TABLE OF CONTENTS

CHAPTER I

INTRODUCTION AND DESIGN OF THE STUDY

1.1 INTRODUCTION

Humans operate computers basically with the help of mouse and keyboard. The mouse is a pointing device that allows users to move a cursor on the computer screen and select or click on different items. The keyboard is used to type in text and commands, as well as perform various functions within software applications. Apart from these two methods, computers can also be operated using physical hand. This project aims to develop a solution for Gesture enabled commands to operate Laptops/PCs for frequently used operations.

Object detection is a computer vision technique that involves identifying and locating objects of interest within an image or video stream. The goal of object detection is to not only recognize what is present in the image, but also to identify where each object is located within the image. Image classification can be defined as the task of categorizing images into one or multiple predefined classes. Although the task of categorizing an image is instinctive and habitual to humans, it is much more challenging for an automated system to recognize and classify images.

It is the process of assigning spectral classes into information classes. Spectral classes are groups of pixels that are uniform with respect to their brightness values in the different spectral channels of data. Information classes are categories of interest. In short, image classification is a process of assigning all pixels in the image to a particular class based on spectral information represented by the digital numbers (Anupam Anand, 2017).

Deep leaning, a subset of machine learning allows computational models to learn by gathering knowledge from experience. Complex concepts can be learnt due to its hierarchical conceptualization. It has an immense influence on fields such as Computer Vision, Natural Language Processing, Speech Recognition etc.

In this project, Mediapipe hands model has been used for analyzing the input video stream and identifying regions of interest that are likely to contain hands. It then processes each region using a deep neural network that predicts the location and confidence score for each hand landmark. The output hand landmarks of the Mediapipe hands model are fed in as input to classify the hand gestures into one of several predefined gesture categories, such as "thumbs up", "thumps down", “okay”, “fist”, “stop”, “rock”, “livelong”, “call me” etc. The pyAutoGUI library provides a set of functions to programmatically control mouse and keyboard actions. Hence, the output of the gesture recognition model i.e classId is connected with the pyautogui library. Overall, it helps to automate certain tasks based on the predicted category of the input data.

1.2 PROBLEM STATEMENT

The problem statement “Developing a solution for Gesture enabled commands for operating Laptops/PCs for frequently used operations on daily basis” has been listed in the official Smart India Hackathon (SIH) portal by DRDO.

For any user, authenticated by face recognition, few gestures could be defined for frequently used tasks- save, exit, print, screen-lock, screen unlock, system shut down, system restart. Save, print and exit operations are context sensitive meaning that it is applicable for current application. For example if word document is open and the gesture for save is done then the document will saved, if print gesture is done then printer dialog will open etc. Similarly a gesture could be defined for close/exit which will close the current application. If no application is opened then it will work as system shut down. It is similar to Alt+F4 key press functionality on windows PC.

1.3 OBJECTIVES OF THE STUDY

1. To develop a solution or software, for operating laptop or PC using physical hand gestures.
2. To use a model that can detect the hand in the image and saves the 21 hand landmarks of the detected hand image.
3. To classify the hand based on the gesture the user performs.
4. To connect the output of the classification model to a python library that allows to automate keyboard and mouse actions.
5. To feed in inputs to the model with webcam and test the results.

1.4 CHAPTER SCHEME

This project report is organized into six chapters as follows:

Chapter I _ Introduction

The chapter deals with the introduction, problem statetement, objectives, and the chapter scheme.

Chapter II – Review of Literature

In this chapter, the most relevant research and theories related to 'Gesture Enabled commands to operate laptops or PC' were discussed.

Chapter III – System Specification

In this chapter, the environment that has been used for developing the solution has been summarized.

Chapter IV – Methodology

This chapter presents the process flow, concepts and techniques that are used in this project.

Chapter V – Testing and Results

In this chapter, details regarding test data, test experiment and test results are summarised.

Chapter VI – Conclusion and Further work

In this chapter, details regarding the work conducted so far and directions to further work, has been presented and summarized.

CHAPTER II

REVIEW OF LITERATURE

2.1 INTRODUCTION

In this chapter, the most relevant research and theories related to ‘Gesture Enabled commands to operate laptops or PC’ were discussed.

2.2 RELATED RESEARCH AND THEORIES

(Jayesh Jidge, 2022) “Hand motion based computer activity control” stated that there are several new interactive methods, such as Sign and speech recognition, visual analysis, virtual reality, touch-free writing, brain activity have come out in recent years to achieve this goal. There is a need an alternative means to control computers and the technology needs to be robust so that it can be scaled on bigger boundaries. So, the proposed algorithm develops an alternative input means which can be useful in real time applications to control the daily computer activities like controlling media player, web browser, file manager, PowerPoint presentations and also for playing games through different gesture commands using a simple web camera. The main aim is to study and analyze the use of gesture recognition techniques for embedded systems by proposing a new model based on neural networks such as 2D and 3D CNNs able to perform gesture control using a simple web camera without any additional expensive hardware.

(Gopi Manoj Vuyyuru, 2021), Performing Basic Tasks on Computer using Hand Gestures & Ultrasonic Sensors stated that innovative approaches like hand gestures play a vital role in this emerging world and evolving time of Industry 4.0 because they are simply accessible and are efficient in detecting nearby objects. The need for hardware components such as the keyboard and mouse can be drastically reduced with this hand gesture technique. We utilize ultrasonic sensors to identify and characterize hand gestures in real-time in this paper. Ultrasonic sensors use ultrasonic sound waves to measure the distance between a target item and then transform the reflected sound into an electrical signal. The main goal of the paper is to use Arduino and Ultrasonic sensors, as well as various Python packages, to enhance the precision and speed of the computer interface.

(Muhammad Sheikh Sadi, 2022) **“Finger-Gesture Controlled Wheelchair with Enabling IoT”** Stated that Modern wheelchairs, with advanced and robotic technologies, could not reach the life of millions of disabled people due to their high costs, technical limitations, and safety issues and proposed a gesture-controlled smart wheelchair system with an IoT-enabled fall detection mechanism to overcome these problems. It can recognize gestures using Convolutional Neural Network (CNN) model along with computer vision algorithms and can control the wheelchair automatically by utilizing these gestures. It maintains safety of the users by performing fall detection with IoT based emergency messaging systems. The development cost of the overall system is cheap and is lesser than USD 300. Hence, it is expected that the proposed smart wheelchair should be affordable, safe, and helpful to physically disordered people in their independent mobility.

(**Brimingham**) **“Gesture Control Technology: An investigation on the potential use in Higher Education”** mentioned that for decades, keyboards and mouse have been the main input devices for computers. However, with the rising popularity of ubiquitous and ambient devices, (i.e. PlayStations), and equipment which allows users to grasp virtual objects, hand or body gesture are becoming essential: gesture controls have become central to the human computer interaction system. Gesture recognition can be seen as a way for computers to begin to understand human body language. Compared to the primitive user interfaces, such as keyboard and mouse, it builds a richer bridge between the computers and humans. Gesture control devices recognize and interpret human body movements, allowing the user to interact with a computer system.

(Ayushi Bhawal) “Arduino Based Hand Gesture Control of Computer” tried building our own Gesture Control Laptop/Computer by combining the power of Arduino and Python. We will use two Ultrasonic sensors to determine the position of our hand and control a media player (VLC) based on the position. We have used this for demonstration, but once we have understood the work, we can do anything by just changing few lines of code and control our favourite application in our favourite way. The concept behind this work is very simple. We will place two Ultrasonic sensors on top of our monitor and will read the distance between the monitor and our hand using Arduino, based on this value of distance we will perform certain actions. To perform actions on our computer we use Python Pyautogui library. The commands from Arduino are sent to the computer through serial port. This data will be then read by python which is running on the computer and based on the read data an action will be performed. The incoming time-domain signals are buffered, and Fourier transform is applied on them. The

Arduino can be connected to the PC/Laptop for powering the module and also for serial communication. The result of this operation is magnitude vectors that are spread equally over the spectral width. After each FFT vector is computed, it is further processed to determine the bandwidth of the signals, speed of gestures and motion detection. The detected motions are then converted to pc commands.

CHAPTER III

SYSTEM SPECIFICATION

3.1 INTRODUCTION

In this chapter, the environment that has been used for developing the solution has been summarized.

3.2 HARDWARE SPECIFICATION

S. N	CATEGORY	SPECIFICATION
1	IDE	PyCharm 2022.3.2 (Community Edition)
2	CPU	Intel(R) Pentium(R) CPU N4200 @ 1.10GHz
3	RAM	4 GB
4	Hard Disk Space	27.9 GB
5	Operating System	Windows 10.0.19044

PyCharm is an integrated development environment (IDE) used for Python programming. It is developed by JetBrains and is available in two editions: Community (free and open-source) and Professional (paid). For this project, the community edition has been used. This solution can run in CPU and GPU enabled machine is optional for this.

