CV Assignment-3

By:- Kwanit Gupta (B19EE046)

Colab Notebook:-
https://colab.research.google.com/drive/19rX9cvFOv6sZe_fcEsfVCWXJQ1evIMsX?usp=sharing

Drive Link:-
https://drive.google.com/drive/folders/18h-YACJRjZEQZSS5JerhRdWqmqcvRfnO?usp=sharing

Q.1 Use the subset of the LFW dataset provided with this assignment, include 1 face photograph of your favorite Indian sportsperson from the web to augment the dataset, and implement Eigen face recognition from scratch. You may use the PCA library, but other functionalities should be originally written. Show top-K Eigen's faces of the favorite Indian sportsperson you considered in for different values of K. The report should also contain a detailed quantitative and qualitative analysis. (Use provided data as train set and a test set will be provided separately)

Solution:-

1. Preprocess the images by converting them to grayscale and resizing them to a common size.

2. Calculate the mean face of the subset of images.
    a. This is done by averaging the pixel values of all images after flattening them.

3. Subtract the mean face from each image to get the deviation from the mean.
    a. This step helps to eliminate the lighting and shading differences in the images.

4. Calculate the covariance matrix of the deviation faces.
    a. This matrix shows how much the faces differ from each other.

5. Compute the eigenvectors and eigenvalues of the covariance matrix.
    a. Eigenvectors determine the directions in which the data varies the most.
    b. Eigenvalues indicate the magnitude of the variation in the respective eigenvector directions.

6. Sort the eigenvectors in decreasing order of their corresponding eigenvalues.
    a. This helps to prioritize the most significant eigenvectors.

7. Select a subset of the eigenvectors with the highest eigenvalues to form the principal components.
    a. The number of components is typically chosen based on the percentage of variance they explain.

8. Project each deviation face onto the principal components to obtain the weights for each image.
    a. This is done by taking the dot product of each deviation face with the principal components.
9. Store the weights and principal components for later use in face recognition.

10. For face recognition, the algorithm takes a new image, preprocesses it, and computes the deviation face.

11. Project the deviation face onto the principal components to obtain the weights for the new image.

12. Compare the weights of the new image with the stored weights of the subset of images using a distance metric, such as Euclidean distance.

13. Find the closest match by selecting the stored image with the minimum distance.

14. If the minimum distance is below a certain threshold, the new image is considered a match with the stored image.

Limitations:

1. EigenFace algorithm may not work well in cases where the illumination and pose of the new image are significantly different from the training images.

2. The algorithm may also be affected by facial expressions, background overlay and occlusions.

Scope of Improvements:

1. Combination of EigenFaces with other face recognition techniques such as Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG).

2. Usage of more advanced machine learning techniques such as Convolutional Neural Networks (CNN) for better recognition accuracy.

Qualitative Analysis:

EigenFaces are a set of eigenvectors of the covariance matrix of the face image dataset. Each EigenFace represents a specific direction in the high-dimensional space of face images. These directions can be interpreted as the most significant features that vary across the face images. Therefore, EigenFaces can be used to represent any face image as a linear combination of these directions.

To analyze the EigenFaces qualitatively, we can visualize them as images. Each EigenFace is a grayscale image that shows the pattern of variation in the dataset captured by that specific direction. For example, the first EigenFace captures the average face of the dataset, while the second EigenFace captures the difference between male and female faces, and so on.

Quantitative Analysis:

To quantify the quality of the EigenFaces, we can compute the variance explained by each EigenFace. The variance explained by an EigenFace is proportional to the corresponding eigenvalue of the covariance matrix. Therefore, we can sort the EigenFaces in descending order of their eigenvalues and plot the cumulative percentage of variance explained as a function of the number of EigenFaces used.

For example, if we use the first five EigenFaces to represent a face image, we can compute the percentage of variance explained by summing the first five eigenvalues and dividing by the total sum of eigenvalues. This percentage represents how much of the variability in the dataset is captured by the first five EigenFaces.

Also, I tested on some of the training as well as unseen “Virat Kohli” Image for EigenFace. But the results weren’t good as the identities associated with those predictions didn’t match with the personalities.

Q.2 Develop an Image Search Engine for CIFAR-10 that takes the image as a query and retrieves top-5 similar images using Visual BoW. Report Precision, Recall, and AP. Draw the P-R curve. Write down each step of implementation in clear and precise terms with an appropriate illustration.

Solution:- Visual BoW is a popular technique in computer vision used for image classification and object recognition. It involves extracting features from images and representing them as a histogram of visual words. These visual words are obtained by clustering similar local features

together using a clustering algorithm such as KMeans. The BoW representation can then be used to train a classifier such as SVM for image classification.

1. Data Preprocessing

    a. We use the CIFAR-10 dataset which consists of 60,000 32x32 color images in 10 classes.
    b. The dataset is split into training and testing sets with 50,000 and 10,000 images respectively.
    c. The images are converted to grayscale and normalized to have zero mean and unit variance.
    d. SIFT (Scale-Invariant Feature Transform) is used as the feature descriptor to extract local features from the images.

2. Feature Extraction

    a. SIFT is a feature descriptor that is invariant to scale, rotation, and affine distortion.
    b. It works by detecting key points in an image and computing a descriptor for each key point based on the local image gradient.
    c. The SIFT descriptors for all the key points in all the images are extracted using the OpenCV library.

3. Clustering and Codebook Generation

    a. The SIFT descriptors from all the images are clustered using the KMeans algorithm to form a codebook of visual words.
    b. The number of clusters (i.e. visual words) is a hyperparameter that needs to be tuned.
    c. The centroids of the KMeans clusters represent the visual words in the codebook.
    d. The codebook is saved for later use during feature encoding.

4. Feature Encoding

    a. For each image, the SIFT descriptors are extracted and assigned to the nearest visual word in the codebook using a nearest neighbor search.
    b. The frequency of each visual word in the image is counted to form a histogram of visual words, which is the BoW representation for that image.
    c. The BoW representations for all the images in the training set are used to train a K-Nearest Neighbor.

5. Classification and Evaluation

a. The trained K-Nearest Neighbor is used to predict the Top-K Images with the given Image as Query.

