COL 780: Computer Vision
Assignment 3
Report

1. Hog Feature Extraction

Step 1: Finding magnitude and direction of the gradient
The compute_gradient function calculates the gradient of an image. It first computes the x and y gradients using the Sobel filter. It then calculates the magnitude of the gradient, which gives the intensity of the change in brightness in the image. It also calculates the orientation of the gradient, which gives the direction of the change in brightness. These two pieces of information, magnitude and orientation, are crucial in many image processing tasks as they provide a simple way to represent the edge information in an image."

Step 2: Finding the histogram
The compute_histogram function calculates a histogram of gradient orientations. It first determines which bin each gradient orientation belongs to and computes the weights for interpolation. It then iterates over each bin and adds the magnitude of the gradients that belong to that bin to the histogram, using interpolation to distribute the magnitude between the current bin and the next bin. This histogram represents the distribution of edge orientations in the image.

Step 3: Block normalization
The normalize_block function normalizes a block of histograms. It concatenates all the histograms in the block into a single 1D array and computes its norm. If the norm is greater than 0, it normalizes the block by dividing it by its norm. This normalization step reduces the effect of illumination and contrast changes. If the norm is 0, it returns the block as it is because a zero vector cannot be normalized. It helps in making the features more robust to changes in illumination and contrast.

Step 4: Finding hog vector
The compute_hog_features function calculates the Histogram of Oriented Gradients (HOG) features of an image. It first computes the gradient magnitude and orientation of the image. It then divides the image into cells and computes a histogram of oriented gradients for each cell. It normalizes these histograms in overlapping blocks to account for changes in illumination and contrast. The normalized histograms are then concatenated into a single 1D array, which represents the HOG features of the image. These features are a crucial component in many computer vision tasks as they capture the shape information present in the image.

2. Extracting ROI

I have used the MediaPipe Hands solution to detect hands in an image and extracting the region of interest (ROI) around the detected hands. It first initializes the MediaPipe Hands object with specific parameters. It then defines a function to increase the size of a bounding box while ensuring it remains within the image boundaries. Further it converts the input image to RGB, detects hand landmarks in the image, computes the bounding box around the hand landmarks, increases the size of the bounding box, and extracts the ROI from the image using the bounding box coordinates.

Step 1: Initializing parameters
The mpHands.Hands object is initialized with specific parameters. static_image_mode=True means the solution treats the input images as a batch of static, possibly unrelated, images. max_num_hands=2 is the maximum number of hands to detect. min_detection_confidence=0.5 and min_tracking_confidence=0.5 are the minimum confidence values for the detection and tracking models respectively.

Step 2: Bounding box
The increase_bbox function takes a bounding box (bbox), a scale factor (scale_factor), and the shape of the image (img_shape) as inputs. It increases the size of the bounding box by the scale factor while ensuring that the bounding box remains within the image boundaries.

Step 3: Extracting ROI
The extract_roi_from_image function takes an image (img) and a scale factor (scale_factor) as inputs. It converts the image to RGB, processes the image using the hands object to detect hand landmarks, and if any hand landmarks are detected, it computes the bounding box around the hand landmarks, increases the size of the bounding box using the increase_bbox function, and extracts the ROI from the image using the bounding box coordinates.

3. Training Model

The training_model implies that training a Support Vector Machine (SVM) classifier on a set of training data and evaluates it on a set of testing data. It first computes the Histogram of Oriented Gradients (HOG) features of the training and testing sets. It then trains the SVM classifier on the training data and uses it to predict the classes of the testing data. It computes various performance metrics of the classifier, including the True Positive Rate (TPR), False Positive Rate (FPR), accuracy, precision, recall, F1-score, and the Area Under the Receiver Operating Characteristic (ROC) Curve (AUROC). It also saves the trained SVM model to a file and plots the ROC curve.