3.3 SOFTWARE SPECIFICATION

PROGRAMMING LANGUAGE		
S. N	**CATEGORY**	**VERSION**
1	Python	3.7.8

SOFTWARE LIBRARIES		
S. N	**CATEGORY**	**VERSION**
1	MediaPipe	0.9.0.1
2	Tensorflow	2.11.0
3	Keras	2.11.0
4	numpy.	1.21.6
5	Matplotlib	3.5.3
6	OS Module	Python standard version
7	OpenCV	4.7.0.72
8	PyAutoGUI	0.9.53
9	DateTime	5.1
10	Pip	22.3.1

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception.**Invalid source specified.**

Mediapipe – It is an open-source framework developed by Google for building machine learning (ML) pipelines. It provides a collection of pre-built ML models and tools allowing developers to easily customize and combine different modules to create new applications or enhance existing ones.

Tensorflow – Software library for Artificial Intelligence

Keras – Deep Learning API written in python running on top of Tensorflow2. This is also a cross-platform software can run on Tensorflow, Theano, CNTK, Pytorch

Numpy – Library used for working with arrays.

Matplotlib – cross platform data visualization and graphical plotting library for python

OS Module - The module contains several useful functions that help us to access, modify, and perform OS-related tasks such as access and modifying directories.

OpenCV – An open-source computer vision and machine learning software library. OpenCV provides a wide range of image and video processing algorithms, as well as machine learning tools, which can be used to analyze and interpret visual data from various sources, such as cameras and image files.

PyAutoGUI – A Python library for automating mouse and keyboard actions on a computer. It provides a simple and intuitive interface for performing tasks such as clicking, scrolling, typing, and taking screenshots using Python code.

Time - The time module in Python provides various time-related functions, including functions for getting the current time, setting time delays, and measuring the execution time of a program. It is a fundamental module in Python used for performing various time-related operations.

CHAPTER IV

METHODOLOGY

4.1 INTRODUCTION

In this chapter, the process flow, concepts and techniques that are used in this project are summarized.

4.2 PROCESS FLOW OF THE PROJECT

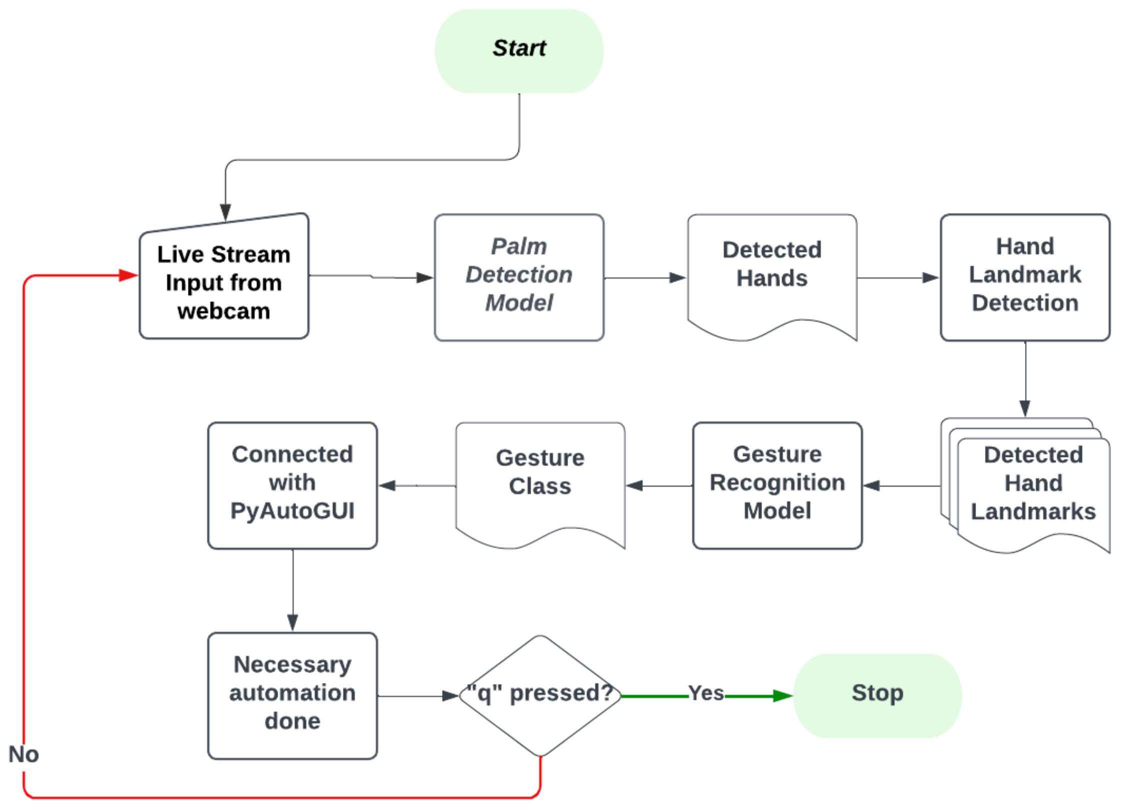


Fig 4.1: Methodology to detect Gesture

Start the Process. The input has been fed to the model via Live stream using web cam. Then the frames from the live stream are fed into the "Palm detection model". The model returns the cropped part of the hand. The hand landmark detection model has been called, it detects all the 21 keypoints of the hand and stores it the landmarks list. The list is then passed to the Gesture Recognition model. It outputs a class. PyAutoGUI has been linked with the class ID to automate necessary keyboard and mouse movements. Then, when the user enters q in the keyboard, the function or the program exits the loop and will stop the process.

4.3 CONCEPT – MEDIAPIPE:

MediaPipe is an open-source framework developed by Google for building machine learning (ML) pipelines to process multimedia data such as video and audio. It provides a collection of pre-built ML models and tools that allow developers to build complex multimedia processing pipelines for a variety of applications, including computer vision, gesture recognition, augmented reality, and more. With MediaPipe, developers can easily build custom ML pipelines using a variety of pre-built modules and tools, and run their ML pipelines on a wide range of devices including desktops, mobile devices, and embedded systems.

4.3.1 MEDIAPIPE HAND LANDMARK DETECTION:

MediaPipe Hand landmark detection (Mediapipe Hands for short) is a computer vision solution developed by Google's MediaPipe team that uses machine learning to detect and track human hands in real-time. It is based on a neural network that was trained on approximately 30K real-world images, as well as several rendered synthetic hand models imposed over various backgrounds, allowing it to accurately identify and track hand landmarks, such as fingertips and palm positions, even in challenging lighting and occlusion conditions.

The model works by first analyzing the input video stream and identifying regions of interest that are likely to contain hands. It then processes each region using a deep neural network that predicts the location and confidence score for each hand landmark. These predictions are then combined and filtered using various techniques to generate a final estimate of the hand pose, including the position, orientation, and shape of each hand. It is available as a pre-trained model that can be easily integrated into custom applications, or as part of the MediaPipe framework, which provides a suite of pre-built modules for a variety of computer vision tasks.

The `mpHands.Hands()` function takes several parameters, some of which are optional. Here's an overview of the parameters:

- `static_image_mode`: A boolean value indicating whether to detect hands in a static image (default is `False`).

- `max_num_hands`: An integer value indicating the maximum number of hands to detect (default is 2).

- `min_detection_confidence`: A float value between 0 and 1 indicating the minimum confidence score required for a hand to be detected (default is 0.5).

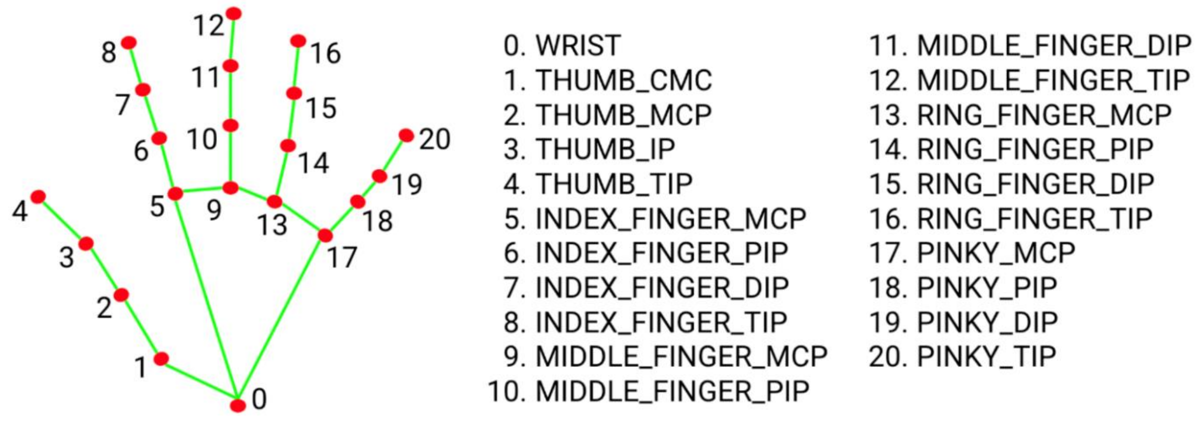
- `min_tracking_confidence`: A float value between 0 and 1 indicating the minimum confidence score required for a hand to continue being tracked (default is 0.5).

MediaPipe Hands actually includes two separate models for detecting and tracking hands:

1. A palm detection model
2. A hand landmark model

The palm detection model in Mediapipe is based on a Single Shot Multibox Detector (SSD) architecture, which is a popular object detection framework. It works by processing an input image through a convolutional neural network (CNN) to generate a set of bounding box predictions and confidence scores for each box. It is responsible for detecting the presence and general location of hands in an input image or video stream. It outputs a set of rectangular bounding boxes, each of which encloses a detected hand. The model is capable of detecting multiple hands in the same image and can also estimate the confidence level of each detection. Once the palm detection model has identified the location of a hand, with desired confidence threshold, the second model, the hand landmark model, takes over.