Following were the query image and top-5 similar images found by the Visual BoW using ORB, AKAZE and SIFT with only 5 main feature descriptor vectors (for faster computation upon lower image size):-

Query Image (1st image):-

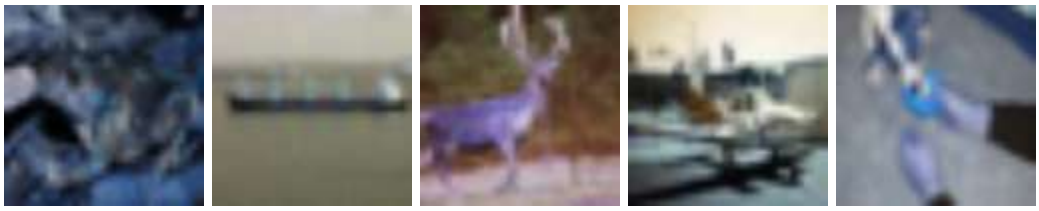
Top-5 Suggestions (By ORB Feature Descriptor)

Top-5 Suggestions (By AKAZE Feature Descriptor)

Top-5 Suggestions (By SIFT Feature Descriptor)

Query Image (500th image):-

Top-5 Suggestions (By ORB Feature Descriptor)

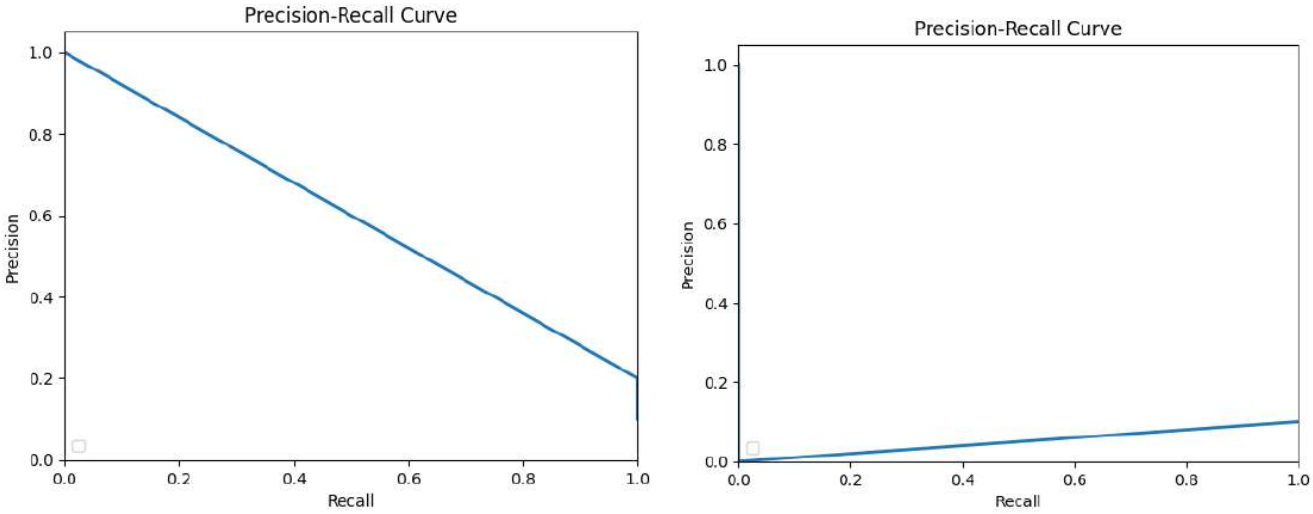


Top-5 Suggestions (By AKAZE Feature Descriptor)

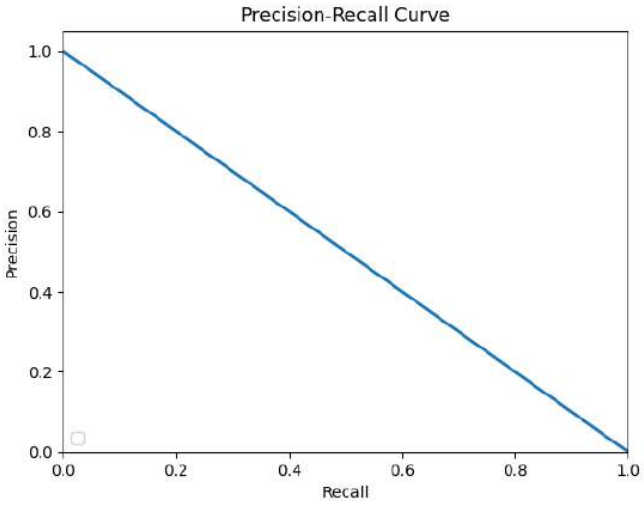
Top-5 Suggestions (By SIFT Feature Descriptor)

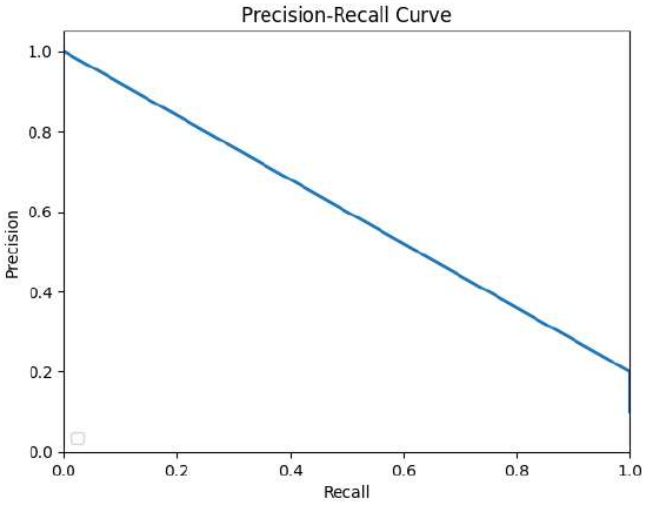
Following are the Precision-Recall Curves for ORB feature descriptors:-

