HAND GESTURE RECOGNITION

A MINOR PROJECT REPORT

Submitted by

GAURAV CHATURVEDI [RA1911031010119]

HIMAYANTH V [RA1911031010088]

Under the Guidance of

Dr. PRAVEENA AKKI

(Assistant Professor O.G.)

in partial fulfillment of the requirements for the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

with specialization in INFORMATION TECHNOLOGY

DEPARTMENT OF NETWORKING AND COMMUNICATIONS

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY

KATTANKULATHUR- 603 203

NOVEMBER 2022

Department of Networking and Communications
SRM Institute of Science & Technology
Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/Course	**:**	B. Tech – CSE in specialization with IT
Student Names	**:**	Gaurav Chaturvedi, Himayanth V
Registration Number	**:**	RA1911031010119, RA1911031010088
Title of Work	**:**	Hand Gesture Recognition

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated and that I / We have met the following conditions:

- Clearly references/lists all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data, etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with theUniversity policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603203

BONAFIDE CERTIFICATE

Certified that this B. Tech project report titled "**HAND GESTURE RECOGNITION**" is the bonafide work of **Mr. GAURAV CHATURVEDI** and **Mr. HIMAYANTH V** who carried out the project work under my/our supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

SIGNATURE

Dr. PRAVEENA AKKI
GUIDE
Assistant Professor
Department of Networking
And Communications

SIGNATURE

DR. ANNAPURANI PANAIYAPPAN. K
HEAD OF THE DEPARTMENT
Professor and HOD
Department of Networking
And Communications

Signature of Internal Examiner

Signature of External Examiner

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr T.V.Gopal**, for his valuable support.

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work. We are incredibly grateful to our Head of the Department**, Dr. Annapurani Panaiyappan.K** Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our program coordinators Dr.TYJ.Naga Malleswar **,** Professor, and Panel Head, **Dr.P.Balamurugan ,** Assistant Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for their inputs during the project reviews and support. We register our immeasurable thanks to our Faculty Advisor**, Dr.P.Supraja,** Assistant Professor,Department of Networking and Communications , SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr.Praveena Akki ,** Assistant Professor Dept.of Networking and Communications for providing me with an opportunity to pursue my project under his/her/their mentorship. He/She/They provided me with the freedom and support to explore the research topics of my interest. Her/His/Their passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Networking and Communications Department staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

Gaurav Chaturvedi

Himayanth V

ABSTRACT

The area of gesture recognition is changing due to the development of new technology. With the introduction of technologies like the Kinect sensor, there is now more potential for human-computer interaction than with the conventional hand gesture recognition approaches, which are more dependent on gloves and sensors and less adaptable. The aim of working with picture datasets for improved recognition is to extract additional features to improve gesture recognition. It has been noted that adding more characteristics makes it much simpler to accurately recognize hand gestures, and accuracy may also be raised by optimizing the classification process. To improve the performance of the gesture recognition model, enhanced feature extraction and hybrid classification are used in this study.

.

TABLE OF CONTENTS

Chapter		Title	Page No.
		ACKNOWLEDGEMENTS	**iv**
		ABSTRACT	**v**
		TABLE OF CONTENTS	**vi**
		LIST OF FIGURES	**viii**
		ABBREVIATIONS	**ix**
1		**INTRODUCTION**	**1**
	1.1	Purpose	2
	1.2	Problem statement	3
	1.3	Scope of the project	3
	1.4	Innovation Idea	3
	1.5	Software Requirements	4
2		**LITERATURE SURVEY**	**5**
	2.1	Methodology	6
	2.2	Challenges to address	8
3		**SYSTEM ARCHITECTURE AND DESIGN**	**9**
	3.1	Module description	10
	3.2	Implementation	12
4		**CODE**	15
5		**RESULT**	**28**
6		**CONCLUSION**	**30**

7	**RELATED WORK**	**31**
8	**REFERENCES**	**34**
APPENDIX		
A	**PAPER PUBLICATION**	**36**

LIST OF FIGURES

1.1	ER Diagram
3.1	ER Diagram
3.2	UML Diagram
3.2.1	Dataset Testing
5.1	Base Paper Result
5.2	Research Paper Result
A.1	Plagiarism Report

ABBREVIATIONS

AES	Advanced Encryption Standard
ANN	Artificial Neural Network
CSS	Cascading Style Sheet
CV	Computer Vision
DB	Data Base
DNA	Deoxyribonucleic Acid
SQL	Structured Query Language
SVM	Support Vector Machine
UI	User Interface

CHAPTER I

INTRODUCTION

Traditional techniques, such as data gloves, attach sensors or markers to the fingers in order to detect hand motions using electro-mechanical or magnetic sensing. These techniques are efficient in measuring hand motions accurately and in real time, but they impede the natural mobility of the hand and cannot be used in non-contact contexts. The equipment also costs a lot for occasional usage and requires complicated calibration.

An alternate solution to the issues is provided by vision-based hand gesture detection techniques, which may be employed naturally in noncontact contexts. However, because of the optical sensors' limitations, the acquired pictures are susceptible to background clutter and poor illumination. As a result, these techniques frequently fail to identify and track the hand accurately.

An alternative possibility is to examine hand gesture detection in a new way with the advancement of depth cameras, such as the Kinect sensor. By combining the depth map and colour picture, the hand gesture may be identified and detected.

After the extraction of images successfully we follow the below framework:

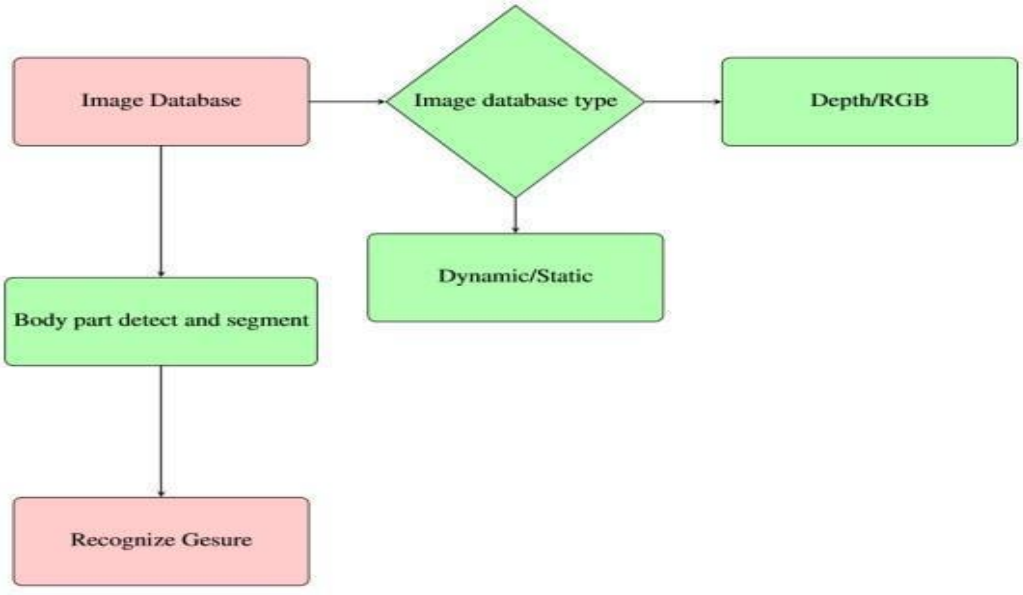


Fig.1.1 ER Diagram

1.1 PURPOSE

Traditional methods attach sensors or markers to the fingers, e.g., the data gloves, to capture hand gestures via electro-mechanical or magnetic sensing. These methods are effective in providing complete and real-time measurements of hand gestures; however, they hinder the natural motion of the hand and are inapplicable in non-contact environments.

Moreover, the devices are expensive for casual use and require complex calibration. The vision-based hand gesture recognition methods give an alternative solution to the problems, which can be used naturally in noncontact environments. However, due to the limitations of the optical sensors, the captured images are sensitive to lighting conditions and cluttered backgrounds.

Thus, these methods usually cannot detect and track the hand robustly. Alternative solution: With the development of depth cameras, e.g., the Kinect sensor, hand gesture recognition can be explored in a new form. The hand gesture can be detected and recognized by the fusion of both the color image and depth map.

1.2 PROBLEM STATEMENT

In the traditional hand gesture recognition methods, lots of dependencies on gloves and sensors makes it more difficult to make some actual use of this field and makes it less flexible, but with development of technologies like Kinect sensor, provides new opportunities for human computer interaction. When we work on image dataset for better recognition the goal is to extract more features so that gesture recognition gives better results. It has been observed that if we extract more features, it is a lot easier to identify the hand gesture with better accuracy as well as accuracy can also be increased by improving the classification process. In this research a better feature extraction and hybrid classification is applied to enhance the result of the gesture recognition model.

1.3 SCOPE OF THE PROJECT

It is observed that gesture detection is used in many fields and it builds a bridge between human and machine interaction and understanding therefore it needs to applied to more areas to take its benefits.Gaming is a very popular trend now a days among youngsters and most of the popular games uses gesture recognition therefore there is a huge possibility of usage presently as well as in future, if we enhance accuracy by applying different techniques like I have proposed in my thesis it might take user experience to a whole new level.

1.4 INNOVATION IDEA

With development of technologies like Kinect sensor, provides new opportunities for human computer interaction. When we work on image dataset for better recognition the goal is to extract more features so that gesture recognition gives better results. It has been observed that if we extract more features, it is a lot easier to identify the hand gesture with better accuracy as well as accuracy can also be increased by improving the classification process. In this research a better feature extraction and hybrid classification is applied to enhance the result of the gesture recognition model.

1.5 SOFTWARE REQUIREMENTS

SOFTWARE:

- OS: Windows 7/8/8.1/10/11, MacOS
- IDE: Sublime Text, Visual Studio Code, Mongo Db

HARDWARE:

- Processor: Intel i5 7th gen or more
- Motherboard requirement: Intel® Chipset motherboard
- Graphic cars- 2GB or more
- Ram requirement: 6GB or more
- Cache required: 4 MB
- Hard-Disk: 256 GB hard disk recommended
- Disk drive: 1.44 MB Floppy Disk Drive
- Monitor display: 1280 x 720 Display
- Processor Speed: 2.6 GHz and more

CHAPTER 2

LITERATURE REVIEW

The plan for this research includes following steps:

Pre-processing phase: - The pre-processing stage of gesture recognition is the initial stage. The picture dataset, which is gathered from an authentic data source, is used as input during this step. Images are transformed to gray scale for further processing after being taken as input.

Segmentation: - Image segmentation is the process of breaking up a digital image into multiple pieces. The primary goal of segmentation is to recognise objects or extract data from photos. The duty of photo examination is made simpler by this method. The process of image segmentation is used to identify objects and the boundaries of images. Differentiated characteristics are shared by pixels with the same label portion in order to label every pixel in an image.

Feature Extraction: - The region of interest is the outcome as of right now. In order to extract the characteristics from this area of interest, this phase is implemented. Feature extraction is the process of removing from an image a group of values referred to as features. For further processing, these attributes offer information about the image. To identify an infection within a plant, a variety of characteristics are often used, including colour, texture, morphology, and colour coherence vector. Different strategies can be used to extract features. A system may be created using these techniques. These techniques include the histogram-based feature extraction approach, colour co-occurrence method, spatial grey-level dependency matrix, and gray-level co-occurrence matrix (GLCM). A statistical method for classifying textures is the GLCM method.

Classification of Data: - The third process, model creation, involves splitting the entire dataset into training and test sets. When compared to the test set, the training set will be larger. The classification model is used, with training and test sets as input. The entire dataset will be split into ratios of 70 and 30. For the gesture recognition system, 70% will be training sets and 30% will be test sets. For the purpose of recognising gestures, the hybrid classification method—a mix of KNN, SVM, and random forest—is used.

2.1 METHODOLOGY

Traditional methods attach sensors or markers to the fingers, e.g., the data gloves, to capture hand gestures via electro-mechanical or magnetic sensing. These methods are effective in providing complete and real-time measurements of hand gestures; however, they hinder the natural motion of hand and are inapplicable in non-contact environments. Moreover, the devices are expensive for casual use and require complex calibration.

The vision-based hand gesture recognition methods give an alternative solution to the problems, which can be used naturally in noncontact environments. However, due to the limitations of the optical sensors, the captured images are sensitive to lighting conditions and cluttered backgrounds. Thus, these methods usually cannot detect and track the hand robustly.

Alternative solution: With the development of the depth cameras, e.g., the Kinect sensor, hand gesture recognition can be explored in a new form. The hand gesture can be detected and recognized by the fusion of both the color image and depth map.
The type of the pictures is determined by the image dataset; if the RGB image dataset, a further step is the conversion of the RGB photos into grayscale images. Following the preprocessing step, ROI (Region of Interest) segmentation is done to focus solely on the needed portion of the image.

Different steps involved in this process are explained as:

Image database: RGB and depth pictures are the two types of images in the collection. A regular camera can capture RGB photos, while depth cameras like Kinect and Leap Motion can capture RGB images and depth images at the same time. The ability to record some spatial information through the use of depth pictures makes it easier to identify and categorise motions.

Gestures are divided into static gestures and dynamic gestures, which determine whether get a single photo or a video
Body Part detection and segmentation: Problems with occlusion, variable light intensity, and light direction throughout the picture gathering process raise the bar for the algorithm's

resilience. As the case for gesture recognition becomes more compelling, more and more algorithms are being developed to solve illumination invariance and occlusion issues.

The two popular methods of gesture segmentation involve CNN (Convolution Neural Network) and deep threshold. Both of these methods have been discussed below:

Gesture Segmentation Based on Convolutional Neural Networks:
Optimization using convolutional neural networks, based on FCN (Full Convolutional Neural Networks) or SegNet, is a component of convolutional neural network-based gesture segmentation. The last CNN layer is replaced by FCN and SegNet with a deconvolution layer, the picture is up-sampled to its original size, and each pixel is forecasted. FCN and SegNet are superior than CNN in that they may take input pictures of any size, no longer need that all images be the same size, and do away with the issue of recurrent storage and convolution computations.FCN, however, also has blatant flaws. The image is not particularly clear when the upsampling factor is large, and the sensitivity to detail has to be increased; the relationship between various pixels is not completely used. The hand area extraction approach incorporates a convolutional neural network to anticipate the middle finger joint of the palm, greatly enhancing the division impact.. Hand area may be divided using SegNet. The toughness of the division is increased by taking a screenshot of the region close to the hand, but the resulting image comprises the edge area, which reduces the identification accuracy.

The segmentation method based on convolutional neural networks is flexible, and a variety of different methods can be combined to perform gesture segmentation.

Gesture Segmentation Based on Depth Threshold Method:

The depth threshold technique extracts an image whose distance is within a preset range by measuring the distance between each pixel and the camera in accordance with the distance between the object and the camera in the depth picture. The depth range for the hand on the depth picture is defined, or the hand is immediately taken into account as the item nearest to the camera, in order to better extract the hand range. With this technique, the preprocessing effect is enhanced, a more precise hand area is obtained, and gesture detection accuracy is increased.However, it has restrictions on the recognition process and a restricted recognition

range. Although the depth threshold approach can segment images quickly and effectively, user behavior is severely constrained, and there is not much room for improvement.

Recognition and classification of gestures: There are two types of gesture recognition: dynamic recognition and static recognition. While dynamic gestures are variations in gesture motions over time, or several successive static gestures, static gestures are gestures made in a single image.

Three types of images are used for gesture recognition: depth maps, RGB maps, and RGB-D maps. The distance between the camera and the object may be accurately represented by the depth map. The grey picture resembles the depiction of the depth map. The depth map differs because each pixel displays the separation between the object and the camera. The RGB-D picture consists of depth images as well as RGB three-channel images.

Although the two images look different, there is a one-to-one correspondence between the pixels.

2.2 CHALLENGES TO ADDRESS

The identification of gestures is the foundation of this research. There are several phases involved in gesture recognition algorithms, including pre-processing, segmentation, form representation, feature matching, and classification. The underlying study uses the deep map approach for pre-processing, threshold-based segmentation for manual segmentation, and the DWT feature matching technique for feature matching. Last but not least, neural network classification approach is used. Low accuracy is provided by the gesture recognition algorithms. The high accuracy gesture recognition method needs to be improved.

