COL780: Computer Vision

Assignment-3 Report

Bogam Sai Prabhath
2023AIB2079
Indian Institute of Technology Delhi
April 3, 2024

Task-1:

Human hand Detection using the HOG descriptors. implementing the human hand recognition task.

1.1 Image Preprocessing:

converted the image into grayscale to work with the 2-dimensional image also, The provided images may contain noise so Gaussian blur is used.

Gaussian blur

Gradient

1.2 Feature Extraction:

For the extraction of features the ROI coordinates are provided using this, the required image is extracted, and the further process is done as below.

1.2.1 gradient extraction:

The ROI is extracted, and the image is resized to a fixed dimension of (250x150) height=250, width=150, and the derivatives in the x direction and y direction are calculated by simple convolution with derivative kernels in both directions. the gradient angle is calculated using the Tan function . the magnitude is calculated using the derivatives.

$$gradient = arctan2(dy, dx)$$
$$magnitude = \sqrt{(dx^2 + dy^2)}$$

1.2.2 HOG feature extraction :

The gradient direction and magnitude is calculated using the derivatives, then the image (250x150) is divided into (4x4) sub-blocks and the histogram is calculated for each (4x4) sub-image. the histogram contains the 8 bins in which the 360° is divided by 45° each. the (4x4) subarray is traversed, and the histogram bin is done by adding up the magnitude value to which bin the pixel value goes by considering the gradient. Finally, the HOG features are generated, and the shape of this is (62x37x8) 8 depth of each position (i,j) representing the features of the (4x4) sub-image.

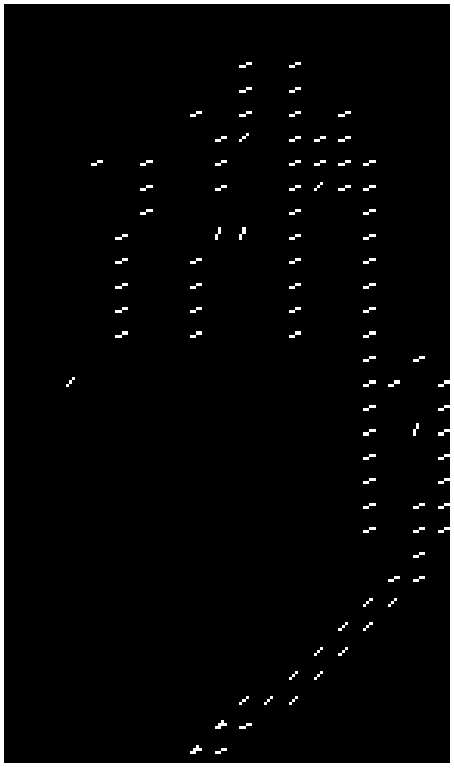
1.3 HOG visualization:

As mentioned above the sub-matrix is of shape (4x4) this is changed to (8x8) for the better visualization. For each sub-matrix of **(8x8),** the center is finded and the arrows are drawn in the direction of **Ɵ** by considering the respective magnitude **m** in the histogram bins.the histogram values are normalized in order to fit the arrow within the (8x8) bin.

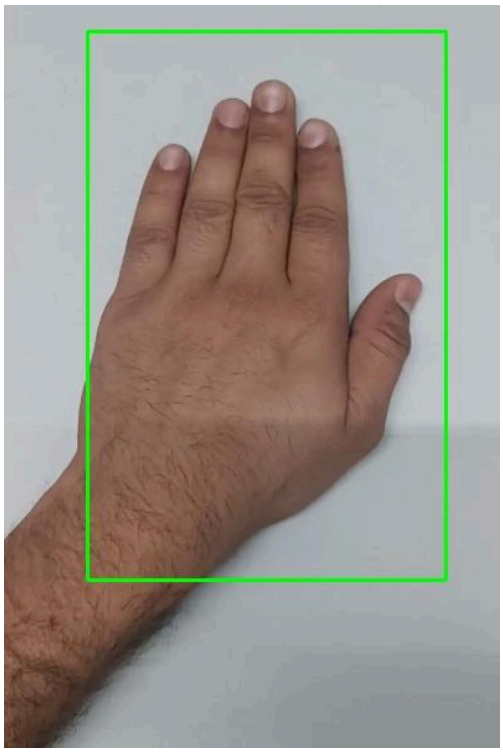
$$\Theta = (i * 22.5°) + 90°$$
$$(x1, y1) = ((i * 8) + 4, (j * 8) + 4)$$
$$(x2, y2) = (m * sin(\Theta), m * cos(\Theta))$$

