Project: Hand Cricket

Submitted in fulfillment of the requirement for

ArIES

2nd-year recruitment project

Mentored by:- Maanas Verma (3rd Year EPH)
Pranjal Mangal (3rd Year EE)

Submitted by:- Akshat Gupta (ECE)
Yatin Mittal (Mech)
Shrashtika Singh (EE)

Brief Description:

Playing Hand Cricket with a computer, with gesture detection.

Ideation Process:

- **Rules of the Game**- Detailed discussions regarding the rules of the game.
  1) The game will offer the players up to six hand gestures (1 to 6) to play the game only with the right hand.
  2) The game will only be a single-player game in which the player plays against the computer. User will do the batting, and the computer will do the balling.
  3) The computer will generate gestures using random functions and whenever that value matches with the player gesture, then the game will be over.

Hand Gesture Detection:

Requirement: *The player should play this game in a well-lighted area as proper finger detection requires sufficient light.*

Studying various online examples and hand detection repositories to come up with the best hand detection method.
Choices between - OpenCV, Yolo, and CNN.

1) **OpenCV**- Less error-prone and a lot faster, but difficult to configure.
2) **Yolo** - Training of data requires a lot of time as it requires labeling of each image.
3) **CNN** - Too slow, and the response time is way ahead of other methods. Not feasible for this project.

Work on OpenCV:

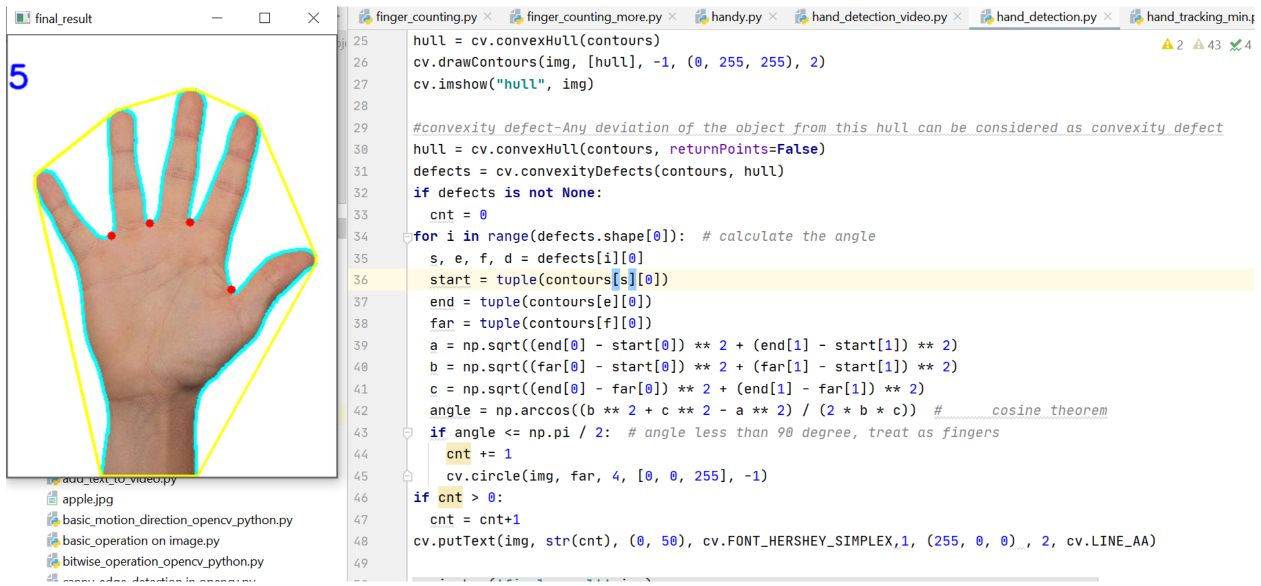
We start with a basic medium article, “**Hand Detection and Finger Counting Using OpenCV-Python.**”

Reference:

https://medium.com/analytics-vidhya/hand-detection-and-finger-counting-using-opencv-python-5b594704eb08

We used simple OpenCV techniques like Skin Masking, Contours, Convex Hull, and Convexity defects and rules of cosine theorem in this algorithm.