Following are the various objectives of this research work:

4.1 To research and evaluate different gesture recognition methods

4.2 Use an approach for gesture recognition based on neural networks.

4.3 To suggest a hybrid classification method for gesture recognition

Use the suggested method and compare it to what is already available in terms of accuracy, precision, and recall.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

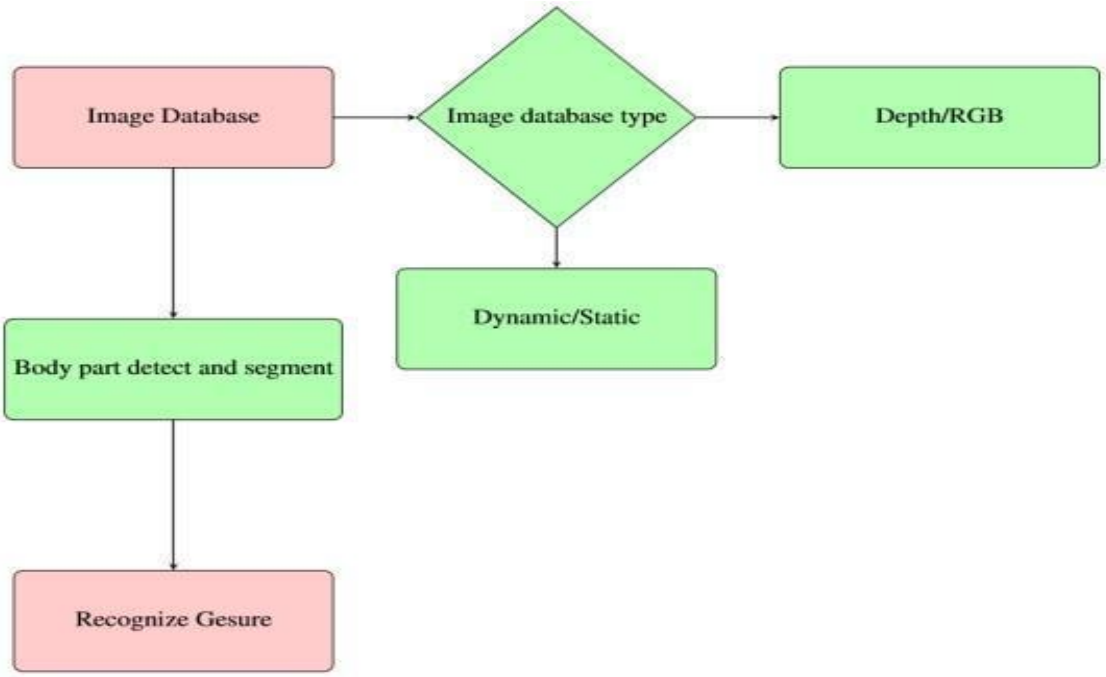


Fig.3.1 ER Diagram

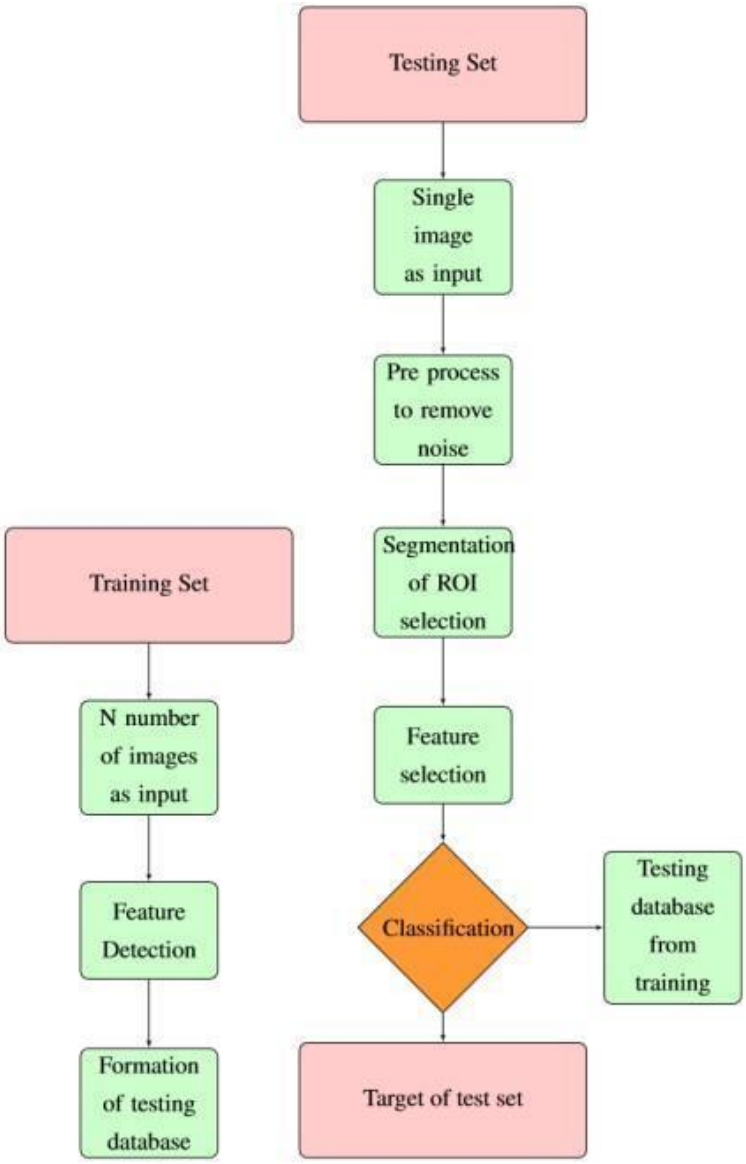


Fig 3.2 UML Diagram

The plan for this research includes following

steps:

Pre-processing phase: - The pre-processing stage of gesture recognition is the initial stage. The picture dataset, which is gathered from an authentic data source, is used as input during this step. Images are transformed to gray scale for further processing after being taken as input.

Segmentation: - Image segmentation is the process of breaking up a digital image into multiple pieces. The primary goal of segmentation is to recognise objects or extract data from photos. The duty of photo examination is made simpler by this method. The process of image segmentation is used to identify objects and the boundaries of images. Differentiated characteristics are shared by pixels with the same label portion in order to label every pixel in an image.

Feature Extraction: - The region of interest is the outcome as of right now. In order to extract the characteristics from this area of interest, this phase is implemented. Feature extraction is the process of removing from an image a group of values referred to as features. For further processing, these attributes offer information about the image. To identify an infection within a plant, a variety of characteristics are often used, including colour, texture, morphology, and colour coherence vector. Different strategies can be used to extract features. A system may be created using these techniques. These techniques include the histogram-based feature extraction approach, colour co-occurrence method, spatial grey-level dependency matrix, and gray-level co-occurrence matrix (GLCM). A statistical method for classifying textures is the GLCM method.

Classification of Data: - The third process, model creation, involves splitting the entire dataset into training and test sets. When compared to the test set, the training set will be larger. The classification model is used, with training and test sets as input. The entire dataset will be split into ratios of 70 and 30. For the gesture recognition system, 70% will be training sets and 30% will be test sets. For the purpose of recognising gestures, the hybrid classification method—a mix of KNN, SVM, and random forest—is used.

3.1 MODULES DESCRIPTION

Following modules are planned and created:-

MediaPipe: Google created a framework for adaptable machine learning solutions called MediaPipe. It is a lightweight, cross-platform framework that is available under an open-source licence. Some pre-trained ML solutions are included with MediaPipe, including face detection, pose estimation, hand recognition, object detection, etc.

TensorFlow: The Google Brains team created the open-source TensorFlow library for machine learning and deep learning. While it may be used to many different tasks, deep neural networks are its main emphasis.

glob: Using precise pattern matching, this is used to locate all file paths.

csv (Comma Separated Values): Databases and spreadsheets may be read and written using these module objects.

skimage: Open source Python tool for image preprocessing that includes features like the greycomatrix (converts RGB image to grey scale)

numpy: This is a core Python library used for complex calculations, big mathematical operations, and the processing of enormous amounts of data.

sklearn: Python library for predictive analysis and classification of data.

pandas:It is a free software programme for manipulating data that is used to filter data frames.

matplotlib:This Python package is used to produce various charts and visualisations..

sys: This Python module gives users access to several variables and functions that are used to modify code in the runtime environment.

3.2 IMPLEMENTATION

Technologies we worked with :-

Dataset:

The fist and palm photos that make up the data set I used are utilised to train the model and to test it later using the same data set.

Testing database:

A testing database with features and a target set will be used to record the training results in a csv file:

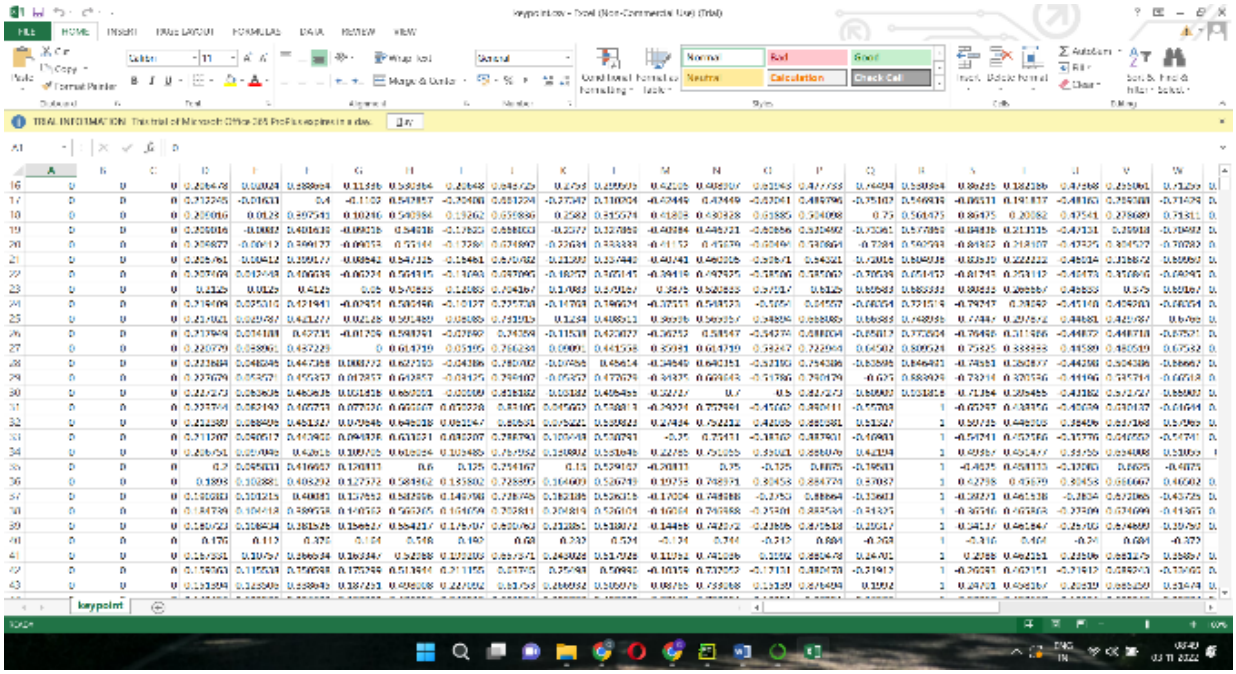


Fig 3.2.1 Dataset Testing

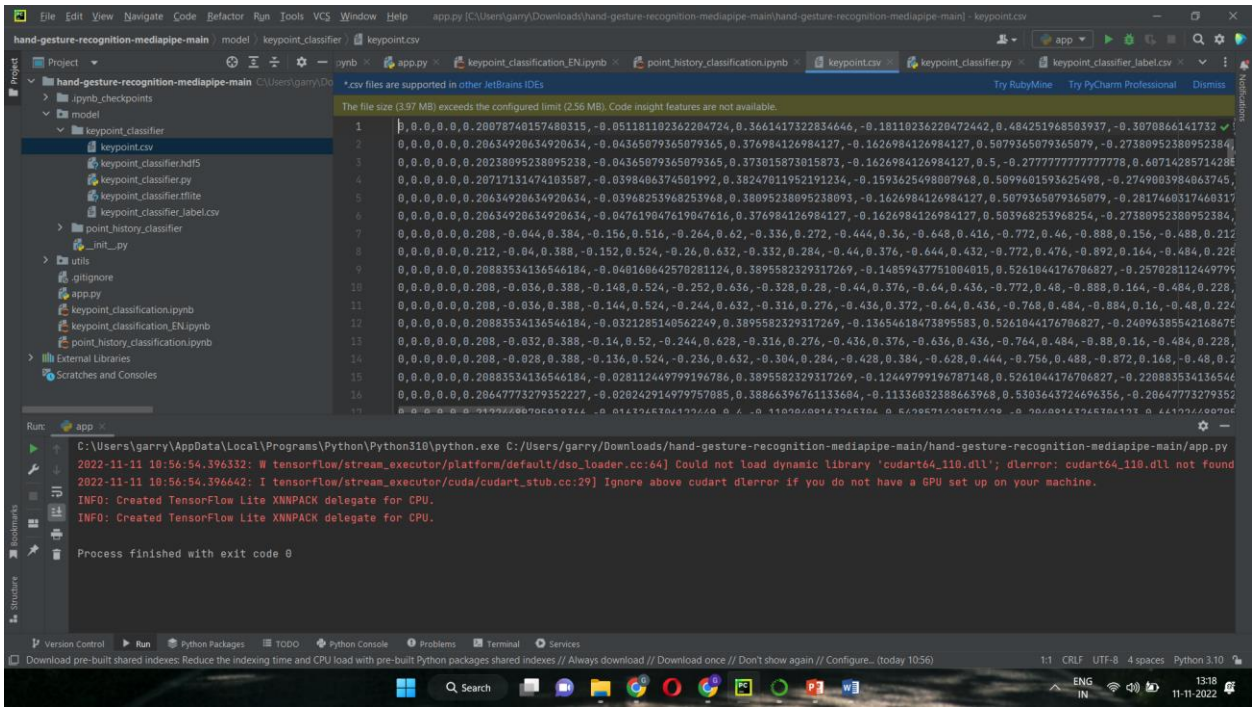
Training: Reading every picture, extracting features including contrast, dissimilarity, homogeneity, ASM, energy, and correlation, translating those features into data frames, creating a target using the random module, and putting the output in a CSV file (testing.csv) for testing.

