CV Assignment-3 Report

Note: All the links for the code and data are provided at the end of the report.

Task: To give solutions and approaches to the given problems.

Objectives:

Question 1.

- **Eigen faces from scratch: 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.**

Procedure:

- This code imports the following libraries:
- cv2 which is OpenCV, a computer vision library
- pyplot from matplotlib, a library for visualizing data
- numpy for numerical operations
- os for interacting with the operating system
- Second code downloads an image of Mahendra Singh Dhoni, also known as MS Dhoni, a former Indian cricketer and captain of the Indian national team, from two different URLs.
- If the file "msd2.jpg" does not exist in the current directory, the code downloads the image from the second URL and saves it as "msd2.jpg" in the directory "data/face-lfw-train/"
- The code also downloads another image of MS Dhoni from the first URL and saves it as "msd.jpg", but this file is not used in the code beyond this point.
- Next Load the data from the data folder.
- Plotting the loaded images

- Combining all images into one vector for further computations.
- Compute the Mean image of all the above images.
- Result of mean image:

- Normalizing the images. And plotting them,

- calculating the covariance matrix of the normalised_images array using NumPy's cov function.
- Next is dividing the covariance matrix by the length of the images list.

- Next is printing the shape of the covariance matrix and its values.
- Calculating the eigenvalues and eigenvectors of the covariance matrix using NumPy's linalg.eig function.
- Sorting the pairs of eigenvectors and eigenvalues.
- Selecting top-K Eigenvectors.
- Plotting K Eigenfaces.

- Finding the weights for further calculations.
- Testing the Recognization, using eigenfaces.
- Loading test image, and computing the distances b/w each image.
- Results:

```python
# matching the test image with the training images
for i in range(len(w)):
    print(f"distance between {labels[i]} and unknown: {np.linalg.norm(w[i] - w_unknown)}")
# NOTE:- THE LOWER THE DISTANCE THE MORE SIMILAR THE IMAGES ARE
```

```
distance between Liu_Ye_0001 and unknown: 647474440.9646686
distance between Eric_Snow_0001 and unknown: 159772127.22479048
distance between Jackie_Chan_0003 and unknown: 359024338.0915352
distance between Jose_Luis_Rodriguez_Zapatero_0001 and unknown: 449932270.9775375
distance between Priyanka_Chopra_0001 and unknown: 363488543.33091617
distance between Atal_Bihari_Vajpayee_0019 and unknown: 312821395.33617574
distance between msd2 and unknown: 0.0
distance between msd and unknown: 177719255.03471318
distance between Elena_Tihomirova_0001 and unknown: 518008318.7754563
distance between Prince_Harry_0001 and unknown: 413434262.2109754
distance between Edward_Belvin_0001 and unknown: 293150920.81556374
distance between Prince_Charles_0002 and unknown: 423085166.3757774
```

- Note:- for the same image the distance b/w images are 0, and also for similar images the distance is minimum.