The arrow is drawn from the two points the initial point is the center of the window and the final is as above, where as m is the normalized magnitude value.the thresholding is done on the m to remove the noise and highlight the high magnitude gradients i have chosen the threshold as 300.
The visualization of the images is as below

$$threshold = 300$$



HOG features



original image

1.3 Model Training:

The hand data is provided in order to train the svm for hand detection i have used the object's data of which contain non_hand data from the kaggle i,e COCO128 data set i have used the sliding window with 100 stride to generate the class 0 data. Using the ROI points given the hands is detected. From the positive class data and the negative class data the HOG features are generated and fed into the SVM inorder to train the classifier for hand detection.to feed each data point into SVM the HOG

features of each image is flattened into a 1d vector of shape (1X18352) i.e, the number of features in the each data point is in 18352 dimension space.we are learning the hyperparameters inorder to classify the data in this dimensional space.

$$Number\ of\ features = 18352$$
$$dimension\ of\ feature\ space = (1x\ 18352)$$

Model can be downloaded from this link

Task-2:

the next task is to recognize whether a detected hand is open or closed. For this task, we are required to implement a hand recognition algorithm using the detected hand regions and HOG features.

2.1 Data Preprocessing:

Same as in Task 1 the data is preprocessed by resizing and applying the Gaussian blur.

2.2 Data Augmentation:

The provided training data is biased toward the position of hand placed so, i have generated the extra data by data augmentation in which the original image is flipped vertically, and horizontally and rotated in clock and anti-clock directions to get extra 8 more images so the overall data trained is 8 times the original dataset which is provided.

2.3 HOG extraction :

As mentioned in task A the HOG features are extracted with the same feature space and SVM is trained with the extracted features of the open hand and the closed hand.

Task-3:
For this task i have **controlled the volume** of my pc using the trained SVM model.

3.1 interaction with environment:
For the volume control the primary camera of the pc is accessed and the frames are capture continuously from the camera.and used for the further process.

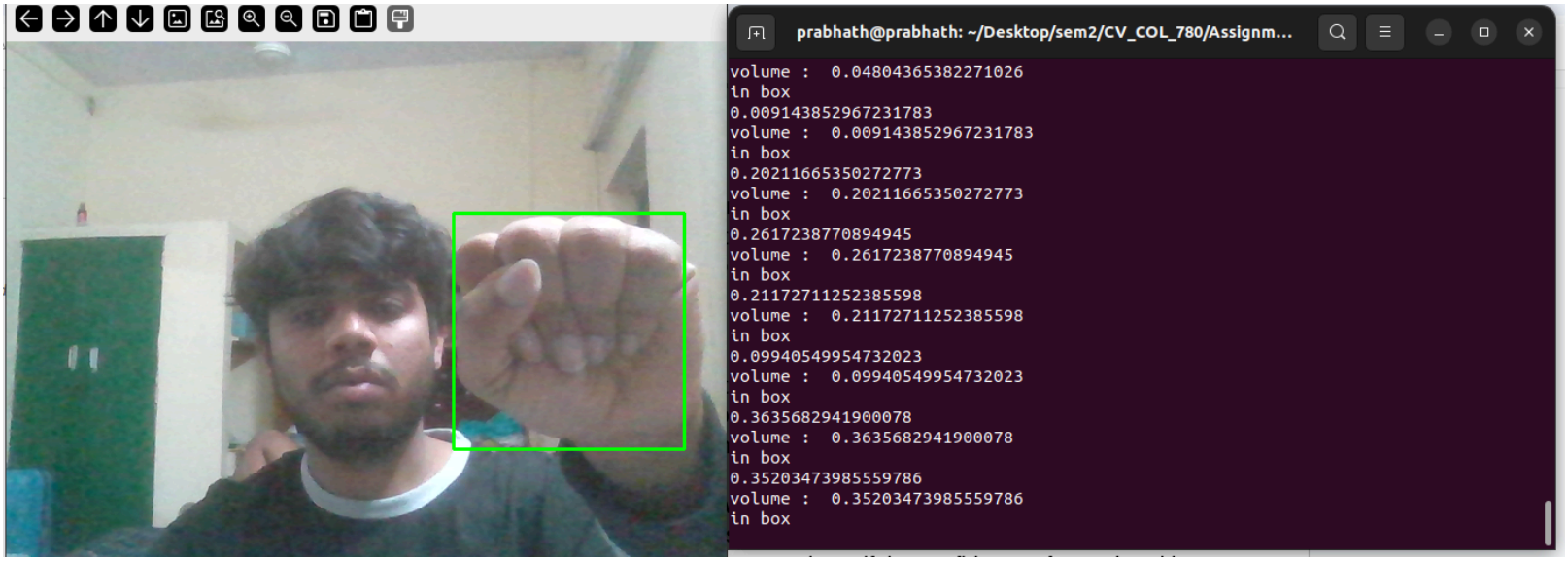
3.2 data extraction:
when ever the hand comes across the camera the ROI features which are the coordinates of the hand in frame are captured by the provided code and the same processing is done in order to generate the HOG features of the frame.