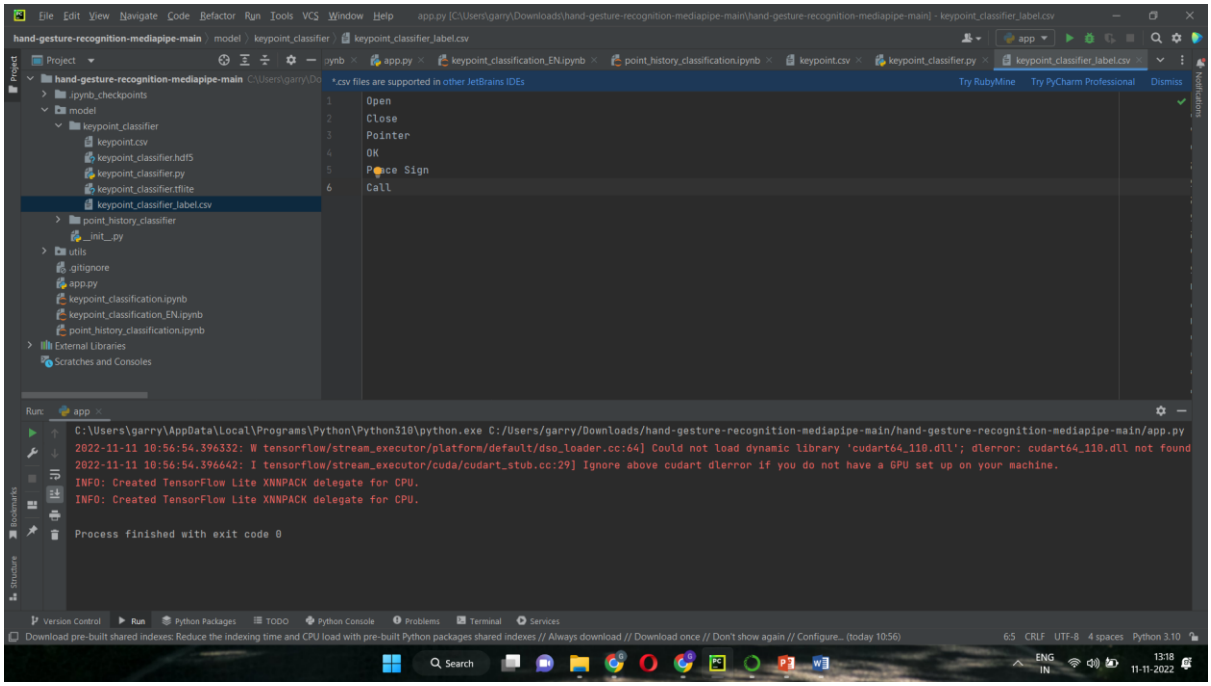
Testing: Reading a picture, doing any necessary preprocessing, and then calculating its contrast, dissimilarity, homogeneity, ASM, energy, and correlation, before reading the CSV file produced during training and forecasting the outcomes. Voting classifiers are employed to

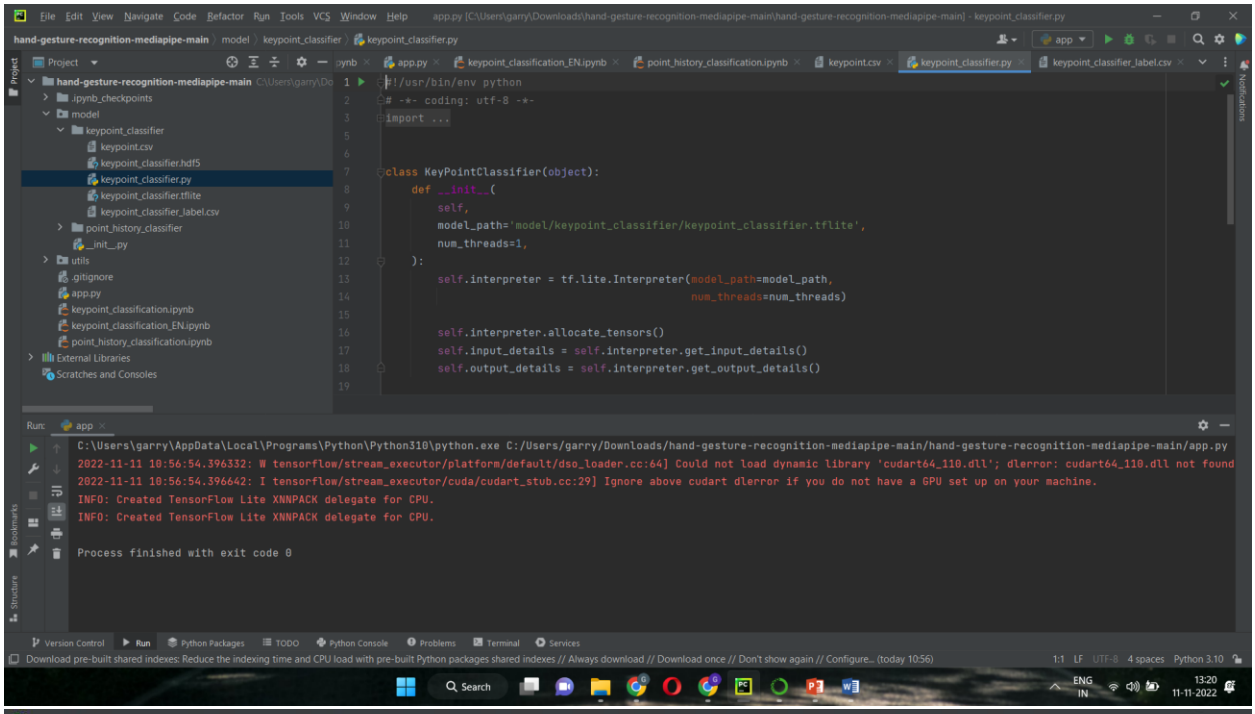
calculate accuracy in this case; hybrid classification, which included GaussianNB, RandomForestClassifier, and LogisticRegression, is proposed.

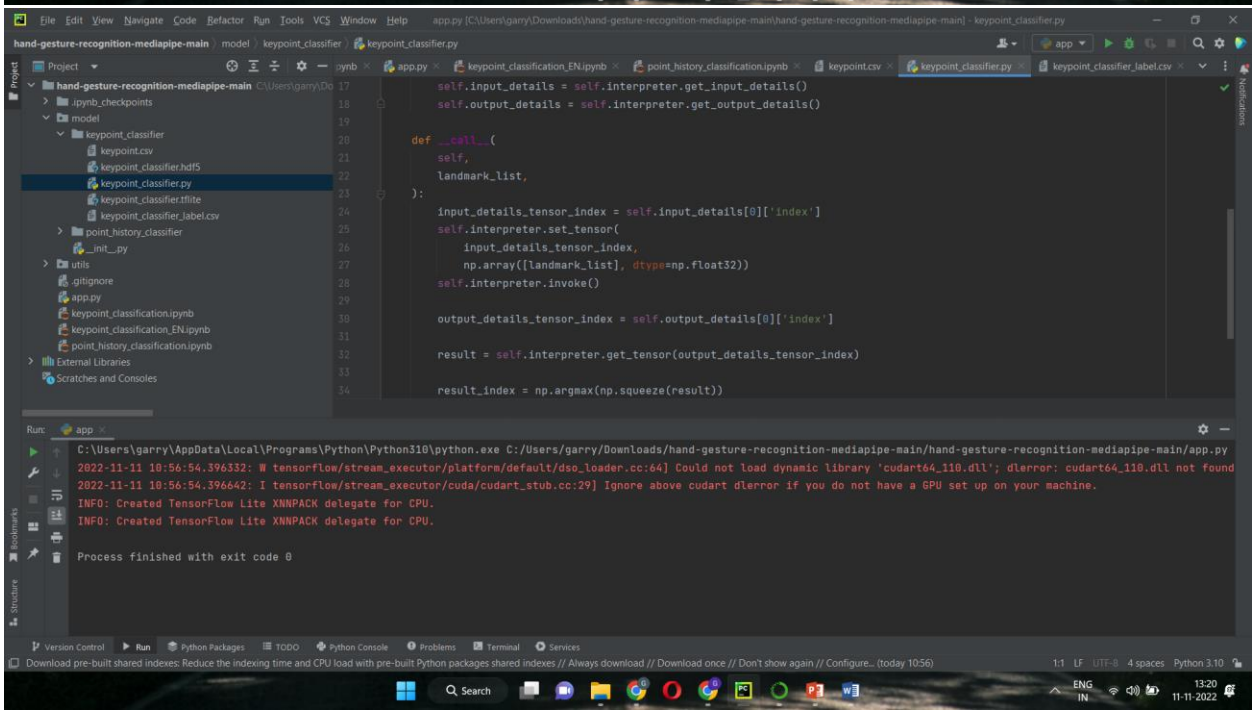
CHAPTER 4

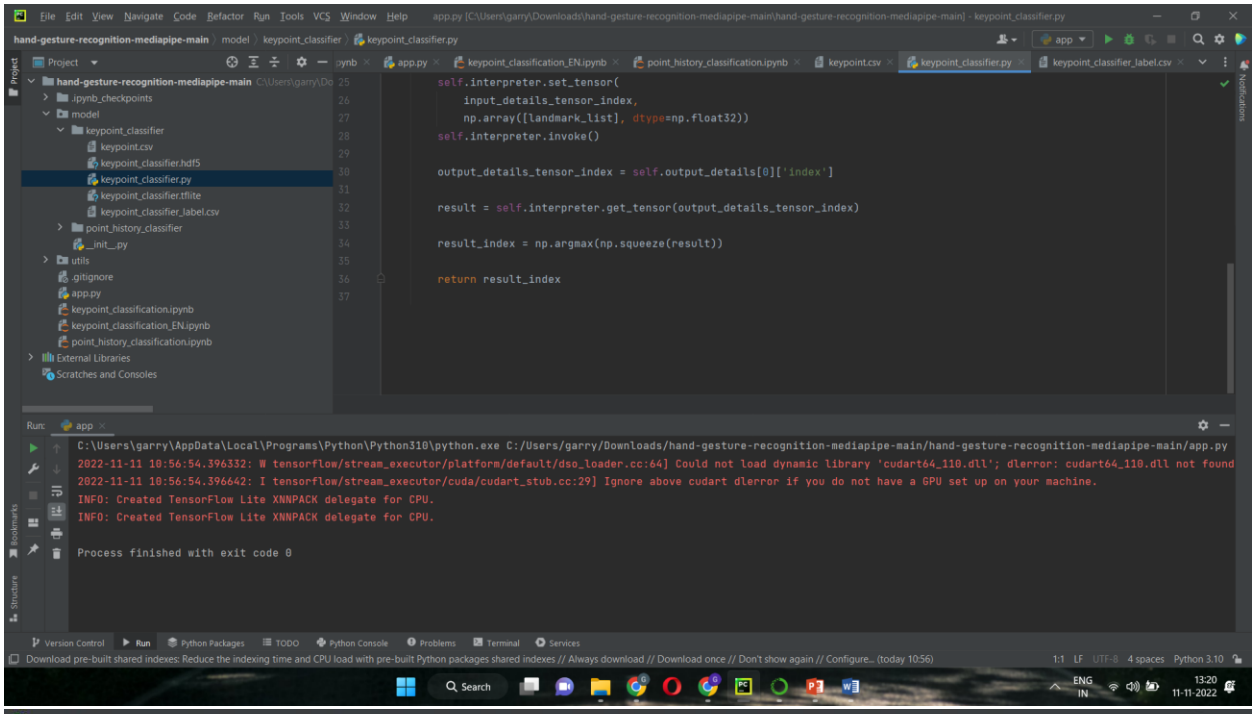
CODE

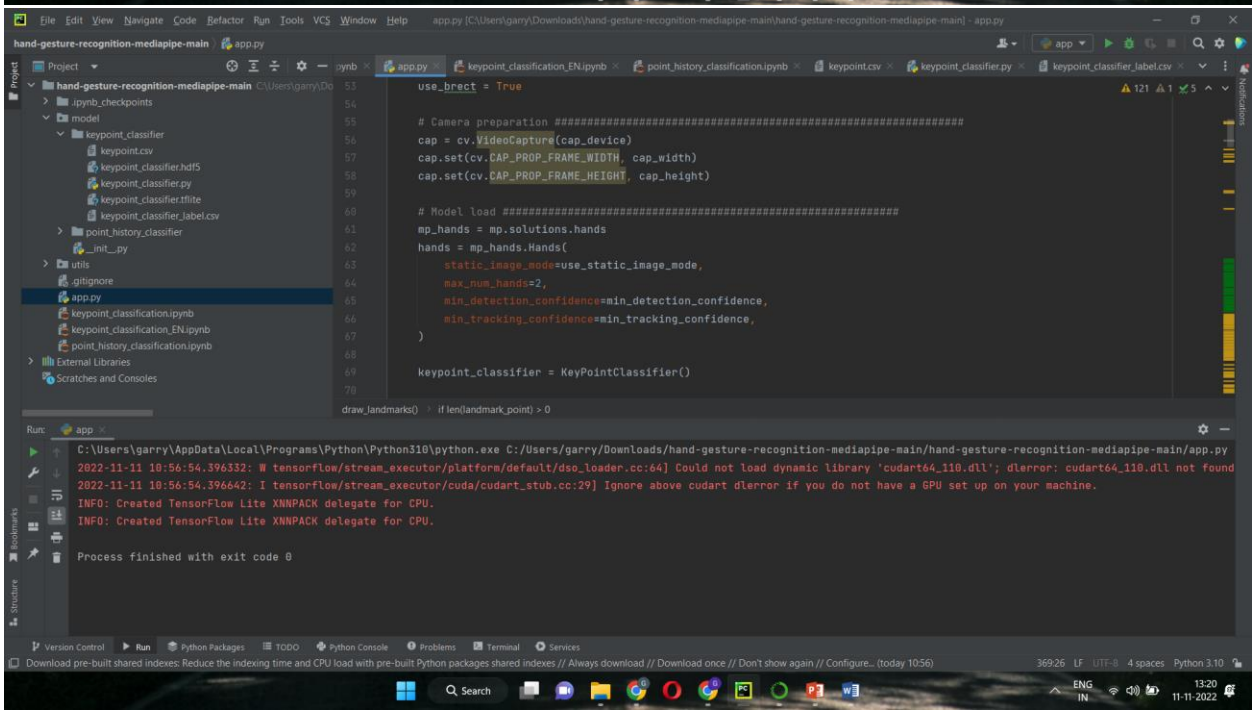


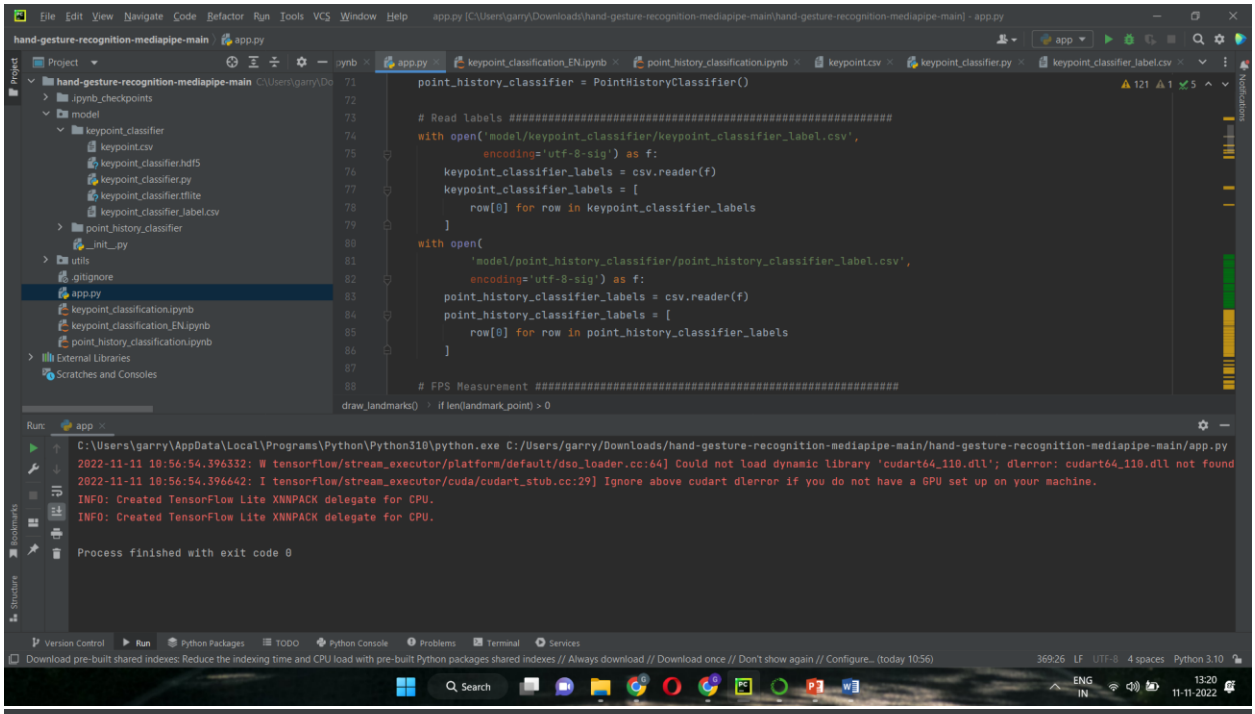

Open
Close
Pointer
OK
Peace Sign
Call
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0