The hand landmark model is responsible for accurately localizing and tracking the landmarks of the hand within the detected palm region. It uses a convolutional neural network (CNN) to predict the 3D coordinates of 21 hand landmarks (refer Image 1), including the tips of the fingers, the base of the palm, and the wrist.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

These landmarks are represented as (x, y, z) coordinates in a 3D coordinate system that is aligned with the camera's view. The x and y coordinates are normalized to [0.0, 1.0] by the image width and height, respectively. The z coordinate represents the landmark depth, with the depth at the wrist being the origin. The smaller the value, the closer the landmark is to the camera. The magnitude of z uses roughly the same scale as x. The landmarks can also be visualized as points or lines on top of the input image or video stream, which can help users to understand the position and movement of their hands in real-time.

FOR EXAMPLE:

```
Landmark #0:
  x            : 0.638852
  y            : 0.671197
  z            : -3.41E-7
Landmark #1:
  x            : 0.634599
  y            : 0.536441
  z            : -0.06984
... (21 landmarks for a hand)
```

Together, these two models work in tandem to accurately detect and track the position, orientation, and shape of a person's hand in real-time, even in challenging lighting and occlusion conditions.

Since running the palm detection model is time consuming, when in video or live stream running mode, Hand Landmarker uses the bounding box defined by the hand landmarks model in one frame to localize the region of hands for subsequent frames. Hand Landmarker only re-triggers the palm detection model if the hand landmarks model no longer identifies the presence of hands or fails to track the hands within the frame. This reduces the number of times Hand Landmarker tiggers the palm detection model.

FEATURES:

- **Input image processing** - Processing includes image rotation, resizing, normalization, and color space conversion.
- **Score threshold** - Filter results based on prediction scores.

TASK INPUTS

The Hand Landmarker accepts an input of one of the following data types:

- Still images
- Decoded video frames
- Live video feed

TASK OUTPUTS

The Hand Landmarker outputs the following results

- Handedness of detected hands
- Landmarks of detected hands in image coordinates
- Landmarks of detected hands in world coordinates

4.3.2 MEDIAPIPE GESTURE RECOGNITION:

MediaPipe Gesture Recognition is a module of the MediaPipe library that enables developers to recognize hand gestures. It provides the recognized hand gesture results along with the landmarks of the detected hands.

This task operates on image data with a machine learning (ML) model, and accepts either static data or a continuous stream. The output hand landmarks of the Mediapipe hands model are also fed in as input to classify the hand gestures into one of several predefined gesture categories, such as "thumbs up", "thumps down", “okay”, “fist”, “stop”, “rock”, “livelong”, “call me” etc. The task outputs hand landmarks in image coordinates, hand landmarks in world coordinates, handedness (left/right hand), and the hand gesture categories of multiple hands.

To use the MediaPipe Gesture Recognizer module, developers can use the MediaPipe Python API or the MediaPipe C++ API to access the hand pose estimation and gesture recognition modules and integrate them into their own applications. The module is highly customizable, allowing developers to adjust the performance and accuracy of the gesture recognition algorithm to fit their specific application requirements.

MediaPipe Gesture Recognizer can be used in a wide range of applications, such as virtual reality, gaming, and sign language recognition. The module is designed to work in real-time on a variety of devices, including smartphones, laptops, and embedded systems.

Overall, the MediaPipe Gesture Recognizer module provides a powerful and flexible way for developers to incorporate gesture recognition capabilities into their applications, and enables a wide range of new user experiences and interaction paradigms.

4.4 CONCEPT - PyAutoGUI:

`PyAutoGUI` is a Python library that allows for GUI automation on various operating systems, such as Windows, macOS, and Linux. It provides a set of functions to programmatically control mouse and keyboard actions, take screenshots, manipulate windows, and more.

SOME OF THE KEY FEATURES OF `PYAUTOGUI` INCLUDE:

1. Mouse and keyboard control: The library can simulate mouse clicks, mouse movement, and keyboard presses and releases.

2. Screen recording: The library can capture screenshots and record the screen in various formats.

3. GUI automation: The library can interact with GUI elements such as buttons, menus, and text boxes to automate tasks.

4. Multi-platform support: The library can be used on Windows, macOS, and Linux.

5. Fail-safe features: The library includes fail-safe mechanisms that can stop an automation script if certain conditions are met, such as the mouse moving to a specific corner of the screen.

PyAutoGUI can be useful for a variety of tasks, such as automating repetitive tasks, creating software demos, and testing graphical user interfaces. However, it's important to use caution when using `pyautogui`, as it has the potential to cause unintended actions on the computer if used improperly.

The x, y coordinates used by PyAutoGUI has the (0, 0) origin coordinates in the top left corner of the screen. The x coordinates increase going to the right (just as in mathematics) but the y coordinates increase going down (the opposite of mathematics). On a screen that is 1366 x 768 pixels in size, coordinates (0, 0) are for the top left while (1366, 768) is for the bottom right.

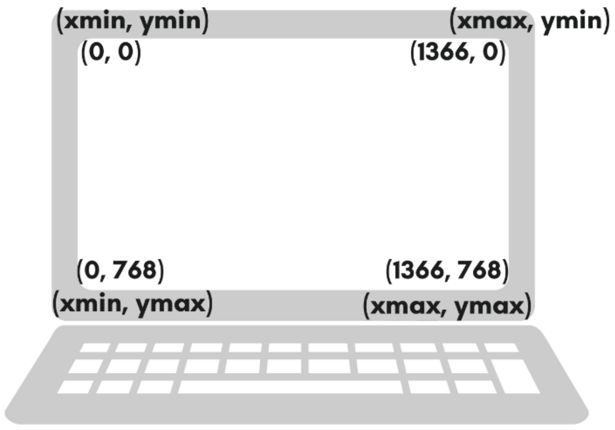


Fig 4.2: Coordinates of a laptop screen

All keyboard presses done by PyAutoGUI are sent to the window that currently has focus, as if the physical keyboard key has been pressed. In our case, the Gesture recognition model, classifies input data into different categories, and then perform desired actions based on the predicted category. The `classID` variable stores the index of the predicted category with the highest probability, obtained using the `argmax` function of the NumPy library.

If the predicted category is 2, the program will simulate a keyboard shortcut to save the current file by pressing the 'Ctrl' and 'S' keys together using the PyAutoGUI library. If the predicted category is 3, the program will simulate a keyboard shortcut to close the current window by pressing the 'Alt' and 'F4' keys together using the PyAutoGUI library. If the predicted category is 0, the program will simulate a key press of the 'Enter' key using the PyAutoGUI library. If the predicted category is 6, the program will simulate a keyboard shortcut to print the current document by pressing the 'Ctrl' and 'P' keys together using the PyAutoGUI library. If the predicted category is 5, the program will simulate a keyboard shortcut to close the current tab or window by pressing the 'Ctrl' and 'W' keys together using the PyAutoGUI library.

All of these commands are performed on the current active window. If the window is not active, then the functions may not work.

CODE SNIPPET:

#Connecting Model’s output with PyAutoGUI

```
classID = np.argmax(prediction)
if (classID == 2):
     pyautogui.hotkey('ctrl', 's')
elif (classID == 3):
     pyautogui.hotkey('Alt', 'f4')
elif (classID == 0):
     pyautogui.press('enter')
elif (classID == 6):
     pyautogui.hotkey('ctrl', 'p')
elif (classID == 5):
     pyautogui.hotkey('ctrl', 'w')
```

4.5 CONVOLUTIONAL NEURAL NETWORK:

Convolutional neural network (Convnet for short is a class of artificial neural network that uses convolution. It uses mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. In mathematics in particular functional analysis convolution is a mathematical operation on two functions (f and g) that produces third function that expresses how the shape of one is modified by the other. The term convolution refers to both the result function and to the process of computing it.

CNN is developed exclusively for the computer vision and image processing A breakthrough in building models for images classification came with the discovery that a convolutional neural network (CNN) could be used to progressively extract higher-level representation of the image content.

It is based on shared weight architecture. In densely connected network we get only one pattern for an image in hand, whereas CNN extracts various local patterns from the data. CNN is also known as shift invariant or space invariant artificial neural network.