1. Class 1 and Class 2

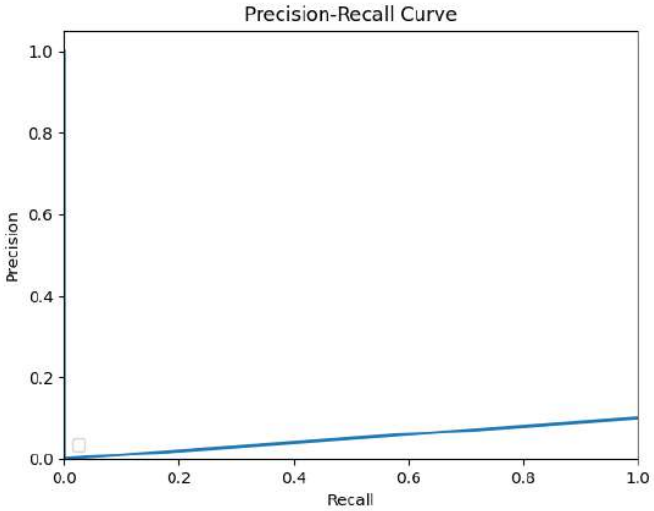


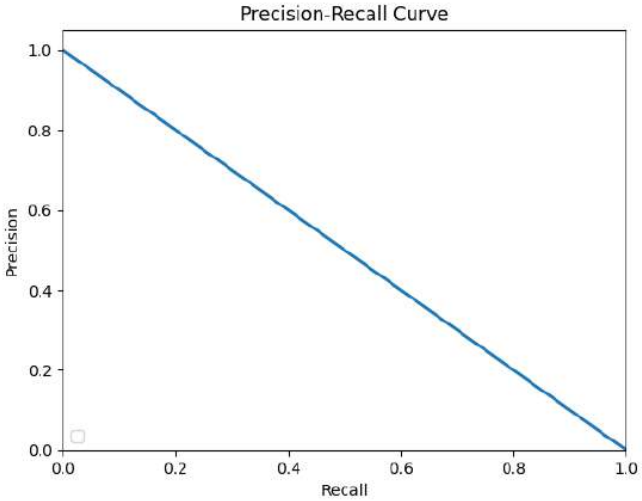
2. Class 3 and Class 4



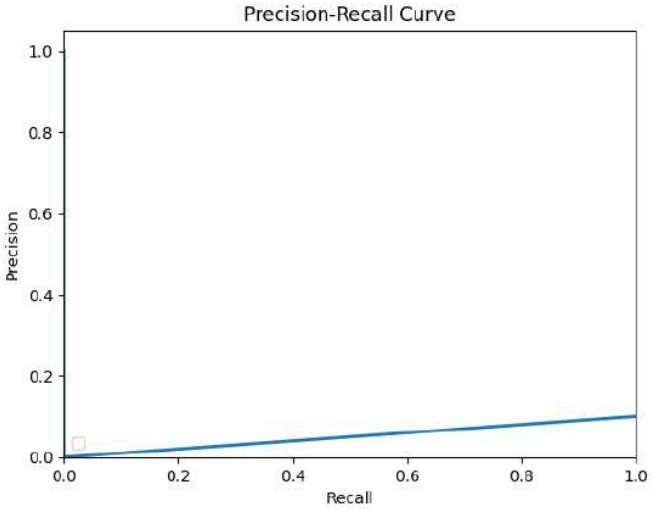


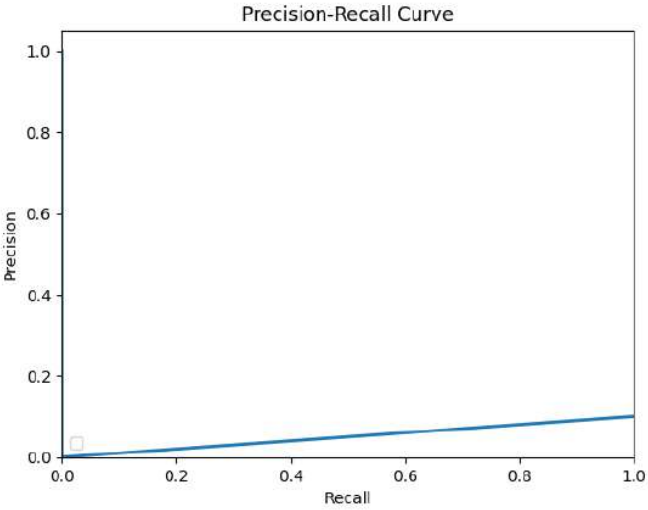
3. Class 5 and Class 6



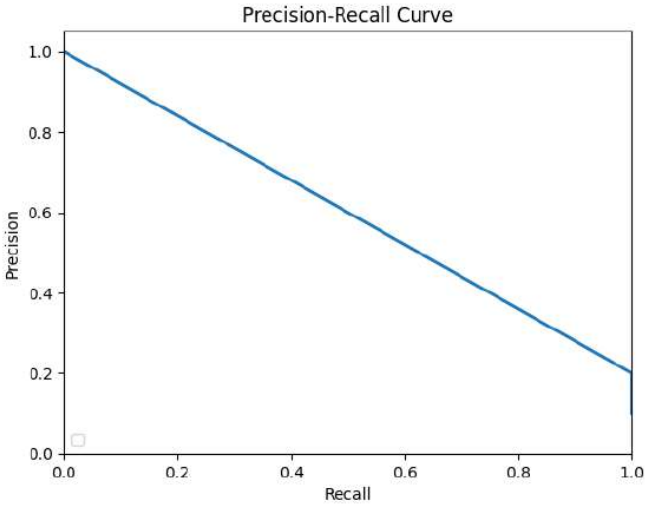


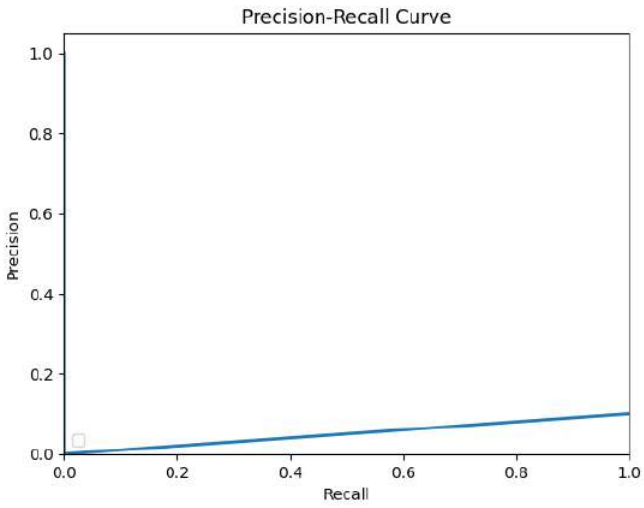
4. Class 7 and Class 8





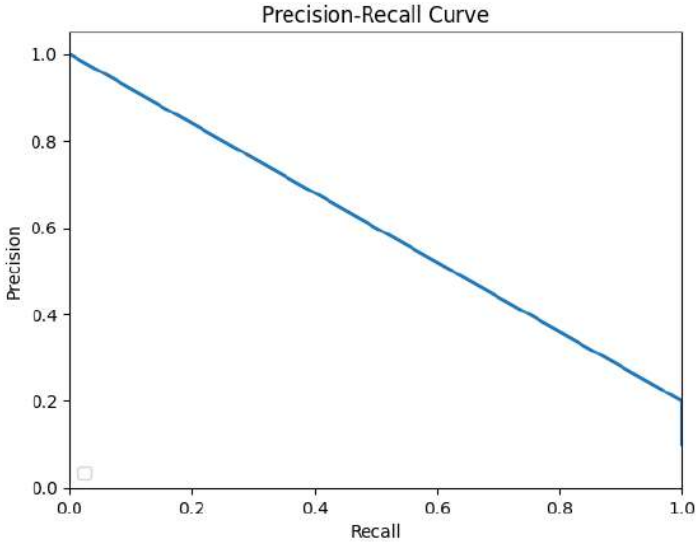
5. Class 9 and Class 10

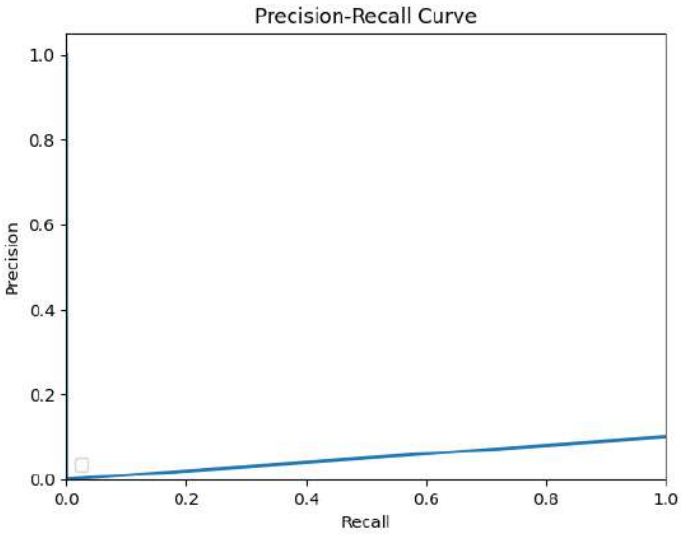




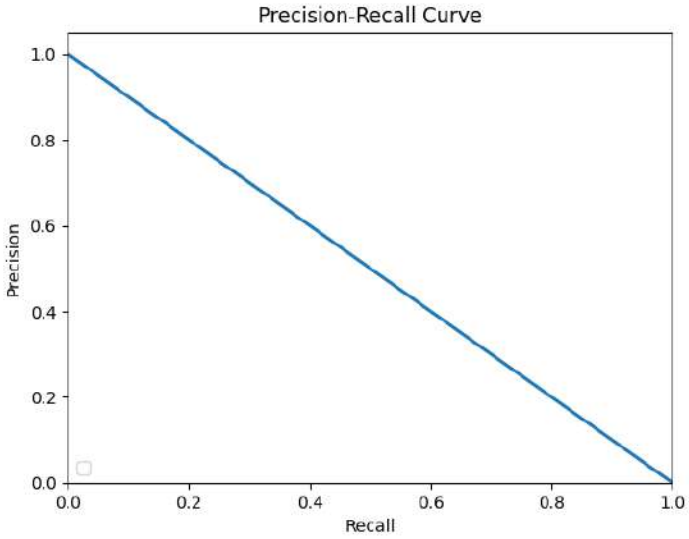
Following are the Precision-Recall Curves for AKAZE feature descriptors:-