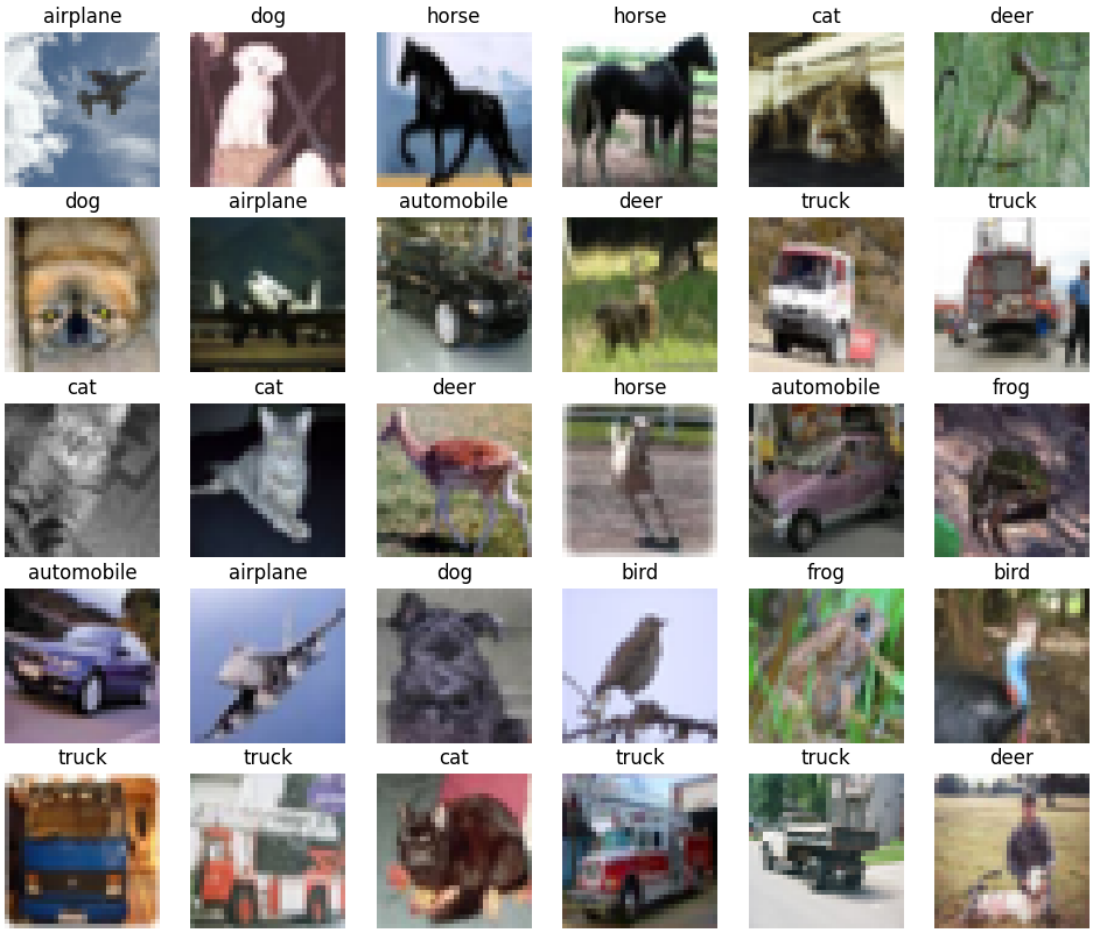
Objectives:

Question 2.

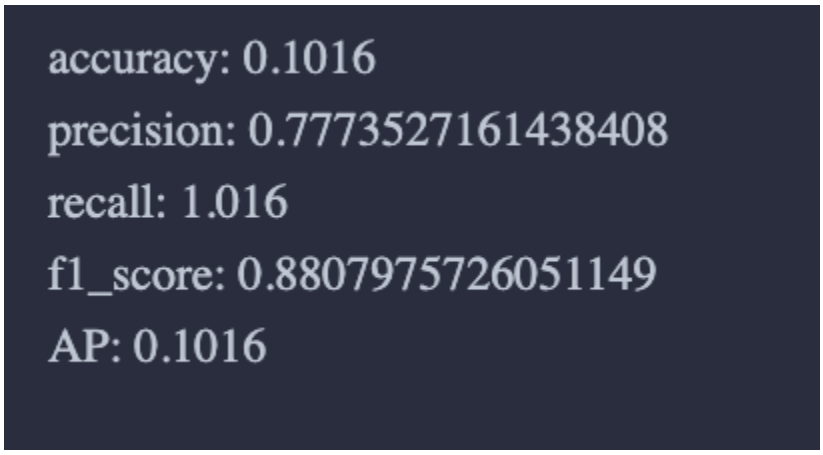
- **Develop an Image Search Engine for CIFAR-10 that takes the image as a query and retrieves top-5 similar images using Visual Bag of Words.**