TWO IMPORTANT PROPERTIES OF CONVNET:

1. The patterns they learn are translation invariant
2. They can learn spatial hierarchy way

Types of operations or layers:

A. Convolution Stage

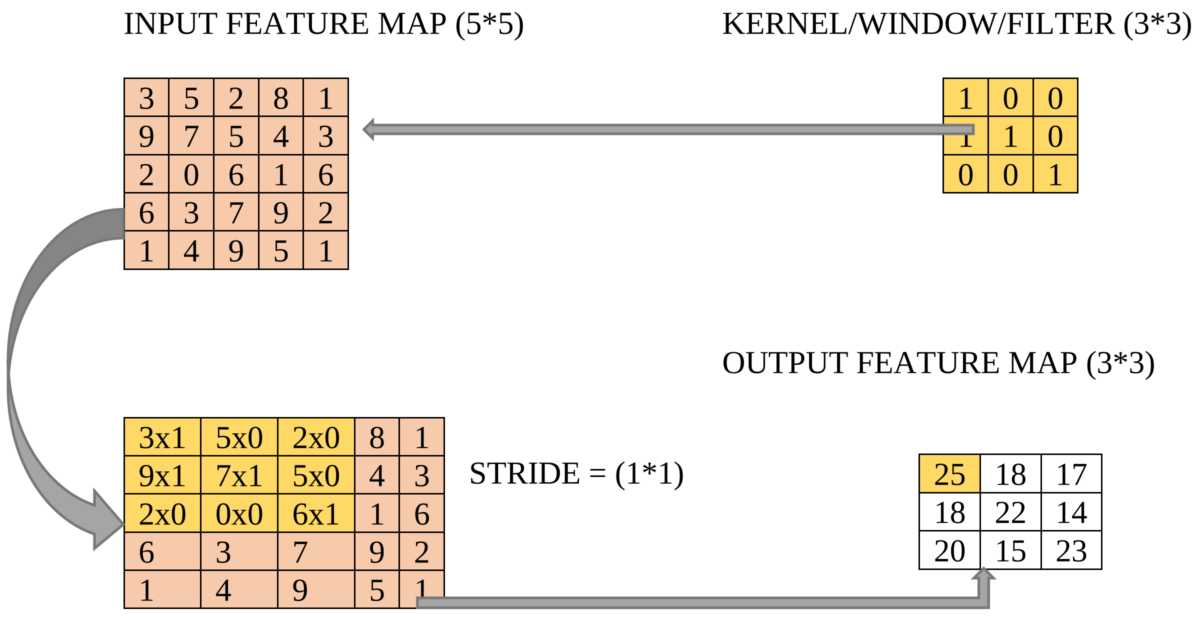
A convolution extracts tiles of the input feature maps and applies filters to them to compute new features, producing an output feature map or convolved feature (which may have a different size and depth than the input feature map).

Convolution are defined by two parameters:

1. Size of the tiles that are extracted typically 3X3 or 5X5
2. The depth of the output feature map which corresponds to the number of filters that are applied.

During convolution layer, the filters effectively slide over the input feature map's grid horizontally and vertically one pixel at a time extracting each corresponding tile.

For Example:



INPUT FEATURE MAP (5*5)

3	5	2	8	1
9	7	5	4	3
2	0	6	1	6
6	3	7	9	2
1	4	9	5	1

KERNEL/WINDOW/FILTER (3*3)

1	0	0
1	1	0
0	0	1

3x1	5x0	2x0	8	1
9x1	7x1	5x0	4	3
2x0	0x0	6x1	1	6
6	3	7	9	2
1	4	9	5	1

STRIDE = (1*1)

OUTPUT FEATURE MAP (3*3)

25	18	17
18	22	14
20	15	23

= 3 + 0 + 0 + 9 + 7 + 0 + 0 + 0 + 6 = 25

From this it is clear that the output value in the feature map does not have to connect to each pixel value in the input image. It only needs to connect to the receptive field where the filter is being applied.

By default, our kernels are only applied where the filter fully fits on top of the input. But I can also control this behavior and the size of our output with Padding.

Padding – Pads the outside of the input 0’s to allow the kernel to reach the boundary pixels. padding = 1 means I have one layer of 0’s around our border

Strides – Controls how far the kernels “steps” over pixels. For a convolution or pooling operation, the stride S denotes the number of pixels by which the window moves after each operation. Strides = (2,2) means our kernel moves 2 data points to the right for each row then moves 2 points down to the next row.

FILTERS:

Using a filter smaller than the input is intentional as we allow the same filter (set of Weights) to be multiplied by the input array multiple times at different points on the input. If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the filter an opportunity to discover that feature anywhere in the image. This capability is commonly referred to as translation invariance.

The Weights in the feature detector remains fixed as it moves across the image, which is also known as parameter sharing (sharing same filter or same kernel across the image). Going through convolutional layers, process features a large amount of data, which makes it hard to train the neural network, to compress the data, I need to go through pooling.

B. POOLING STAGE

Helps to reduce complexity, improve efficiency and the risk of overfitting. Receives the result form a convolution layer and compress it. Filter of pooling layer is smaller than the feature map. Down sampling operation typically applied after a convolution layer which does some spatial invariance.

As a result, I get pooled feature maps that are a summarized version of the features detected in the input. It reduces the number of dimensions of the feature map, while still preserving the most critical feature information.

Two types of pooling:

1. Max pooling

Aggregating using maximum of the window

2. Average pooling

Aggregating using average of the window.

Pooling operation takes two paraments:

1. **size** – the pooling filter (typically 2x2)
2. **stride** – The distance in pixels separating each extracted tile.

FULLY CONNECTED LAYER:

The flattened output is fed to a feed – forward neural network and back propagation is applied at every iteration of training.

Job – perform classification based on the feature extracted by the convolutions or previous layers.

GENERAL OVERVIEW:

1. A convolution layer is responsible for recognizing features in pixels.
2. A pooling layer is responsible for making these features more abstract.
3. A fully connected layer is responsible for using the acquired features for prediction

CHAPTER V

TESTING AND RESULTS

5.1 INTRODUCTION

In this chapter, details regarding test data, test experiment and test results are summarised.

5.2 TEST DATA

The Hand Landmarker accepts an input of Still images, Decoded video frames or Live video feed**.** In this project, live video feed from the webcam has been fed into the model for further processing.

In case of users, who have their webcam damaged or not in a good condition, other alternatives can be used. One of the alternatives is using Iriun web cam. It is a software application that turns the mobile device (Android or iOS) into a wireless webcam for the computer. The Iriun app uses the mobile device's camera to capture video and audio, and streams it to the computer via WiFi or USB connection. This allows users to use their phone or tablet as a high-quality webcam for video conferencing, streaming, or recording purposes.

To use Iriun, download and install the Iriun Webcam app on mobile device from the Google Play Store or App Store. Then, download and install the Iriun Webcam software on computer, which is available for Windows and macOS. Once both the app and software are installed, connect the mobile device to the computer via WiFi or USB and start using it as a webcam. If connection is Via WiFi, then both the devices have to be connected with the same network.

OVERVIEW OF THE PROCESS:

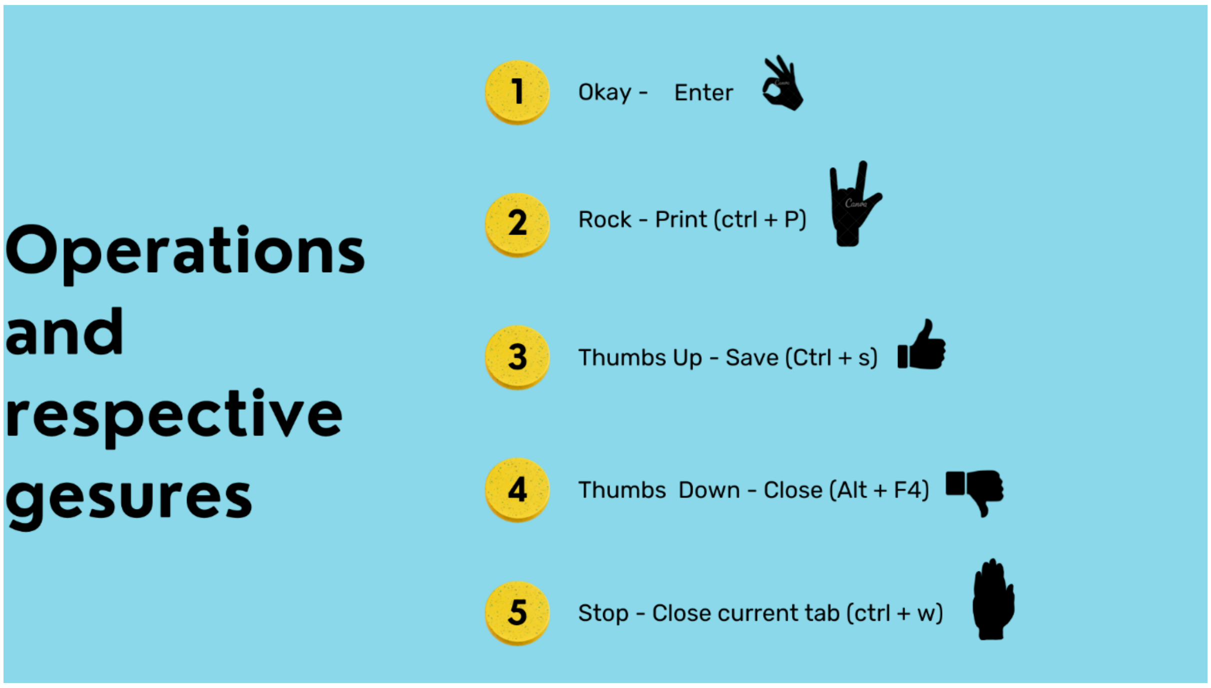
1. **Palm detection:** The first step in the process is Palm detection. MediaPipe Hands uses a multi-stage neural network to detect hands or Palms in each video frame. This neural network analyses the image data and outputs a set of hand landmarks, which are key points on the hand such as the fingertips, knuckles, and wrist.
2. **Hand Landmark Detection:** Once the hands have been detected, the next step is hand Landmark detection. MediaPipe Hands uses an algorithm called Kalman filtering to

track the hand landmarks across frames. This helps to ensure that the hand landmarks remain stable and accurate even when the hand is moving or the camera is shaking.