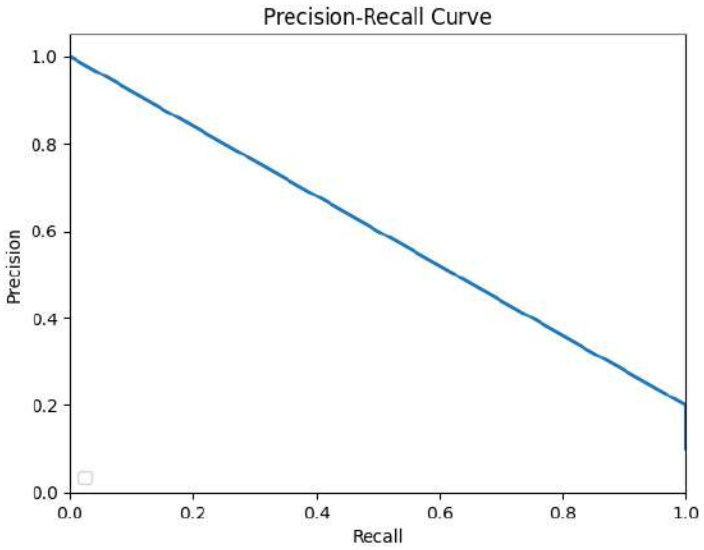
1. Class 1 and Class 2



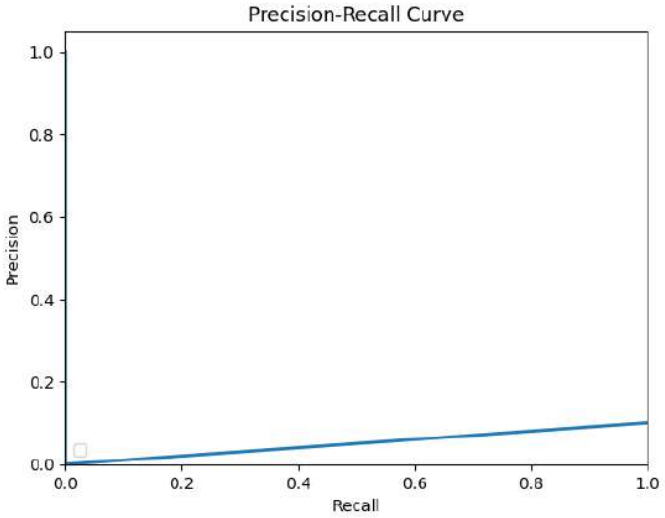


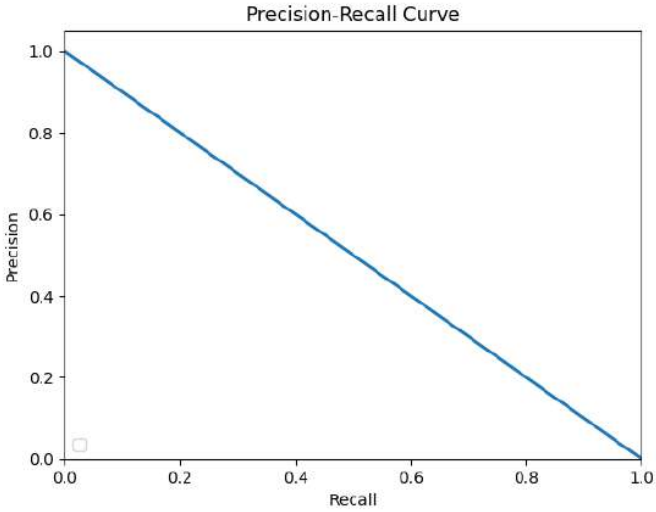
2. Class 3 and Class 4



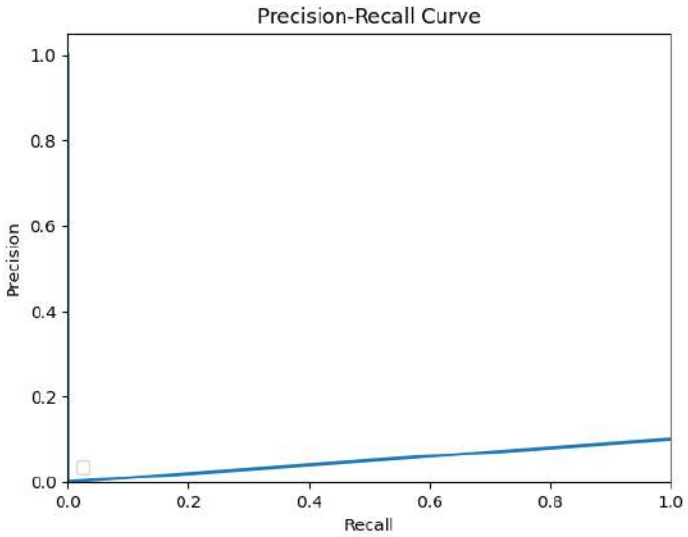


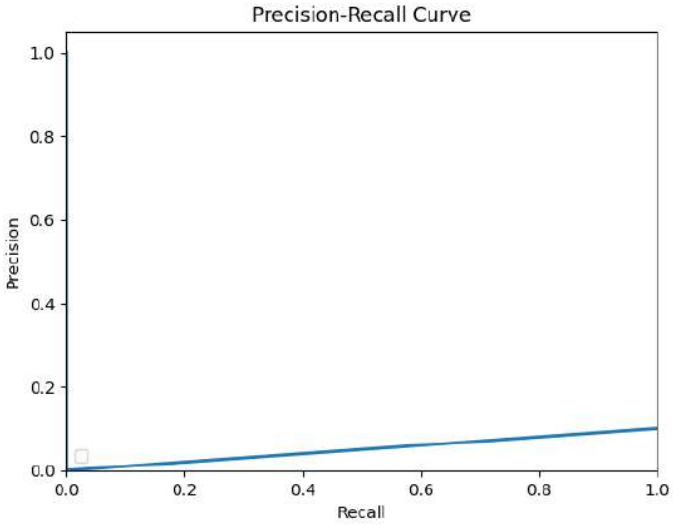
3. Class 5 and Class 6



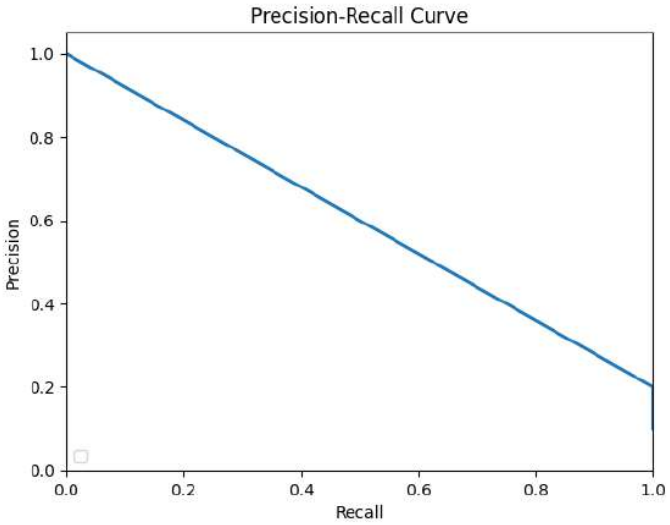