And we got the results as shown below:



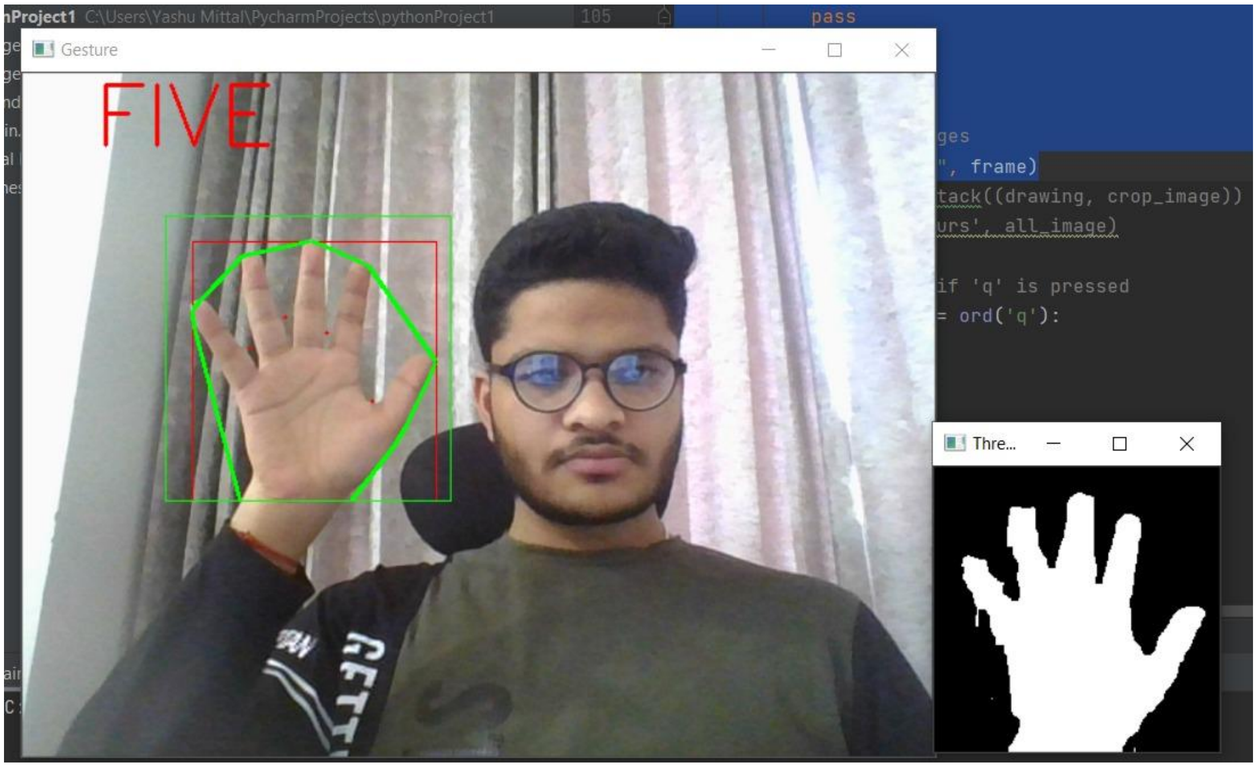
```
hull = cv.convexHull(contours)
cv.drawContours(img, [hull], -1, (0, 255, 255), 2)
cv.imshow("hull", img)

#convexity defect-Any deviation of the object from this hull can be considered as convexity defect
hull = cv.convexHull(contours, returnPoints=False)
defects = cv.convexityDefects(contours, hull)
if defects is not None:
  cnt = 0
for i in range(defects.shape[0]):  # calculate the angle
  s, e, f, d = defects[i][0]
  start = tuple(contours[s][0])
  end = tuple(contours[e][0])
  far = tuple(contours[f][0])
  a = np.sqrt((end[0] - start[0]) ** 2 + (end[1] - start[1]) ** 2)
  b = np.sqrt((far[0] - start[0]) ** 2 + (far[1] - start[1]) ** 2)
  c = np.sqrt((end[0] - far[0]) ** 2 + (end[1] - far[1]) ** 2)
  angle = np.arccos((b ** 2 + c ** 2 - a ** 2) / (2 * b * c))  #      cosine theorem
  if angle <= np.pi / 2:  # angle less than 90 degree, treat as fingers
    cnt += 1
    cv.circle(img, far, 4, [0, 0, 255], -1)
if cnt > 0:
  cnt = cnt+1
cv.putText(img, str(cnt), (0, 50), cv.FONT_HERSHEY_SIMPLEX,1, (255, 0, 0) , 2, cv.LINE_AA)
```