4. Results and Conclusion

Train Images:	Closed (1672), Open(3123)
Test Images:	Closed (256), Open(498)

True Positive Rate (TPR): 0.9607843137254902

False Positive Rate (FPR): 0.004918032786885246

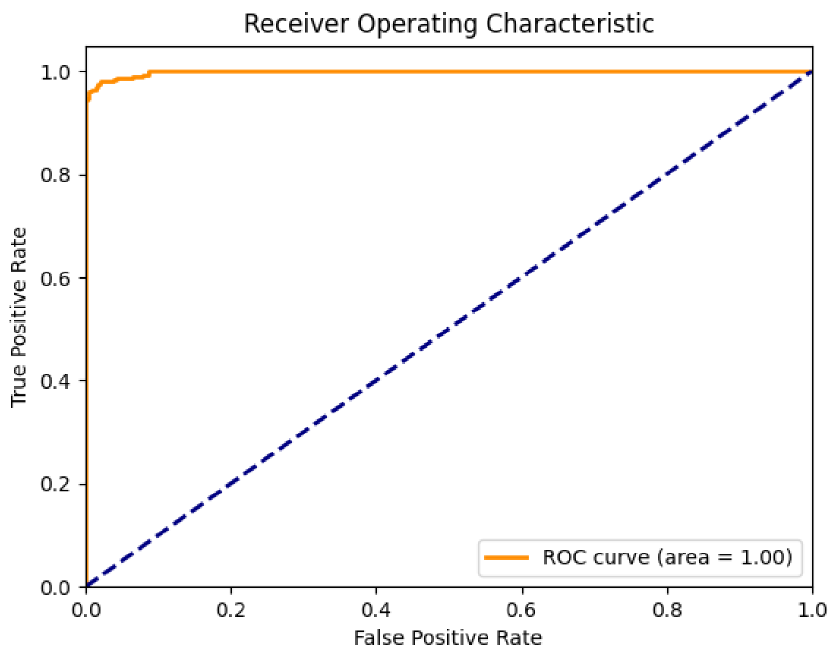
Accuracy: 0.9836244541484717

```
              precision   recall  f1-score   support

           0     0.98      1.00     0.99      610
           1     0.99      0.96     0.98      306

    Accuracy                        0.98      916
   macro avg     0.99      0.98     0.98      916
weighted avg     0.98      0.98     0.98      916
```

AUROC: 0.9983445837351334



5. Issues and Challenges during Training the Model

a) Limited Dataset:

- Training such algorithms requires a substantial amount of diverse data to ensure robustness.
- Due to constraints on available data, the model have not achieved optimal performance, lacking the necessary variability for effective training.

b) Variability in Hand Gestures:

- Recognizing accurate hand gestures necessitates a wide range of hand shapes and sizes, which can vary significantly among individuals.
- Limited diversity in the dataset had hindered the model's ability to generalize across different hand types, leading to potential inaccuracies in gesture recognition.

c) Computational Complexity of HOG Feature Extraction:

- Implementation of the HOG feature extractor from scratch had introduced computational overhead.
- The process of computing the HOG vector for each image had consumed considerable time, impacting the overall efficiency of the algorithm.

d) Static Variables and Adaptability:

- Certain variables within the algorithm are hardcoded for specific purposes, which limits their adaptability to varying scenarios.
- Enhancing the algorithm's flexibility by making these variables dynamic would enable it to adapt more effectively to different contexts and environments.

e) Training Time of Support Vector Machine (SVM):

- The size of the HOG vector influences the duration required for training the SVM model.
- Larger HOG vectors may necessitate extended training times, potentially affecting the algorithm's responsiveness and usability.

6. Application of above algorithm(Bonus Part)

Music Control System

Requirement: Music Player should be running in Background.

At this point in time, I've been exploring the utilization of the Histogram of Oriented Gradients (HOG) feature for the purpose of recognizing hand gestures. Essentially, the HOG feature extraction technique provides us with a set of features that characterize both open and closed hand gestures. Once we have these features, we pass them through a predictor model, which is essentially a machine learning algorithm trained to classify input data into predefined categories. In our case, the predictor model categorizes these features into either open or closed hand gestures.