3.3 SVM confidence:
For the given input feature of the SVM it returns the confidence percentage for the belonging class which is in 0 to 1.i have used this confidence to control the volume.

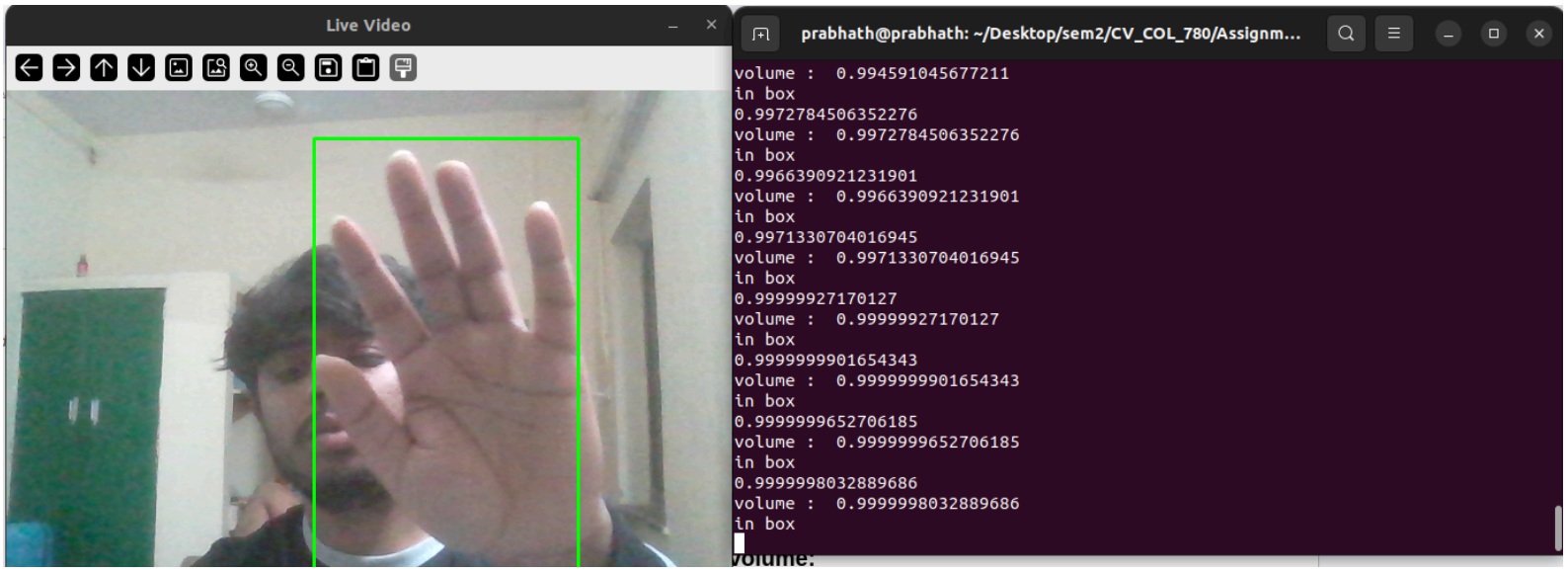
3.4 hand mapping to volume:
The open hand signifies the high volume i,e 100% and the closed hand signifies the low or 0% volume.if the confidence of open hand is inbetween then the volume is adjusted according to the confidence of the classifier.