3. **Output:** Finally, MediaPipe Hands outputs the processed hand landmarks of 21 key points with x,y,z co-ordinates in floating point.
4. **Gesture Recognizer:** Output from the above model is fed to the gesture recognizer and eventually, the model recognizes the gesture from the input and classifies it to one of the predefined hand gesture classes.
5. **Controlling Apps:** Once the model classifies the gesture, the if-else statement associated will run and enables the defined PyAutoGUI functionalities, thereby automating the mouse and keyboard.

5.3 CONTROLLING APPS

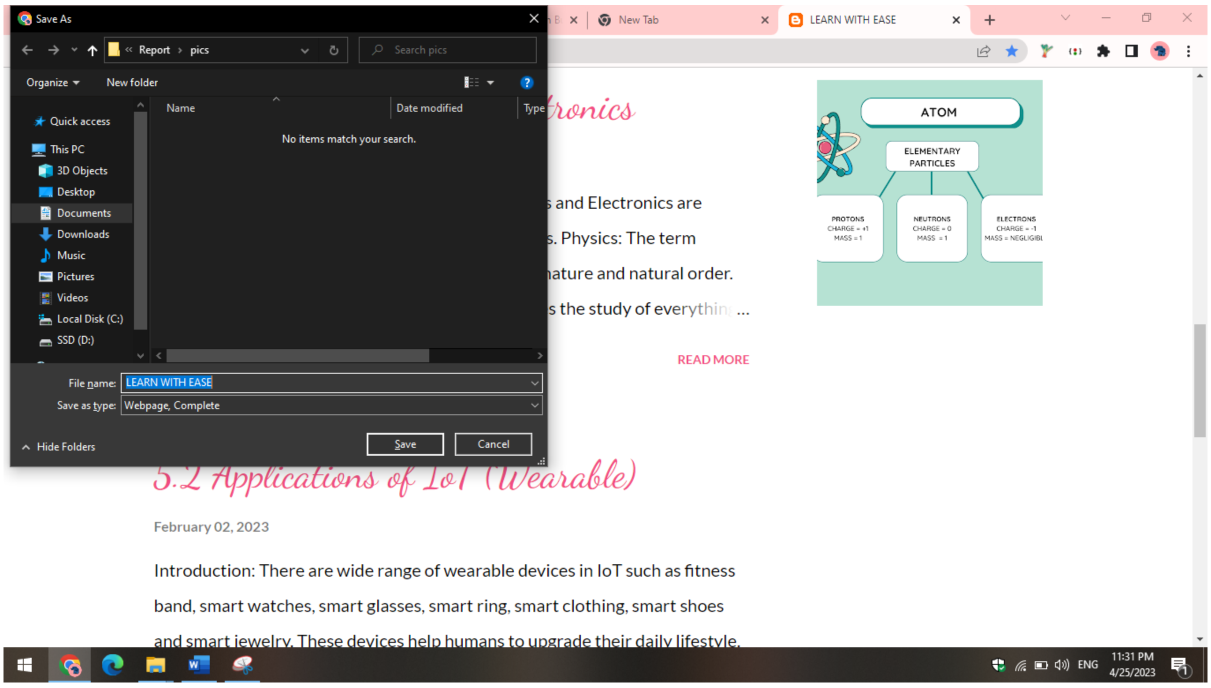
Fig 5.1: Gesture list and respective operations


Operations and respective gesures
1 Okay - Enter
2 Rock - Print (ctrl + P)
3 Thumbs Up - Save (Ctrl + s)
4 Thumbs Down - Close (Alt + F4)
5 Stop - Close current tab (ctrl + w)

SCENARIO:

Imagine a chrome browser with 5 tabs, an excel workbook, a word document and a notepad file has also been opened. Any of the above gestures performed will impact the current active window. In our case, if chrome with five tabs is the current active window, then performing stop gesture would close the current tab. Doing it for four more times would close the chrome window.

Fig 5.2: Thumbs Up in a webpage



ronics

s and Electronics are s. Physics: The term nature and natural order. s the study of everythin ...

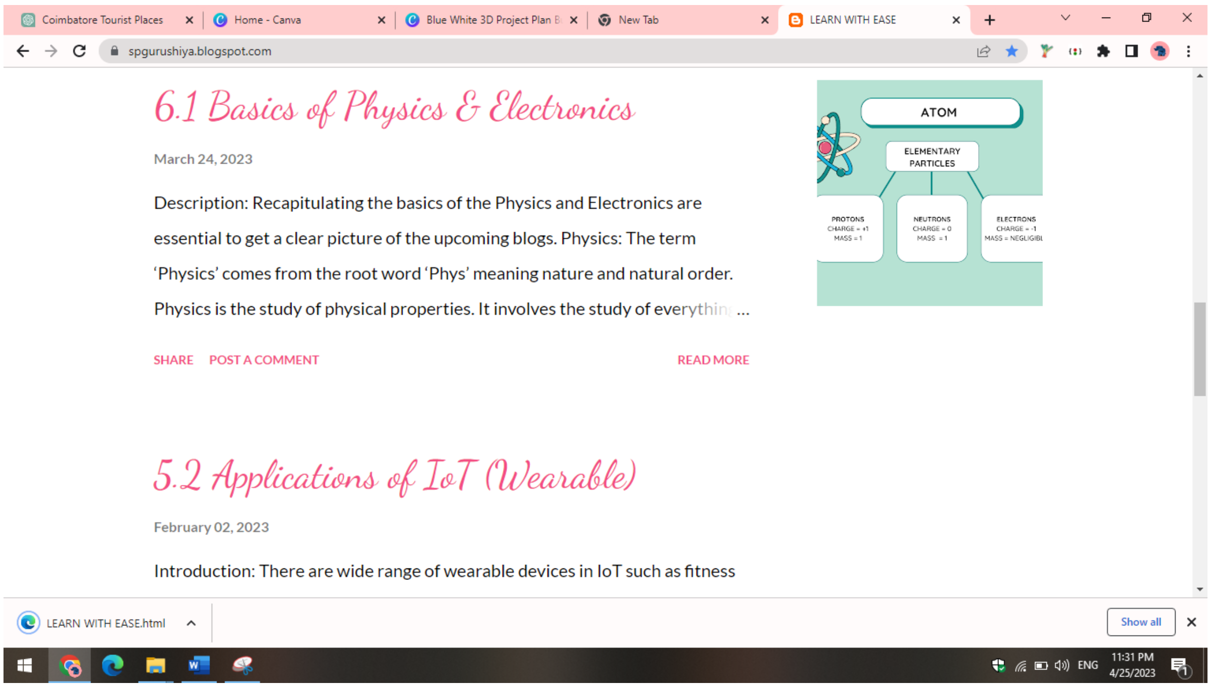
5.2 Applications of IoT (Wearable)

February 02, 2023

Introduction: There are wide range of wearable devices in IoT such as fitness band, smart watches, smart glasses, smart ring, smart clothing, smart shoes and smart jewelry. These devices help humans to upgrade their daily lifestyle.

If “thumbs up” gesture is performed meanwhile, then the ‘Save As dialog box’ prompts to save the webpage.

Fig 5.3: Okay sign in a webpage



6.1 Basics of Physics & Electronics

March 24, 2023

Description: Recapitulating the basics of the Physics and Electronics are essential to get a clear picture of the upcoming blogs. Physics: The term 'Physics' comes from the root word 'Phys' meaning nature and natural order. Physics is the study of physical properties. It involves the study of everythin ...

5.2 Applications of IoT (Wearable)

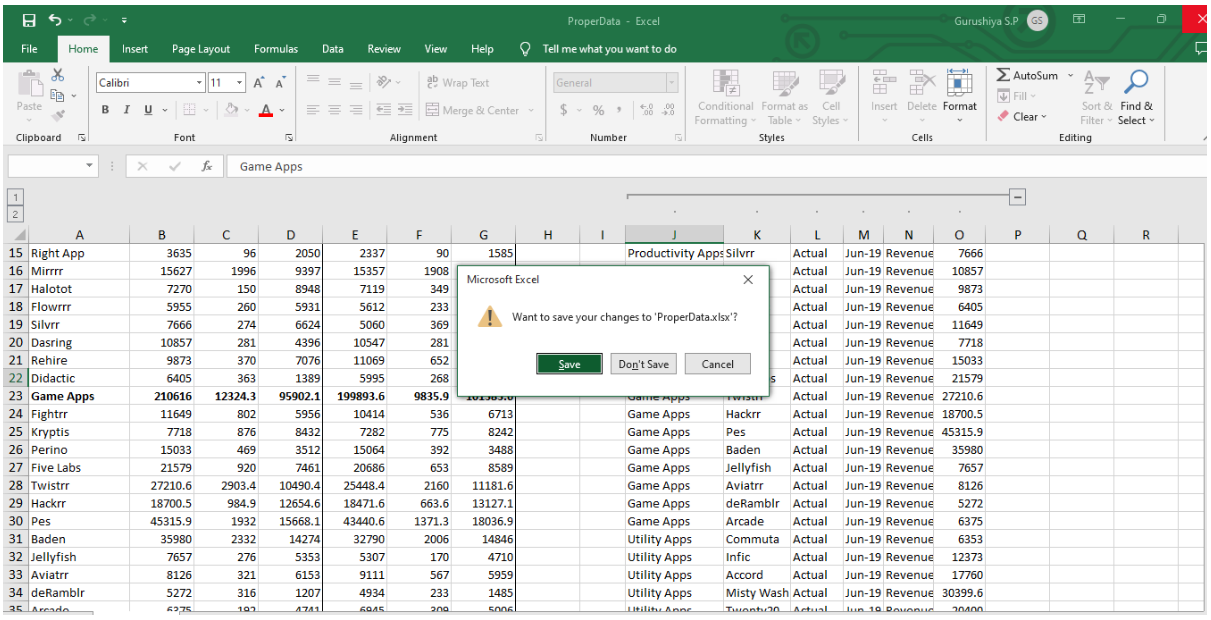
February 02, 2023

Introduction: There are wide range of wearable devices in IoT such as fitness

If 'Okay' gesture has performed, the webpage starts downloading.

Then, if Excel workbook turns to be the current active window, then 'thumbs down" sign would close the workbook, if there are no changes made and if user made any changes, then a dialog box with a question "Want to save changes to your "filename"? with three options: 1. Save, 2. Don't save. 3. Cancel.

Fig 5.4: Okay sign in a Excel workbook



Game Apps

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
15	Right App	3635	96	2050	2337	90	1585			Productivity Apps	Silvrr	Actual	Jun-19	Revenue	7666
16	Mirrrr	15627	1996	9397	15357	1908						Actual	Jun-19	Revenue	10857
17	Halotot	7270	150	8948	7119	349						Actual	Jun-19	Revenue	9873
18	Flowrrr	5955	260	5931	5612	233						Actual	Jun-19	Revenue	6405
19	Silvrr	7666	274	6624	5060	369						Actual	Jun-19	Revenue	11649
20	Dasring	10857	281	4396	10547	281						Actual	Jun-19	Revenue	7718
21	Rehire	9873	370	7076	11069	652						Actual	Jun-19	Revenue	15033
22	Didactic	6405	363	1389	5995	268						Actual	Jun-19	Revenue	21579
23	**Game Apps**	**210616**	**12324.3**	**95902.1**	**199893.6**	**9835.9**	[illegible]					Actual	Jun-19	Revenue	27210.6
24	Fightrr	11649	802	5956	10414	536	6713			Game Apps	Hackrr	Actual	Jun-19	Revenue	18700.5
25	Kryptis	7718	876	8432	7282	775	8242			Game Apps	Pes	Actual	Jun-19	Revenue	45315.9
26	Perino	15033	469	3512	15064	392	3488			Game Apps	Baden	Actual	Jun-19	Revenue	35980
27	Five Labs	21579	920	7461	20686	653	8589			Game Apps	Jellyfish	Actual	Jun-19	Revenue	7657
28	Twistrr	27210.6	2903.4	10490.4	25448.4	2160	11181.6			Game Apps	Aviatrr	Actual	Jun-19	Revenue	8126
29	Hackrr	18700.5	984.9	12654.6	18471.6	663.6	13127.1			Game Apps	deRamblr	Actual	Jun-19	Revenue	5272
30	Pes	45315.9	1932	15668.1	43440.6	1371.3	18036.9			Game Apps	Arcade	Actual	Jun-19	Revenue	6375
31	Baden	35980	2332	14274	32790	2006	14846			Utility Apps	Commuta	Actual	Jun-19	Revenue	6353
32	Jellyfish	7657	276	5353	5307	170	4710			Utility Apps	Infic	Actual	Jun-19	Revenue	12373
33	Aviatrr	8126	321	6153	9111	567	5959			Utility Apps	Accord	Actual	Jun-19	Revenue	17760
34	deRamblr	5272	316	1207	4934	233	1485			Utility Apps	Misty Wash	Actual	Jun-19	Revenue	30399.6

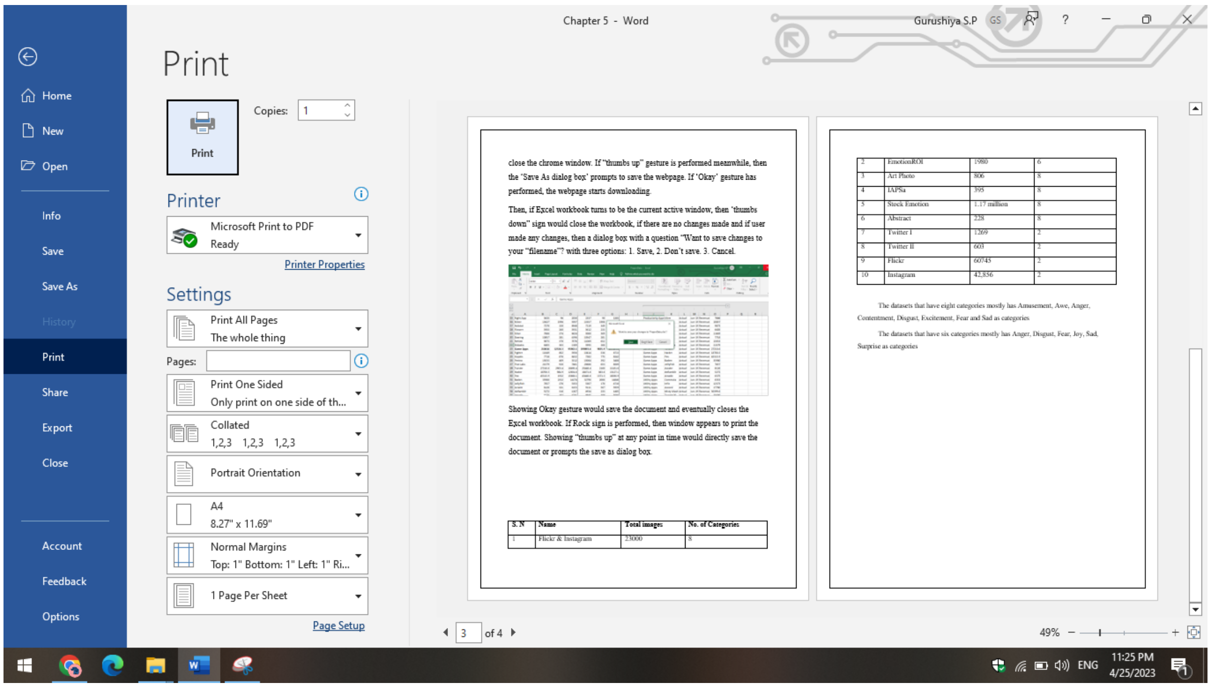
Microsoft Excel

Want to save your changes to 'ProperData.xlsx'?

Save | Don't Save | Cancel

Showing Okay gesture would save the document and eventually closes the Excel workbook.

Fig 5.5: Rock sign in a word document



Print

Copies: 1

Printer: Microsoft Print to PDF — Ready

Printer Properties

Settings

Print All Pages — The whole thing

Pages:

Print One Sided — Only print on one side of th...

Collated — 1,2,3 1,2,3 1,2,3

Portrait Orientation

A4 — 8.27" x 11.69"

Normal Margins — Top: 1" Bottom: 1" Left: 1" Ri...

1 Page Per Sheet

Page Setup

3 of 4

49%

close the chrome window. If "thumbs up" gesture is performed meanwhile, then the 'Save As dialog box' prompts to save the webpage. If 'Okay' gesture has performed, the webpage starts downloading.

Then, if Excel workbook turns to be the current active window, then 'thumbs down" sign would close the workbook, if there are no changes made and if user made any changes, then a dialog box with a question "Want to save changes to your "filename"? with three options: 1. Save, 2. Don't save. 3. Cancel.

Showing Okay gesture would save the document and eventually closes the Excel workbook. If Rock sign is performed, then window appears to print the document. Showing "thumbs up" at any point in time would directly save the document or prompts the save as dialog box.