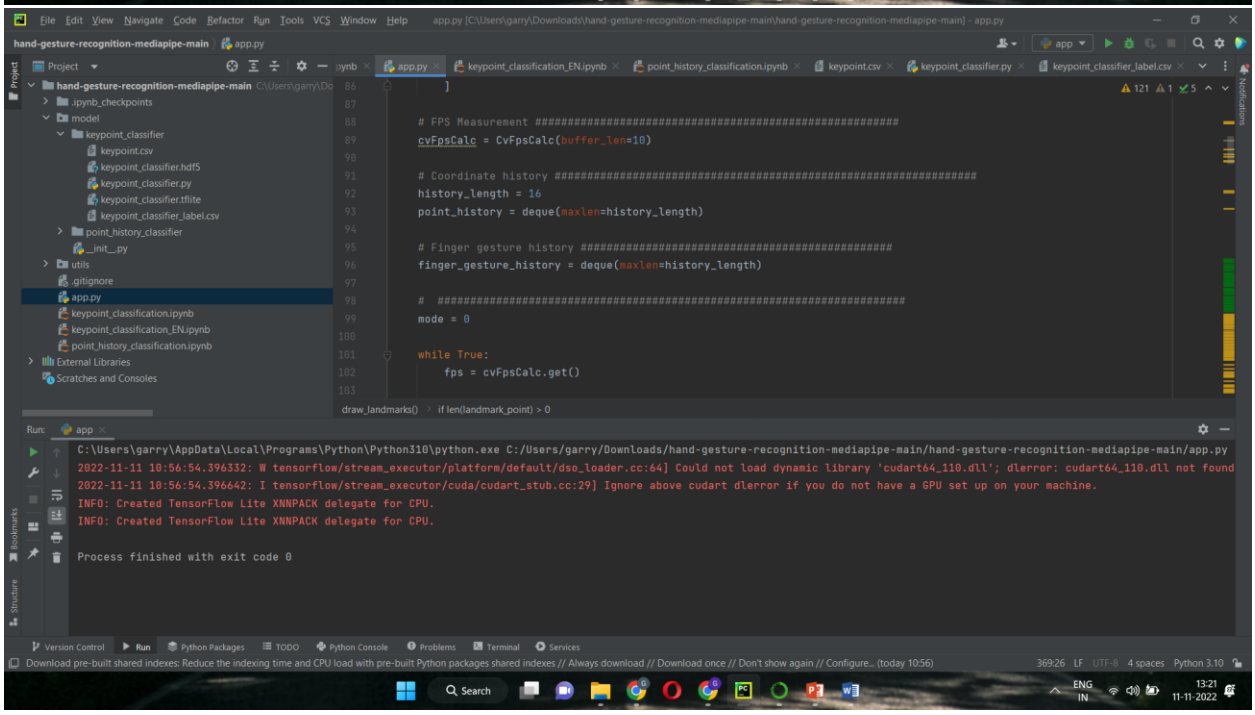

File Edit View Navigate Code Refactor Run Tools VCS Window Help
app.py [C:\Users\garry\Downloads\hand-gesture-recognition-mediapipe-main\hand-gesture-recognition-mediapipe-main] - keypoint_classifier.py
hand-gesture-recognition-mediapipe-main model keypoint_classifier keypoint_classifier.py
app.py keypoint_classification_EN.ipynb point_history_classification.ipynb keypoint.csv keypoint_classifier.py keypoint_classifier_label.csv
#!/usr/bin/env python
-*- coding: utf-8 -*-
import ...
class KeyPointClassifier(object):
def __init__(
self,
model_path='model/keypoint_classifier/keypoint_classifier.tflite',
num_threads=1,
):
self.interpreter = tf.lite.Interpreter(model_path=model_path,
num_threads=num_threads)
self.interpreter.allocate_tensors()
self.input_details = self.interpreter.get_input_details()
self.output_details = self.interpreter.get_output_details()
Run: app
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0
Version Control Run Python Packages TODO Python Console Problems Terminal Services
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (today 10:56)
1:1 LF UTF-8 4 spaces Python 3.10
Search
ENG IN
13:20 11-11-2022

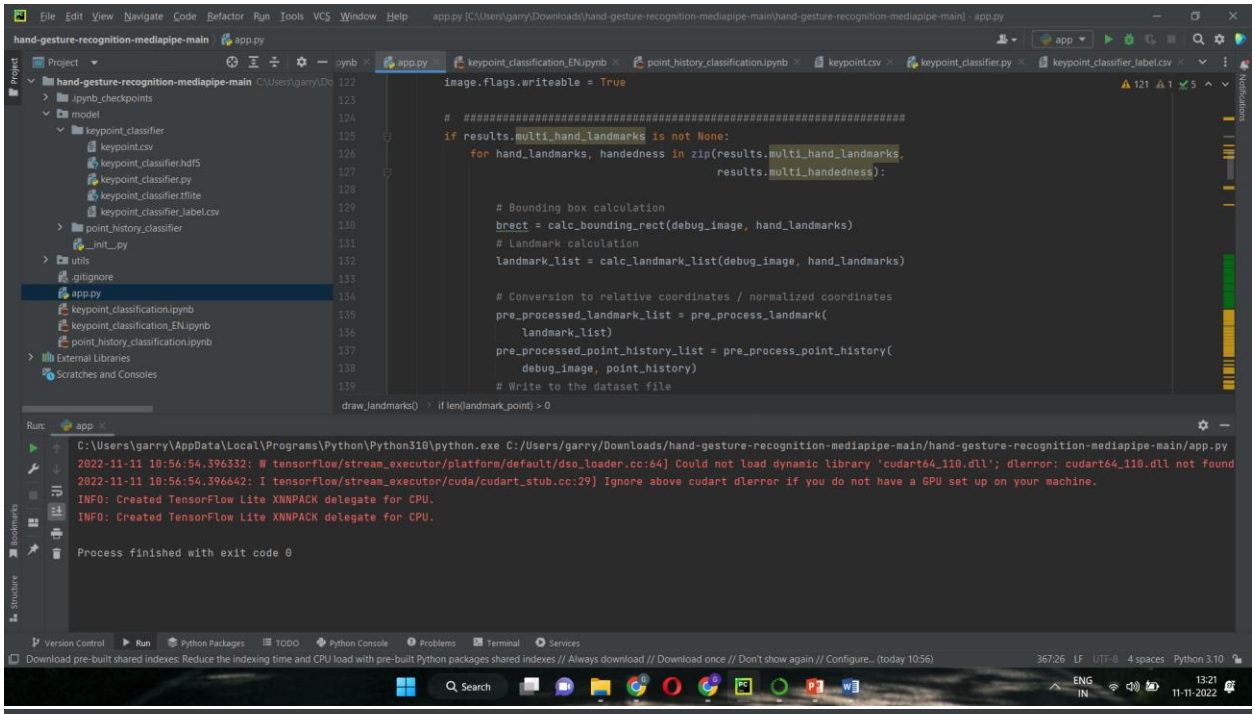

File Edit View Navigate Code Refactor Run Tools VCS Window Help
app.py [C:\Users\garry\Downloads\hand-gesture-recognition-mediapipe-main\hand-gesture-recognition-mediapipe-main] - keypoint_classifier.py
hand-gesture-recognition-mediapipe-main model keypoint_classifier keypoint_classifier.py
app.py keypoint_classification_EN.ipynb point_history_classification.ipynb keypoint.csv keypoint_classifier.py keypoint_classifier_label.csv
self.input_details = self.interpreter.get_input_details()
self.output_details = self.interpreter.get_output_details()
def __call__(
self,
landmark_list,
):
input_details_tensor_index = self.input_details[0]['index']
self.interpreter.set_tensor(
input_details_tensor_index,
np.array([landmark_list], dtype=np.float32))
self.interpreter.invoke()
output_details_tensor_index = self.output_details[0]['index']
result = self.interpreter.get_tensor(output_details_tensor_index)
result_index = np.argmax(np.squeeze(result))
Run: app
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0
Version Control Run Python Packages TODO Python Console Problems Terminal Services
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (today 10:56)
1:1 LF UTF-8 4 spaces Python 3.10
Search
ENG IN
13:20 11-11-2022

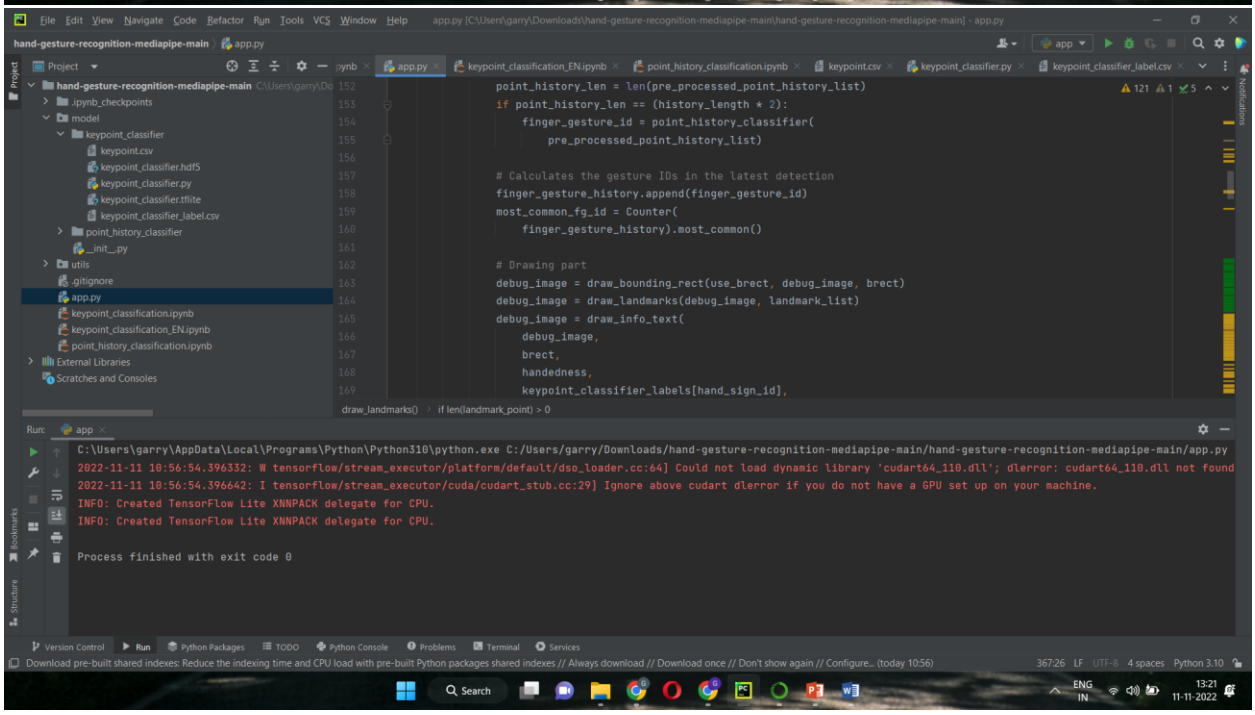

self.interpreter.set_tensor(
input_details_tensor_index,
np.array([landmark_list], dtype=np.float32))
self.interpreter.invoke()
output_details_tensor_index = self.output_details[0]['index']
result = self.interpreter.get_tensor(output_details_tensor_index)
result_index = np.argmax(np.squeeze(result))
return result_index
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0

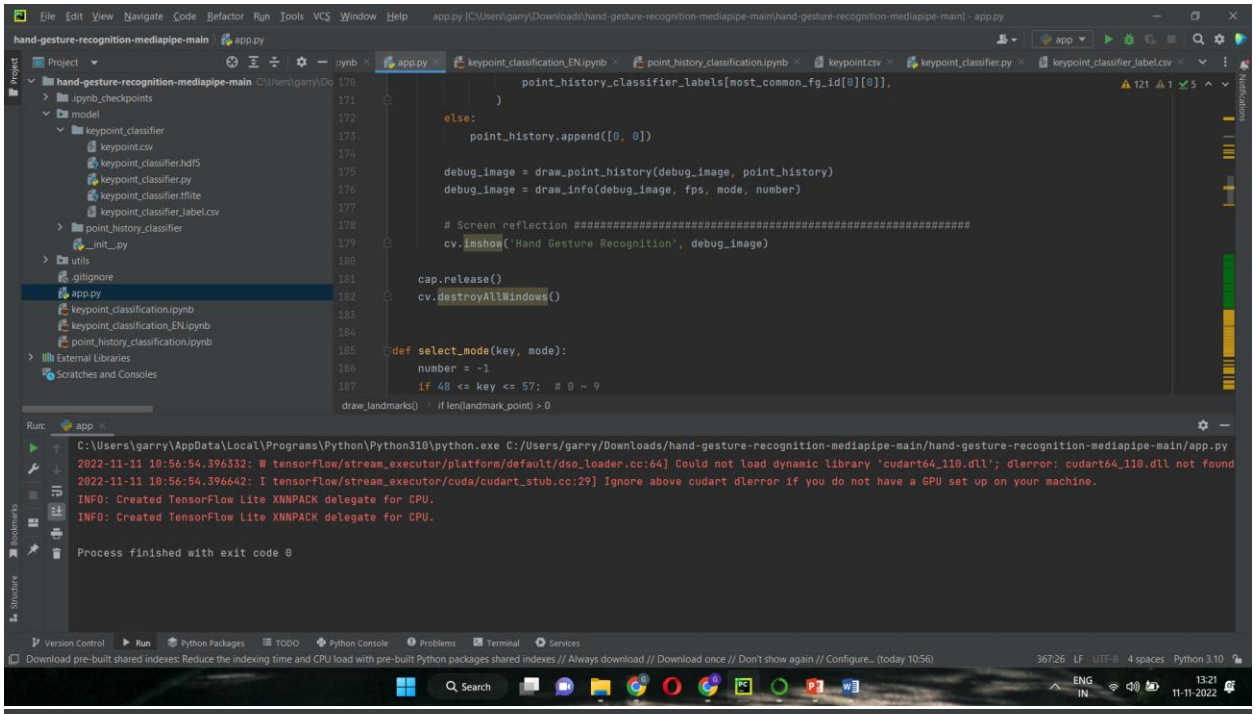

use_brect = True
# Camera preparation ###
cap = cv.VideoCapture(cap_device)
cap.set(cv.CAP_PROP_FRAME_WIDTH, cap_width)
cap.set(cv.CAP_PROP_FRAME_HEIGHT, cap_height)
# Model load ###
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(
static_image_mode=use_static_image_mode,
max_num_hands=2,
min_detection_confidence=min_detection_confidence,
min_tracking_confidence=min_tracking_confidence,
)
keypoint_classifier = KeyPointClassifier()
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0

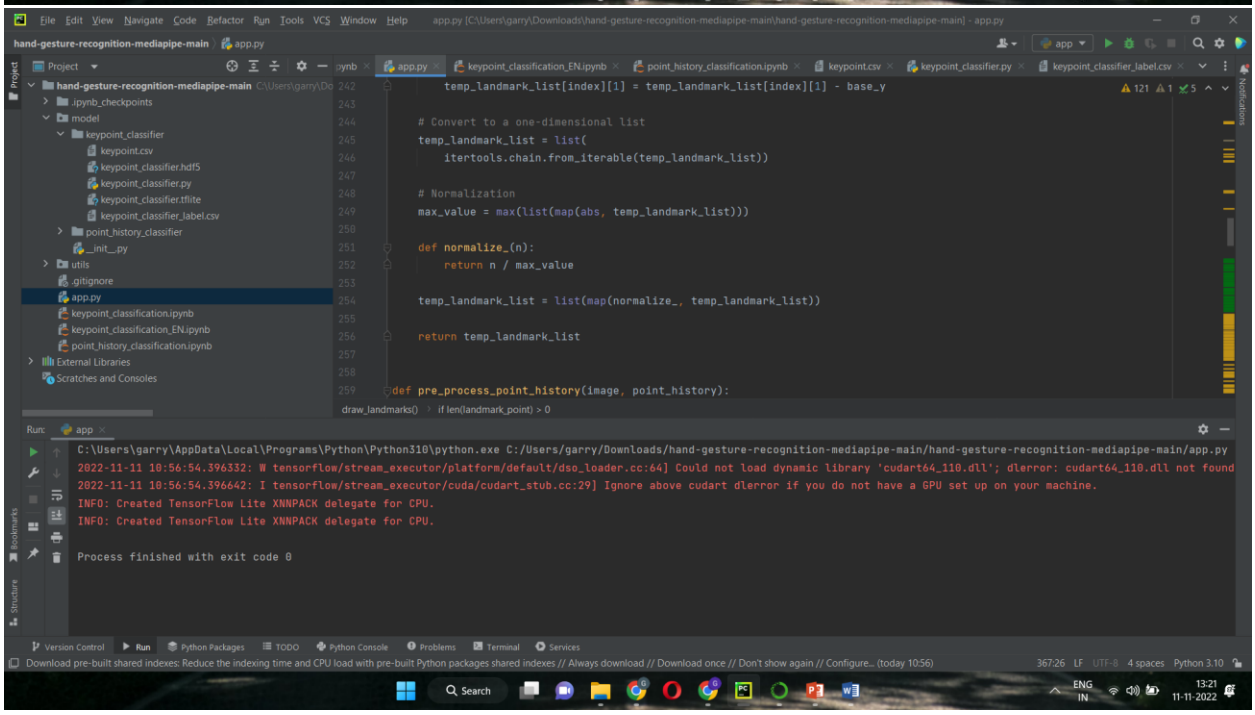

point_history_classifier = PointHistoryClassifier()
# Read labels ###
with open('model/keypoint_classifier/keypoint_classifier_label.csv',
encoding='utf-8-sig') as f:
keypoint_classifier_labels = csv.reader(f)
keypoint_classifier_labels = [
row[0] for row in keypoint_classifier_labels
]
with open(
'model/point_history_classifier/point_history_classifier_label.csv',
encoding='utf-8-sig') as f:
point_history_classifier_labels = csv.reader(f)
point_history_classifier_labels = [
row[0] for row in point_history_classifier_labels
]
FPS Measurement
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0