Code available at:

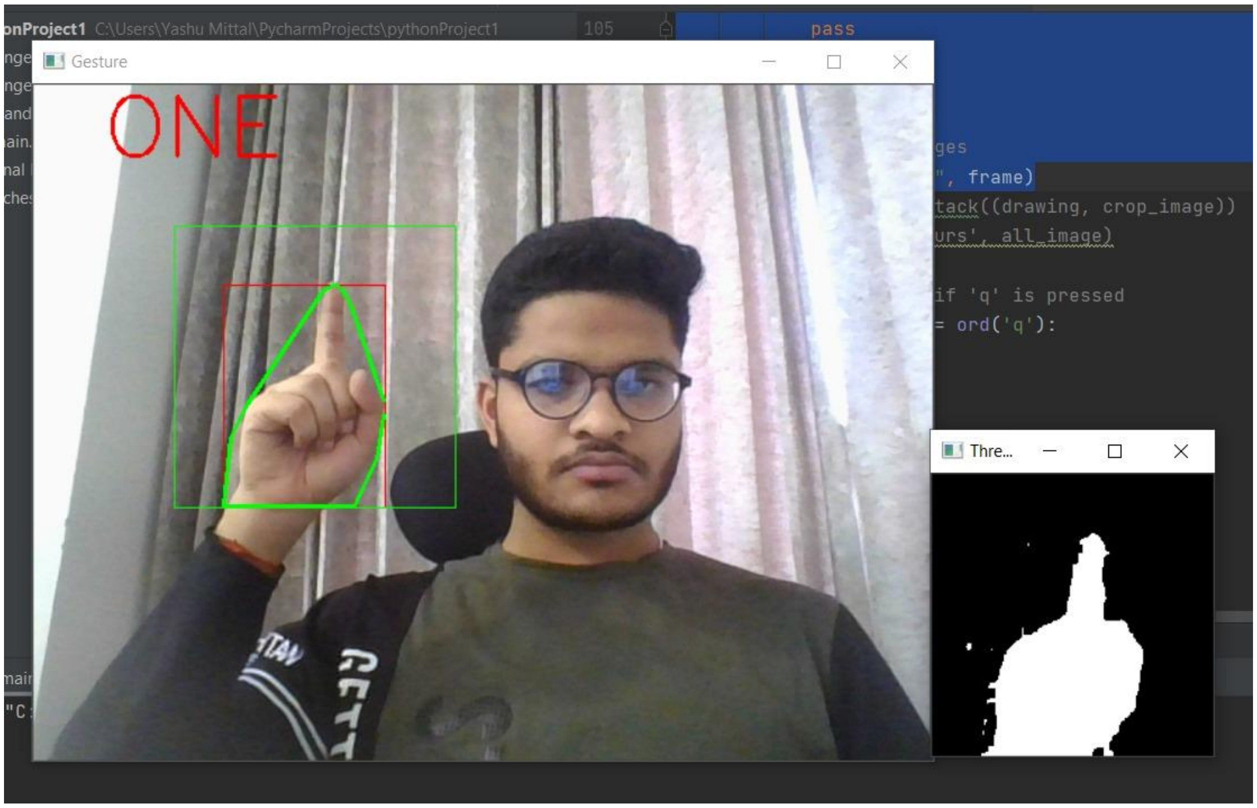
https://github.com/ariesiitr/Hand-Cricket/blob/shrashtika/hand_detection.py

After that, we implemented this algorithm on video frames as our project needs to work in real-time detection.


nProject1 C:\Users\Yashu Mittal\PycharmProjects\pythonProject1
Gesture
FIVE
Thre...

```
pass
ges
", frame)
tack((drawing, crop_image))
urs', all_image)

if 'q' is pressed
= ord('q'):
```


onProject1 C:\Users\Yashu Mittal\PycharmProjects\pythonProject1
Gesture
ONE
Thre...

```
pass
ges
", frame)
tack((drawing, crop_image))
urs', all_image)

if 'q' is pressed
= ord('q'):
```

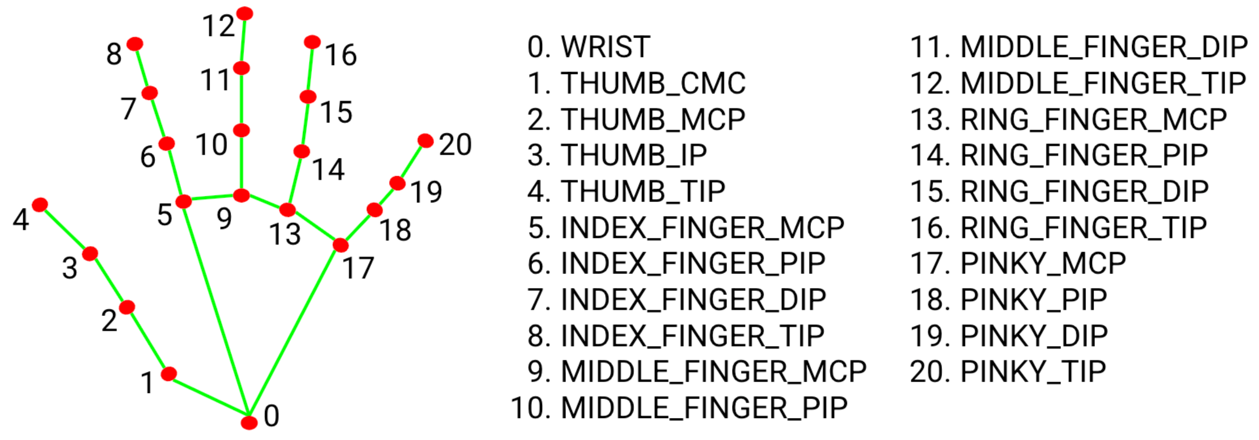
Code available at:
https://github.com/ariesiitr/Hand-Cricket/blob/yatin/main.py

In this code, we found that detection is good, but fluctuations are very high then; we tried to find some method to reduce this fluctuation or a new method with very fewer fluctuations.

In a new method, we use ***MediaPipe***.
MediaPipe offers customizable Python solutions as a prebuilt Python package on PyPI, which can be installed simply with ***pip install mediapipe***. It also provides tools for users to build their solutions.

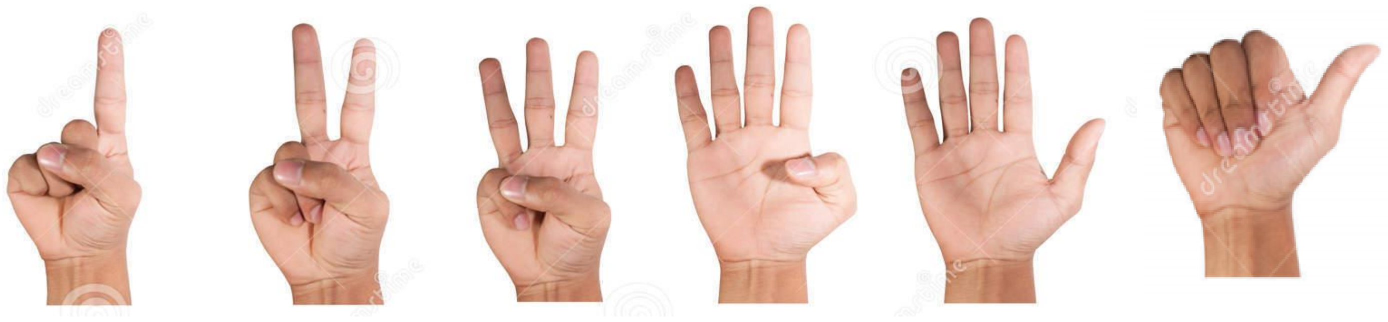
Firstly for palm detection, we use a pre-built ***Palm detection model*** in MediaPipe. MediaPipe Hands utilizes an ML pipeline consisting of multiple models working together. A palm detection model that operates on the full image and returns an oriented hand bounding box. A hand landmark model operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand keypoints.

After the palm detection over the whole image, our 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.



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

Link for the reference:
https://google.github.io/mediapipe/solutions/hands
To implement this algorithm first, we create a class "***handDetector***" available at:
https://github.com/ariesiitr/Hand-Cricket/blob/shrashtika/HandTrackingModule.py
We use this class in our actual model.
Using MediaPipe, the accuracy rate is high and better than the previous algorithm.
First, we write a code to simply count fingers 0 to 5 instead of detecting a particular hand pose.

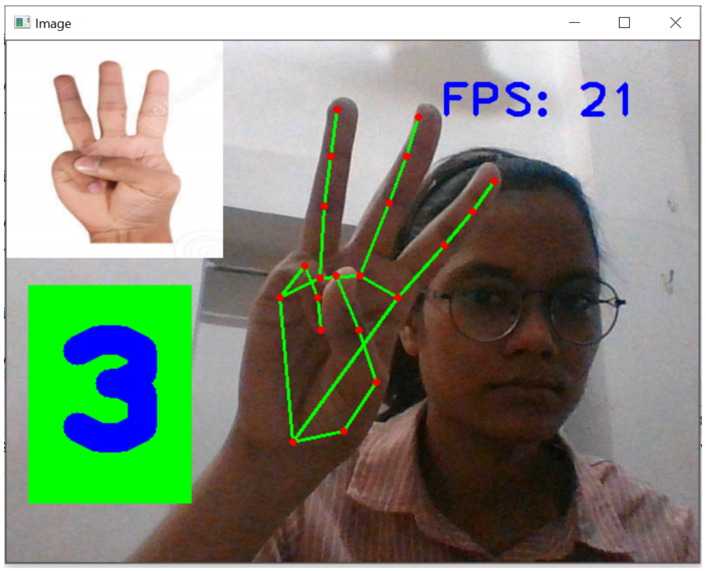


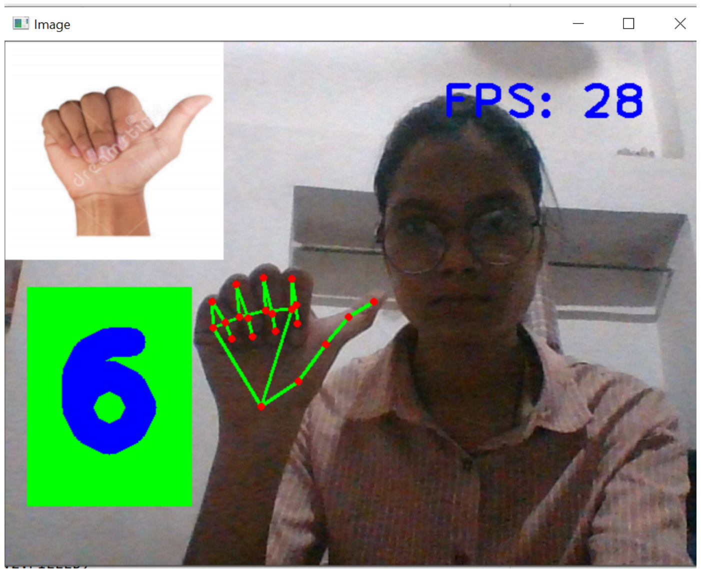
For detecting particular six hand poses shown in the above snap to make our model more effective, we write unique code lines for each particular hand pose.