Procedure:

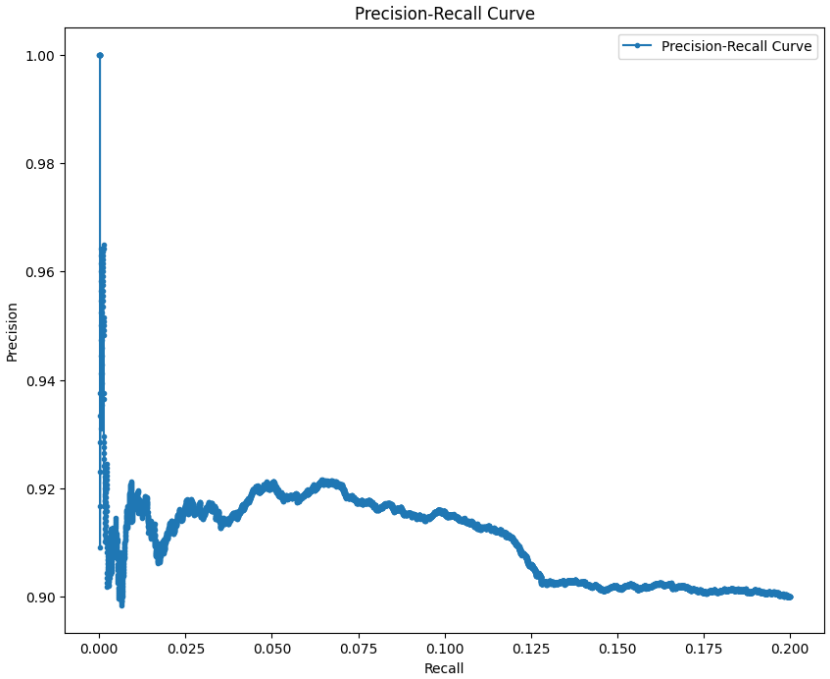
- This code imports the following libraries:
- cv2 which is OpenCV, a computer vision library
- pyplot from matplotlib, a library for visualizing data
- numpy for numerical operations
- os for interacting with the operating system
- Sklean for classification, and cluttering.
- Download the CIFAR-10 dataset from the following link: 'https://www.cs.toronto.edu/~kriz/cifar.html' and extract the dataset in the same folder as the code.
- Load the dataset with numpy load function.
- Combine all train batches to make one train dataset.
- Plotting random samples from the dataset.



- Applying Feature extraction using the SIFT_create method from open-cv
- Saving descriptor for the dataset.
- Computing histograms using descriptors from the above step i.e our vocabulary.
- Once the computed histogram, any type of classification algorithm can be used to identify the input image and get similar images.
- I've used Linear-SVM, giving below results.



- Note:- the performance of the Model is not good, but for our use, we want to find out similar images not classify the image hence the model can be used.
- Precision-Recall curve:

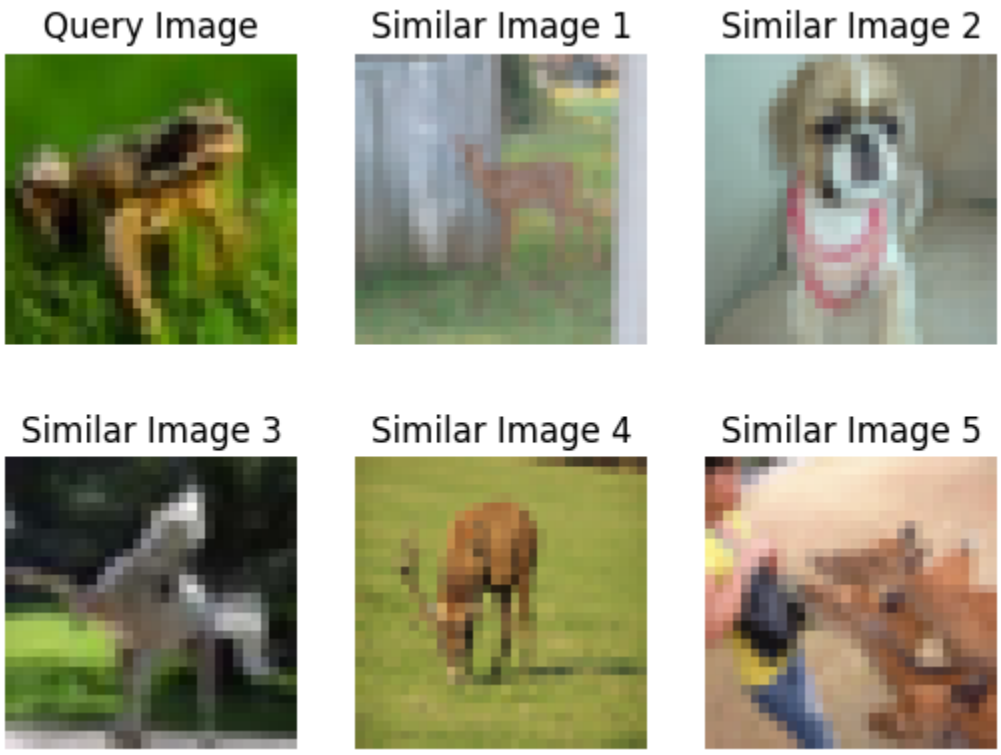


- At last testing the model will give results:-

```
print(f"Predicted class: {id_to_label[y_pred[0]]}")
print(f"Actual class: {id_to_label[test_img_label]}")
```

```
Predicted class: frog
Actual class: frog
```

Fig: Showing the prediction for the query image.



- Note:- the query Image has a green background with the frog on it, and all the other images have either a green component or a “frog-like” component in it.

Objectives:

Question 3.

- **Viola Jones Face detection: Write down Viola Jones's face detection steps in detail.**

Procedure:

- Computation of the Integral Image: The procedure begins by computing the integral image representation of the input image, which entails computing the sum of all pixel intensities for each potential rectangular region in the image. This can be calculated effectively using a method known as dynamic programming.

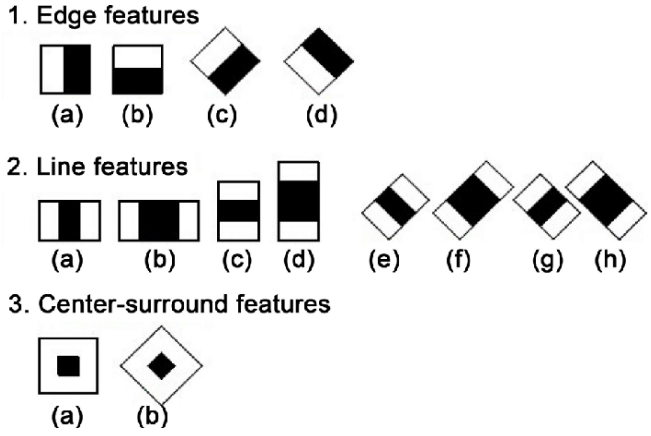
1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