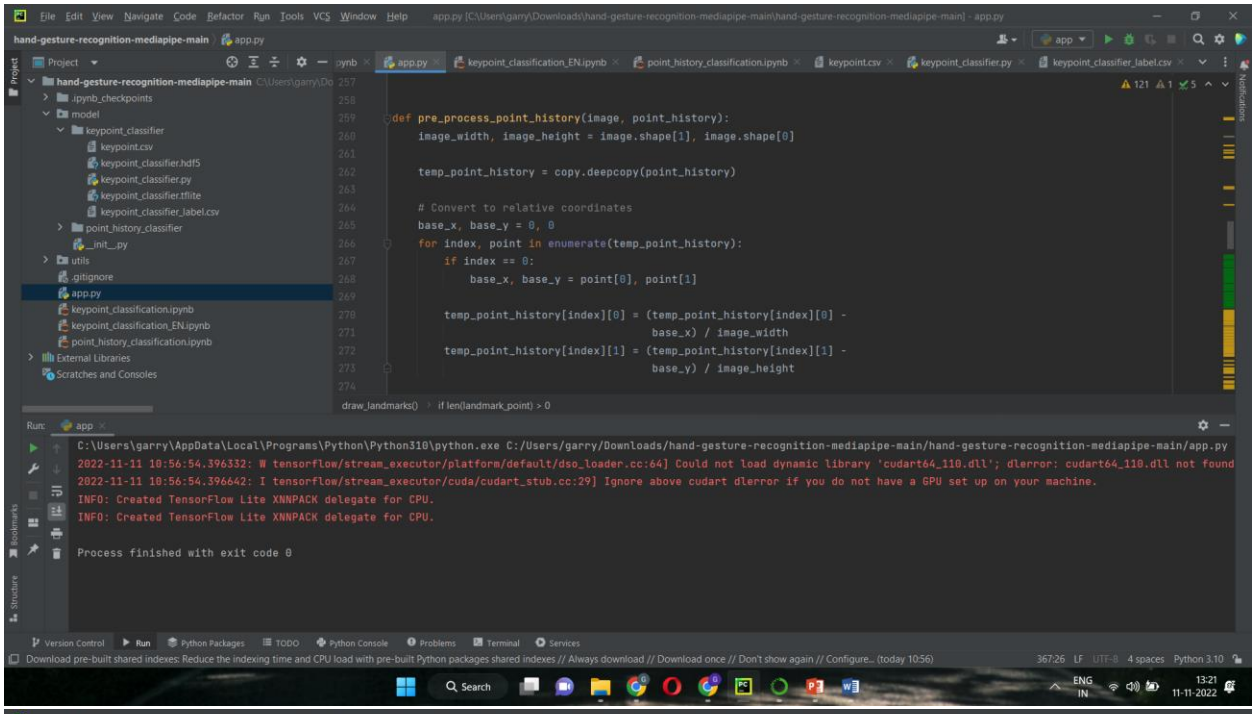

]
# FPS Measurement ##
cvFpsCalc = CvFpsCalc(buffer_len=10)
# Coordinate history ###
history_length = 16
point_history = deque(maxlen=history_length)
# Finger gesture history ##
finger_gesture_history = deque(maxlen=history_length)
# ###
mode = 0
while True:
fps = cvFpsCalc.get()
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0

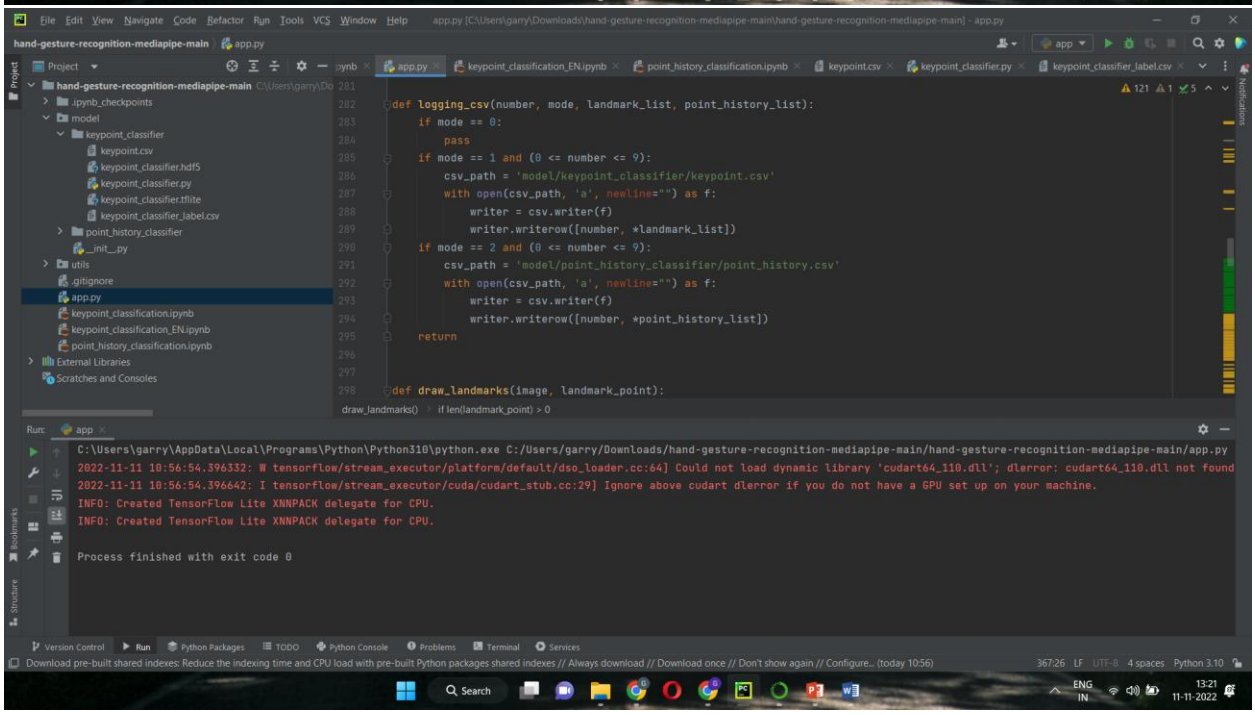

File Edit View Navigate Code Refactor Run Tools VCS Window Help
app.py [C:\Users\garry\Downloads\hand-gesture-recognition-mediapipe-main\hand-gesture-recognition-mediapipe-main] - app.py
hand-gesture-recognition-mediapipe-main
app.py
Project
hand-gesture-recognition-mediapipe-main
.ipynb_checkpoints
model
keypoint_classifier
keypoint.csv
keypoint_classifier.hdf5
keypoint_classifier.py
keypoint_classifier.tflite
keypoint_classifier_label.csv
point_history_classifier
__init__.py
utils
.gitignore
app.py
keypoint_classification.ipynb
keypoint_classification_EN.ipynb
point_history_classification.ipynb
External Libraries
Scratches and Consoles
keypoint_classification_EN.ipynb
point_history_classification.ipynb
keypoint.csv
keypoint_classifier.py
keypoint_classifier_label.csv
image.flags.writeable = True
if results.multi_hand_landmarks is not None:
for hand_landmarks, handedness in zip(results.multi_hand_landmarks,
results.multi_handedness):
# Bounding box calculation
brect = calc_bounding_rect(debug_image, hand_landmarks)
# Landmark calculation
landmark_list = calc_landmark_list(debug_image, hand_landmarks)
# Conversion to relative coordinates / normalized coordinates
pre_processed_landmark_list = pre_process_landmark(
landmark_list)
pre_processed_point_history_list = pre_process_point_history(
debug_image, point_history)
Write to the dataset file
draw_landmarks() if len(landmark_point) > 0
Run: app
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0
Version Control
Run
Python Packages
TODO
Python Console
Problems
Terminal
Services
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (today 10:56)
367:26 LF UTF-8 4 spaces Python 3.10
Search
ENG
IN
13:21
11-11-2022

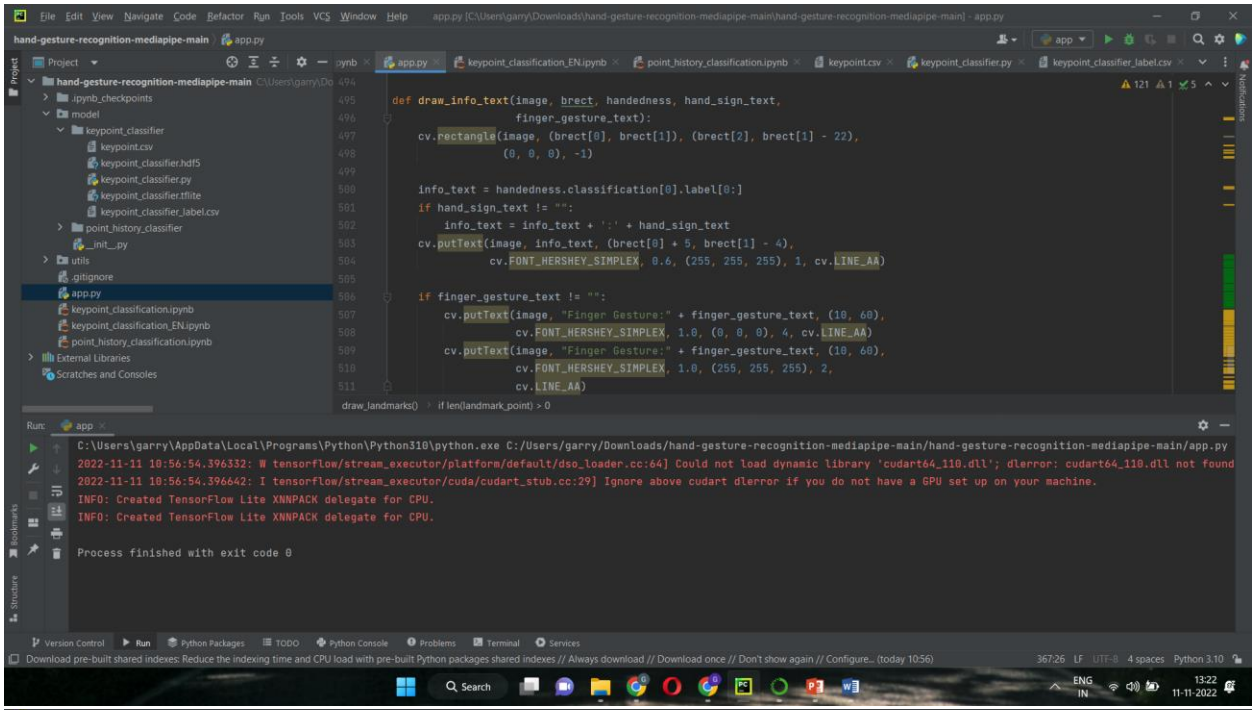

File Edit View Navigate Code Refactor Run Tools VCS Window Help
app.py [C:\Users\garry\Downloads\hand-gesture-recognition-mediapipe-main\hand-gesture-recognition-mediapipe-main] - app.py
hand-gesture-recognition-mediapipe-main
app.py
Project
hand-gesture-recognition-mediapipe-main
.ipynb_checkpoints
model
keypoint_classifier
keypoint.csv
keypoint_classifier.hdf5
keypoint_classifier.py
keypoint_classifier.tflite
keypoint_classifier_label.csv
point_history_classifier
__init__.py
utils
.gitignore
app.py
keypoint_classification.ipynb
keypoint_classification_EN.ipynb
point_history_classification.ipynb
External Libraries
Scratches and Consoles
keypoint_classification_EN.ipynb
point_history_classification.ipynb
keypoint.csv
keypoint_classifier.py
keypoint_classifier_label.csv
point_history_len = len(pre_processed_point_history_list)
if point_history_len == (history_length * 2):
finger_gesture_id = point_history_classifier(
pre_processed_point_history_list)
# Calculates the gesture IDs in the latest detection
finger_gesture_history.append(finger_gesture_id)
most_common_fg_id = Counter(
finger_gesture_history).most_common()
# Drawing part
debug_image = draw_bounding_rect(use_brect, debug_image, brect)
debug_image = draw_landmarks(debug_image, landmark_list)
debug_image = draw_info_text(
debug_image,
brect,
handedness,
keypoint_classifier_labels[hand_sign_id],
draw_landmarks() if len(landmark_point) > 0
Run: app
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0
Version Control
Run
Python Packages
TODO
Python Console
Problems
Terminal
Services
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (today 10:56)
367:26 LF UTF-8 4 spaces Python 3.10
Search
ENG
IN
13:21
11-11-2022