<u>Algorithm used</u>:

To detect the particular hand pose of the ***right hand*** we write unique code for each hand pose, the algorithm used for 4 fingers is up-down. If the top point of any of the fingers is above the bottom point of finger that means the finger is open otherwise close. For thumb, we used the left-right algorithm. If the thumb of the right hand is the right side of the screen then it would be considered as open otherwise close.

Results:



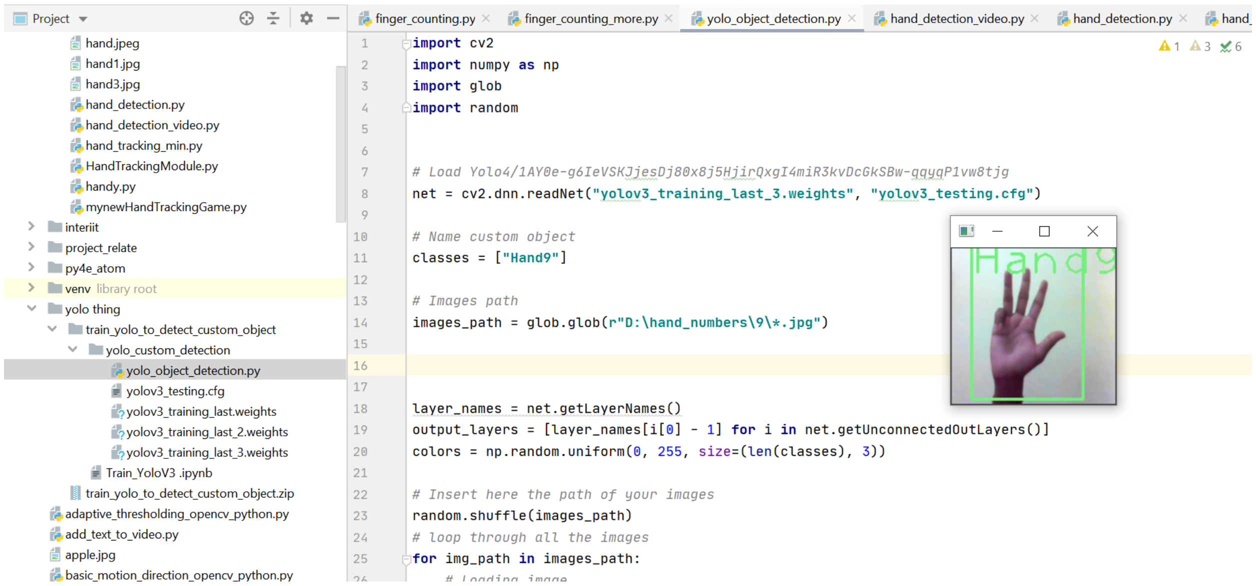


Code available at:
https://github.com/ariesiitr/Hand-Cricket/blob/shrashtika/finger_counting_1to6.py
Till now, we have found this code is suitable for our model. And the problem of background detection and uneven background was also resolved. Now the gestures can be recognized in a variety of different backgrounds.

YOLO Work and why we are selecting OpenCV over YOLO:

Reference:
https://pysource.com/2020/04/02/train-yolo-to-detect-a-custom-object-online-with-free-gpu/
We needed to prepare an image dataset of 10 different hand gestures and for each hand gesture minimum of 350 images. And then specify where the custom object is located on the specific image.
For this operation, we used an external software: ***LabelImg*.**
We need to specify the custom objects in each specific image i.e., in 350*10 images.