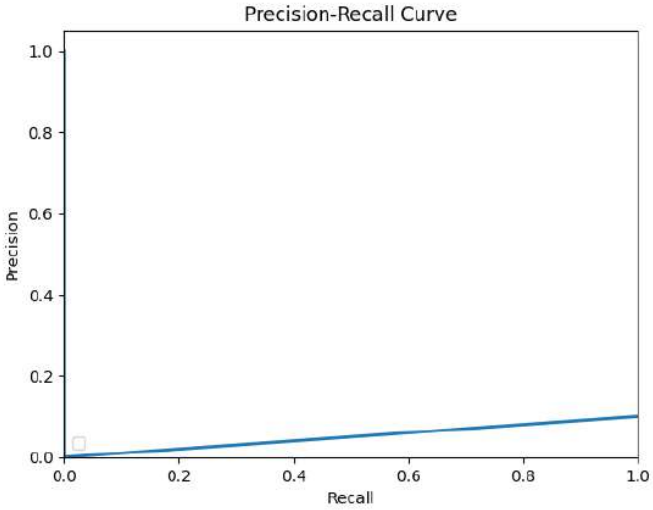
4. Class 7 and Class 8





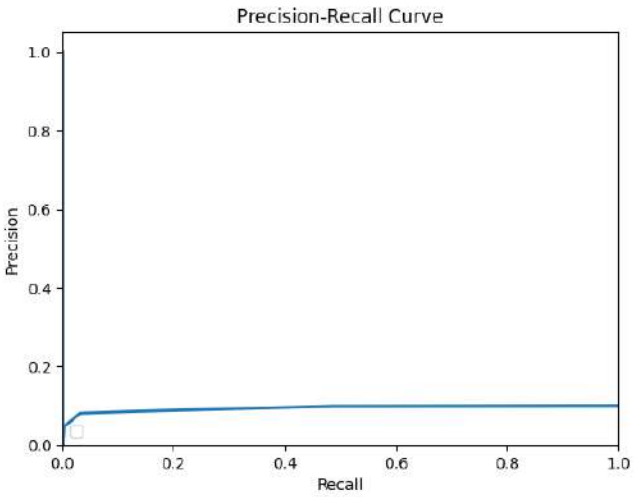
5. Class 9 and Class 10

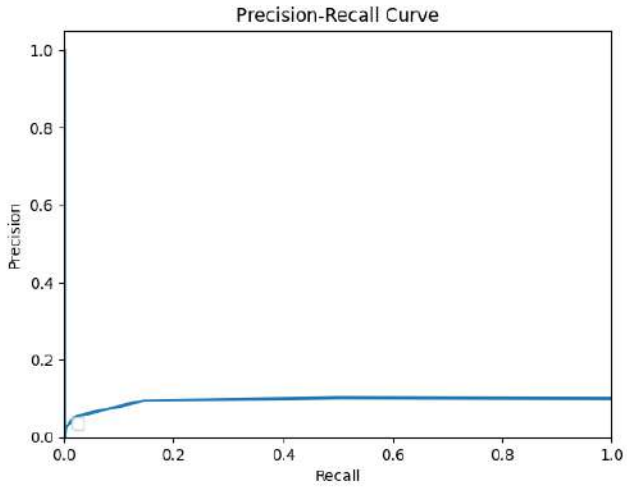




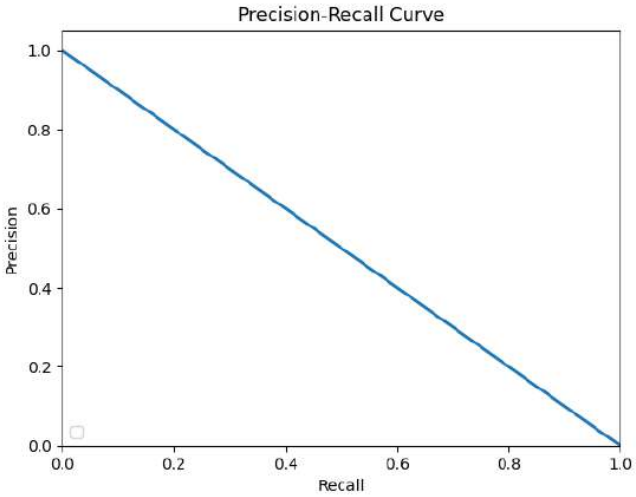
Following are the Precision-Recall Curves for SIFT feature descriptors:-