input image

0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

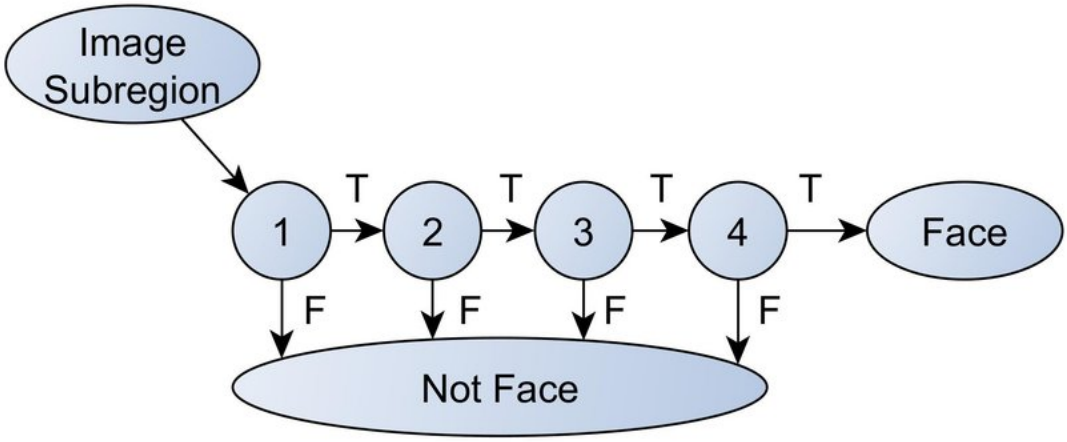
integral image

- Haar-like Feature Selection: The algorithm selects a set of Haar-like features to use for detecting faces. These features are rectangular patterns of pixels that differ in brightness or color, and are chosen based on their ability to discriminate between faces and non-faces.

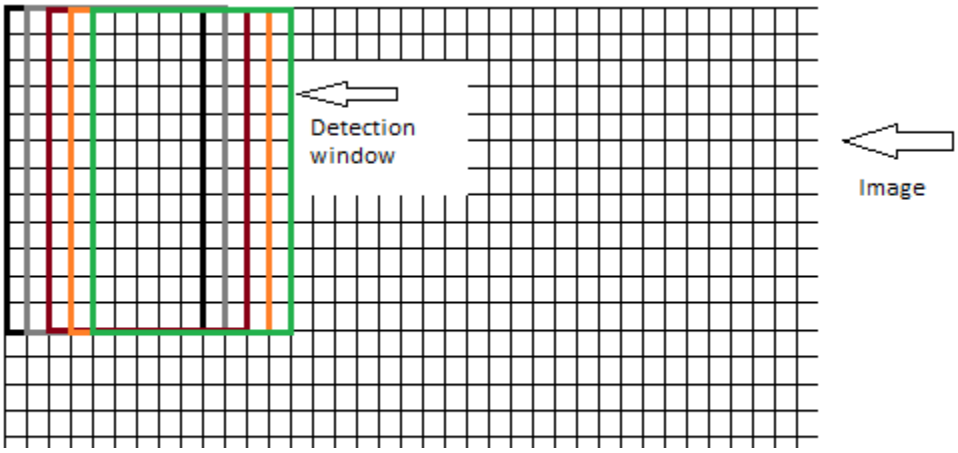


- Adaboost Training: The algorithm uses Adaboost, a machine learning algorithm, to select the most discriminative Haar-like features and to train a strong classifier. Adaboost works by iteratively training weak classifiers and adding them to the strong classifier based on their classification accuracy.
- Cascading Classifiers: The algorithm uses a cascading classifier to improve performance and reduce false positives. The cascading classifier consists of multiple stages, each with its own classifier. The image is first passed through the first stage, and if it passes, it

is passed on to the next stage, and so on. If the image fails any stage, it is immediately rejected, avoiding the need to compute all features for every window.



- Sliding Window: The algorithm applies the cascading classifier to every possible window in the image. A window is a rectangular region of the image, typically ranging in size from 24x24 to 80x80 pixels. The algorithm slides the window across the image, evaluating the classifier at each position.



- Non-maximum Suppression: Finally, the algorithm applies non-maximum suppression to remove overlapping detections and select the best candidate faces. This involves discarding detections that have low confidence scores or are too close to other detections.

