INDIAN HAND SIGN CLASSIFICATION

Name: Yatharth Nanda ID:2021AAPS1291G

Objective: Classify images of English alphabets using google mediapipe library's hand detection model

Main objective: Using mediapipe hands, I aim to extract hand landmarks and build a model for hand sign classification.

Initial setup and flow: I have used the following two datasets

https://www.kaggle.com/datasets/soumyakushwaha/indian-sign-language-dataset
https://www.kaggle.com/datasets/prathumarikeri/indian-sign-language-isl?resource=download

The images from this set were not being picked up by the Google model, so different configurations of the model were tried to see what works best. Finally, the following criterion were decided:

Furthermore, Images were removed from the dataset for the following reasons:

1.) Inability of the API to detect the hands.

2.) Wrong "handedness" detection of hands. Two rights when shown two hands insider of right and left, right and left are flipped etc

3.) Wrong number of hands being detected.

Current approach of processing hand landmarks:

1.) Get 21 landmarks (x and y coordinates) from each hand. So, 42 2 dimensional landmarks = 82 datapoints in all.

2.) Preprocess the obtained hands. If there are two hands, then we define the average of the wrists as a common point and declare them as base coordinates. Then we subtract these base coordinates from the coordinates in the hand and feed them into the vector.

3.) If there is only one hand, then we can subtract the wrist point using that as the base reference and feed the other non-existent hand as 0's. This forms the 82-input vector to the neural network which we are trying to do.

28th Feb 2024

Task: Generating new data to further train the model

1.) Generated rotations of varying degrees [5,7,9,10,12]. The number of images was capped at around 100 so sometimes lower degree variations like 5-degree clock and anticlock was omitted.

2.) Filtered valid images if the model was detecting two hands when it was supposed to detect one or if the model is detecting the wrong "handedness" from the image.

3.) Split training data into 4 folders uniformly, used 3 for training and 1 as test.

4.) Accuracy during cross-validation ranging between 88 to 90.2 percent.

6th March 2024

Task: Generating new data from YouTube and initial testing of model)

1.) Took screenshots of popular ISL videos and added my own for H, J, and Y (H isn't matching up in any videos so need to find out about that).

2.) Converted to JPEG format using image_conversion.py.

3.) Checked if mediapipe is working on the character pictures.

4.) Produced rotations and checked if mediapipe is working on them.

5.) Combined rotated images under youtube_data for testing.

27th March 2024

Task: (Reporting on loss and accuracy by testing against different sub sets)

Cross-validation values:

After the discussion, the model size was reduced to the following one. The data recorded below was for seed=42 and batch size =64.

Cross-validation data (using 3 keypoint_new files for training and the other one for testing):

File being used as test data	Loss	Accuracy
Keypoint4new.csv	0.26	0.9401
Keypoint3new.csv	0.6568	0.8978
Keypoint2new.csv	0.4821	0.9127
Keypoint1new.csv	0.6594	0.8557

Cross-Validation data (using 5 files in all, 4 keypointnew files and one keypoint yt file):

File being used as test data	Loss	Accuracy
Keypoint4new.csv	0.2718	0.9476
Keypoint3new.csv	0.6694	0.9052
Keypoint2new.csv	0.4740	0.9077
Keypoint1new.csv	0.7540	0.8831
Keypointyt.csv	0.4730	0.9099

Avg accuracy = 0.9107

27th March 2024

Task : Report on performance if only hand (right) is retained per gesture

1. For this in case both hands are present in the sign , the left hand was dropped
2. In case only hand is present in the sign it was retained and sent to the model for processing
3. preprocessing on the hand happened as described in the method description for single handed signs wherein the base wrist point was subtracted and then the distances were normalised
4. 5 files keypointsingle 1, 2 3 4 and keypointytsingle were used from the using the 4waydatasetnew format
   Here are the losses and cross validation (seed = 42 and batch size 64)
   Here is the data compiled in a table