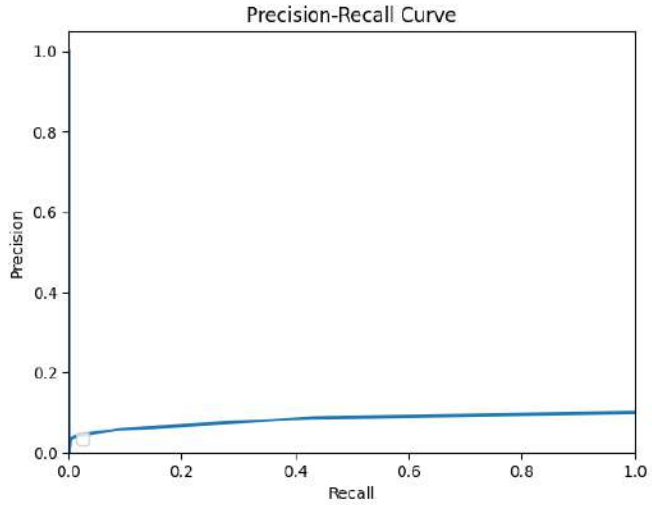
1. Class 1 and Class 2



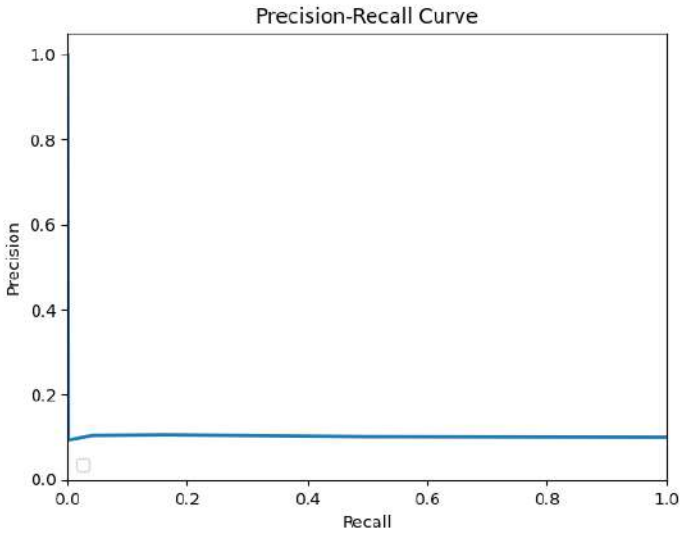


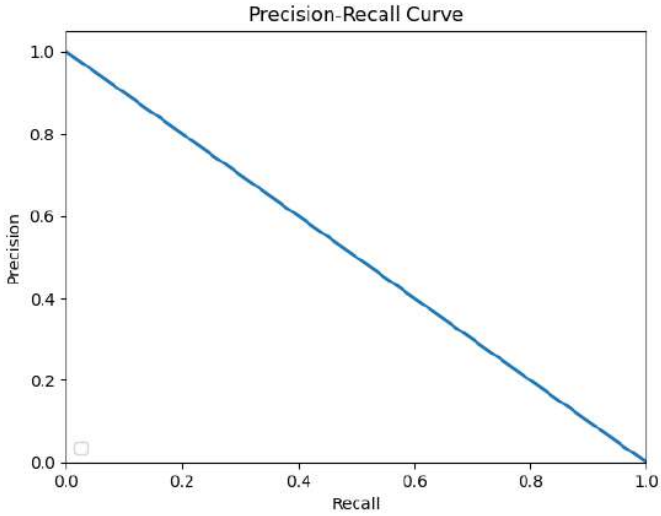
2. Class 3 and Class 4



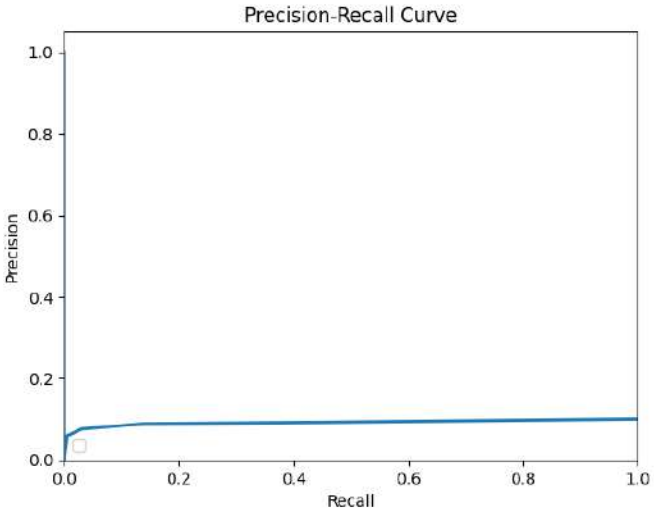


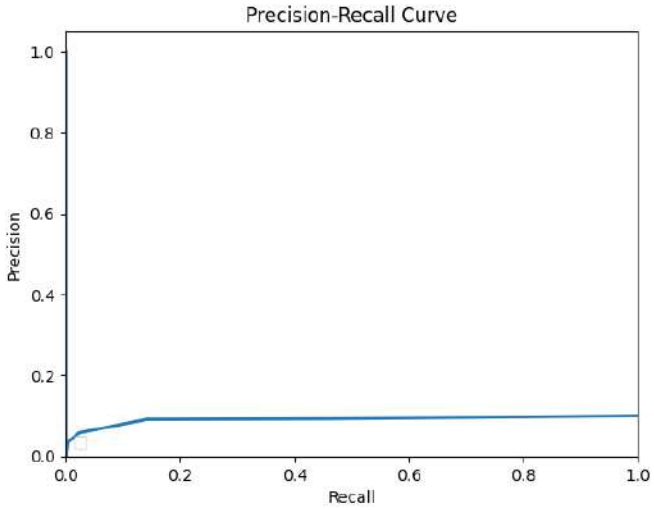
3. Class 5 and Class 6



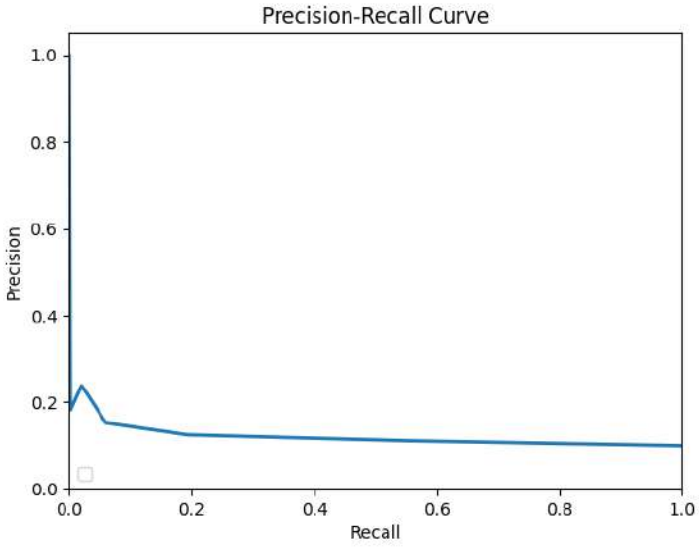


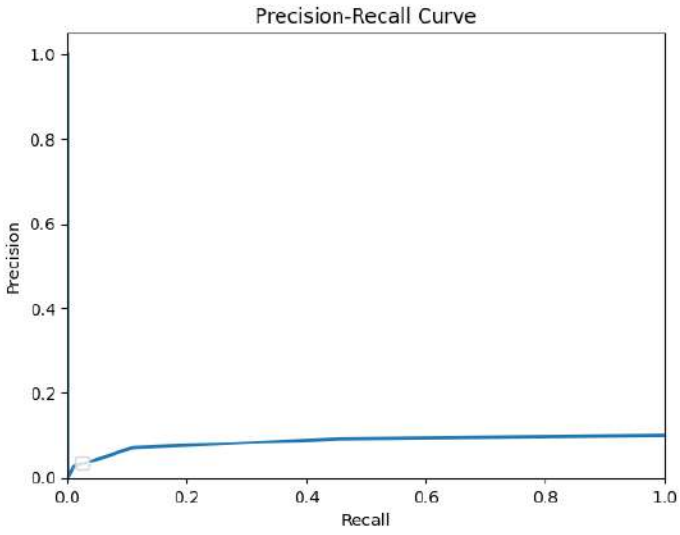
4. Class 7 and Class 8





5. Class 9 and Class 10





Limitations and Scope of Improvements

- SIFT is a computationally expensive feature descriptor and may not be suitable for real-time applications.
- The performance of the classifier heavily depends on the quality and quantity of the training data.
- The hyperparameters such as the number of clusters and SVM regularization parameter need to be carefully tuned for optimal performance.
- Other feature descriptors such as SURF, ORB, and HOG can be used as alternatives to SIFT for faster and more robust feature extraction.
- More advanced techniques such as deep learning can be used for feature extraction and classification to achieve state-of-the-art performance.

Q.3 Write down Viola Jones's face detection steps in detail.

Solution:- The Viola-Jones algorithm is a machine learning approach for face detection that uses a cascade of classifiers to identify facial features in an image. It was introduced in 2001 by Paul Viola and Michael Jones and has since become a popular algorithm for real-time face detection in images and videos.

Step 1: Haar-like features

(a) The algorithm uses Haar-like features to detect faces. These features are rectangular patterns that capture differences in pixel intensities in the image.
