Blackjack Advisor - EE250 Final Project

Abir Bhatt and Vrinda Gandhi

Demo Video

1. Project Description

In blackjack, the game starts with the player being dealt 2 face-up cards, and the dealer being dealt one face-up card and one face-down card. This is the point in the game that requires the most amount of strategy, as you have to decide whether to hit, stand, double, or split.

As you play more and more consecutive hands, an advanced strategy used is card counting using the Hi-Lo method. In this strategy, a value is assigned to each card that is placed throughout the game. Low cards (2 -6) increase the count, and high cards like face cards and aces decrease it. A high running count indicates high cards remaining in the deck, which is favorable for the player. This count is used to calculate the expected value (EV) of each possible action, which is the average outcome that the player would get if they played that action many times. The action with the highest EV would be the optimal decision for the player. You can then use the Kelly Criterion formula to determine how much to bet.

To aid inexperienced players with these decisions, we chose to exploit the mathematical component of blackjack and built a system that automates it. Our EE250 Final Project is a blackjack advisor that detects the player's two cards and the dealer's face-up card, computes the statistically optimal action using EV analysis over the current deck composition, and then displays the recommendation both over a UI and in an email to the player.

2. System Block Diagram

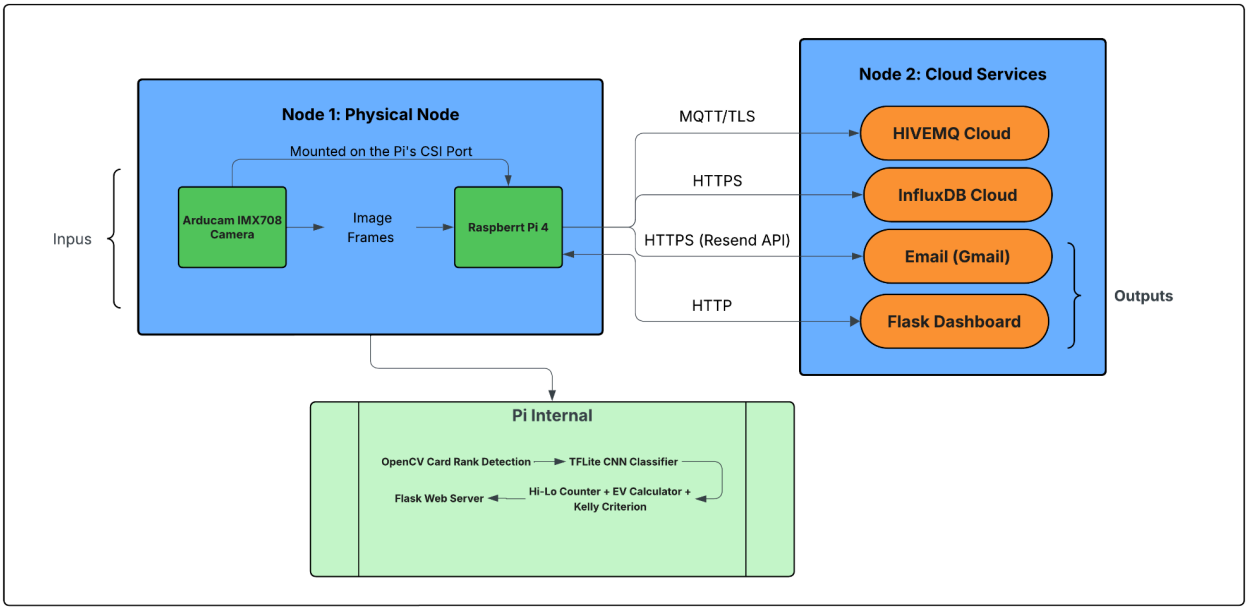

Fig. 1. Block Diagram of the Entire System with Inputs and Outputs

3. Components, Protocols, and Design Choices

Hardware: We used a Raspberry Pi 4 as the central processing node. An Arducam IMX708 Pi Camera is connected to the Pi and mounted on top of a lamp so it captures a birds-eye view of the table.

CV: To detect the cards, we used an OpenCV pipeline that compares each frame against a baseline image of the empty table and then subtracts the background by isolating any pixels that changed (i.e cards). The resulting image is converted to grayscale and smoothed with a Gaussian blur to reduce external noise. OpenCV's contour detection thresholds each pixel, finds the outlines of those white regions, resulting in a bounding box for each detected card.

ML: To classify the cards by rank, we used a MobileNetV2 CNN architecture, which we chose over traditional CNNs since it used lesser parameters and proved more compatible with our components. We took 8 pictures of each card, changing the orientation and position of each, and then organized all of the processed images into folders by rank. We then uploaded them to Google Colab, and ran the MobileNetV2 model on a T4 GPU. The trained model was then converted to a Tensorflow Lite format so it could be run on the Pi.

Counting/EV: Once all three cards are detected and processed, the system updates the Hi-Lo count and computes the EV of each possible action (hit, stand, double, split). The Kelly Criterion then determines the size of the bet based on the situation.

Communication: The Pi communicates over 3 protocols. Card events and recommendations get published to HiveMQ Cloud over MQTT. The running count, true count, and bet recommendation get logged to InfluxDB Cloud using the REST HTTPS API. We chose InfluxDB over MQTT because it is used for time-series data, especially after multiple hands. The recommendations are displayed via a Flask dashboard running on the Pi, and emailed to the player using the Resend Email API (REST HTTPS).

We used Claude (Anthropic) to assist with writing the code, but all design decisions were our own.

4. Limitations and Lessons Learned

Dealer's face down card: Our original design tried to account for the dealer's face-down card to simulate a real blackjack deal. Unfortunately, since our model was only trained on face-up cards, when it would see the face-down card, it would guess a random rank since it always had to output a prediction. Since adding a stricter confidence filter did not improve its performance, we simplified the layout to the three face-up cards, with the face-down card being dealt off-screen.

Card positioning: Our initial design of sorting the detected boxes from left to right because player cards are dealt above the dealer's and not strictly left-first. We now track placed-card coordinates and identify the new card as the detected box more than 150 pixels from all previously seen cards.

Continuing past the opening sequence: If the player were to keep hitting past the opening sequence, each new card would need to be placed in a separate box in the frame, which has limited space. We attempted to solve this by creating two piles and stacking new cards on top of existing ones, but detecting replaced cards was inconsistent. Our main future goal for this project would be to support a full game.