Objectives:

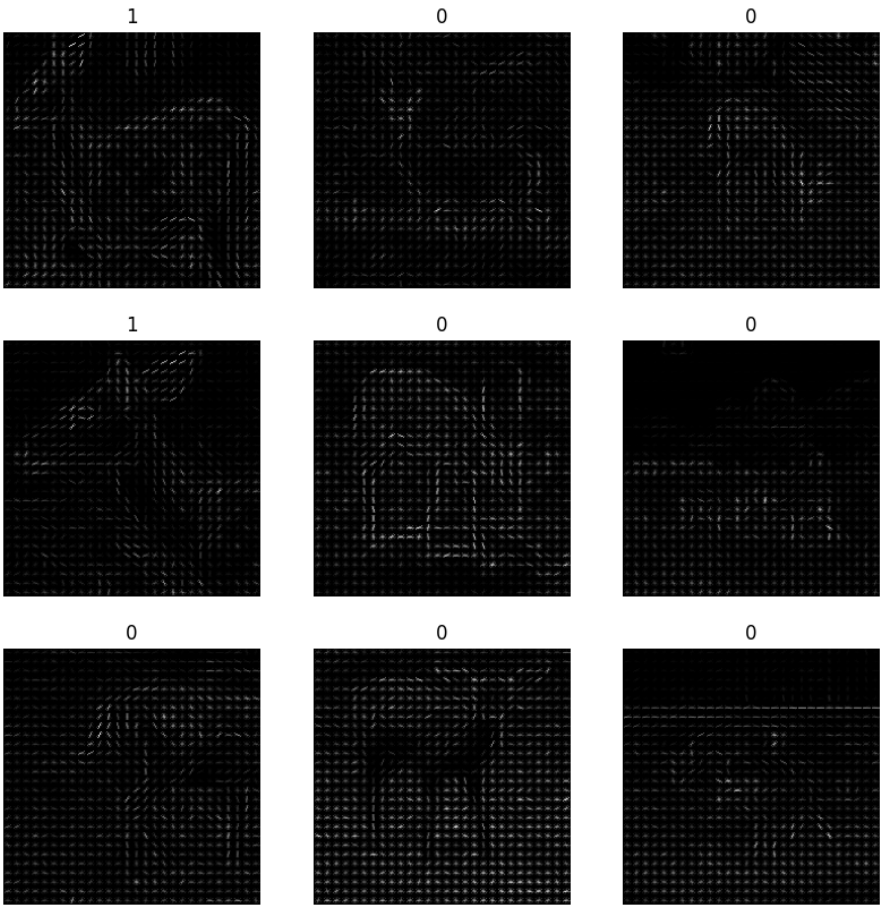
Question 4.

- **Sliding window object detection using HOG**
- **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.**

Procedure:

- The necessary libraries are imported, including OpenCV, NumPy, Matplotlib, scikit-image, and SVM from Scikit-learn.
- The training dataset is read and preprocessed. The images are resized to 224x224, stored in a list, and the corresponding labels are extracted from the file names. A dictionary is created to map the labels to numeric values.
- The code also includes a commented-out section for random cropping of the images to generate more training data.
- Nine random images from the training dataset are plotted, along with their labels.

- HOG descriptors are computed for each image in the training dataset. A 9-orientation HOG descriptor is computed using 8x8 pixels per cell and 2x2 cells per block. Both the HOG descriptors and the corresponding hog images are stored in separate lists.



- Nine random hog images from the training dataset are plotted along with their labels.
- A Support Vector Machine (SVM) classifier with a radial basis function (RBF) kernel is created, with a regularization parameter C=1.0.
- The test dataset is read and preprocessed. The images are resized to 1024x1024, stored in a list.

Fig: random samples from test dataset

- Four random images from the test dataset are plotted.
- Sliding windows of size 224x224 are applied to the test images with a step size of 112 pixels. For each window, a HOG descriptor is computed, and the SVM classifier is

applied to determine if the window contains a deer or not. If a deer is detected, a bounding box is drawn around the window in the original image.