(b) The features are defined as the difference between the sum of pixel intensities in a white rectangle and the sum of pixel intensities in a black rectangle of the same size and shape.
(c) The basic Haar-like features are:
    (i) Two-Rectangle Feature: Two adjacent rectangles with opposite intensities.
    (ii) Three-Rectangle Feature: Three adjacent rectangles with the central rectangle having opposite intensity.
    (iii) Four-Rectangle Feature: Four adjacent rectangles with two opposite pairs of rectangles.
(d) Haar-like features are computed efficiently using the integral image.

Step 2: Integral images

(a) To efficiently compute Haar-like features, the algorithm uses integral images.
(b) An integral image is a 2D matrix where each pixel contains the sum of all pixels above and to the left of it in the original image.
(c) Integral images can be computed in O(n) time, where n is the number of pixels in the original image.
(d) It can be calculated efficiently in one pass using dynamic programming.
(e) The integral image is denoted as II(x,y).
(f) The value of II(x,y) represents the sum of all pixels in the rectangle with the top-left corner at (0,0) and bottom-right corner at (x,y).
(g) The integral image is calculated as follows:
(h) II(x,y) = II(x,y-1) + II(x-1,y) - II(x-1,y-1) + I(x,y), where I(x,y) represents the intensity value of the original image at (x,y).

Step 3: AdaBoost training

(a) The Viola-Jones algorithm uses a variant of the AdaBoost algorithm to train a set of weak classifiers that can identify facial features.
(b) AdaBoost is a boosting algorithm that combines many weak classifiers into a strong classifier.

(c) During training, the algorithm selects the best Haar-like feature and threshold for each weak classifier that minimizes the classification error.