File in use as test data	Loss	Accuracy
Keypointytsingle	1.3375	0.6986
Keypointsingle1	0.9643	0.7960
Keypointsingle2	0.9580	0.8005
Keypointsingle3	0.9933	0.7955
Keypointsingle4	0.6911	0.8254

3rd April 2024

Task: I added an additional input field to the vector with number of hands to test whether it improves the model's ability to not confuse sign using 2 hands with signs using 1 hand . Cross validation data with 85 input vector size (using 5 files in all, 4 keypointnew files and one keypoint yt file):

File being used as test data	Loss	Accuracy
Keypoint4new.csv	0.2918	0.9401
Keypoint3new.csv	0.6298	0.8928
Keypoint2new.csv	0.5052	0.9177
Keypoint1new.csv	0.6348	0.8905
Keypointyt.csv	0.5465	0.9070

Average over 5 test sets= (0.90962)

12th April 2024

Task : To build a classifier program that takes in images through the webcam , processes and stores them to generate more data

Description of code :

1. **logging_csv(character, image_name, csv_path):**

   - Logs the character label associated with an image name into a CSV file.

   - Parameters: character, image_name, csv_path.

2. **get_screen_dimensions():**

   - Retrieves screen dimensions.

   - Returns: Half of the screen width and full screen height.

3. **pre_process_landmark(both_hand_landmarks):**

- Preprocesses detected hand landmarks.

- Parameters: both_hand_landmarks.

- Returns: Preprocessed landmark list.

4. **extract_label(string_representation):**

- Extracts the category name from a string representation.

- Parameters: string_representation.

- Returns: Extracted category name.

5. **draw_landmarks(image, landmark_point):**

- Draws landmarks on an image.

- Parameters: image, landmark_point.

- Returns: Image with landmarks drawn.

6. **calc_landmark_list(image_local, hand_landmarks_local):**

- Calculates landmark points from detected hand landmarks.

- Parameters: image_local, hand_landmarks_local.

- Returns: List of landmark points.

7. **main Block:**

- Initializes variables, camera capture, and processes captured frames with a 5 second timer triggered by pressing the spacebar key .

- Detects hand landmarks, preprocesses them, classifies gestures, and logs results into a CSV file.

- Displays

- Captures frames, processes them, and exits when the ESC key is pressed.

Files in use and their basic description along with methods :

Sure, here are more detailed descriptions for each method in the provided code in processing_folders_for_landmarks python file :

1. pre_process_landmark_dist(both_hand_landmarks): This function takes in a dictionary of hand landmarks. If there are no landmarks or more than two sets of landmarks, it returns None. If there is one set of landmarks, it calculates the Euclidean distance of each landmark from the base point (the wrist), normalizes these distances by dividing each by the maximum absolute distance, and returns a list of these normalized distances. If there are two sets of landmarks, it calculates the average base point, computes the Euclidean distances of each landmark from this average base point, normalizes these distances, and returns a list of these normalized distances.
2. logging_csv(character, preprocess_landmark_list): This function takes in a character and a list of preprocessed landmark distances. It opens a CSV file located at a specified path and appends a row to this file. The row contains the character and the formatted landmark distances. The landmark distances are formatted to 16 decimal places.
3. pre_process_landmark(both_hand_landmarks): This function takes in a dictionary of hand landmarks. If there are no landmarks or more than two sets of landmarks, it returns None. If there is one set of landmarks, it calculates the relative coordinates of each landmark from the base point (the wrist), normalizes these coordinates by dividing each by the maximum absolute coordinate, and returns a list of these normalized coordinates. If there are two sets of landmarks, it calculates the average base point, computes the relative coordinates of each landmark from this average base point, normalizes these coordinates, and returns a list of these normalized coordinates.
4. extract_label(string_representation): This function takes in a string representation, splits it by comma and space characters, and iterates over each part. If a part contains 'category_name', it extracts the category name and returns it.
5. draw_landmarks(image, landmark_point): This function takes in an image and a list of landmark points. It draws lines and circles on the image to represent the landmarks of the hand. It uses different colors and thicknesses for different parts of the hand.
6. calc_landmark_list(image_local, hand_landmarks_local): This function takes in an image and a list of hand landmarks. It calculates the width and height of the image, initializes an empty list of landmark points, and iterates over each hand landmark. For each hand landmark, it calculates the x and y coordinates, and appends them to the list of landmark points. It returns the list of landmark points. Sure, here are more detailed descriptions for methods 7 to 10 in the provided code:

7. process_directory(main_folder_path): This function takes a path to a main directory as input. It creates a HandLandmarker object using the vision.HandLandmarkerOptions from the mediapipe library. It then iterates over each subdirectory (character folder) in the main directory. If the subdirectory is a valid character folder (A-Z), it processes each file in the character folder. For each file, it reads the image, detects hand landmarks using the HandLandmarker object, preprocesses the landmarks, and logs them into a CSV file. If the image cannot be read, it prints an error message. If the main directory cannot be found, it prints a "Folder not found!" message.
8. process_folder(char_folder_path): This function takes a path to a character folder as input. It creates a HandLandmarker object using the vision.HandLandmarkerOptions from the mediapipe library. It then processes each file in the character folder. For each file, it reads the image, detects hand landmarks using the HandLandmarker object, preprocesses the

landmarks, and logs them into a CSV file. If the image cannot be read, it prints an error message.

9. process_dir_singlehand(main_folder_path): This function takes a path to a main directory as input. It creates a HandLandmarker object using the vision.HandLandmarkerOptions from the mediapipe library. It then iterates over each subdirectory (character folder) in the main directory. If the subdirectory is a valid character folder (A-Z), it processes each file in the character folder. For each file, it reads the image, detects single hand landmarks using the HandLandmarker object, preprocesses the landmarks, and logs them into a CSV file. If the image cannot be read, it prints an error message. If the main directory cannot be found, it prints a "Folder not found!" message.
10. process_folder(char_folder_path): This function takes a path to a character folder as input. It creates a HandLandmarker object using the vision.HandLandmarkerOptions from the mediapipe library. It then processes each file in the character folder. For each file, it reads the image, detects hand landmarks using the HandLandmarker object, preprocesses the landmarks, and logs them into a CSV file. If the image cannot be read, it prints an error message.

16th + 24th April 2024

Task : Test the model for proof of work

Users : Hostel students of BITS GOA

Description : The users were taught hand signs and then the classifier program was used . The results and average accuracy of the model was compiled in the compiled excel file name "results from users(hostel)"

Results : Average accuracy – roughly 81 percent across 7 students and 11-14 images per user

FOLDER AND FILE DESCRIPTIONS USED THROUGHOUT THE PROJECT

1. Folder name : initial_sop_19thfeb
    a. Indian modified : dataset with 40 images per character from https://www.kaggle.com/datasets/prathumarikeri/indian-sign-language-isl?resource=download
    b. ISL_Dataset_new : combination of the usable images from both datasets
    c. Combined_dataset_new : generating new data for the files using rotations by varying degrees
    d. 4 way dataset new : Latest data split into 4 folders so that 3 can be used as training and one as test for cross validation.
        i. Further contains files keypoint1.csv , keypoint2.csv etc which correspond to hand landmarks data from split1 ,split2 and so forth

2. Folder name :6th march 2024

    a. youtube_data_raw :Data compiled from YouTube videos

    b. youtube_data : rotations of above data to make a comprehensive dataset

3. Folder name : 26th March 2024

    a. youtube_data : Data compiled from YouTube ISL videos
    b. confusion matrices for single handed data : screenshots for
    c. keypointsingle1 ,keypointsingle2 : Right hand key points recorded for all gestures

4. Folder name : 3rd April 2024

    a. modified _dataprocessing: to make the input vector as 85 sized now instead

    84 sized, the value is 5 for a single hand and 10 for two hands

5. Folder name: 9th April + 24th April

    a. captured images – captured images from users

    b. 9april_classifier.py – code for classifier program that takes in an image after a 5 sec timeout on pressing the spacebar, processes it and displays the results

    c. classification: initial excel with some test image results

    d. compiled results: manually entered entries, with actual classes vs model predictions