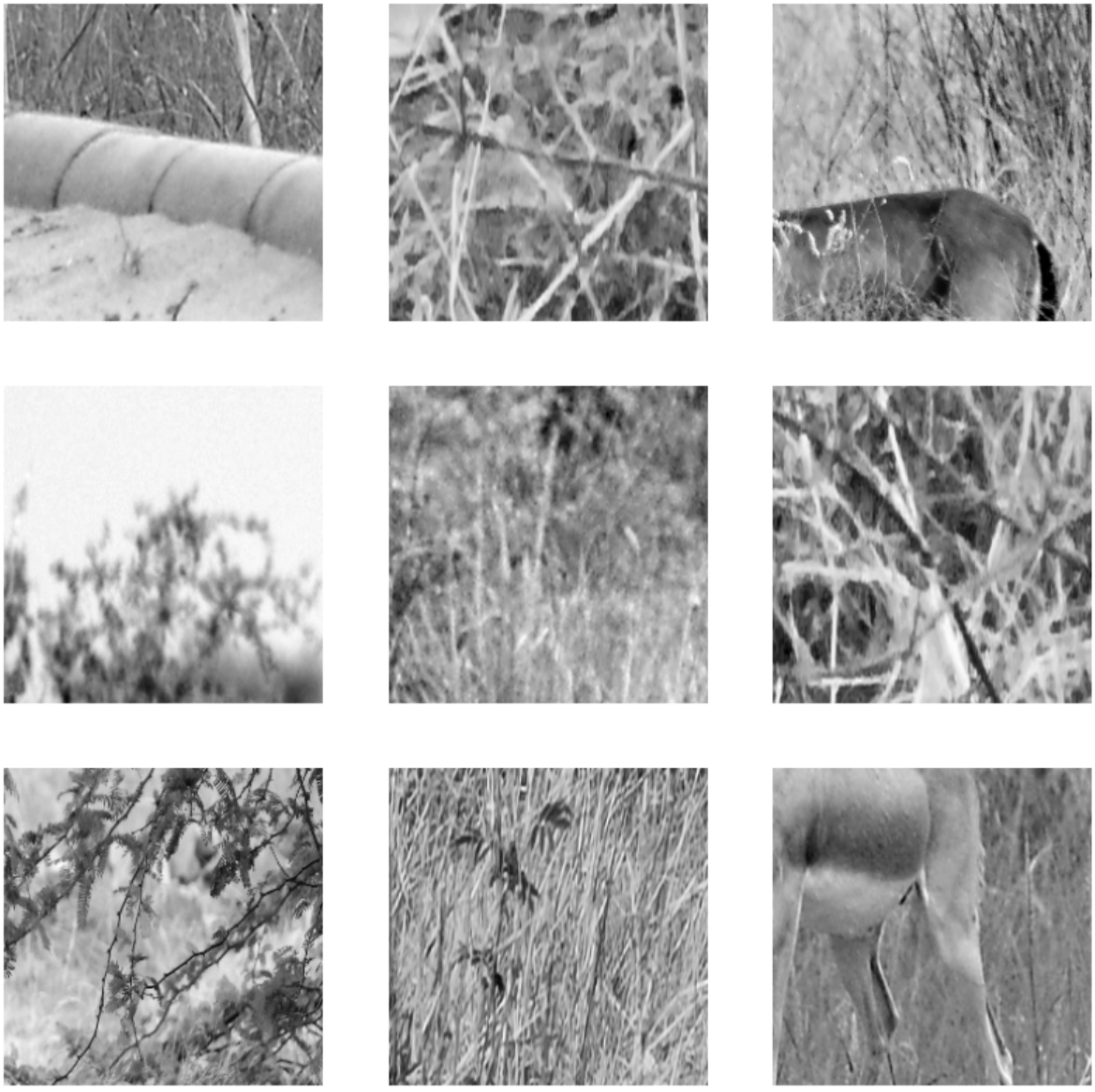


Fig: Patches of size 224,224 for sliding window

- The resulting images with bounding boxes are plotted for visualization.

