SVM algorithm works by finding a hyperplane that maximally separates the data points of different classes in a feature space. The hyperplane is defined by a set of support vectors, which are the data points closest to the hyperplane. The distance between the support vectors and the hyperplane is called the margin, and the SVM algorithm seeks to find the hyperplane that maximizes the margin.

In cases where the data is not linearly separable, the SVM algorithm can use a kernel function to map the data into a higher-dimensional space where it becomes linearly separable. Popular kernel functions include the linear kernel, polynomial kernel, and radial basis function (RBF) kernel.

SVM classifiers have several advantages over other classifiers, such as logistic regression and decision trees, including their ability to handle high-dimensional data and their resistance to overfitting. They are commonly used in applications such as image classification, text classification, and bioinformatics.

Algorithm and Procedure:

1. As given in the question, I have made two folders “Deer” and “Non-Dear”. The Deer folder contains the images of the deer only and the Non-Deer folder contains the images of the gras, mountains other than the deer.
2. The Deer folder is made from cropping the image in the given dataset.
3. Read all the images and calculate the Hog-feature on argument orientations = 9, pixels_per_cell = (1, 1), cells_per_block = (1, 1), flatten = True. I tried with the different values but the boundary box obtained on the test images is not correct. Given above parameter, gives the result that will be shown in the result part.For the deer images, we have assigned the ground truth as 1 and for the non-deer images 0.
4. Then fit the feature of the image into the **SVM** classifier.
5. At last, traverse all the images of the test folder and resize the images to (128,128). Define the windows size to (64,64) and step size=4 for slide over the images. Then for the given window size, we calculate the feature and then do the prediction using **SVM.** If SVM gives the result to 1 then draw a rectangle on the

image of the window size. Here we have taken windows size is (64*64) because while calculating the **HOG** feature for the SVM training, we have taken the images size as (64,64), otherwise if we take different windows sizes it will produce errors like SVM expecting some other feature size.

Result: