ARCHITECTURE DIAGRAM

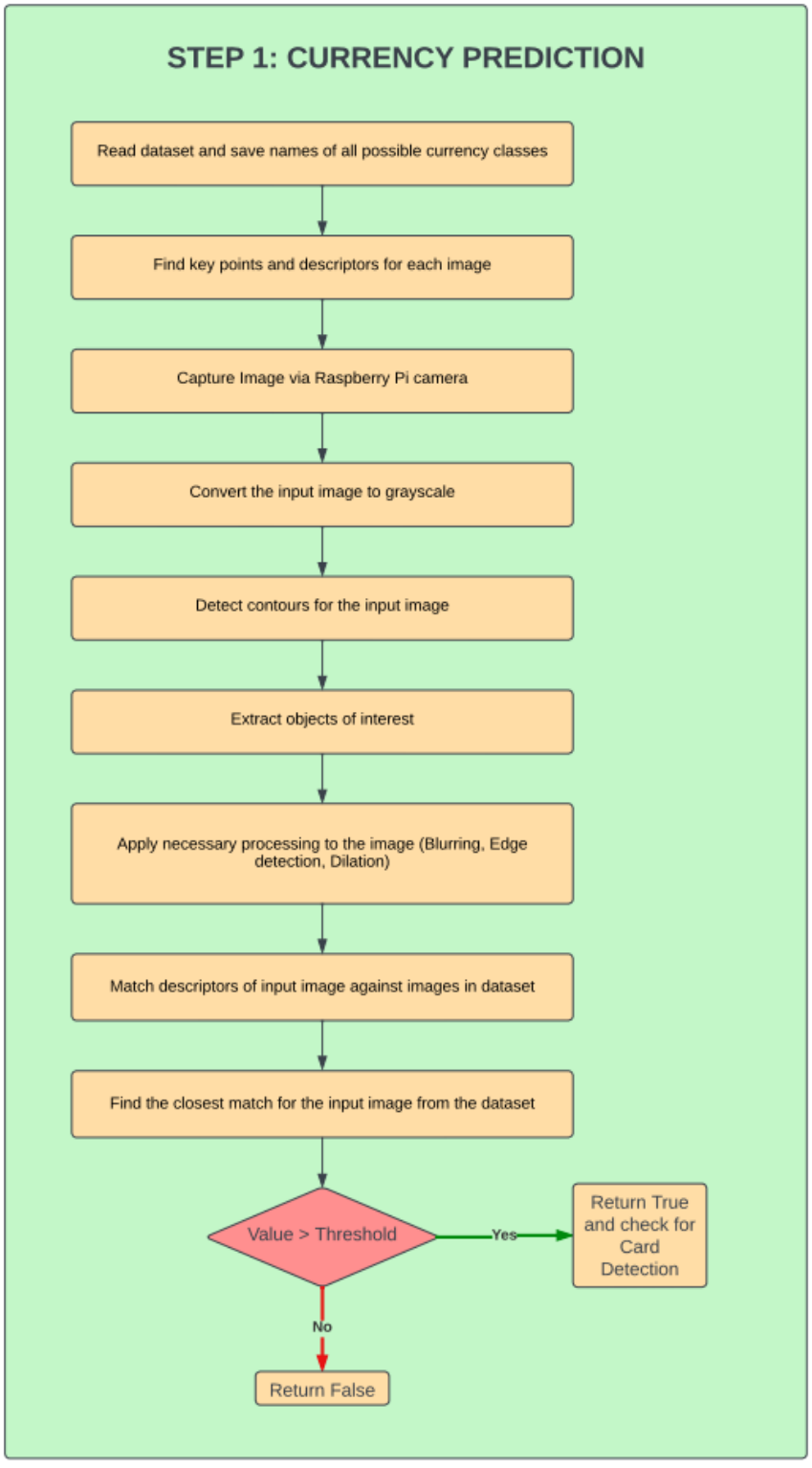

fig a: The Architecture Diagram for Currency Classification

EXPLANATION OF CURRENCY CLASSIFICATION ARCHITECTURE (fig a)

1. Read dataset and save names of all possible currency classes:

The first step is to read the dataset that contains images of different currency notes. The names of all possible currency classes are then saved. This step is crucial as it helps to identify and categorize the currency notes correctly.

2. Find key points and descriptors for each image:

The key points and descriptors for each image in the dataset are identified using a feature detection algorithm such as SIFT, SURF or ORB. These key points represent the unique features of each currency note, and the descriptors are the numeric values that describe these key points.

3. Capture Image via Raspberry Pi camera:

The next step is to capture an image of a currency note using a Raspberry Pi camera. This image will be used to compare against the images in the dataset.

4. Convert the input image to grayscale:

The input image is then converted to grayscale. This step helps to simplify the image and reduce the amount of data that needs to be processed.

5. Detect contours for the input image:

Contours are then detected in the grayscale image. Contours represent the boundaries of the objects in the image, and they can be used to extract the objects of interest.

6. Extract objects of interest:

The objects of interest, in this case, the currency note, are then extracted from the image using the contours detected in the previous step. This step helps to isolate the currency note from the background.