S. N	Name	Total images	No. of Categories
1	Flickr & Instagram	23000	8

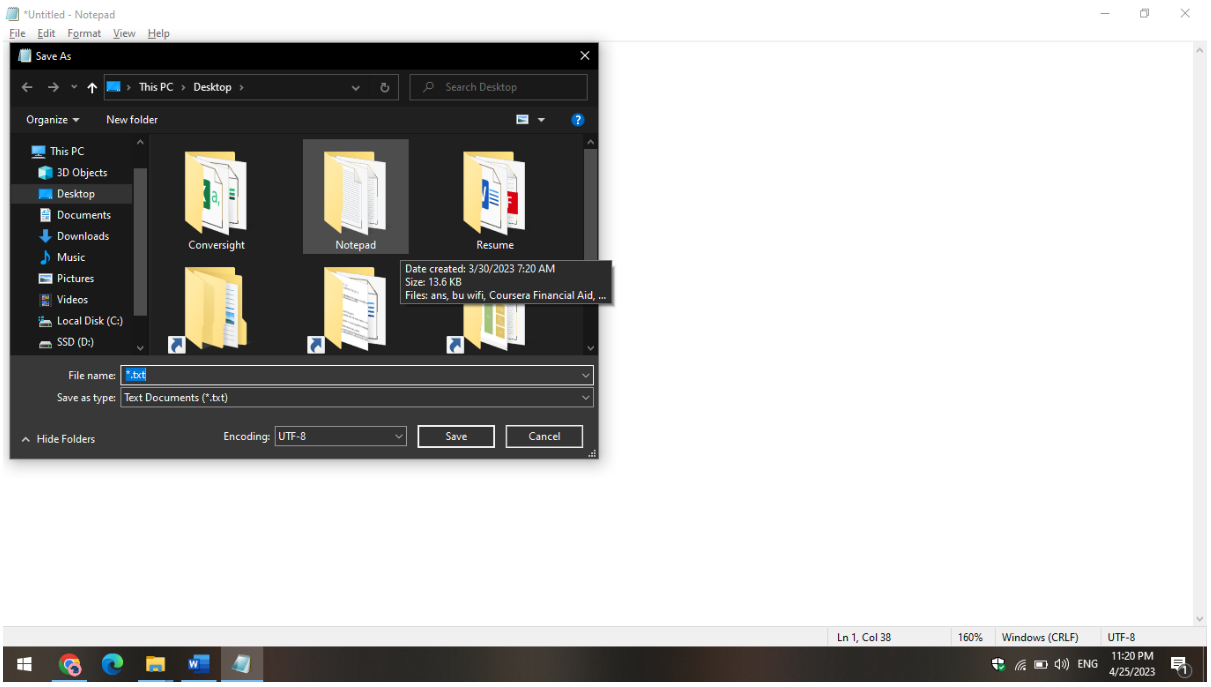
2	EmotionROI	1980	6
3	Art Photo	806	8
4	IAPSa	395	8
5	Stock Emotion	1.17 million	8
6	Abstract	228	8
7	Twitter I	1269	2
8	Twitter II	603	2
9	Flickr	60745	2
10	Instagram	42,856	2

The datasets that have eight categories mostly has Amusement, Awe, Anger, Contentment, Disgust, Excitement, Fear and Sad as categories

The datasets that have six categories mostly has Anger, Disgust, Fear, Joy, Sad, Surprise as categories

If Rock sign is performed, then window appears to print the document.

Fig 5.6: Thumbs Up in a word document



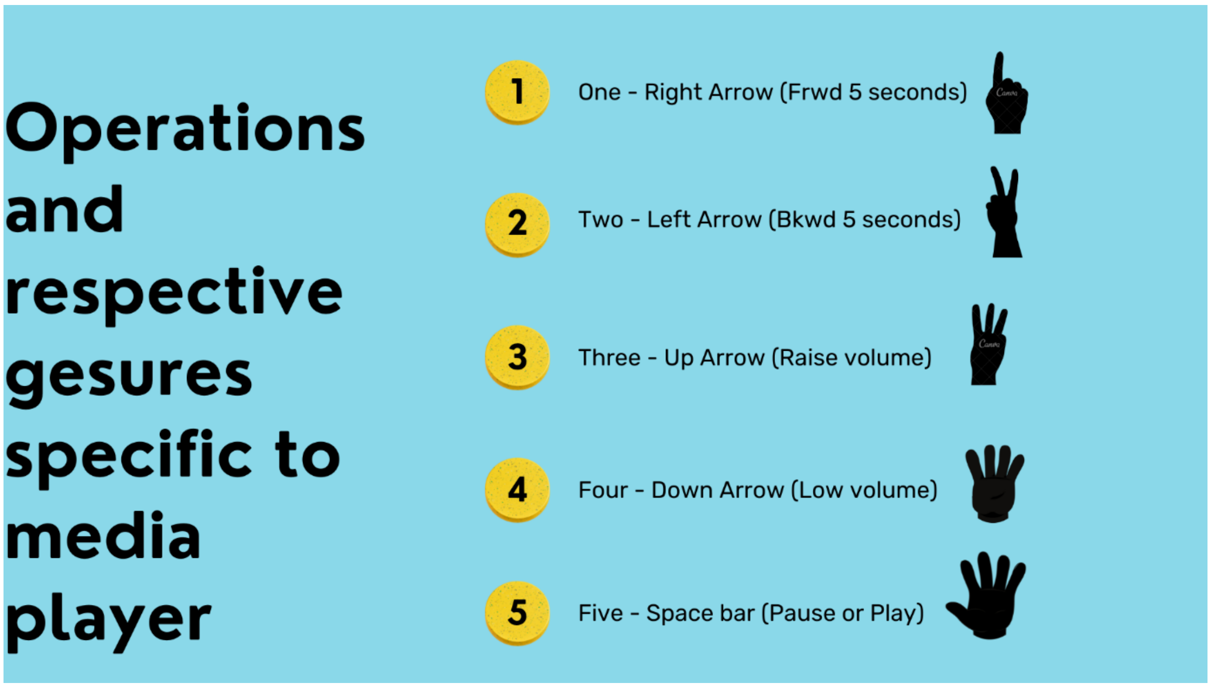
Date created: 3/30/2023 7:20 AM
Size: 13.6 KB
Files: ans, bu wifi, Coursera Financial Aid, ...

File name: *.txt
Save as type: Text Documents (*.txt)
Encoding: UTF-8

Showing "thumbs up" at any point in time would directly save the document or prompts the save as dialog box.

5.4 CONTROLLING MEDIA PLAYER

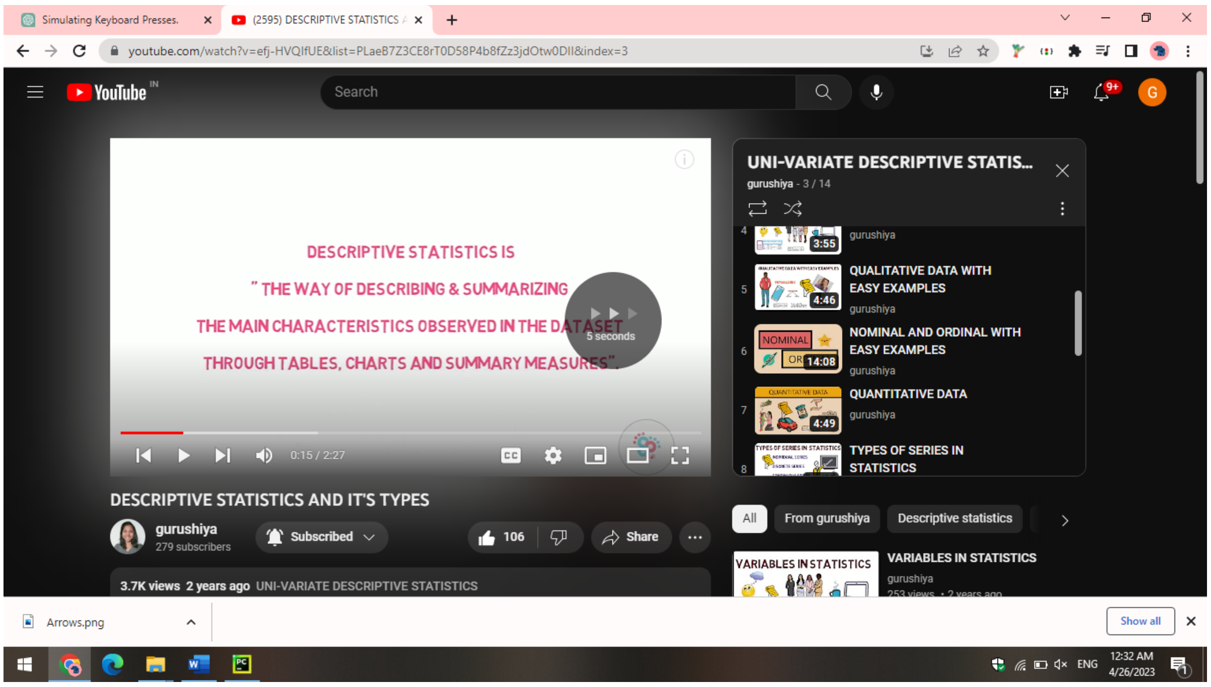
Fig 5.7: Gesture list and respective operations


Operations and respective gesures specific to media player
1
One - Right Arrow (Frwd 5 seconds)
2
Two - Left Arrow (Bkwd 5 seconds)
3
Three - Up Arrow (Raise volume)
4
Four - Down Arrow (Low volume)
5
Five - Space bar (Pause or Play)

Here, to control the mediaplayer, PyAutoGUI library in Python has been used to simulate keyboard presses based on the count of the fingers. The "if" statements check if "cnt" is equal to a specific number and if so, simulate a keyboard press using the "pyautogui.press()" function.