Presently, I've been delving into controlling an active music system using this approach. Upon capturing hand gestures through the HOG feature and subsequent classification by the predictor model, the system responds accordingly to manage the music playback.

Specifically, when an open hand gesture is recognized, the system initiates a pause command, halting the playback of the current song. Conversely, upon detecting a closed hand gesture, it triggers the next song in the playlist, seamlessly transitioning to the subsequent track for continued enjoyment.

This integration of hand gesture recognition into music playback control adds an interactive dimension to the user experience, allowing for intuitive and hands-free management of the music listening session. It exemplifies the potential of combining computer vision techniques with real-world applications to enhance user interaction and convenience.

Type 1: Webcam Input
When opting for webcam input, the system activates the webcam to capture an image of the hand gesture. After execution of the algorithm, the webcam interface is launched, allowing the system to wait for 4 seconds before capturing the image. Once the image is captured, it undergoes preprocessing and analysis to determine the appropriate action based on the recognized hand gesture.

Type 2: Direct Image Input
Alternatively, users have the option to directly input an image containing the hand gesture. In this scenario, the image is provided as input to the system without requiring interaction with a webcam. The algorithm processes the image directly, performing the necessary preprocessing steps before proceeding with the classification of the hand gesture and subsequent execution of the designated task.

These two input methods cater to different user preferences and scenarios, offering flexibility and convenience in utilizing the hand gesture recognition system. Whether capturing real-time gestures through a webcam or analyzing pre-existing images, the system adapts to meet the user's needs for seamless interaction and control.

The **challenges encountered** during the development and implementation of the hand gesture recognition system include:

a) Difficulty in Recognizing Hand Gestures in Complex Backgrounds:

- The performance of the webcam model is subpar due to challenges in distinguishing hand gestures against backgrounds containing distinct objects.
- The presence of clutter or varied backgrounds complicates the task of accurately identifying and isolating hand gestures, leading to reduced effectiveness of the recognition algorithm.

b) Limited Diversity in Hand Gesture Dataset:

- The model's training dataset lacked diversity in hand gestures, making it challenging to effectively categorize and recognize the various types of hand gestures.
- Variations in hand shapes, sizes, and orientations among different individuals were not adequately represented in the dataset, limiting the model's ability to generalize across diverse scenarios.

c) Latency Issues in Hand Gesture Recognition:

- The application experiences latency during the recognition and capturing of hand gestures, compromising its responsiveness and real-time performance.

- Delays in processing and analyzing input data contribute to the latency, potentially impacting the user experience and usability of the system.

7. Libraries Used:

a) Pickle: Python module used for serializing and deserializing Python objects. It allows objects to be saved to a file and loaded later.

b) Numpy: Fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

c) Scipy: An ecosystem of open-source software for mathematics, science, and engineering. It builds on top of Numpy and provides additional functionality for optimization, integration, interpolation, linear algebra, and more.

d) Sklearn: Scikit-learn is a simple and efficient tool for data mining and data analysis. It provides a wide range of supervised and unsupervised learning algorithms, as well as utilities for model selection and evaluation, data preprocessing, and more.

e) Matplotlib: A comprehensive library for creating static, interactive, and animated visualizations in Python. It provides a MATLAB-like interface and supports a wide variety of plots and charts.

f) Pyautogui: A cross-platform GUI automation library for Python. It allows you to control the mouse and keyboard to automate tasks on your computer, such as clicking, typing, and taking screenshots.

g) Mediapipe: A framework for building perception pipelines to process multimodal data, such as images, audio, and video. It provides pre-built components for tasks like pose estimation, hand tracking, object detection, and more.

References:

a) Youtube: https://www.youtube.com/watch?v=Ptc4dEnPwt8
b) Histograms of Oriented Gradients for Human Detection: Navneet Dalal and Bill Triggs