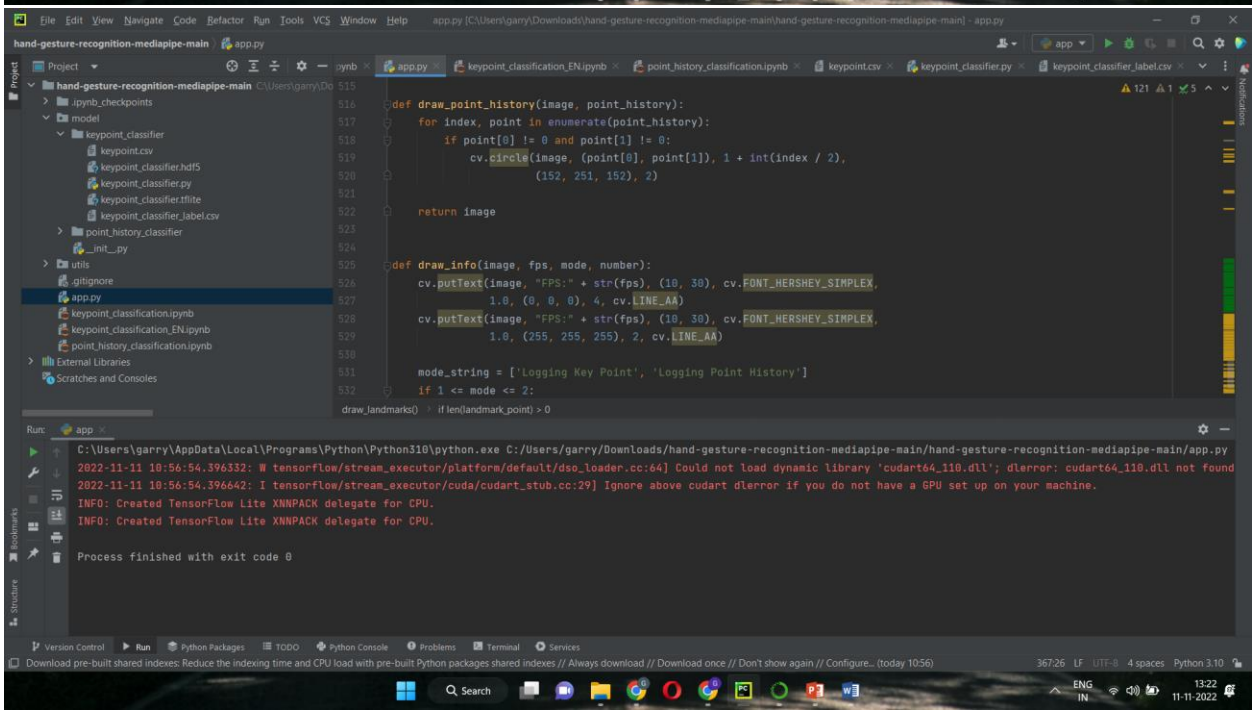

File Edit View Navigate Code Refactor Run Tools VCS Window Help
app.py [C:\Users\garry\Downloads\hand-gesture-recognition-mediapipe-main\hand-gesture-recognition-mediapipe-main] - app.py
hand-gesture-recognition-mediapipe-main
app.py
app
Project
hand-gesture-recognition-mediapipe-main C:\Users\garry\Do
.ipynb_checkpoints
model
keypoint_classifier
keypoint.csv
keypoint_classifier.hdf5
keypoint_classifier.py
keypoint_classifier.tflite
keypoint_classifier_label.csv
point_history_classifier
__init__.py
utils
.gitignore
app.py
keypoint_classification.ipynb
keypoint_classification_EN.ipynb
point_history_classification.ipynb
External Libraries
Scratches and Consoles
.ipynb
app.py
keypoint_classification_EN.ipynb
point_history_classification.ipynb
keypoint.csv
keypoint_classifier.py
keypoint_classifier_label.csv
point_history_classifier_labels[most_common_fg_id[0][0]],
)
else:
point_history.append([0, 0])
debug_image = draw_point_history(debug_image, point_history)
debug_image = draw_info(debug_image, fps, mode, number)
# Screen reflection ###
cv.imshow('Hand Gesture Recognition', debug_image)
cap.release()
cv.destroyAllWindows()
def select_mode(key, mode):
number = -1
if 48 <= key <= 57: # 0 ~ 9
draw_landmarks() > if len(landmark_point) > 0
Run:
app
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0
Version Control
Run
Python Packages
TODO
Python Console
Problems
Terminal
Services
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (today 10:56)
367:26 LF UTF-8 4 spaces Python 3.10
Search
ENG
IN
13:21
11-11-2022

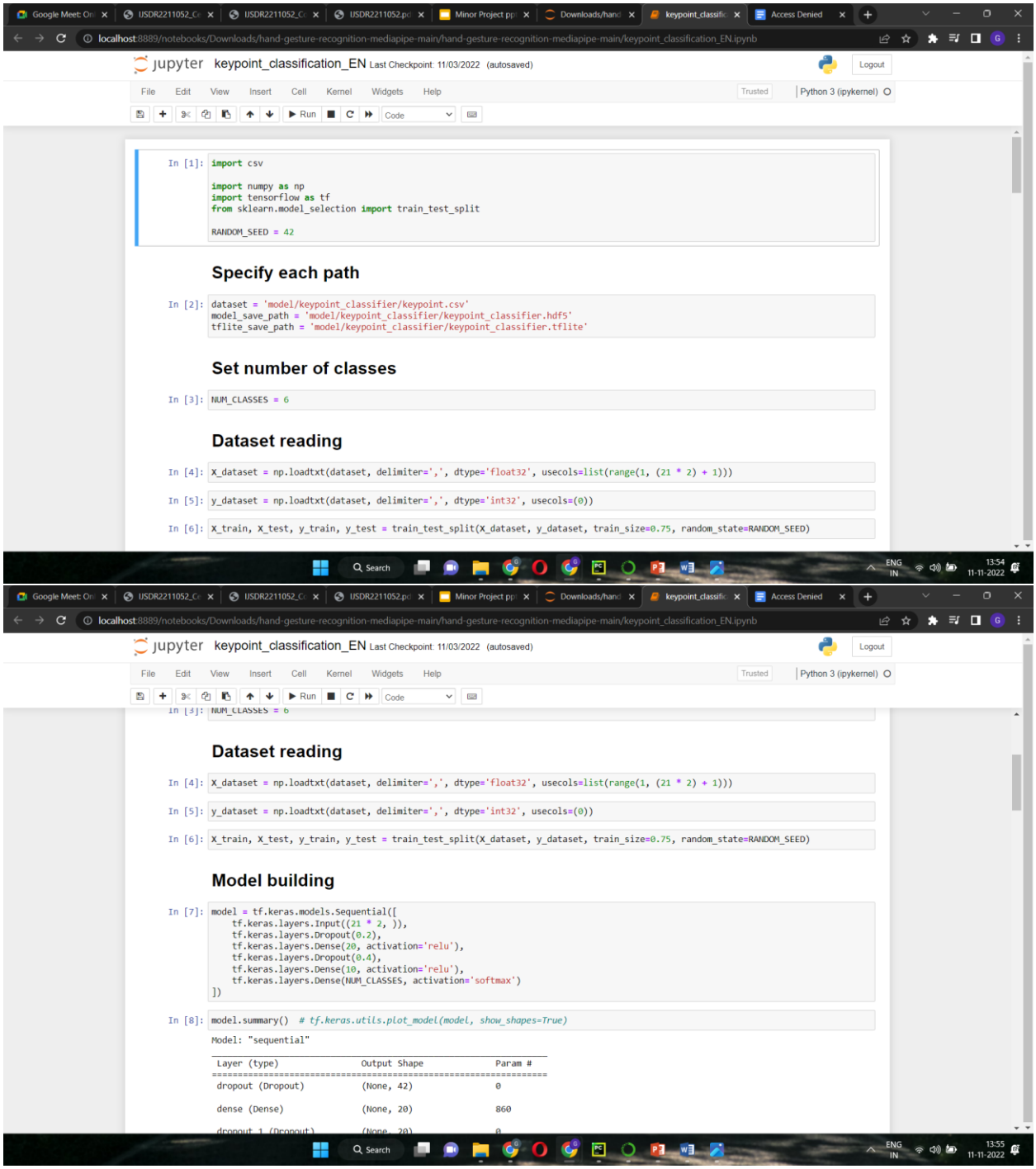

File Edit View Navigate Code Refactor Run Tools VCS Window Help
app.py [C:\Users\garry\Downloads\hand-gesture-recognition-mediapipe-main\hand-gesture-recognition-mediapipe-main] - app.py
hand-gesture-recognition-mediapipe-main
app.py
app
Project
hand-gesture-recognition-mediapipe-main C:\Users\garry\Do
.ipynb_checkpoints
model
keypoint_classifier
keypoint.csv
keypoint_classifier.hdf5
keypoint_classifier.py
keypoint_classifier.tflite
keypoint_classifier_label.csv
point_history_classifier
__init__.py
utils
.gitignore
app.py
keypoint_classification.ipynb
keypoint_classification_EN.ipynb
point_history_classification.ipynb
External Libraries
Scratches and Consoles
.ipynb
app.py
keypoint_classification_EN.ipynb
point_history_classification.ipynb
keypoint.csv
keypoint_classifier.py
keypoint_classifier_label.csv
temp_landmark_list[index][1] = temp_landmark_list[index][1] - base_y
# Convert to a one-dimensional list
temp_landmark_list = list(
itertools.chain.from_iterable(temp_landmark_list))
# Normalization
max_value = max(list(map(abs, temp_landmark_list)))
def normalize_(n):
return n / max_value
temp_landmark_list = list(map(normalize_, temp_landmark_list))
return temp_landmark_list
def pre_process_point_history(image, point_history):
draw_landmarks() > if len(landmark_point) > 0
Run:
app
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0
Version Control
Run
Python Packages
TODO
Python Console
Problems
Terminal
Services
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (today 10:56)
367:26 LF UTF-8 4 spaces Python 3.10
Search
ENG
IN
13:21
11-11-2022

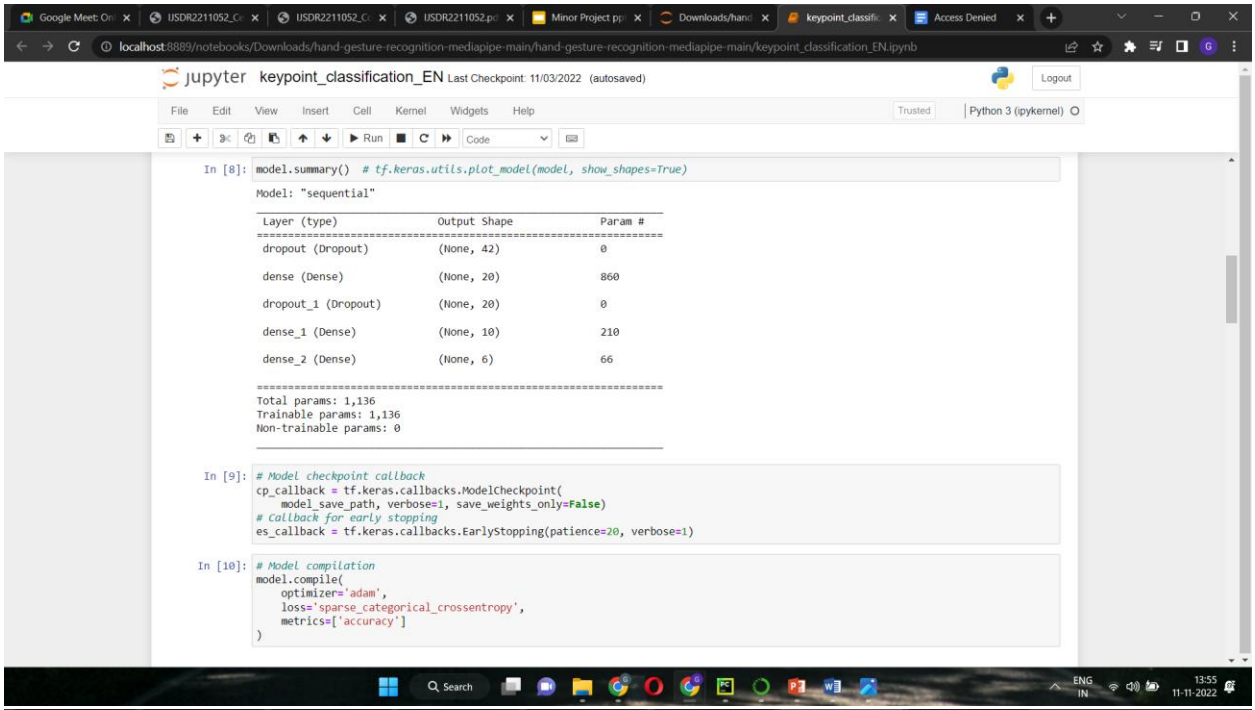

def pre_process_point_history(image, point_history):
image_width, image_height = image.shape[1], image.shape[0]
temp_point_history = copy.deepcopy(point_history)
# Convert to relative coordinates
base_x, base_y = 0, 0
for index, point in enumerate(temp_point_history):
if index == 0:
base_x, base_y = point[0], point[1]
temp_point_history[index][0] = (temp_point_history[index][0] -
base_x) / image_width
temp_point_history[index][1] = (temp_point_history[index][1] -
base_y) / image_height
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0

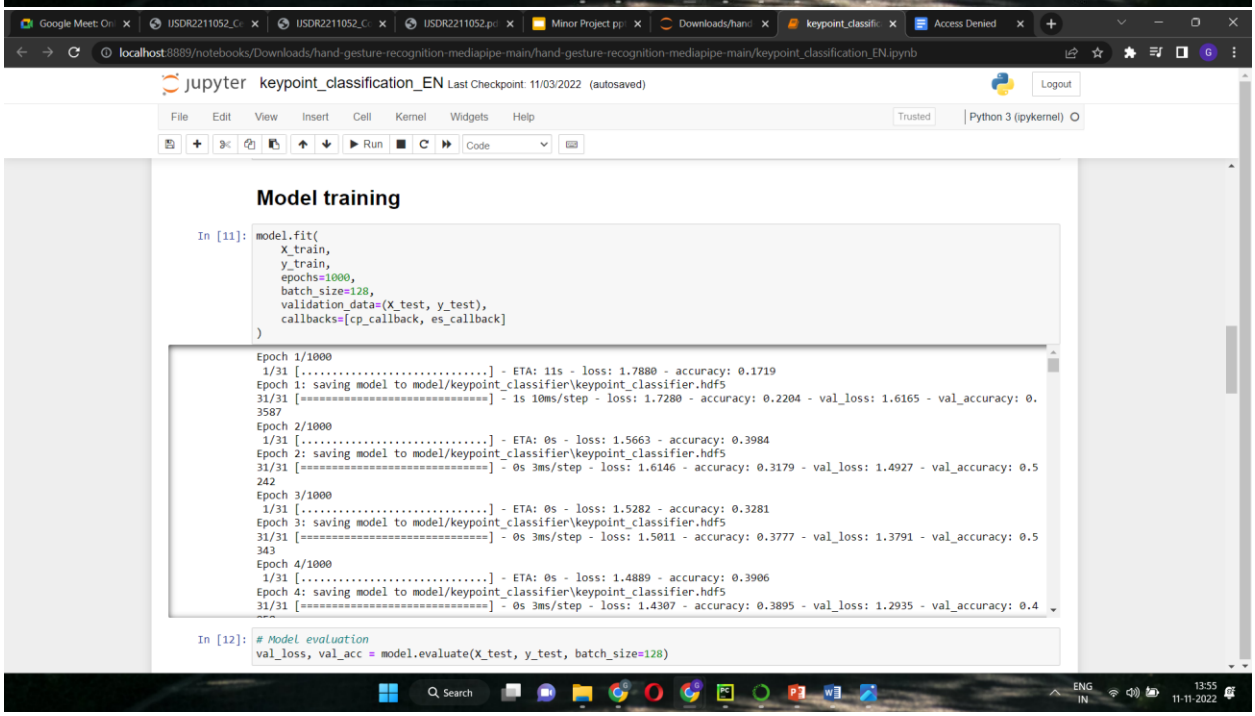

def logging_csv(number, mode, landmark_list, point_history_list):
if mode == 0:
pass
if mode == 1 and (0 <= number <= 9):
csv_path = 'model/keypoint_classifier/keypoint.csv'
with open(csv_path, 'a', newline="") as f:
writer = csv.writer(f)
writer.writerow([number, *landmark_list])
if mode == 2 and (0 <= number <= 9):
csv_path = 'model/point_history_classifier/point_history.csv'
with open(csv_path, 'a', newline="") as f:
writer = csv.writer(f)
writer.writerow([number, *point_history_list])
return
def draw_landmarks(image, landmark_point):
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0