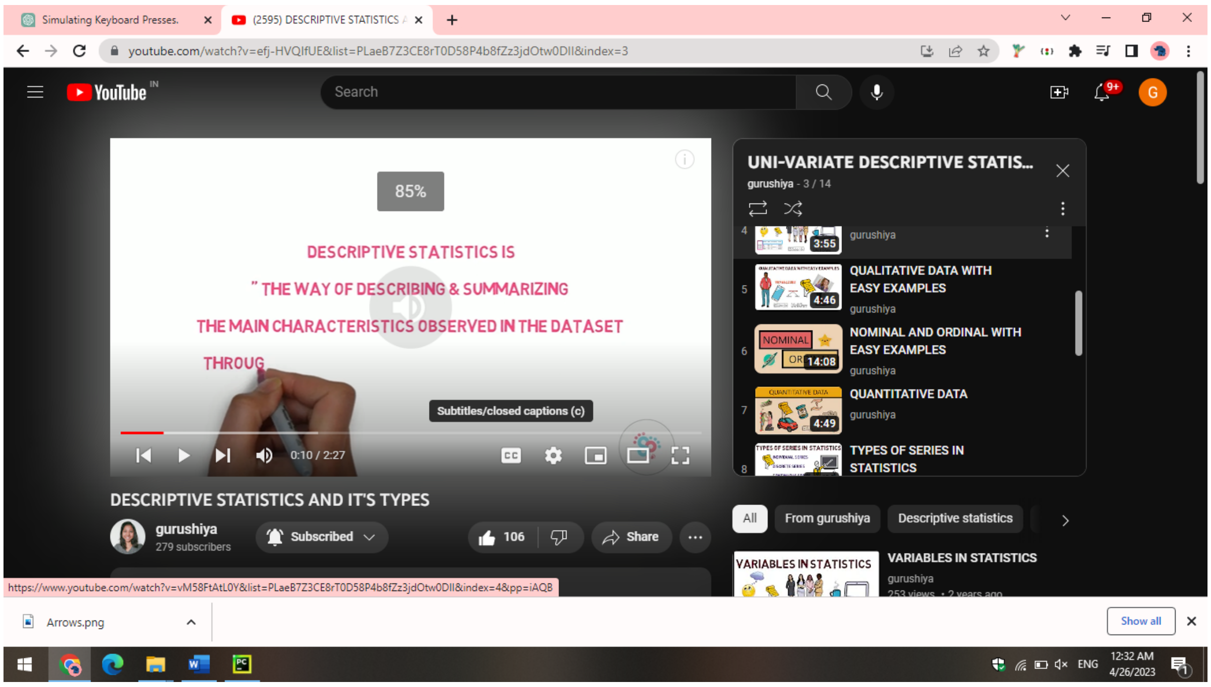
If "cnt" is equal to 1, it simulates a press of the "right" arrow key, thereby, forwards the video by five seconds. If "cnt" is equal to 2, it simulates a press of the "left" arrow key and backwards by five seconds. If "cnt" is equal to 3, it simulates a press of the "up" arrow key, therefore volume is increased by 5 percent. If "cnt" is equal to 4, it simulates a press of the "down" arrow key, volume is decreased by 5 percent. If "cnt" is equal to 5, it simulates a press of the "space" bar. It means the video will be played or paused accordingly.

Fig 5.8: Showing one – right arrow


DESCRIPTIVE STATISTICS IS
" THE WAY OF DESCRIBING & SUMMARIZING
THE MAIN CHARACTERISTICS OBSERVED IN THE DATASET
THROUGH TABLES, CHARTS AND SUMMARY MEASURES"
5 seconds
DESCRIPTIVE STATISTICS AND IT'S TYPES
UNI-VARIATE DESCRIPTIVE STATIS...

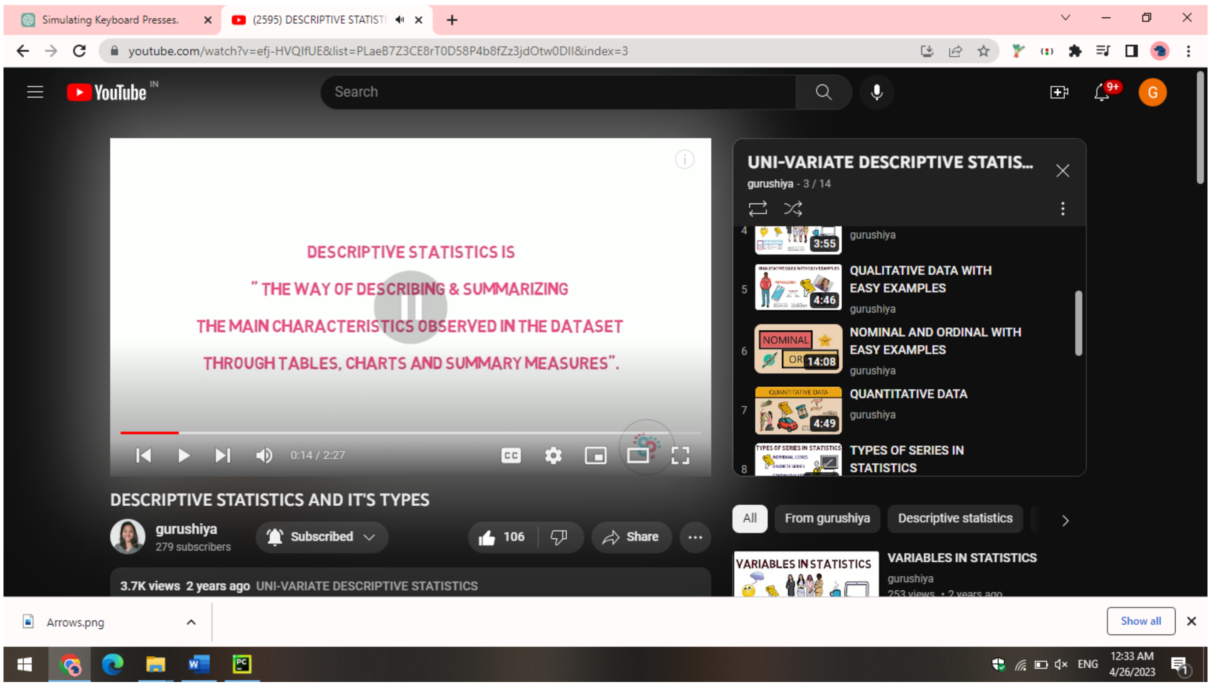
If "cnt" is equal to 1, it simulates a press of the "right" arrow key, thereby, forwards the video by five seconds.

Fig 5.9: Showing three – up arrow


85%
DESCRIPTIVE STATISTICS IS
" THE WAY OF DESCRIBING & SUMMARIZING
THE MAIN CHARACTERISTICS OBSERVED IN THE DATASET
Subtitles/closed captions (c)
DESCRIPTIVE STATISTICS AND IT'S TYPES
UNI-VARIATE DESCRIPTIVE STATIS...

If "cnt" is equal to 3, it simulates a press of the "up" arrow key, therefore volume is increased by 5 percent.

Fig 5.10: Showing five – space bar


DESCRIPTIVE STATISTICS IS
" THE WAY OF DESCRIBING & SUMMARIZING
THE MAIN CHARACTERISTICS OBSERVED IN THE DATASET
THROUGH TABLES, CHARTS AND SUMMARY MEASURES".
DESCRIPTIVE STATISTICS AND IT'S TYPES
UNI-VARIATE DESCRIPTIVE STATIS...

If "cnt" is equal to 5, it simulates a press of the "space" bar. It means the video will be played or paused accordingly.

CHAPTER VI

CONCLUSION AND FURTHER WORK

6.1 INTRODUCTION

In this chapter, details regarding the work conducted so far and directions to further work, has been presented and summarized.

6.2 CONCLUSION

The results show that deep learning does provide promising results on the gesture recognition task. The input feed from webcam has been loaded into the mediapipe hands model for hand tracking and landmarks detection. The hand landmarks have been fed into the Gesture recognition model for further classification. The output from the classification model has been successfully connected with the PyAutoGUI library for further automation of tasks in the PC. This solution is an extra feature from the perspective techies and for non-techies, this solution can ease their way of approaching laptops or PC.

6.3 FURTHER WORK

This work doesn’t cover physically challenged people into account. So, a customised solution can be built for them. Instead of hand gestures, various facial emotions can be considered for automating the task i,e., an emotion classification algorithm can be connected with PyAutoGUI library to make this project function.

BIBLIOGRAPHY

1. (Jayesh Jidge, Sanika Badwaik, Rutika Bhoir, Prof. Kanchan Doke, 2022) “Hand motion-based computer activity control”

2. (Gopi Manoj Vuyyuru, Malvika Ramesh Shirke, 2021), Performing Basic Tasks on Computer using Hand Gestures & Ultrasonic Sensors
3. (Muhammad Sheikh Sadi, Mohammed Alotaibi, Md. Repon Islam, 2022) “Finger-Gesture Controlled Wheelchair with Enabling IoT”
4. (Brimingham University) “Gesture Control Technology: An investigation on the potential use in Higher Education”
5. (Ayushi Bhawal, Debaparna Dasgupta, Arka Ghosh, Koyena Mitra, Surajit) “Arduino Based Hand Gesture Control of Computer”
6. https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer
7. https://developers.google.com/mediapipe/solutions/vision/hand_landmarker
8. https://pypi.org/project/PyAutoGUI/#:~:text=PyAutoGUI%20is%20a%20cross%2Dplatform,pip%20install%20pyautogui
9. https://www.jetbrains.com/help/pycharm/getting-started.html
10. https://developers.google.com/mediapipe/solutions/vision/hand_landmarker/python
11. https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/
12. https://developers.google.com/ml-kit/vision/object-detection