To train the image dataset, we used the free server offered by google colab(Free GPU). For this, we used a notebook named ***“Train_YoloV3.ipynb”***.
Notebook Available at:
https://github.com/ariesiitr/Hand-Cricket/blob/shrashtika/Train_YoloV3.ipynb
We are required to complete at least 1000 iterations of training. We used a pre-built python project to test your model with OpenCV.



```
        hand.jpeg
        hand1.jpg
        hand3.jpg
        hand_detection.py
        hand_detection_video.py
        hand_tracking_min.py
        HandTrackingModule.py
        handy.py
        mynewHandTrackingGame.py
    > interiit
    > project_relate
    > py4e_atom
    > venv  library root
    v yolo thing
        v train_yolo_to_detect_custom_object
            v yolo_custom_detection
                yolo_object_detection.py
                yolov3_testing.cfg
                yolov3_training_last.weights
                yolov3_training_last_2.weights
                yolov3_training_last_3.weights
            Train_YoloV3 .ipynb
        train_yolo_to_detect_custom_object.zip
    adaptive_thresholding_opencv_python.py
    add_text_to_video.py
    apple.jpg
    basic_motion_direction_opencv_python.py
```

```python
import cv2
import numpy as np
import glob
import random


# Load Yolo4/1AY0e-g6IeVSKJjesDj80x8j5HjirQxgI4miR3kvDcGkSBw-qqyqP1vw8tjg
net = cv2.dnn.readNet("yolov3_training_last_3.weights", "yolov3_testing.cfg")

# Name custom object
classes = ["Hand9"]

# Images path
images_path = glob.glob(r"D:\hand_numbers\9\*.jpg")



layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))

# Insert here the path of your images
random.shuffle(images_path)
# loop through all the images
for img_path in images_path:
    # Loading image
```

Code available at:
https://github.com/ariesiitr/Hand-Cricket/blob/shrashtika/yolo_object_detection.py
This model was working well to some extent. Still, the problem is that it is very time-consuming, like the specification of the custom object using Labeling and training of dataset on google colab. Due to this, we switch to OpenCV.

Finally, we decided to go with the OpenCV.

<u>Time and Random module</u>:

We used a timer to provide the player with sufficient time to be ready with his/her hand gesture using Timer module. We have set

the timer for 2 seconds. Timer module is an inbuilt module in python.
We used **random.randint**() function of random module for generating the computer's number. Random module is an inbuilt module in python. The **random.randint**() method returns a random integer between the specified integers.

We also add two more functions. First, for displaying the number shown by the player and generated by the computer in one round. And second, for displaying the final score of the player and who is the winner.
We are simply using numpy library in these functions.

After fulfilling all the requirements of the game we integrate all the above-discussed functions and modules on a single python file.
Python file available at:
https://github.com/ariesiitr/Hand-Cricket/blob/shrashtika/final.py

How the game looks like and how it works:

In this hand cricket game, user will do the batting i.e. in every round the user will always choose a number at first. After that computer will generate its number using random module. This will happen again and again until the user number and computer's number are unequal. And display
Internally, the code will add the user's score and computer's score in every round. When the values of user and computer will match then the game will over and the screen will display the final score of the user and display who is the winner.

How to play:

1) Run the python file

2) A window ‘a’ display, press ‘q’ to start a timer of 2 seconds. User should be ready with hand gestures.
3) After 2 seconds hand gesture is detected and displayed on the other window ‘Image’.
4) Press close ‘x’ to close the ‘Image’ window. Press ‘esc’ to close the window ‘a’.
5) After that computer number displays on a new window ‘Image’. Press ‘x’ to close the window.
6) A new window ‘Image’ pops up which displays the instant score of both computer and user in that round. Press ‘x’ to close the window.
7) Again window ‘a’ pop up for the user’s gesture detection and the same process will repeat until the value of user and computer will match.
8) When the values match, then after displaying the instant score, a new window pops up and displays the total score of the user and the winner. Press ‘x’ to close the open window and the program stops.