7. Apply necessary processing to the image (Blurring, Edge detection, Dilation):

The extracted image is then processed to improve the accuracy of the currency note detection. Techniques such as blurring, edge detection, and dilation are

applied to the image to remove noise and enhance the features of the currency note.

8. Match descriptors of input image against images in dataset:

The descriptors of the input image are then matched against the descriptors of the images in the dataset. This step helps to identify which currency note the input image most closely resembles.

9. Find the closest match for the input image from the dataset:

The closest match for the input image is then identified based on the similarity between the descriptors. The currency note that has the most similar descriptors to the input image is selected.

10. If the value is greater than threshold, return true; else, return false:

The final step is to compare the similarity score to a predefined threshold. If the similarity score is greater than the threshold, the input image is classified as the identified currency note, and the system returns a positive result. If the similarity score is below the threshold, the system returns a negative result, indicating that the input image does not match any currency note in the dataset.

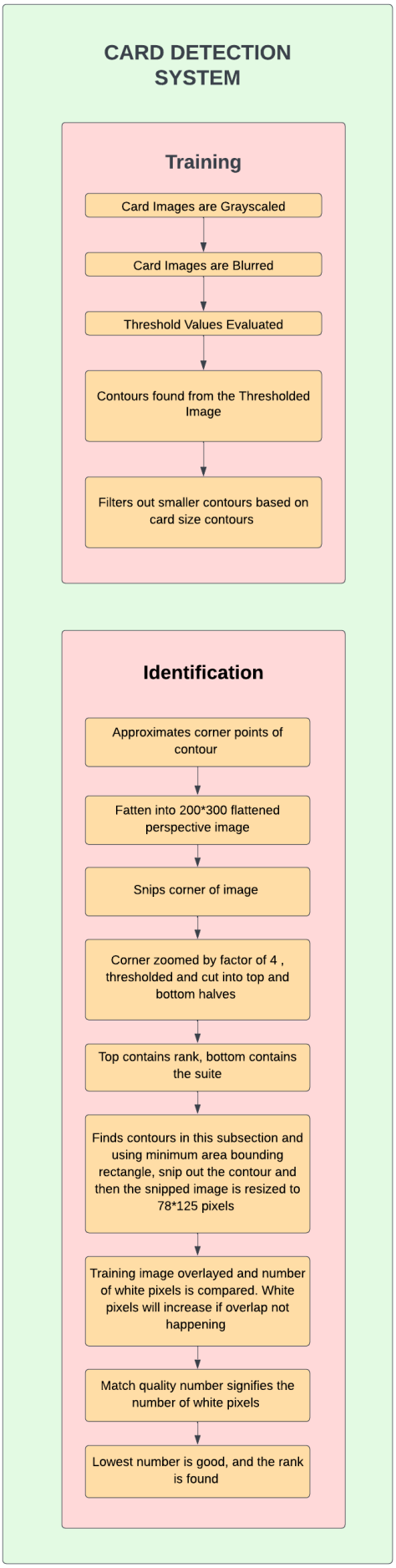

fig b: The Architecture Diagram for Playing Card Classification

EXPLANATION OF PLAYING CARD CLASSIFICATION ARCHITECTURE (fig b)

The algorithm to detect playing cards involves two major stages: detection and identification.

Detection:

1. **Grayscaled**: The first step is to convert the input image to grayscale, which reduces the amount of data that needs to be processed and simplifies the image.

2. **Blurred**: A Gaussian blur is applied to the grayscale image, which helps to reduce noise and improve edge detection.

3. **Thresholded**: The blurred image is then thresholded to convert it into a binary image with clear contours.

4. **Finds contours and thresholded image:** Contours are then detected in the thresholded image.

5. **Filters out smaller contours based on card size contours:** Contours that are smaller than a predefined size is filtered out as they are unlikely to be playing cards.

6. **Now knows how many cards are present in the image**: The number of remaining contours is then used to determine how many cards are present in the image.

Identification:

1. **Identifies rank and suit and compares to predefined rank and suit from images:** The rank and suit of each card are identified by comparing the card's contours to predefined images of card ranks and suits.

2. **Approximates corner points of contour:** The corners of each card are then approximated using the contours detected in the previous step.

3. **Uses that to flatten into 200*300 flattened perspective image:** The image is then flattened into a 200 x 300 pixel image by applying a perspective transformation to the corners of the card.

4. **Snips corner of image:** The corners of the flattened image are then cropped to isolate the rank and suit of the card.

5. **Corner zoomed by factor of 4, thresholded and cut into top and bottom halves:** The corner is then zoomed in by a factor of 4, thresholded, and divided into the top half, which contains the rank, and the bottom half, which contains the suit.

6. **Top contains rank, bottom contains the suite:** The rank and suit are then detected in each half by finding contours and using a minimum area bounding rectangle to crop the contour.

7. **Finds contours in this subsection and using minimum area bounding rectangle, snip out the contour and then the snipped image is resized to 78*125 pixels:** The cropped rank and suit images are then resized to a predefined size (78 x 125 pixels) to allow for comparison to training images.