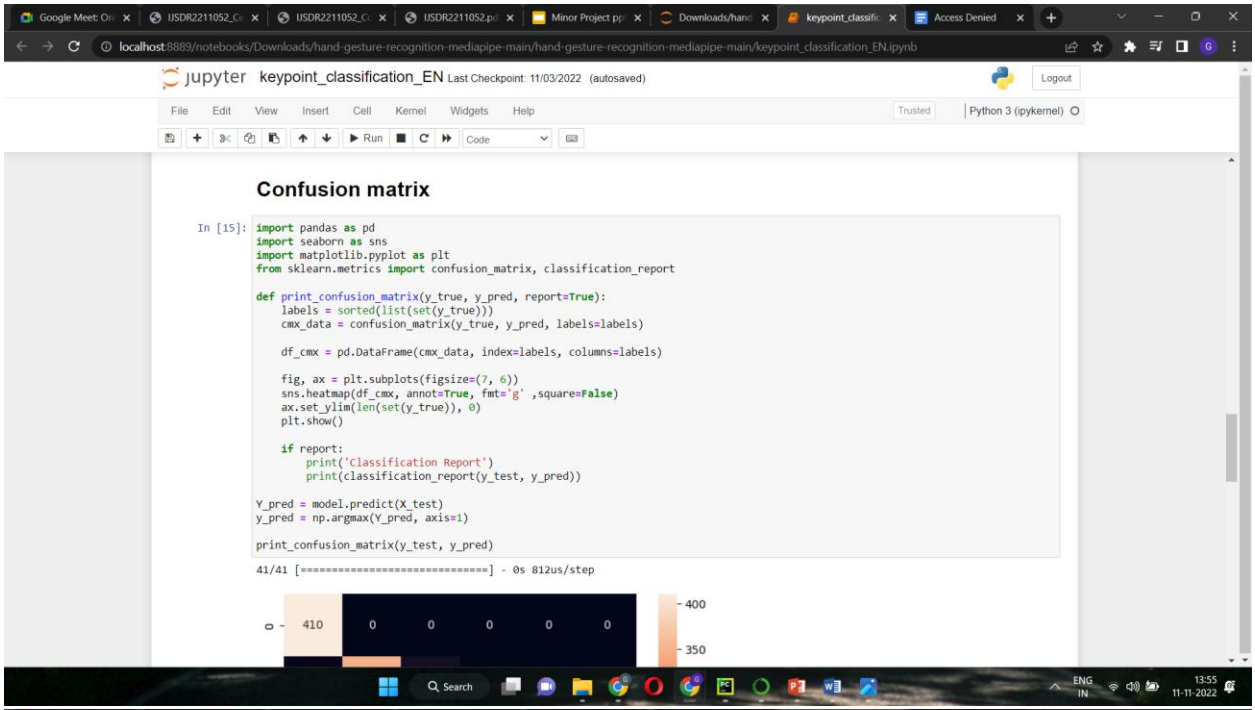

def draw_info_text(image, brect, handedness, hand_sign_text,
finger_gesture_text):
cv.rectangle(image, (brect[0], brect[1]), (brect[2], brect[1] - 22),
(0, 0, 0), -1)
info_text = handedness.classification[0].label[0:]
if hand_sign_text != "":
info_text = info_text + ':' + hand_sign_text
cv.putText(image, info_text, (brect[0] + 5, brect[1] - 4),
cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1, cv.LINE_AA)
if finger_gesture_text != "":
cv.putText(image, "Finger Gesture:" + finger_gesture_text, (10, 60),
cv.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 0), 4, cv.LINE_AA)
cv.putText(image, "Finger Gesture:" + finger_gesture_text, (10, 60),
cv.FONT_HERSHEY_SIMPLEX, 1.0, (255, 255, 255), 2,
cv.LINE_AA)
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0

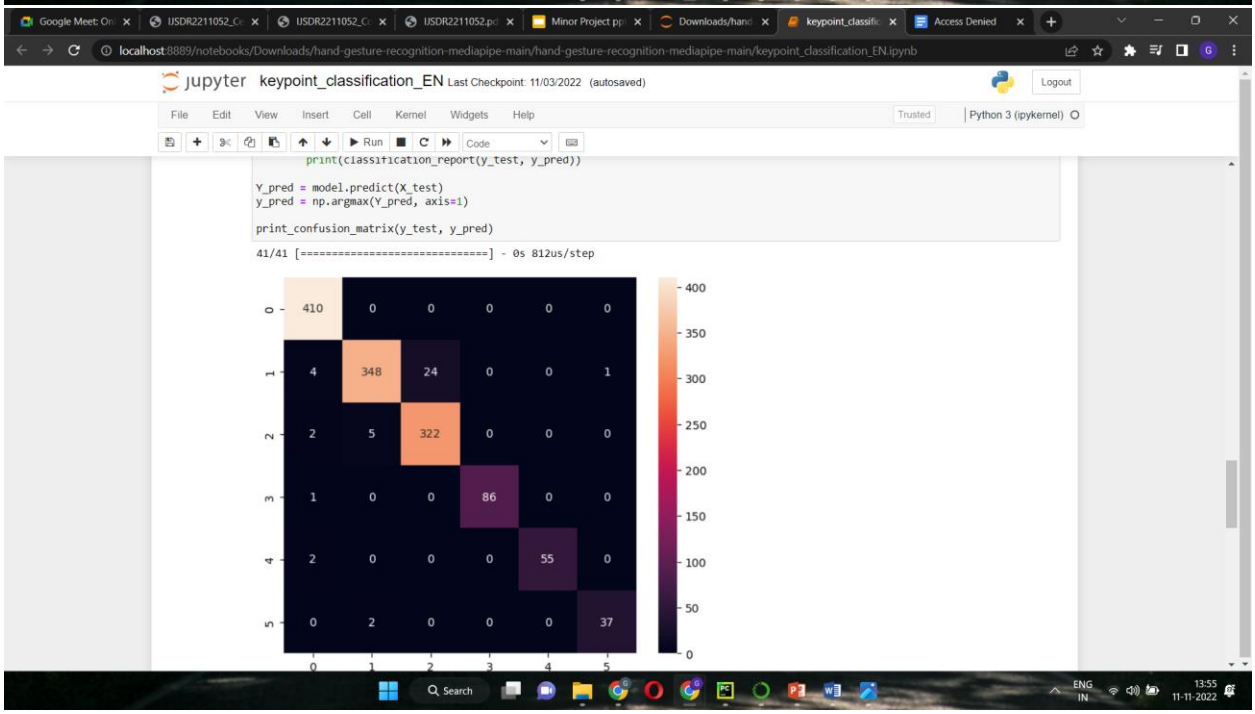

def draw_point_history(image, point_history):
for index, point in enumerate(point_history):
if point[0] != 0 and point[1] != 0:
cv.circle(image, (point[0], point[1]), 1 + int(index / 2),
(152, 251, 152), 2)
return image
def draw_info(image, fps, mode, number):
cv.putText(image, "FPS:" + str(fps), (10, 30), cv.FONT_HERSHEY_SIMPLEX,
1.0, (0, 0, 0), 4, cv.LINE_AA)
cv.putText(image, "FPS:" + str(fps), (10, 30), cv.FONT_HERSHEY_SIMPLEX,
1.0, (255, 255, 255), 2, cv.LINE_AA)
mode_string = ['Logging Key Point', 'Logging Point History']
if 1 <= mode <= 2:
C:\Users\garry\AppData\Local\Programs\Python\Python310\python.exe C:/Users/garry/Downloads/hand-gesture-recognition-mediapipe-main/hand-gesture-recognition-mediapipe-main/app.py
2022-11-11 10:56:54.396332: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-11 10:56:54.396642: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Process finished with exit code 0

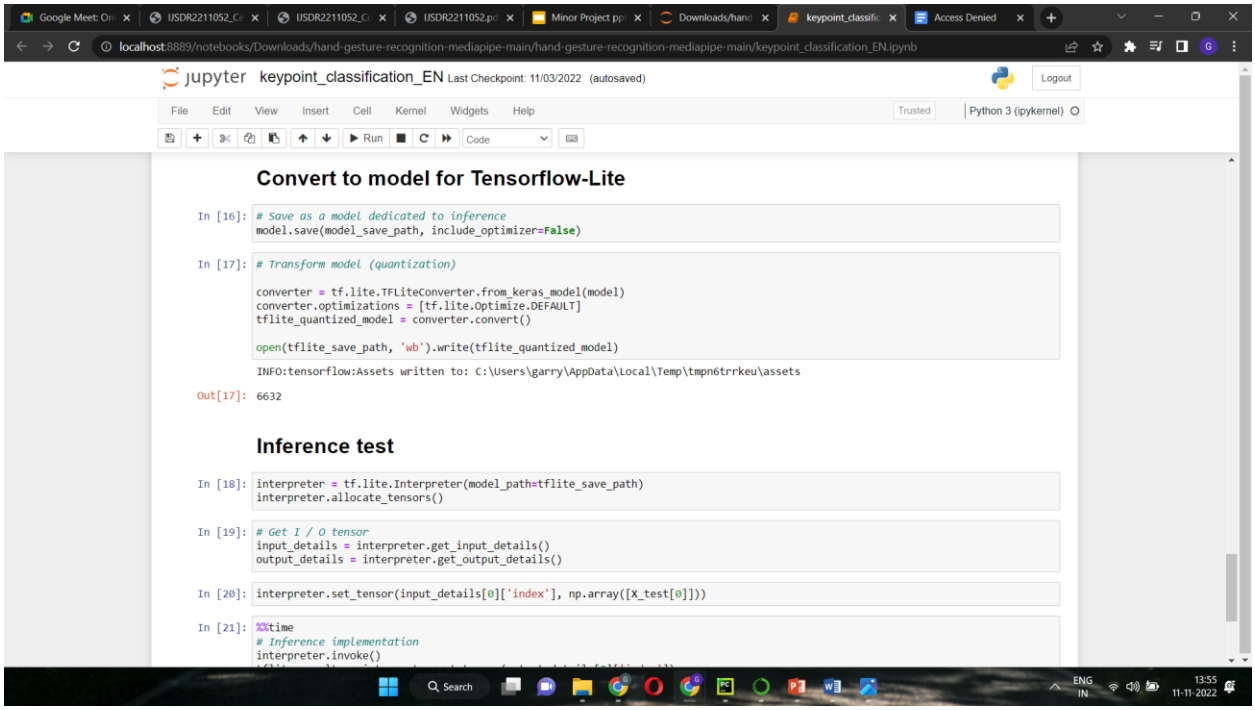

jupyter keypoint_classification_EN Last Checkpoint: 11/03/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [1]:
import csv
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
RANDOM_SEED = 42
Specify each path
In [2]:
dataset = 'model/keypoint_classifier/keypoint.csv'
model_save_path = 'model/keypoint_classifier/keypoint_classifier.hdf5'
tflite_save_path = 'model/keypoint_classifier/keypoint_classifier.tflite'
Set number of classes
In [3]:
NUM_CLASSES = 6
Dataset reading
In [4]:
X_dataset = np.loadtxt(dataset, delimiter=',', dtype='float32', usecols=list(range(1, (21 * 2) + 1)))
In [5]:
y_dataset = np.loadtxt(dataset, delimiter=',', dtype='int32', usecols=(0))
In [6]:
X_train, X_test, y_train, y_test = train_test_split(X_dataset, y_dataset, train_size=0.75, random_state=RANDOM_SEED)
Model building
In [7]:
model = tf.keras.models.Sequential([
tf.keras.layers.Input((21 * 2,)),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(20, activation='relu'),
tf.keras.layers.Dropout(0.4),
tf.keras.layers.Dense(10, activation='relu'),
tf.keras.layers.Dense(NUM_CLASSES, activation='softmax')
])
In [8]:
model.summary() # tf.keras.utils.plot_model(model, show_shapes=True)
Model: "sequential"
Layer (type) Output Shape Param #
dropout (Dropout) (None, 42) 0
dense (Dense) (None, 20) 860
dropout_1 (Dropout) (None, 20) 0


jupyter keypoint_classification_EN Last Checkpoint: 11/03/2022 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
In [8]:
model.summary() # tf.keras.utils.plot_model(model, show_shapes=True)
Model: "sequential"
Layer (type) Output Shape Param #
dropout (Dropout) (None, 42) 0
dense (Dense) (None, 20) 860
dropout_1 (Dropout) (None, 20) 0
dense_1 (Dense) (None, 10) 210
dense_2 (Dense) (None, 6) 66
Total params: 1,136
Trainable params: 1,136
Non-trainable params: 0
In [9]:
# Model checkpoint callback
cp_callback = tf.keras.callbacks.ModelCheckpoint(
model_save_path, verbose=1, save_weights_only=False)
# Callback for early stopping
es_callback = tf.keras.callbacks.EarlyStopping(patience=20, verbose=1)
In [10]:
Model compilation
model.compile(
optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy']
)


jupyter keypoint_classification_EN Last Checkpoint: 11/03/2022 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Model training
In [11]:
model.fit(
X_train,
y_train,
epochs=1000,
batch_size=128,
validation_data=(X_test, y_test),
callbacks=[cp_callback, es_callback]
)
Epoch 1/1000
1/31 [..............................] - ETA: 11s - loss: 1.7880 - accuracy: 0.1719
Epoch 1: saving model to model/keypoint_classifier\keypoint_classifier.hdf5
31/31 [==============================] - 1s 10ms/step - loss: 1.7280 - accuracy: 0.2204 - val_loss: 1.6165 - val_accuracy: 0.3587
Epoch 2/1000
1/31 [..............................] - ETA: 0s - loss: 1.5663 - accuracy: 0.3984
Epoch 2: saving model to model/keypoint_classifier\keypoint_classifier.hdf5
31/31 [==============================] - 0s 3ms/step - loss: 1.6146 - accuracy: 0.3179 - val_loss: 1.4927 - val_accuracy: 0.5242
Epoch 3/1000
1/31 [..............................] - ETA: 0s - loss: 1.5282 - accuracy: 0.3281
Epoch 3: saving model to model/keypoint_classifier\keypoint_classifier.hdf5
31/31 [==============================] - 0s 3ms/step - loss: 1.5011 - accuracy: 0.3777 - val_loss: 1.3791 - val_accuracy: 0.5343
Epoch 4/1000
1/31 [..............................] - ETA: 0s - loss: 1.4889 - accuracy: 0.3906
Epoch 4: saving model to model/keypoint_classifier\keypoint_classifier.hdf5
31/31 [==============================] - 0s 3ms/step - loss: 1.4307 - accuracy: 0.3895 - val_loss: 1.2935 - val_accuracy: 0.4
In [12]:
Model evaluation
val_loss, val_acc = model.evaluate(X_test, y_test, batch_size=128)


jupyter keypoint_classification_EN Last Checkpoint: 11/03/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Confusion matrix
In [15]:
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
def print_confusion_matrix(y_true, y_pred, report=True):
labels = sorted(list(set(y_true)))
cmx_data = confusion_matrix(y_true, y_pred, labels=labels)
df_cmx = pd.DataFrame(cmx_data, index=labels, columns=labels)
fig, ax = plt.subplots(figsize=(7, 6))
sns.heatmap(df_cmx, annot=True, fmt='g' ,square=False)
ax.set_ylim(len(set(y_true)), 0)
plt.show()
if report:
print('Classification Report')
print(classification_report(y_test, y_pred))
Y_pred = model.predict(X_test)
y_pred = np.argmax(Y_pred, axis=1)
print_confusion_matrix(y_test, y_pred)
41/41 [==============================] - 0s 812us/step
410 0 0 0 0 0
400
350
Search
ENG IN
13:55 11-11-2022

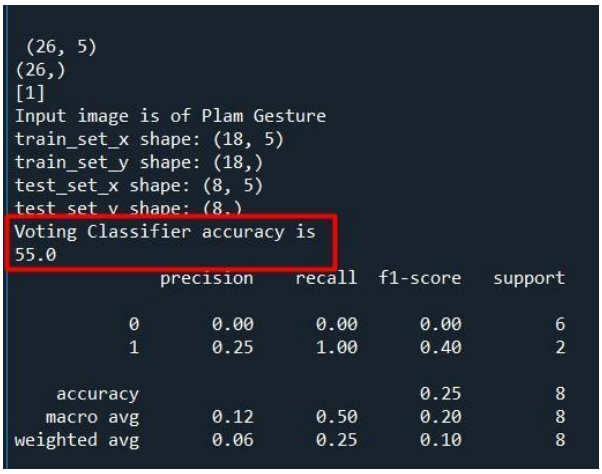

jupyter keypoint_classification_EN Last Checkpoint: 11/03/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
print(classification_report(y_test, y_pred))
Y_pred = model.predict(X_test)
y_pred = np.argmax(Y_pred, axis=1)
print_confusion_matrix(y_test, y_pred)
41/41 [==============================] - 0s 812us/step
410 0 0 0 0 0
4 348 24 0 0 1
2 5 322 0 0 0
1 0 0 86 0 0
2 0 0 0 55 0
0 2 0 0 0 37
0 1 2 3 4 5
400
350
300
250
200
150
100
50
0
Search
ENG IN
13:55 11-11-2022


jupyter keypoint_classification_EN Last Checkpoint: 11/03/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Convert to model for Tensorflow-Lite
In [16]:
# Save as a model dedicated to inference
model.save(model_save_path, include_optimizer=False)
In [17]:
# Transform model (quantization)
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_quantized_model = converter.convert()
open(tflite_save_path, 'wb').write(tflite_quantized_model)
INFO:tensorflow:Assets written to: C:\Users\garry\AppData\Local\Temp\tmpn6trrkeu\assets
Out[17]:
6632
Inference test
In [18]:
interpreter = tf.lite.Interpreter(model_path=tflite_save_path)
interpreter.allocate_tensors()
In [19]:
# Get I / O tensor
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
In [20]:
interpreter.set_tensor(input_details[0]['index'], np.array([X_test[0]]))
In [21]:
%%time
# Inference implementation
interpreter.invoke()
Search
ENG
IN
13:55
11-11-2022

CHAPTER 5

RESULTS

5.1 RESULT OF BASE PAPER IMPLEMENTATION



```
 (26, 5)
(26,)
[1]
Input image is of Plam Gesture
train_set_x shape: (18, 5)
train_set_y shape: (18,)
test_set_x shape: (8, 5)
test_set_y shape: (8,)
Voting Classifier accuracy is
55.0
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         6
           1       0.25      1.00      0.40         2

    accuracy                           0.25         8
   macro avg       0.12      0.50      0.20         8
weighted avg       0.06      0.25      0.10         8
```

Fig 5.1 Base paper Result

Base paper python implementation gives 55% accuracy on the image dataset

5.2 Result of new implementation

New implementation gives higher accuracy with proposed changes on the same dataset

Result Analysis

```
Classification Report
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       392
           1       0.97      0.96      0.96       383
           2       0.95      0.97      0.96       340
           3       0.94      1.00      0.97        77
           4       1.00      1.00      1.00        65

    accuracy                           0.97      1257
   macro avg       0.97      0.98      0.97      1257
weighted avg       0.97      0.97      0.97      1257
```

Fig 5.2 Research Paper result

The implementation results shown in the images above demonstrate how utilizing hybrid classification, the base paper's implementation results on the same dataset may be improved.