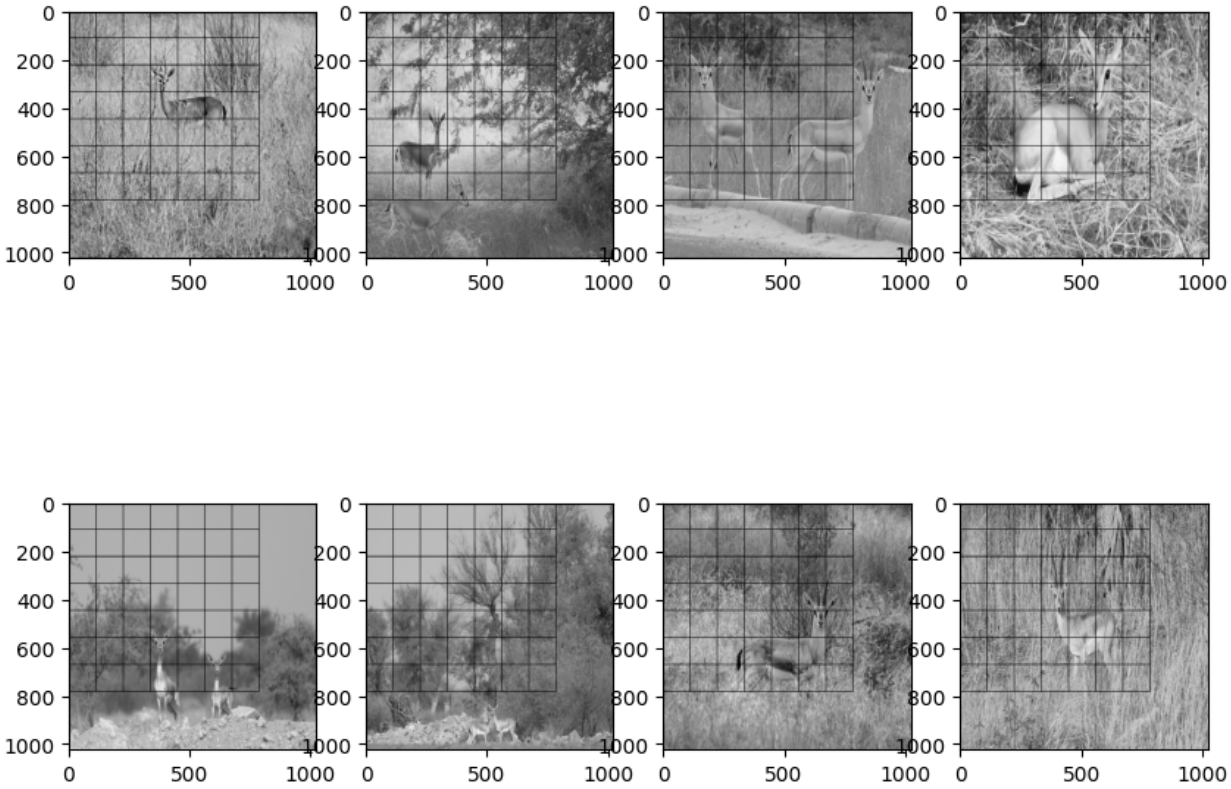


Fig: Bounding boxes on deer

References:

https://docs.opencv.org/4.x/
https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
https://numpy.org/doc/stable/reference/index.html
https://stackoverflow.com/
https://www.geeksforgeeks.org/opencv-python-tutorial/
https://scikit-learn.org/
https://scikit-learn.org/stable/modules/classes.html
https://scikit-learn.org/stable/tutorial/index.html
https://docs.python.org/3/library/os.html
https://docs.python.org/3/library/glob.html

Links:

ColabNotebooks + Data:- CV A3

Note: Use institute id only for accessing the above link.