3.5 volume control:
I have use the python library which can control the volume by interaction with the svm confidence percentage.

The demo images are as below the right terminal shows the volume



Closed hand decreasing the volume



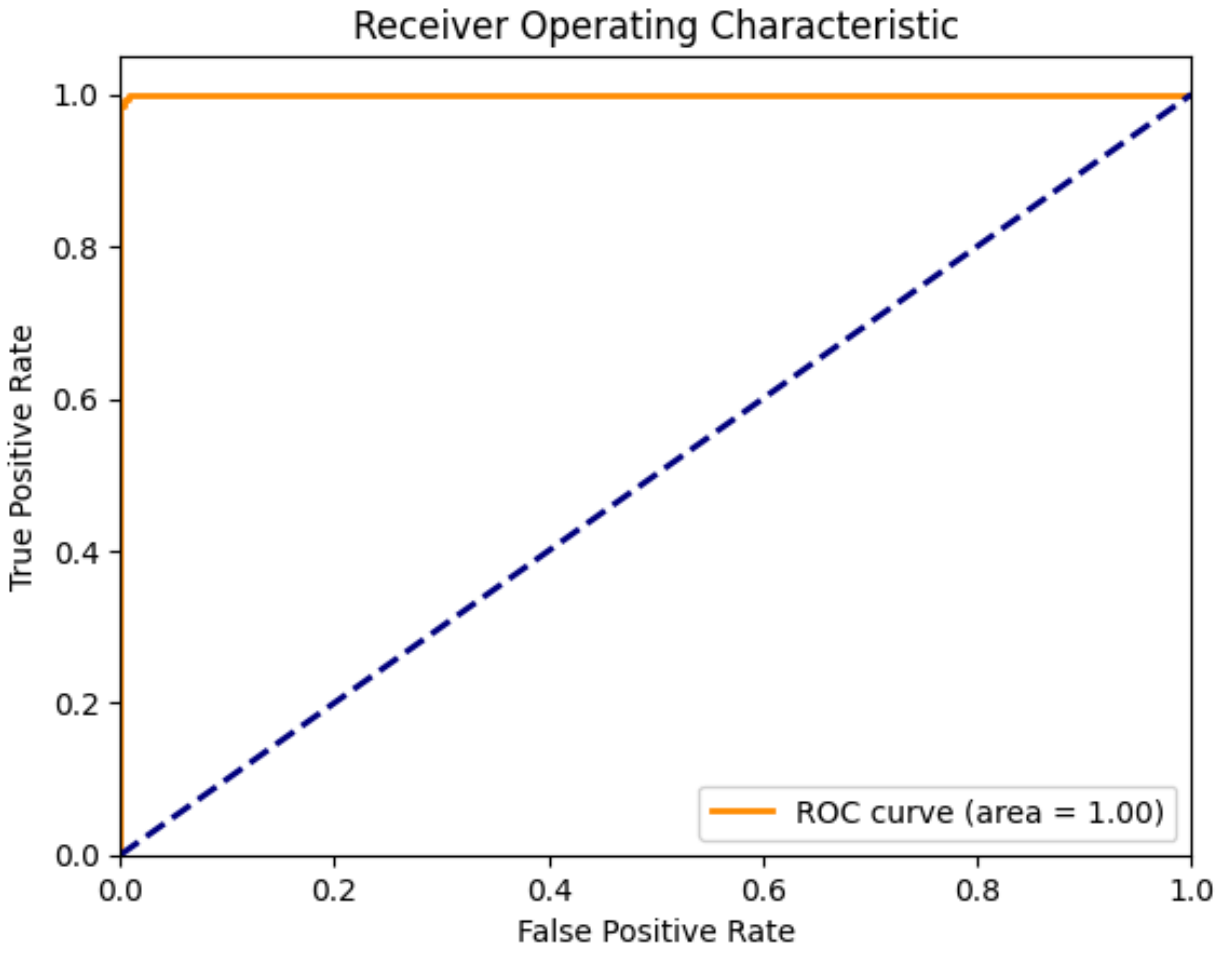
Open hand increasing the volume

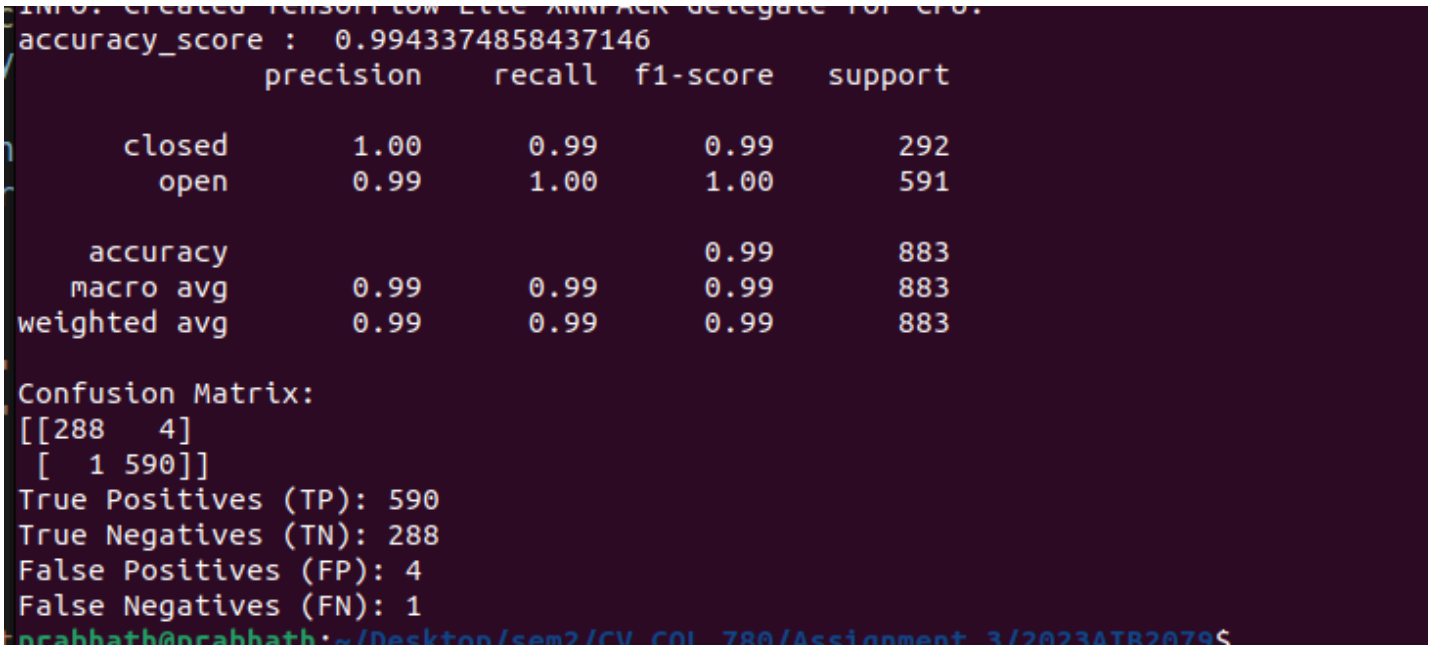
Results :
The precision, recall, and F1-score are calculated for the validation data the results are as below:

	precision	recall	f1-score
Open hand	1.00	0.99	0.99
Closed hand	0.99	1.00	1.00

Accuracy=0.99

AUROC curve:





Conclusion:

Accuracy=0.99
AUROC score=1.00
True Positive=590
True Negative=288
False Positive=4
False Negative=1

Issues and Challenges Faced:

1. As Task A is about hand detection valid data is not provided in order to train the SVM we need a negative class too so i took data from kaggle of objects that are non_hand to done Task A.
2. For training the SVM, the complete data needs to to loaded into the RAM for images with high pixels it is too complex to train the model as SVM can not be trained in batches.i have reduced the image size too small in order to fit into ram and also boost the computation time.
3. Also wore code to randomly take images from the train folder with some given ratio say 0.2 20% of images are only taken to train the model with again reduces the time.

Tools and Technology Used:

1. **Numpy** for array manipulation
2. **Cv2** fror image read and write and for convolution.
3. **Matplotlib** for plotting curves
4. **sklearn** for svm model and metrics calculation
5. **Joblib** to save the model
6. **Pulsectl** to access and control the system volume
7. **os** to read path
8. **csv** to generate the csv file

References:

1. https://link.springer.com/content/pdf/10.1007/978-3-642-24403-2_15.pdf