Simply modifying the method and using the suggested classifier combinations resulted in a significant gain in accuracy in the same context.

By adhering to the research plan, the results were improved. It was discovered that capturing more features improves classification and provides better accuracy than working with minimal features, so more features like contrast, dissimilarity, homogeneity, ASM, energy, and correlation were targeted for capture.

CHAPTER 6

CONCLUSION

It has been noted that gesture detection is utilised in many industries and creates a bridge between human and computer connection and comprehension; as a result, it has to be applied to new domains in order to reap its advantages. More effective object identification and detection techniques can be utilised to minimise the complexity in the future since gesture recognition uses advanced machine learning algorithms, which are quite difficult in terms of time complexity and efficiency. Gaming is a very common trend among young people today, and since most popular games use gesture recognition, there is a significant chance that they will be used in the future. If we can improve accuracy by using various techniques, as I have suggested in my thesis, it could elevate user experience to a whole new level.

CHAPTER 7

RELATED WORK

Xing Guo, et.al (2019) in order to solve the problems of low recognition rate and less recognition gesture categories caused by incomplete artificial feature extraction information in traditional static gesture recognition methods. a deep CNN framework is designed by using the principle of convolution neural network (Convolutional Neural Network, CNN) to recognize static gesture movements. Combined with a variety of optimal structures of convolution neural network in deep learning, a model with independent static gesture recognition function is realized. The model method can not only ensure the high accuracy and robustness of the recognition results, but also achieve the speed of smooth recognition.[1]

Roman Golovanov, et.al (2020) presents a combined hand gesture recognition system that uses a hand detector to detect hand in the frame and then switches to gesture classifier if a hand was detected. The paper illustrates the proposed combined algorithm. Descriptions of used hand detector and gesture recognition algorithms also are given. Equations for the evaluation of potential performance increase and experimental results are presented. The proposed system is tested on publicly accessible gesture bases and on video sequences prepared by the authors. The experimental results are consistent with theoretical estimates and demonstrate the benefits of the proposed gesture recognition system design.[2]

Qinglian Yang, et.al (2020) proposes a new gesture recognition system based on Deep Neural Network (DNN) and Leap Motion [10]. The palm model is reconstructed to obtain the feature data, and then the feature data of all experimenters are obtained by the Leap Motion controller. The data are finally put into the DNN model for training to implement the recognition of specific hand gesture after being normalized. By testing with 30000 gesture frames of five volunteers, it is found that the recognition accuracy of the proposed scheme can reach 98\%, which indicates that the scheme can complete the specific gesture recognition with a high average recognition rate.[3]

Depeng Zhu, et.al (2019) a soldier identification intelligent recognition system was designed. First, the system performs data normalization and endpoint detection on the collected gesture information. Then, feature extraction of the processed data such as mean, peak-to-peak and root mean square values in the time domain. Finally, the dynamic time warping (DTW) algorithm is used to calculate the similarity between the test gesture and the template gesture for the extracted feature parameters, and then the recognition result is obtained. The experimental results show that the system has the characteristics of high recognition accuracy. Therefore, it is suitable for gesture recognition and communication when performing combat missions.[4]

Xunlei Zhang, et.al (2020) paper adopts 3D separable convolution as an alternative solution. To further extract semantic and action information of gestures, we combine attentional mechanisms and long- and short-term memory. To further extract semantic and action information of gestures, we combine attentional mechanisms and long- and short-term memory (LSTM) networks for gesture recognition in this paper. We conducted experiments on the ChaLearn Large-Scale Gesture Recognition Dataset (IsoGD), and the experimental results validate the effectiveness of our method.[5]

Xuexiang Zhang, et.al (2019) the use of gesture recognition in industrial production to control robots is mostly a gesture-oriented teaching method, lacking a systematic description of dynamic gestures and static gestures, making it difficult for the robot to understand the complete intention expressed by the operator. The static gesture real-time recognition is centered and the computer vision control in the complex environment realizes the space movement and posture movement of the robot. This way of using the human body language to directly control the robot movement, the human experience, the intention and the manipulator's high efficiency The combination of sustainability and other advantages can accomplish tasks that cannot be accomplished by people or robots alone.[6]

Liying Wang, et.al (2019) a novel method of high precision fine-grained gesture recognition is proposed based on a terahertz radar, which is able to sense any gesture movement when its range of motion is greater than 5mm. First, High Resolution Range Profile (HRRP) sequences are extracted from radar echoes. Then, the HRRP features are fed to Random Forest classifier after dimensionality reduction by PCA. In order to verify the proposed method is effective to

detect fine-grained gestures, four kinds of similar gestures with multi-fingers are designed for experiments. The results indicate the recognition rate exceeds 99.7%, which demonstrates a great prospect in fine-grained gesture recognition using a terahertz radar.[7] Xingxiu He, et.al (2020) designs a number gesture recognition system based on Kinect. The system uses the depth image captured by Kinect sensor to extract gesture information for recognition. In the process of recognition, the AdaBoost algorithm in machine learning is adopted firstly, by the help of Visual Gesture Builder to train the number gesture classifier, and export the classifiers to the recognition program to recognize the current gesture. Finally, numbers are outputted according to the recognized gesture, and the basic functions of number gesture recognition system are realized. Through the experimental verification, the recognition system has enough accuracy.

CHAPTER 8

REFERENCES

[1] Xing Guo, Wu Xu, Wen Quan Tang, Cong Wen, “Research on Optimization of Static Gesture Recognition Based on Convolution Neural Network”, 2019, 4th International Conference on Mechanical, Control and Computer Engineering (ICMCCE).

[2] Roman Golovanov, Dmitry Vorotnev, Darina Kalina, “Combining Hand Detection and Gesture Recognition Algorithms for Minimizing Computational Cost”, 2020, 22th International Conference on Digital Signal Processing and its Applications (DSPA).

[3] Qinglian Yang, Weikang Ding, Xingwen Zhou, Dongdong Zhao, Shi Yan, “Leap Motion Hand Gesture Recognition Based on Deep Neural Network”, 2020, Chinese Control and Decision Conference (CCDC).

[4] Depeng Zhu, Ranran Wei, Weida Zhan, Ziqiang Hao, “Individual Soldier Gesture Intelligent Recognition System”, 2019, IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS).

[5]Xunlei Zhang, Yun Tie, Lin Qi, “Dynamic Gesture Recognition Based on 3D Separable Convolutional LSTM Networks”, 2020, IEEE 11th International Conference on Software Engineering and Service Science (ICSESS).

[6] Xuexiang Zhang, Xuncheng Wu, “Robotic Control of Dynamic and Static Gesture Recognition”, 2019, 2nd World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM).

[7] Liying Wang, Zongyong Cui, Zongjie Cao, Shengping Xu, Rui Min, “Fine-Grained Gesture Recognition Based on High Resolution Range Profiles of Terahertz Radar”, 2019, IEEE International Geoscience and Remote Sensing Symposium. Xingxiu He, Jia Zhang, “Design and Implementation of Number Gesture Recognition System Based on Kinect”, 2020, 39th Chinese Control Conference (CCC).

APPENDIX A

PAPER PUBLICATION

INTERNATIONAL JOURNAL OF SCIENCE & ENGINEERING DEVELOPMENT RESEARCH

An International Open Access Journal ISSN: 2455-2631

Certificate of Publication

The Board of

International Journal of Science & Engineering Development Research

Is hereby awarding this certificate to

Gaurav Chaturvedi

In recognition of the publication of the paper entitled

Hand Gesture Recognition

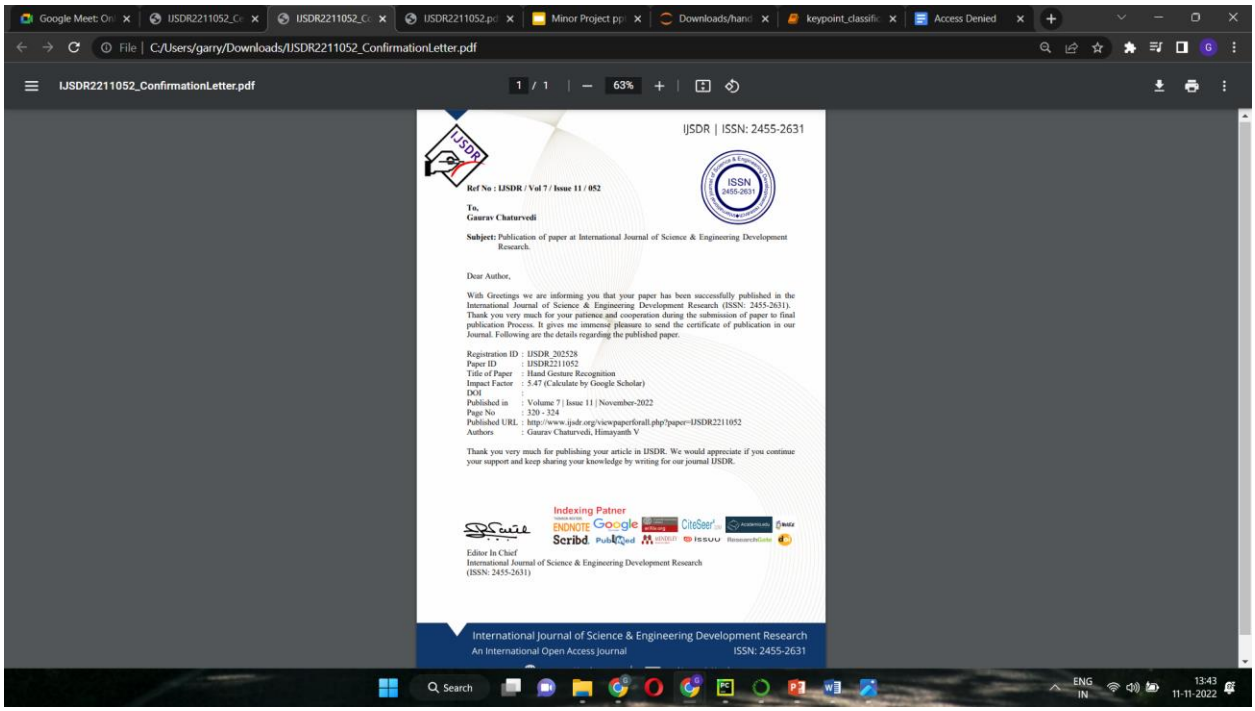
Published in Volume 7 Issue 11, November-2022

Co-Authors - Himayanth V

Paper ID - IJSDR2211052

Editor-In Chief

www.ijsdr.org | editor@ijsdr.org



IJSDR | ISSN: 2455-2631

Ref No : IJSDR / Vol 7 / Issue 11 / 052

To,
Gaurav Chaturvedi

Subject: Publication of paper at International Journal of Science & Engineering Development Research.

Dear Author,

With Greetings we are informing you that your paper has been successfully published in the International Journal of Science & Engineering Development Research (ISSN: 2455-2631). Thank you very much for your patience and cooperation during the submission of paper to final publication Process. It gives me immense pleasure to send the certificate of publication in our Journal. Following are the details regarding the published paper.

Registration ID : IJSDR_202528
Paper ID : IJSDR2211052
Title of Paper : Hand Gesture Recognition
Impact Factor : 5.47 (Calculate by Google Scholar)
DOI :
Published in : Volume 7 | Issue 11 | November-2022
Page No : 320 - 324
Published URL : http://www.ijsdr.org/viewpaperforall.php?paper=IJSDR2211052
Authors : Gaurav Chaturvedi, Himayanth V

Thank you very much for publishing your article in IJSDR. We would appreciate if you continue your support and keep sharing your knowledge by writing for our journal IJSDR.

Indexing Patner

Editor In Chief
International Journal of Science & Engineering Development Research
(ISSN: 2455-2631)

International Journal of Science & Engineering Development Research
An International Open Access Journal ISSN: 2455-2631

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University u/s 3 of UGC Act, 1956)

Office of Controller of Examinations

REPORT FOR PLAGIARISM CHECK ON THE SYNOPSIS/THESIS/DISSERTATION/PROJECT REPORTS

1	Name of the Candidate **(IN BLOCK LETTERS)**	GAURAV CHATURVEDI HIMAYANTH V
2	Address of the Candidate	107A, Ganpati Vihar, Line par, Moradabad 244001 **Mobile Number: 8939116857** 114, Ashwaraopalle, Jangaon, Telangana 506244 **Mobile Number: 9553763384**
3	Registration Number	RA1911031010119 RA1911031010088
4	Date of Birth	11th March 2002 27th May 2000
5	Department	Dept. of Networking and Communication
6	Faculty	Faculty of Engineering and Technology
7	Title of the Synopsis/ Thesis/ Dissertation/Project	Hand Gesture Recognition
8	Name and address of the Supervisor / Guide	Dr. Praveena Akki **Mail ID:** **praveena2@srmist.edu.in** **Mobile Number: 9502565299**
9	Name and address of the Co-Supervisor / Co- Guide (if any)	**N.A.**
10	Software Used	Turnitin
11	Date of Verification	12.11.2022

12	**Plagiarism Details: (to attach the final report)**			
Chapter	**Title of the Chapter**	**Percentage of similarity index (including self-citation)**	**Percentage of similarity index (Excluding self-citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1	Introduction	<1%	<1%	<1%
2	Literature Survey	<1%	<1%	<1%
3	System Architecture Design	<1%	<1%	<1%
4	Methodology	4%	4%	4%
5	Coding and Testing	<1%	<1%	<1%
6	Results and Discussion	<1%	<1%	<1%
7	Conclusion	<1%	<1%	<1%
8				
9				
10				
Thesis abstract		<1%	<1%	<1%
Appendices				

I / We declare that the above information has been verified and found true to the best of my / our knowledge.

Signature of the Candidate	**Signature of the Supervisor / Guide**
Signature of the Co-Supervisor/Co-Guide	**Signature of the HOD / DRCC Chairperson**

plag file

ORIGINALITY REPORT

4%	0%	2%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of West London Student Paper	3%
2	Xing Guo, Wu Xu, Wen Quan Tang, Cong Wen, "Research on Optimization of Static Gesture Recognition Based on Convolution Neural Network", 2019, (ICMCCE). Publication	1%
3	Roman Golovanov, Dmitry Vorotnev, Darina Kalina, "Combining Hand Detection and Gesture Recognition Algorithms for Minimizing Computational Cost", 2020, 22th International Conference on Digital Signal Processing and its Applications (DSPA). Publication	<1%
4	Qinglian Yang, Weikang Ding, Xingwen Zhou, Dongdong Zhao, Shi Yan, "Leap Motion Hand Gesture Recognition Based on Deep Neural Network", 2020, Chinese Control and Decision Conference (CCDC)	<1%