COL780 Assignment 3

Rishit Singla

April 2024

1 Introduction

The objective of this project report is to explore the application of Histogram of Oriented Gradients (HOG) descriptors in conjunction with Support Vector Machine (SVM) classification for the task of detecting hand gestures indicating open or closed states. HOG descriptors are employed to capture distinctive patterns of gradient orientations in localized image regions, providing an effective representation of hand gestures. Leveraging the discriminative power of SVMs, the system is trained to differentiate between open and closed hand gestures based on these features. The report presents the methodology, experimental setup, results, and analysis, demonstrating the effectiveness of the proposed approach in accurately recognizing hand gestures in real-world scenarios.

2 Tools and Technology Used

The project leverages various tools and technologies to accomplish the task of detecting hand gestures indicating open or closed states. The following libraries and frameworks were utilized:

- **OS and shutil Libraries:** These libraries were employed to create directories and check for existing paths, facilitating efficient management of project resources.
- **NumPy Library:** NumPy played a crucial role in handling arrays and data, providing efficient data manipulation and computation capabilities.
- **Pickle:** The Pickle module was utilized for storing and loading machine learning models, enabling easy serialization and deserialization of trained models.
- **OpenCV:** OpenCV was utilized for reading, writing, and processing images. It provided essential functionalities for image manipulation and processing tasks.
- **MediaPipe:** MediaPipe was employed for detecting hands in images, offering pre-trained models and pipelines for efficient hand detection.

- **Matplotlib:** Matplotlib was used for plotting and visualization purposes, enabling the generation of various plots and graphs to analyze results.
- **scikit-learn (sklearn):** scikit-learn was utilized for Support Vector Machine (SVM) classification and Receiver Operating Characteristic (ROC) curve generation, providing a comprehensive set of machine learning tools and algorithms for classification tasks.

These tools and technologies collectively facilitated the development of an effective system for detecting hand gestures using a combination of image processing, machine learning, and visualization techniques.

3 Brief Algorithm

3.1 Hand Detection and Dataset Creation

Initially, hand detection is performed on each image, resulting in the extraction of the detected hand as a separate image. In cases where an image contains multiple hands, individual hand images are retrieved, each corresponding to a detected hand in the image. Subsequently, each extracted hand image is resized to dimensions of (64, 128). These resized hand images are then saved to generate a new dataset containing single-hand images, each standardized to the same resolution.

3.2 Preprocessing

During preprocessing, every image undergoes conversion to grayscale, followed by the application of a Gaussian blur with a kernel size of 3.

3.3 HOG Descriptor

To create the HOG descriptor, we first compute the gradient magnitude and direction at each pixel using a Sobel kernel. Then, for each cell with a shape of (8,8), we perform voting among 9 bins to construct a histogram. With each block consisting of two rows and two columns of cells, we normalize the descriptor to ensure brightness invariance. Using a stride of 1 cell, we compute the HOG descriptor by stacking and reshaping all the histograms of each block. We have (8, 16) cells and histograms corresponding to each of the 9 bins. Each block comprises 4 cells, resulting in a vector length of 4x9 = 36. With a total of 7x15 = 105 blocks, the HOG descriptor features 3780 in total.

3.4 PCA

To reduce the number of features from 3780 to 1024, Principal Component Analysis (PCA) is utilized. First, the covariance matrix is computed from the dataset. Subsequently, the top 1024 eigenvectors with the largest eigenvalues are

selected. These eigenvectors are employed to perform a linear transformation of the input space, resulting in a reduced-dimensional feature representation while preserving as much variance as possible within the data. Following class handles PCA:

```
class PCA():
    def __init__(self, reduced_dim=1024):
        self.red_dim = reduced_dim

    # Finding transformation as per train data
    def fit(self, train_data):
        self.mean = np.mean(train_data, axis=0)
        centered_data = train_data - self.mean

        # Calculating Covariance Matrix
        cov_matrix = np.cov(centered_data, rowvar=False)

        # Eigen Decomposition
        eigen_val, eigen_vec = np.linalg.eig(cov_matrix)
        self.eigen_vec = eigen_vec[:, np.argsort(-eigen_val)
    ][:, :self.red_dim]

        return self.transform(train_data)

    # Transforming Data
    def transform(self, data):
        return np.matmul(data-self.mean, self.eigen_vec)

    # Saving PCA
    def save(self, address):
        np.savez_compressed(address, mean=self.mean, eig_vec
    =self.eigen_vec)

    # Loading PCA
    def load(self, address):
        pca = np.load(address)
        self.mean = pca['mean']
        self.eigen_vec = pca['eig_vec']
```