Step 4: Cascade of classifiers

(a) Once the weak classifiers are trained, the algorithm combines them into a cascade of classifiers.
(b) Each cascade consists of several stages, where each stage contains a subset of the weak classifiers.
(c) At each stage, the algorithm evaluates the image using the weak classifiers in that stage. If the image passes all the weak classifiers in a stage, it is considered a face and the algorithm moves on to the next stage. Otherwise, it is rejected as a non-face and the algorithm stops.

Step 5: Non-maximum suppression

(a) To remove false positives and overlapping detections, the algorithm applies non-maximum suppression to the output of the cascade.
(b) Non-maximum suppression removes overlapping bounding boxes by selecting the box with the highest confidence score and discarding the rest.

Limitations:- The Viola-Jones algorithm has some limitations, including:

(a) It can struggle with detecting faces in images with complex backgrounds or lighting conditions.
(b) It may produce false positives or miss some faces.
(c) It may be computationally expensive to evaluate many weak classifiers on a large number of image windows.
(d) These limitations can be addressed by using more sophisticated feature representations, such as deep learning models, or by optimizing the algorithm for specific use cases.

Scope of improvements:- Some improvements to the Viola-Jones algorithm include:

(a) Using more advanced feature representations, such as Haar wavelets or local binary patterns.
(b) Optimizing the algorithm for specific use cases, such as face recognition in low-light environments or with occlusions.
(c) Combining the algorithm with other techniques, such as deep learning models or domain-specific knowledge, to improve performance.

Q.4 You are given a few deer train images with this assignment. Manually crop them to find out tight bounding boxes for Deer and also obtain some non-deer image patches of different sizes and aspect ratios. Compute HOG features for deer and non- deer image patches and build an SVM classifier to classify deer vs non-deer. Now, implement a sliding window object detection to

find out deer in the test images. Write down each step in the report. Also, objectively evaluate your detection performance.

Solution:-

1. Data collection and preprocessing

    a. Collect the RGB images of deer and non-deer from various sources.
    b. Introduce random augmentations
    c. Create positive and negative training samples by extracting the region of interest (ROI) containing deer and non-deer objects respectively.

2. Feature Extraction using HOG

    a. Extract HOG features from the training samples.
    b. Define the HOG parameters such as the number of orientations, cell size, block size, etc.
    c. Convert the images to grayscale and compute the HOG descriptors.

3. Train the SVM classifier

    a. Train the SVM classifier using the extracted HOG features from the positive and negative samples.
    b. Split the dataset into training and validation sets.

4. Implement sliding window algorithm

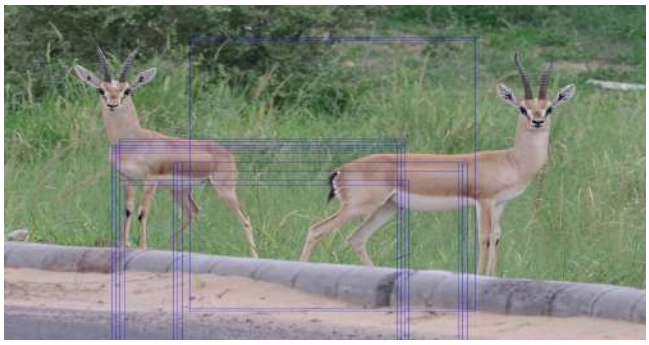
    a. Define a sliding window of fixed size and slide it across the test image.
    b. At each window location, extract the HOG features and classify the window using the trained SVM classifier.

Following are the results of the above pipelined implementation;-

- 1st Image with 1/3rd Image Size for Sliding Window (formed near grass)

- 2nd Image with 3/5th Image Size for Sliding Window (formed at left side of deer)

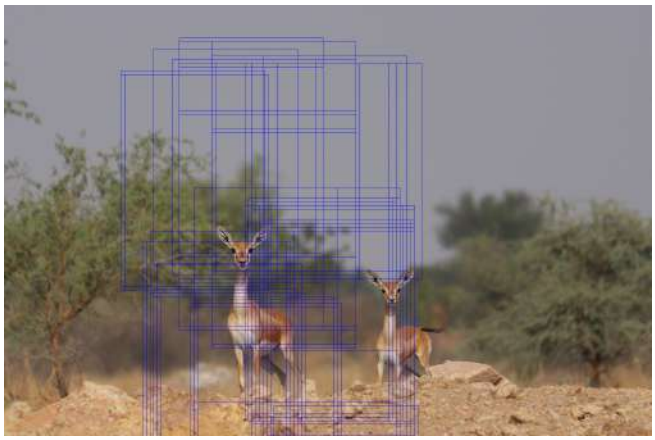
- 3rd Image with 3/5th Image Size for Sliding Window (formed at middle of deers)



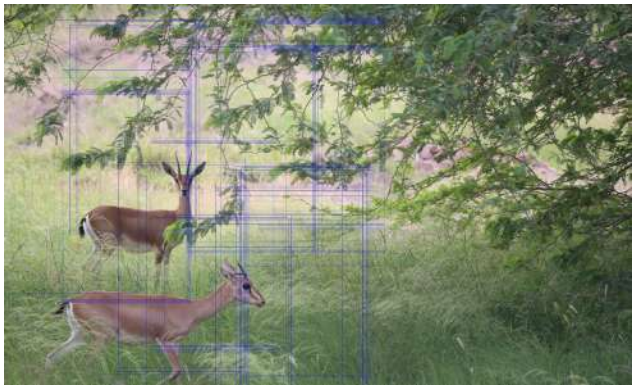
- 4th Image with 60%, 50% and 40% Image Sizes for Sliding Window (not formed)

- 5th Image with 1/5th Image Size for Sliding Window (some formed at deers too)

- 6th Image with 1/3rd Image Size for Sliding Window (majorily formed at deers)



- 7th Image with 1/3rd Image Size for Sliding Window (majorily formed at deers)



- 8th Image with any of the Image Sizes for Sliding Window (not formed anywhere)

5. Limitations:

    a. The algorithm is dependent on the quality of the training dataset.
    b. It may not perform well in complex backgrounds or in cases where the deer object is partially occluded.
    c. The algorithm may produce false positive or false negative results depending on the choice of HOG parameters and the threshold values.

6. Scope of improvements:

    a. Use more advanced object detection techniques such as Faster R-CNN or YOLO.
    b. Implement data augmentation techniques to increase the size and diversity of the training dataset.
    c. Use ensemble methods such as bagging or boosting to improve the accuracy of the classifier.