REPORT

TITLE

Rock-Paper-Scissor game based on Hand Gesture Recognition

MADE BY

AISHWARY SHUKLA

E-MAIL: aishwarystark@gmail.com

ABSTRACT

The aim of this project was to build an interactive version of the classic Rock-Paper-Scissors game using state of the art Computer Vision to detect and recognise the hand gesture of the player using their device's built-in camera. So that instead of using keyboard or mouse as input device, the user can directly play with their hands without wearing any sort of motion tracking gloves.

The technology stack used is MediaPipe (open-source framework by Google) to detect hands. Python3, Open-CV and scikit-learn library is used to build the Machine Learning model which classifies the gestures. This is then integrated with the game elements such as score keeping, number of rounds for game and result display. This report explains what I did and learned during the entire course of mini-project . It also gives an insight into the MediaPipe framework which plays a crucial part in this project.

KEYWORDS: Machine learning, Computer Vision, Game, Gesture recognition, Python

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

"Rock-Paper-Scissor game based on Hand Gesture Recognition" can be defined as a simple and convenient way for an average computer user to entertain themselves and have the kind of fun which they usually get from playing with other humans. After the pandemic due to lockdown most people are now fed up from watching movies and playing video games. They want to do something involving other intelligent beings, ideally a human. But because of social distancing, we are a bit short on those for close proximity activities. So what's the next best thing?

Artificial Intelligence (A.I.) of course! So why not build an A.I. that would play whenever user wants. This project uses Computer Vision to recognise hand gestures shown by user, namely 'Rock', 'Paper' and 'Scissors'. It also randomly generates it's own move, compare it with the user's move, and then shows result according to rules of the game.

1.2 Computer Vision

Computer Vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyse thousands of products or processes in a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

1.3 Architecture

At the core of this project is the image classifier which detects the hand and puts the gesture into one of three categories (Rock, paper and scissor). This is made up of two parts: Hand detection and gesture recognition part. In the hand detection part Google's open source cross-platform MediaPipe library is used which is very fast and accurate in detecting 21 hand landmarks. The co-ordinates of these hand-landmarks are provided to the gesture recognition part which uses a machine learning model, using the Stochastic Gradient Descent Algorithm to output the prediction. This prediction is then compared with the computers move, and then the result is shown. Apart from the core (image classifier), rest of the game is also written in python3.

CHAPTER 2

HARDWARE & SOFTWARE REQUIREMENTS

2.1 Software Requirements

- Operating System : MacOS or Windows 7+
- IDE : PyCharm
- Libraries used : mediapipe, pandas, sklearn, cv2, scipy
- Language : Python 3

2.2 Hardware Requirements

- Processor : Intel core i3 & above
- RAM : 4 GB
- Free disk space: 10 MB
- Keyboard : standard qwerty
- Mouse : Touchpad or 2/3 button mouse
- Webcam : HD webcam recommended
- Monitor : Any

CHAPTER 3

SOFTWARE INSTALLATION

3.1 Installing Python3

1. To download and install python3, visit the official website of Python https://www.python.org/downloads and choose suitable version.

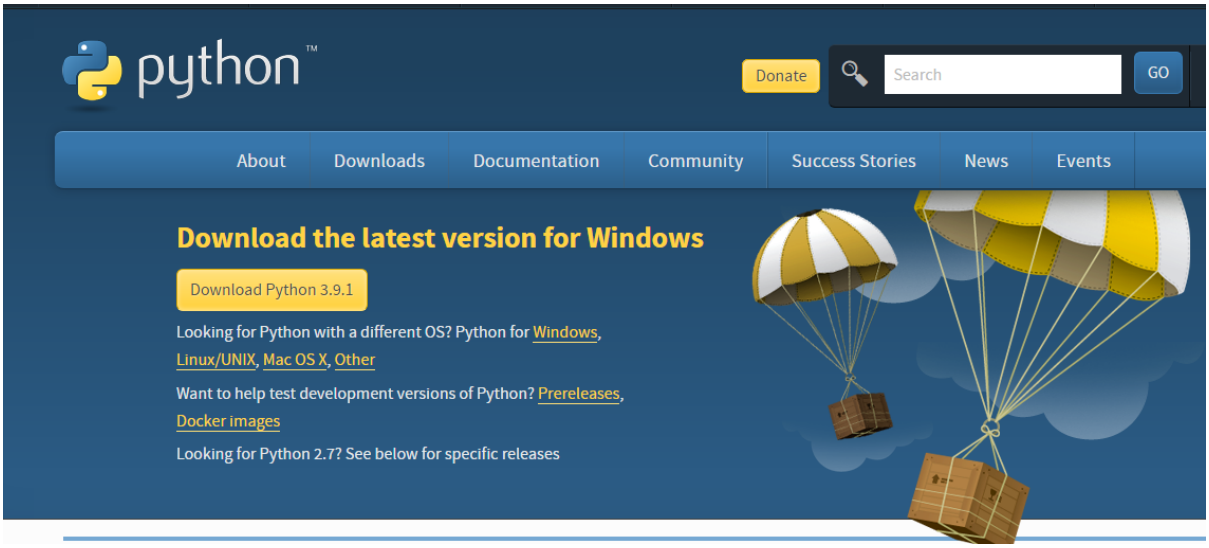


Fig.1: Snap of Python3 installation in the system

2. Once the download is complete, run the exe for install Python. Now click on Install Now.
3. You can see Python installing at this point.
4. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

3.2 Installing PyCharm

1. To download PyCharm, visit the website https://www.jetbrains.com/pycharm/download and click the download button under the Community Section.

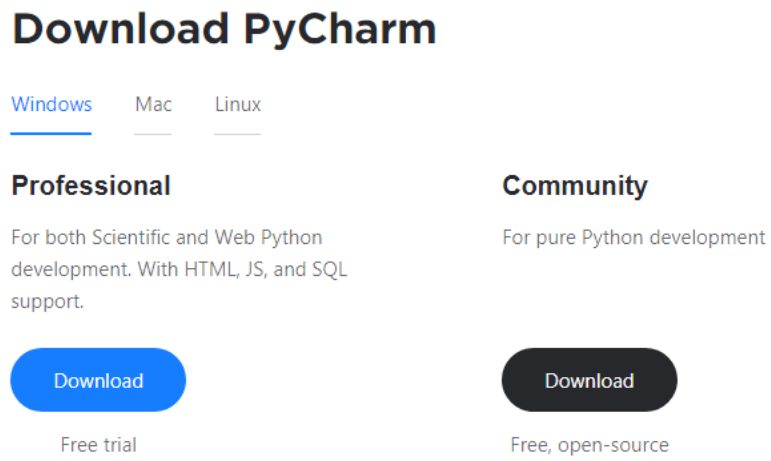


Fig. 2: Snap of PyCharm installation in the system

2. Once the download is complete, run the exe file to start installation. The setup wizard should have started. Click 'Next'.
3. Choose suitable option and wait for the installation to finish.
4. After you click on "Finish," the Following screen will appear

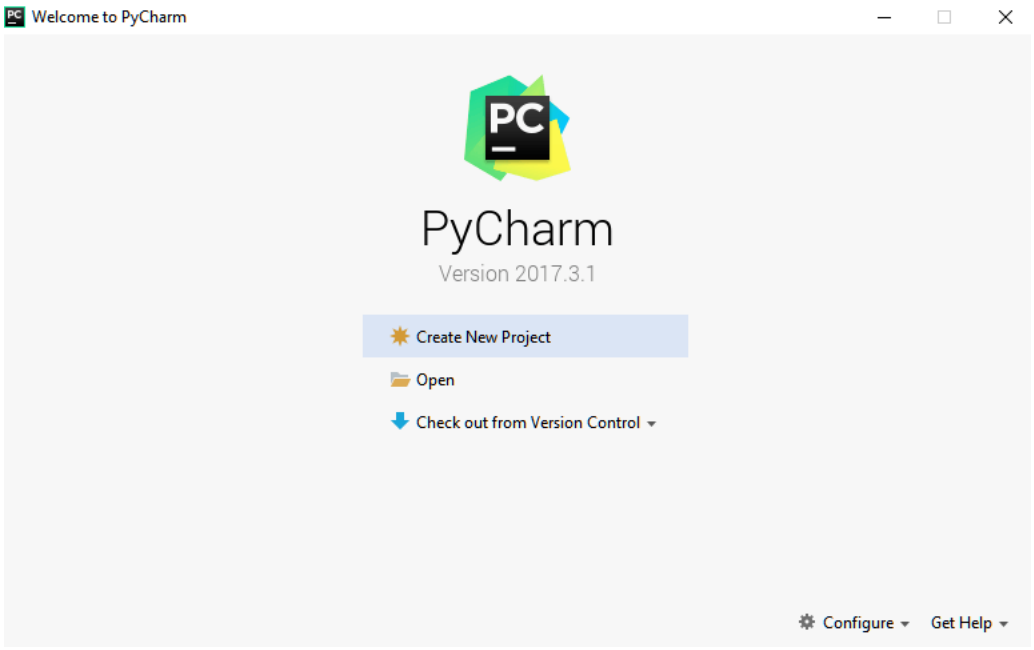


Fig. 3: Opening window of PyCharm

3.3 Installing Packages

1. Some packages are required for proper running of the project.
2. Open the command prompt, anaconda prompt or Terminal as administrator.
3. In the prompt window, open the project path and type “pip3 install <package name>” which you want to install. Libraries needed are mentioned in software requirements.

Example: pip3 install pandas

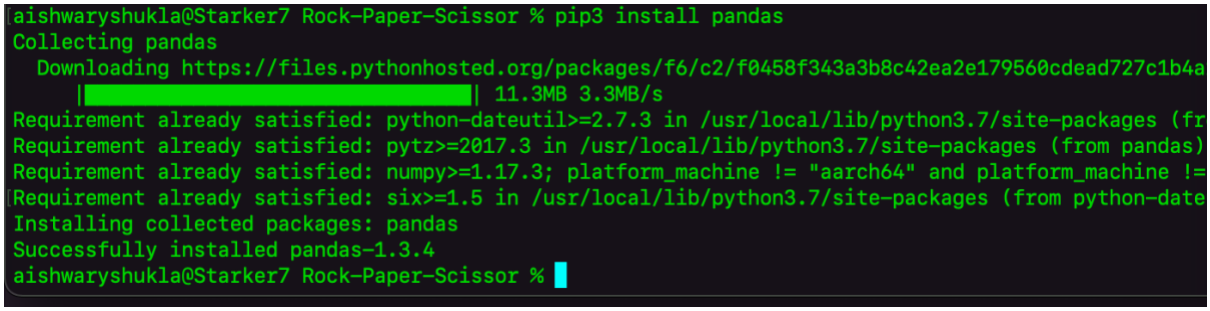


Fig. 4: Installation of numpy

Example: pip3 install scipy

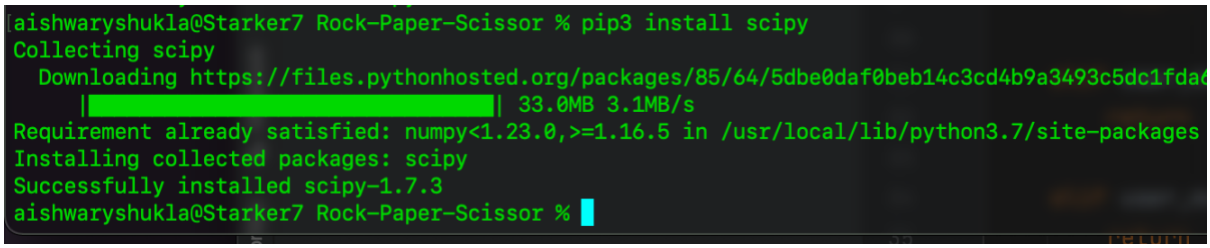


Fig. 5: Installation of scipy

CHAPTER 4

MODULES

4.1 data_generation.py

The objective of this module is to generate data which will be fed into the machine learning model. This script only needs to be run once, i.e., before training ML model.

- 500 images of each gesture are taken using the webcam
- hand_detection_module provides the coordinates of 21 landmarks on the hand
- using id_distance module, the distance between all 21 landmarks is calculated and stored in .csv file

4.2 hand_detection_module.py

This module is an implementation of the MediaPipe library as suitable for this project.
It is called during data generation to find coordinates of hands (for training) and during the running of game to find coordinates of hands (for evaluating gesture).

4.2.1 MediaPipe Hands

MediaPipe Hands is a high-fidelity hand and finger tracking solution from Google. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame. Whereas current state-of-the-art approaches rely primarily on powerful desktop environments for inference, this method can achieves real-time performance on a mobile phone, and even scales to multiple hands.

MediaPipe Hands utilizes an ML pipeline consisting of multiple models working together: A palm detection model that operates on the full image and returns an oriented hand bounding box. A hand landmark model that operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand keypoints.

Providing the accurately cropped hand image to the hand landmark model drastically reduces the need for data augmentation (e.g. rotations, translation and scale) and instead allows the network to dedicate most of its capacity towards coordinate prediction accuracy. In addition, the crops can also be generated based on the hand landmarks identified in the previous frame, and only when the landmark model could no longer identify hand presence is palm detection invoked to relocalize the hand.

Palm Detection Model:

To detect initial hand locations, a single-shot detector model optimized for mobile real-time uses in a manner similar to the face detection model in MediaPipe Face Mesh. Detecting hands is a decidedly complex task: lite model and full model have to work across a variety of hand sizes with a large scale span (~20x) relative to the image frame and be able to detect occluded and self-occluded hands. Whereas faces have high contrast patterns, e.g., in the eye and mouth region, the lack of such features in hands makes it comparatively difficult to detect them reliably from their visual features alone. Instead, providing additional context, like arm, body, or person features, aids accurate hand localization.

MediaPipe's method addresses the above challenges using different strategies. First, it trains a palm detector instead of a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using square bounding boxes ignoring other aspect ratios.

With the above techniques, an average precision of 95.7% is achieved in palm detection.

Hand Landmark Model:

After the palm detection over the whole image, the subsequent hand landmark model performs precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression, that is direct coordinate prediction. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.

To obtain ground truth data, MediaPipe's developers have manually annotated ~30,000 real-world images with 21 3D coordinates. To better cover the possible hand poses and provide additional supervision on the nature of hand geometry, they have also rendered a high-quality synthetic hand model over various backgrounds and map it to the corresponding 3D coordinates.

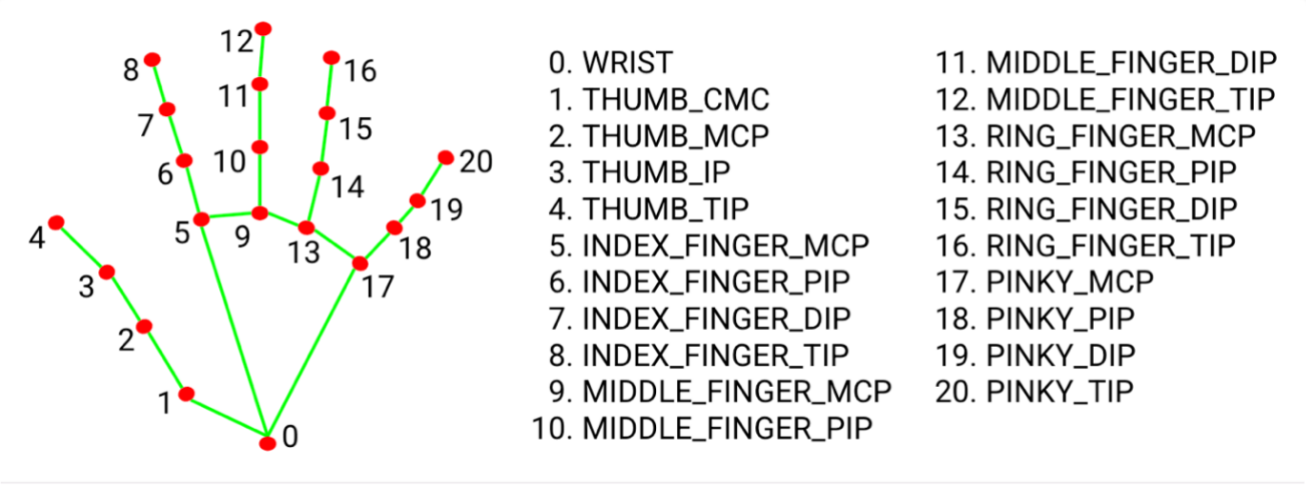


Fig. 6: The 21 Hand Landmarks detected by MediaPipe

4.3 train_sgd.py

- This script splits the .csv data (created from data_generation.py) into train and test sets
- Then it trains the ML model which is a stochastic gradient descent classifier from scikit-learn
- This is the ML model which takes the distances in image taken during gameplay and predicts the gesture.
- The trained model is saved using the pickle.dump() function as hand_model.sav So that training the model is not required again during game playing
- The model accuracy is displayed after the training is over, which came around 99.7% during my run.

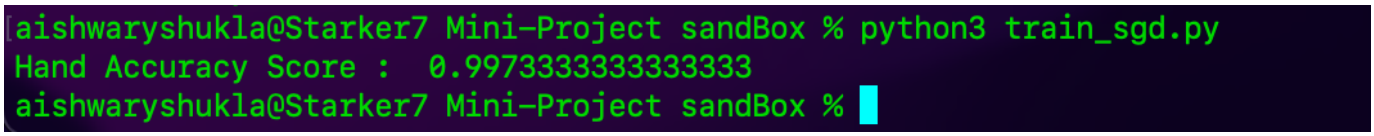


Fig. 7: Accuracy of ML model displayed after training

4.4 main.py

This is the driver script which runs during the game.

- It contains the findout_winner function which defines the rules of the game. (For example: if user's move is 'paper' and computer's move is 'rock' then user will win)
- It has all the required User Interface functions like for displaying an animated computer's hand, showing the winner screen, etc.
- It reads the camera frames and asks the user to enter number of rounds to be played after which it starts a timer. The user shows their hand gesture after the timer and then result of that round (including score) is displayed using the other modules.
- After displaying the final result, if user quits the game, then it ensures that all game windows are closed and the camera is released.

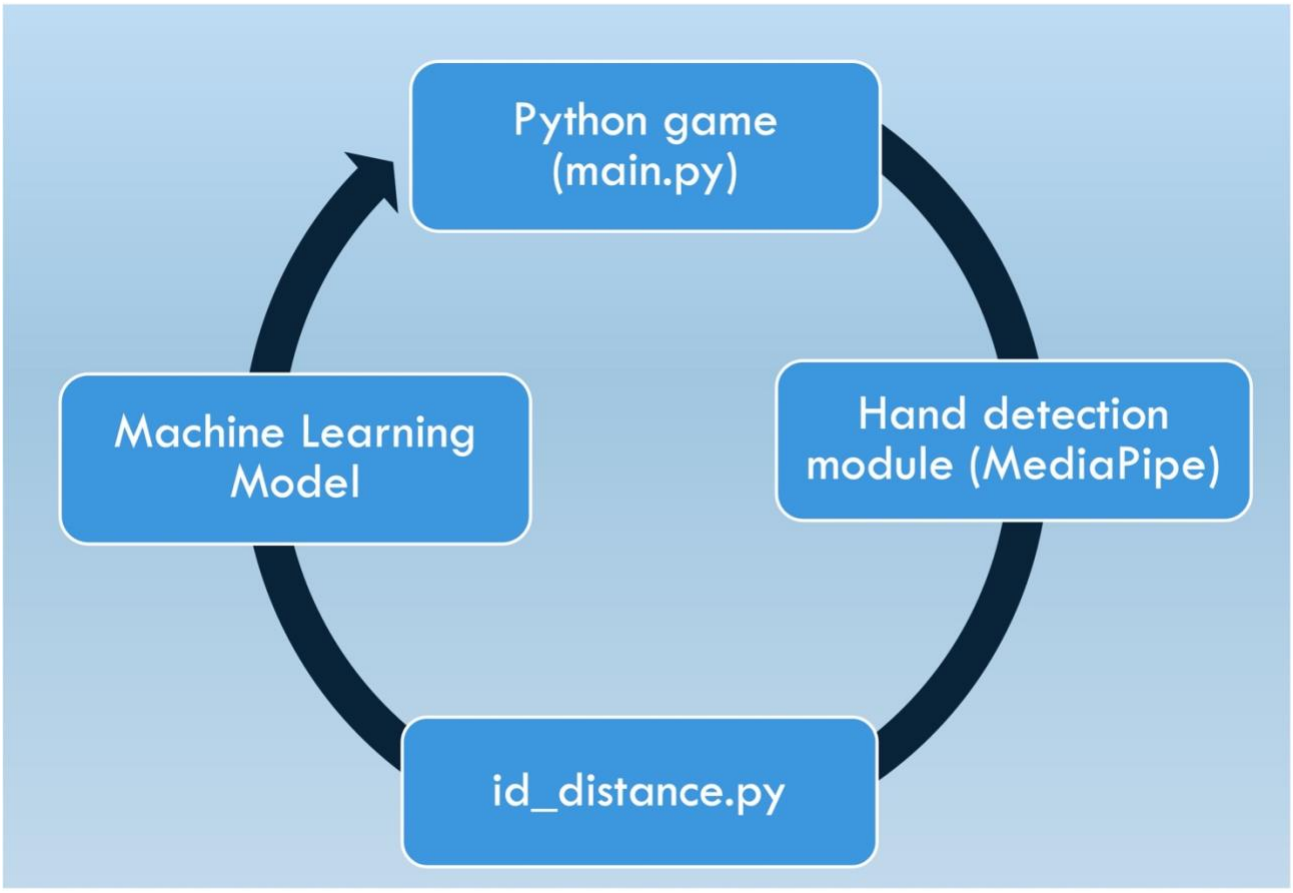


Fig. 8: Diagramatic representation of the architecture of the game

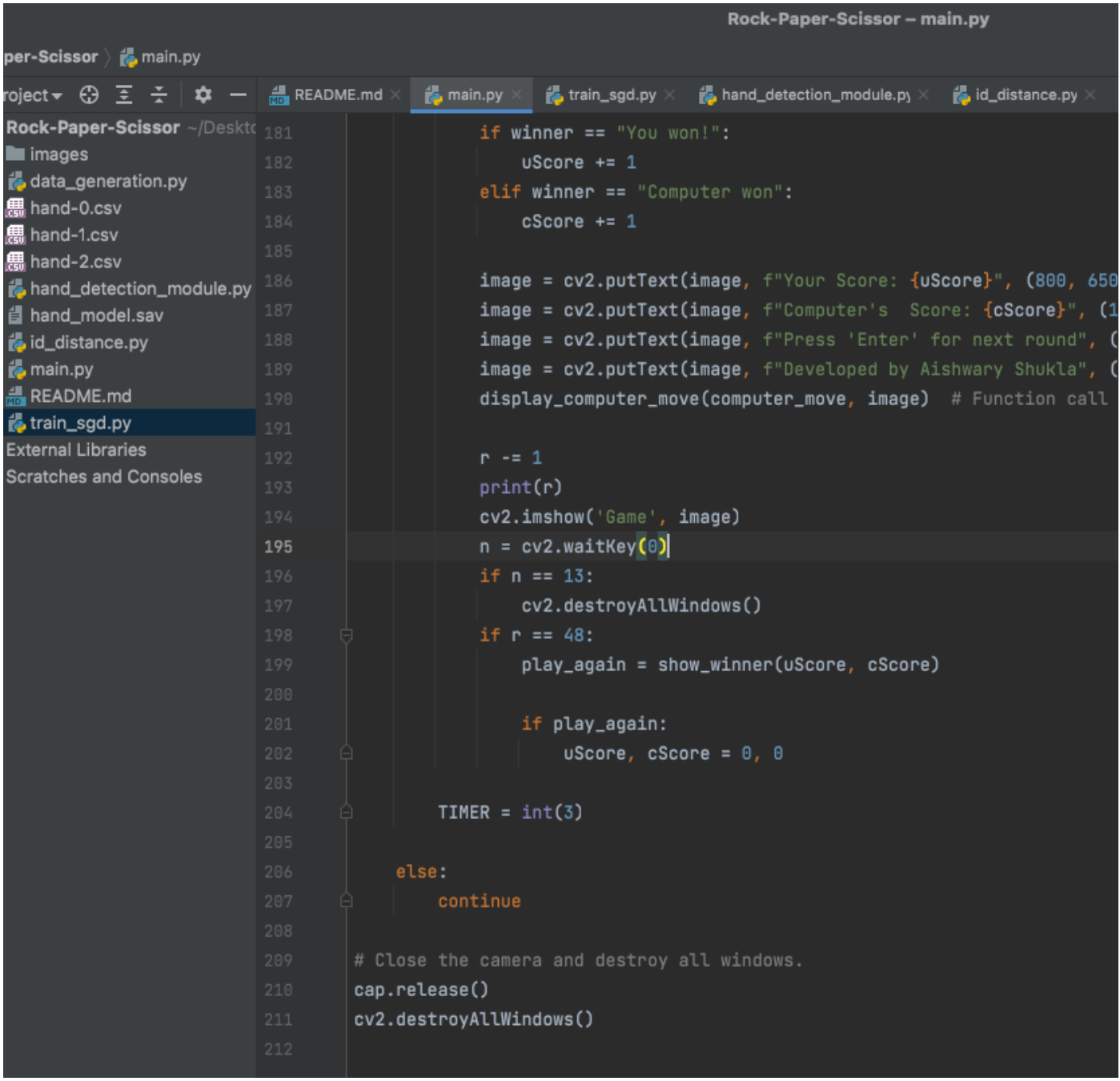


Fig. 9: Snap of main.py file in PyCharm IDE

CHAPTER 5

STEPS FOR EXECUTING THE PROJECT

1. Place the game files in desired path.
2. Ensure that python is installed in your system.
3. Go to the ‘Rock-Paper-Scissor’ folder path in your terminal.
4. Execute this command ‘python3 main.py’ or ‘python main.py’ (in case you don’t have python3).

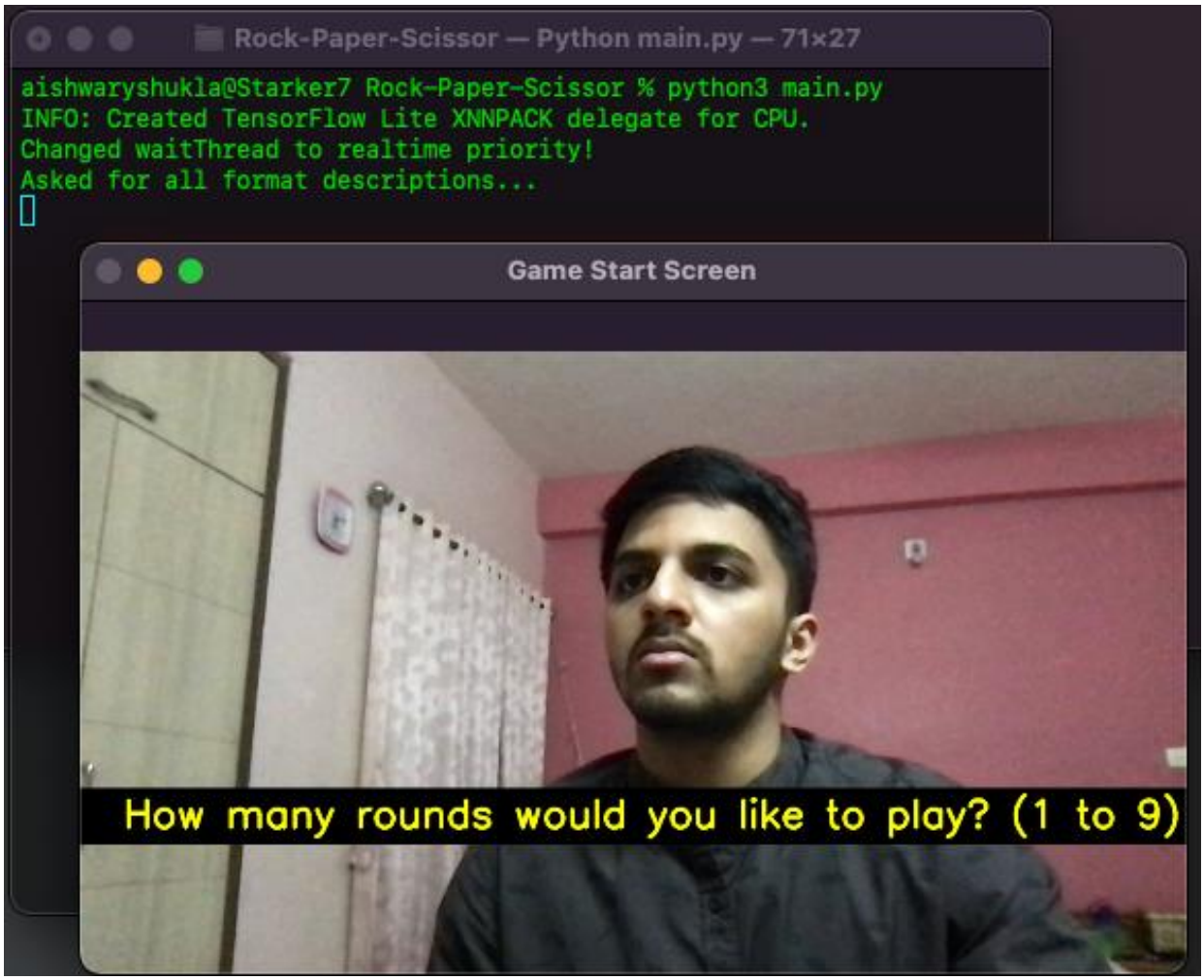


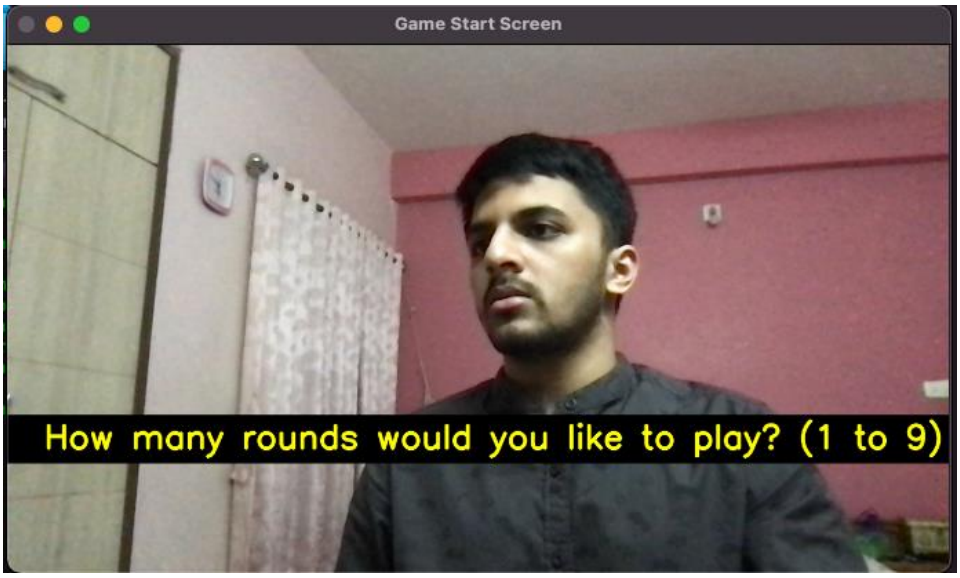
Fig. 10: Snap after running command ‘python3 main.py’

CHAPTER 6

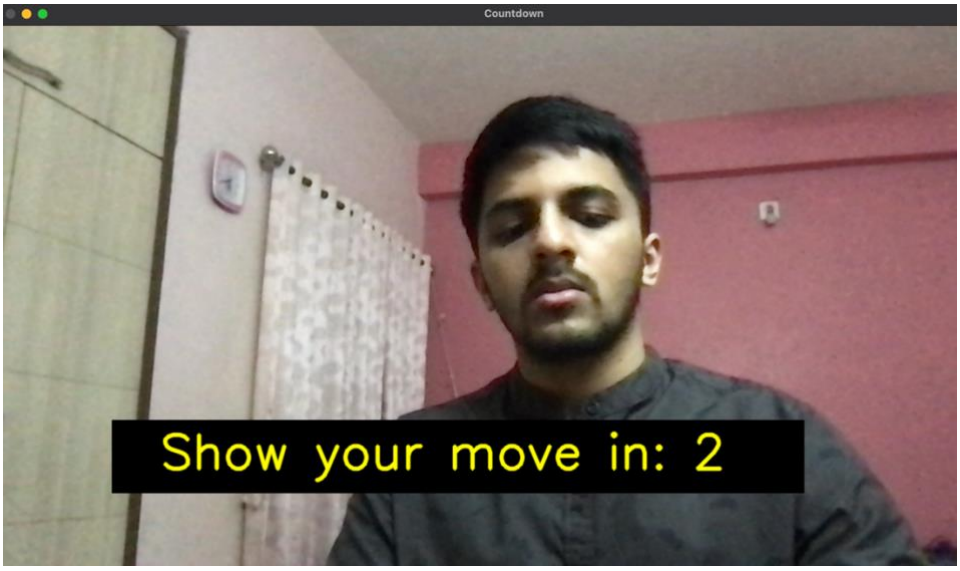
RESULT

After around a month of learning and building, the project meets all of its objectives such as accurate real time hand gesture detection and recognition, smooth and simple yet beautiful game user interface. Taking interactiveness to the next level, this game truly gives the player a one of its kind experience.

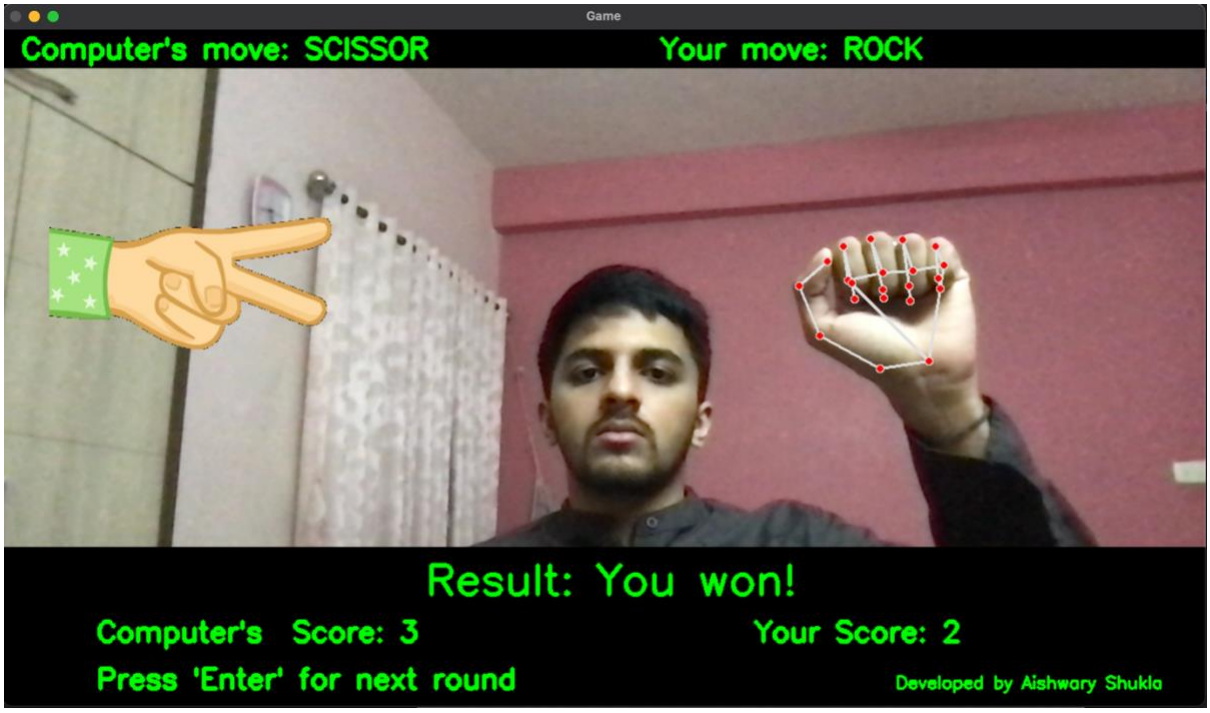
The screen captures of different stages of the application are shown below:



The game start screen where the user inputs the number of rounds of the game he/she wants to play



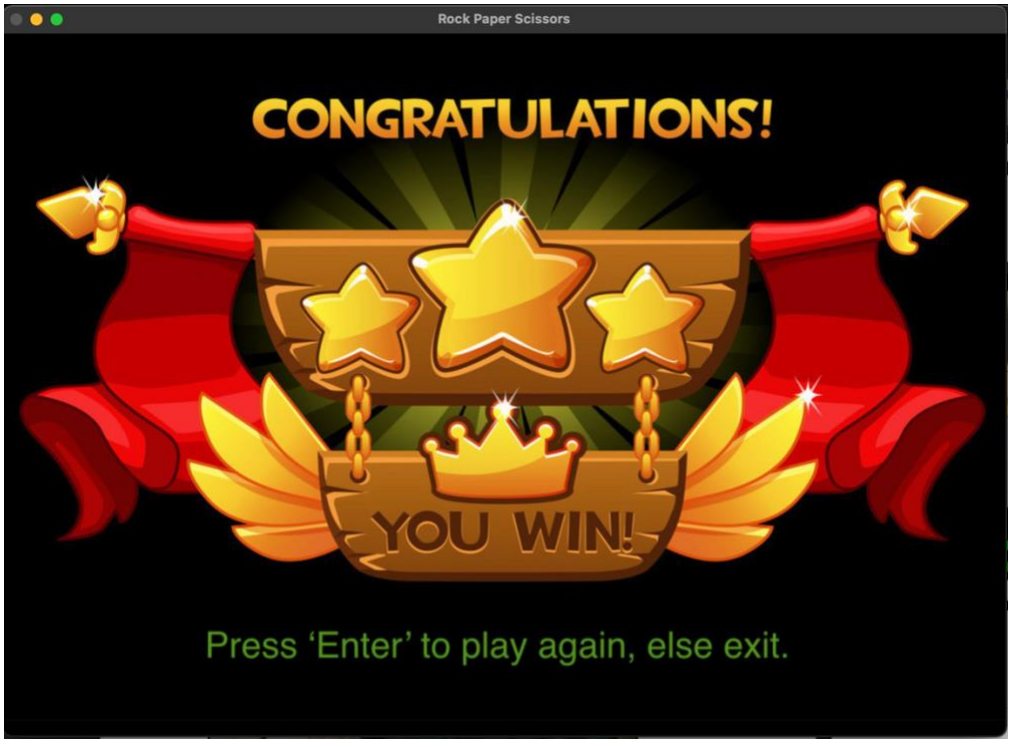
After the user enters the number of rounds, a countdown of 3 seconds starts



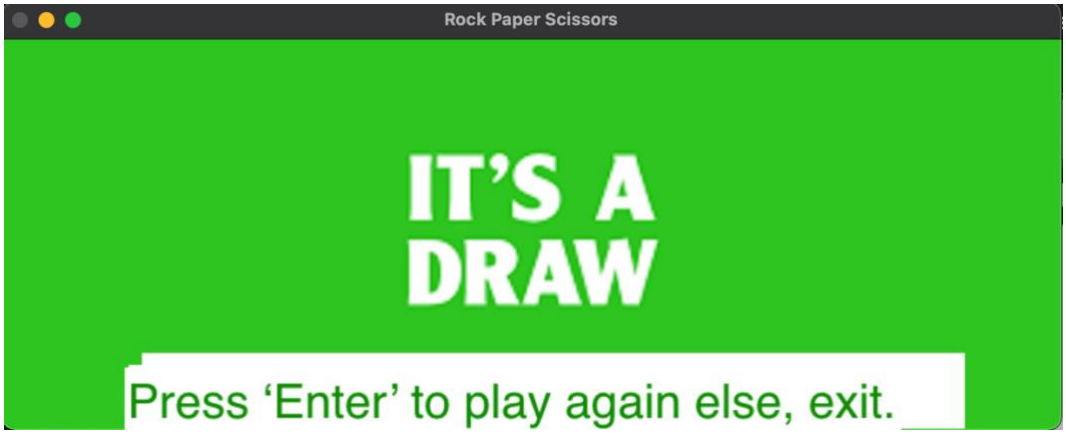
A screenshot where the user has shown ‘rock’ while computer has shown ‘scissor’.

User can play for maximum 10 rounds continuously. After each round, the score will be updated. When the no. of rounds selected by the user gets over. A result screen declaring the final winner of all rounds is displayed.

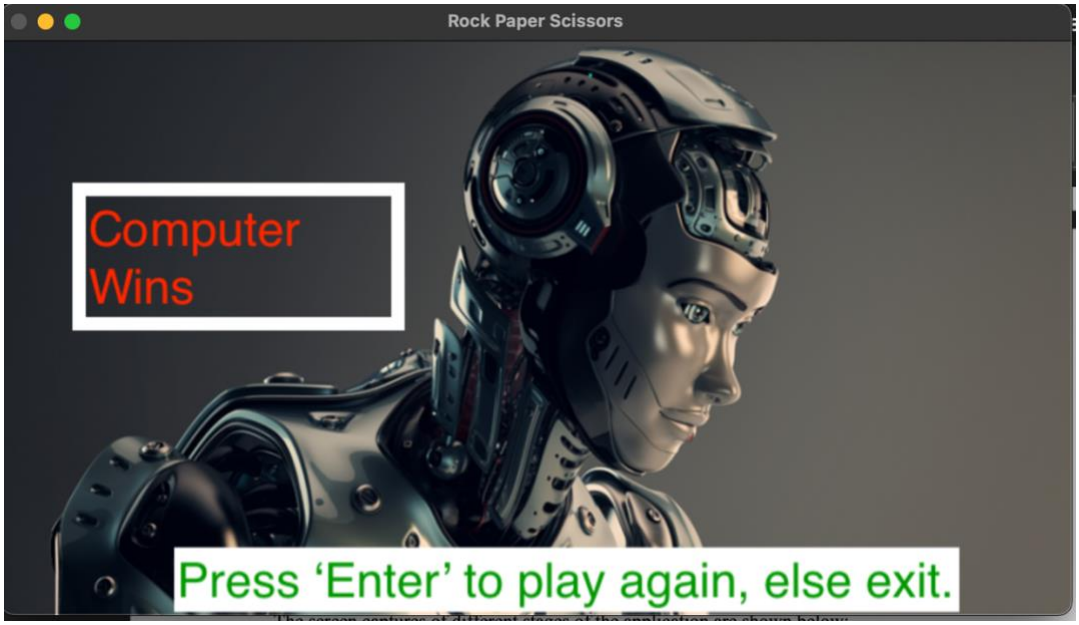
Example of such a screen is shown below:



Screen captured when user wins the majority of rounds



Screen captured when user's score is equal to computer's score at the end of all rounds



Screen captured when computer's score is greater than user's score at the end of all rounds.

CHAPTER 7

CONCLUSION

This mini-project gave me excellent hands on experience in the field of computer vision and python project development. I got the opportunity to explore and use the hot technologies of today such as Machine Learning, OpenCV, MediaPipe library, etc.. I could have implemented the hand gesture classifier without MediaPipe using my machine learning model, but using MediaPipe increased the power of my classifier many folds, as it detected the hands accurately and quickly without much processing power. I also got the opportunity to apply my knowledge of machine learning to create a model for real world project and it gives me immense pleasure to see that model giving high accuracy. I feel excited to work with bleeding edge tools in Artificial Intelligence and cannot wait to delve deeper in this field.

With the rapid advent of AI and Interactive User Experiences, the demand for professionals in this field is increasing and this has created a huge job opportunity. I am very sure that the knowledge I gained during this mini-project will be beneficial.

CHAPTER 8

REFERENCES (in IEEE format)

1. Google LLC “MediaPipe Hands” google.github.io https://google.github.io/mediapipe/solutions/hands (accessed Oct. 11, 2021).

2. Bernard Marr “7 Amazing Examples Of Computer And Machine Vision In Practice” forbes.com https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/?sh=63809bef1018 (accessed Nov. 1, 2021).

3. Taha Anwar “Playing Rock, Paper, Scissors with AI” learnopencv.com https://learnopencv.com/playing-rock-paper-scissors-with-ai/ (accessed Oct. 25, 2021).