3.5 SVM

All the extracted HOG descriptors are utilized to train a Support Vector Machine (SVM). Subsequently, this trained SVM is employed to predict labels as well as probabilities corresponding to each label for unseen data. Various performance metrics including precision, recall, F1 score, and accuracy are computed based on the predicted labels. Additionally, an ROC curve is plotted to visualize the trade-off between true positive rate and false positive rate, and the Area Under the ROC Curve (AUCROC) is calculated to quantify the model's performance in distinguishing between classes. Following class handle SVM:

```
class SVM():
    def __init__(self):
        self.svm = sklearn.svm.SVC(probability=True)

    def train(self, data, labels):
        self.svm.fit(data, labels)

    def predict(self, data):
        return self.svm.predict_proba(data)[:, 1], self.svm.
    predict(data)
```

4 Result and Conclusion

The Performance is concluded in the below model:

	Precision	Recall	F1 Score	Accuracy
Train	0.9998	0.9999	0.9998	0.9999
Validation	0.9954	0.9954	0.9954	0.9958

Table 1: Performance Metrics

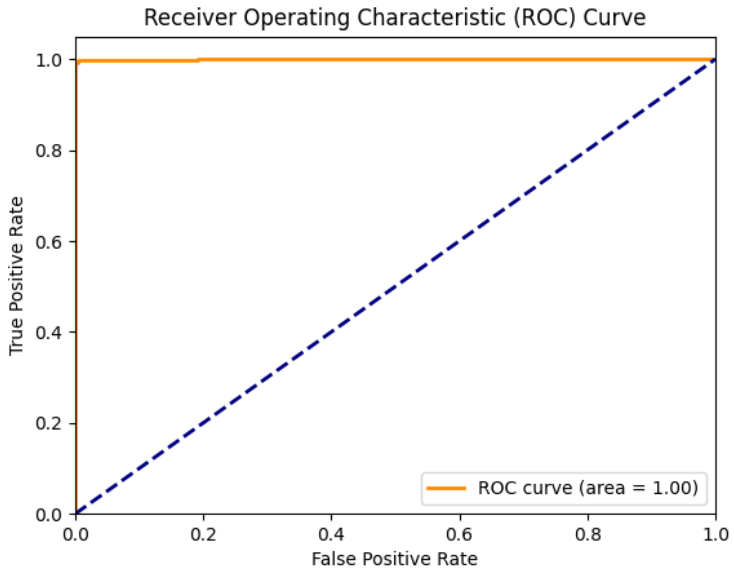

Figure 1: ROC Curve

ROCAUC: 0.9997

Based on the provided metrics and plots, it is evident that our model exhibits exceptional performance on the given dataset, as indicated by the high scores obtained for both training and validation sets. However, it's important to note that the similarity between images in both the training and validation datasets may pose challenges when the model encounters images with different characteristics, potentially affecting its performance.

5 How to run the code

Execute the following command in terminal when dataset is present in current directory to execute the code: `python3 src/main.py`If you dont want to save intermediate models and data then you can just change the save_bool to False in the following snippet:

```
no_box_dataset_addr = "data_no_box"
cropped_dataset = "data"
hog_addr = "temp/hog.npz"
pca_addr = "temp/pca.npz"
svm_addr = "temp/svm.pkl"
roc_addr = "roc"

start_time = time.time()
overwrite = False
save_bool = True
```

If you want to overwrite all the existing data and model then you can just change the overwrite to True in the above snippet.

6 Issues and Challenges Faced

Dataset comprises images with similar characteristics, which could potentially result in erroneous predictions when applied to real-world scenarios. Additionally, noise is introduced in the dataset due to false detections by MediaPipe, which incorrectly identifies certain regions of the image as hands.

References

[1] Scikit-learn: Support Vector Machines (SVM), https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html