8. **The isolated rank is called query image and compared to the training image:** The isolated rank image is then compared to the corresponding training image using a technique called template matching.

9. **Training image overlayed and number of white pixels is compared. White pixels will increase if overlap not happening:** The query image is overlaid onto the training image, and the number of white pixels in the overlap region is compared.

10. **Match quality number signifies the number of white pixels**: The number of white pixels is used as a match quality number, with lower numbers indicating a better match.

11. **Lowest number is good, and the rank is found:** The rank with the lowest match quality number is selected as the most likely match, and the rank of the card is identified.

LAYERED ARCHITECTURE

The architecture diagram consists of four layers:

- Physical
- Network
- Process
- Application

Physical layer

- We use the Wi-Fi card to allow the device to communicate with concerned authorities and report its findings.
- The GPS sensor is used to report the exact location where the laws are being violated.
- Raspberry Pi is the device used to process the live footage and detect playing cards and currency notes.
- A smart camera is used to obtain the live footage and fed into the device.
- Memory card is used to store key frames from the live footage and help store data in case of loss of connection.

Network layer

- 802.11 protocol is used for communication through the WiFi module.
- HTTP along with TCP protocol offers good control and security for data transmission.

Process layer

- The device performs object detection on the live video footage to detect playing cards and cash to classify gambling is taking place or not.
- Location is detected to report the exact coordinates where the concerned authorities need to reach and arrest the wrongdoers.
- Alert management is used to recognise gambling and prepare the required report to be sent to the concerned authorities**.**

Application layer

- We use the cloud to store data for future references
- Google maps is used to provide pins for the current location of the device
- Web server is used for communication between the devices and the concerned law enforcement officers.

METHODOLOGY

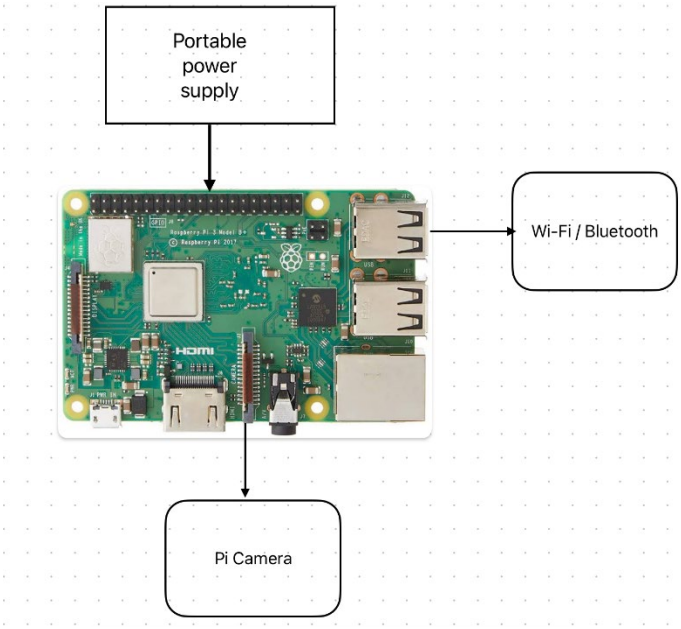

fig c: Raspberry Pi Connections

The project will be implemented using a Raspberry Pi which is a small but capable processing hardware. It will be connected to a Pi camera which allows it to get real time video data of the surroundings to detect gambling. Its highly portable nature and efficiency makes it a good choice to be used in disguise as different props possible. Upon detection, it can notify higher authorities to take the required action. To perform these roles, it is made portable, with its own power supply and Bluetooth or Wi-Fi module for communication.

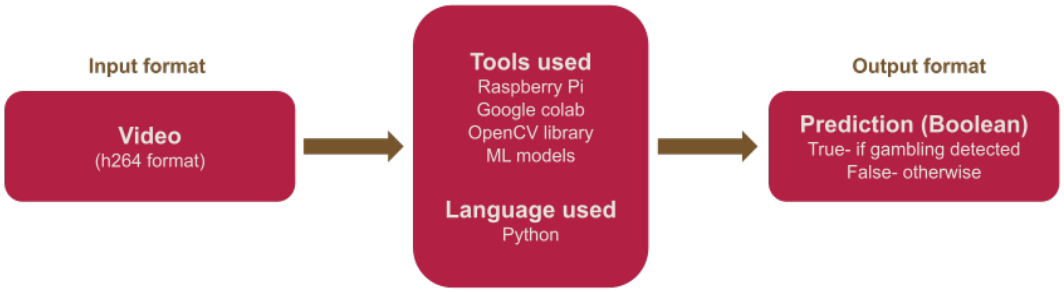

fig d: Process Diagram of Gambling Detection

The analysis is done using OpenCV which is a huge open-source library for computer vision, machine learning and image processing. Here, we use OpenCV to grab the real time data and perform object detection for cards and money which together could signify gambling taking place. The whole project will be implemented using Python on a Jupyter notebook and can be ran on Raspberry Pi or windows machine.