# Playing Card Recognition and Q&A System

Xinshu Zhao, Hang Xu, Ruoqi Yang
Rice University
xinshu@rice.edu, hx38@rice.edu, ry30@rice.edu

## Abstract

*The project we are doing about is for recognizing playing cards and evaluating the specific poker hands from the real images, which is planned to be able to use in the real time. A multi-task convolutional neural network(CNN) model is fine-tuned based on a pre-trained model(EfficientNet-B3). The model provides two outputs which are rank and suit with a joint loss function combining both loss from rank and loss from suit. To be able to recognize multiple cards in one image, an open-CV module is utilized to find the contour and perspective transformation, so that the model can evaluate a decent image after the image processing. Then we add the rule-based function module to evaluate what kind of poker hands is there in the image.*

## 1. Introduction

Recognizing playing cards from images requires very high accuracy. When multiple cards appear in a single image, the overall accuracy drops exponentially, as the probability of getting all predictions correct is given by:

$$P(\text{All correct}) = \text{accuracy}^n$$

where $n$ is the number of cards in the image. Since we aim to recognize up to 5 cards at once, achieving around 90% overall accuracy requires at least 99% accuracy on single card prediction.

To achieve this goal, we use convolutional neural network (CNN) techniques, which are effective and powerful in feature extraction and image evaluation. We also combine this visual recognition model with an OpenCV-based card detection module, which allows the system to isolate and process each card individually.

Our initial attempt at building a card classification model used the CardDataSet, which contains around 8,000 card images covering 52 different classes (combinations of rank and suit). This model achieved a relatively low accuracy of around 78 percent, which is far from sufficient—since the accuracy of correctly identifying all 5 cards in a hand would drop to approximately 30 percent.

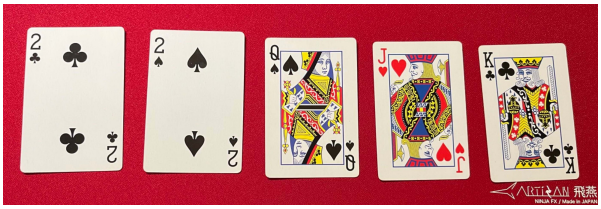


Figure 1: Original Image

Figure 2: Here we show an example how the images with 5 cards get evaluated separately.

To improve this, we came up with the idea of using two separate models: one for predicting the card's suit, and the other for predicting the card's rank. In other words, instead of changing the model architecture, we simply modified the dataset and trained each model to focus on just one of the card's attributes. This approach significantly boosted accuracy to around 99 percent, which would result in approximately 90 percent accuracy when evaluating 5-card hands image.

However, since we aim to build a real-time poker-hand evaluation system, we need a model with faster response time. Running two separate models doubles the computational cost. To address this, we combined the ideas into a single model with two output branches—one for rank and one for suit—allowing us to achieve comparable accuracy while cutting the inference time roughly in half.

Once the types and suits of all cards are identified, we use a rule-based reasoning system to determine the poker hand. If the visual recognition step is accurate, this logic-based analysis can achieve nearly 100% correctness.

## 2. Related Work

Image classification and object detection have been extensively studied in computer vision. Traditional methods based on handcrafted features, such as SIFT and HOG, were

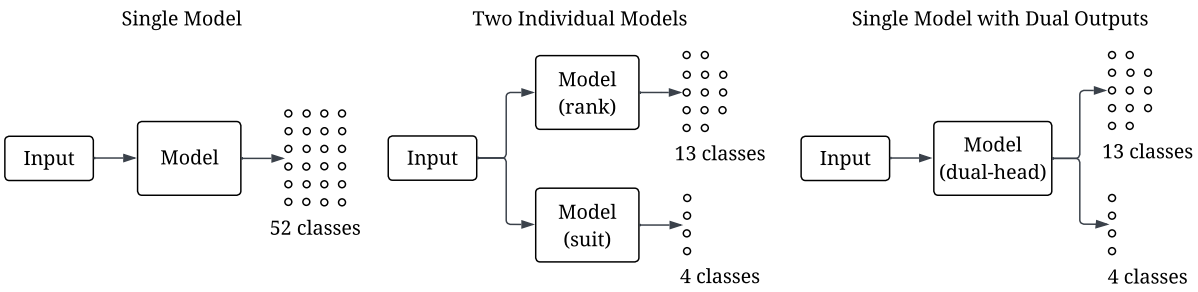



Figure 3: Model Comparison

| | Accuracy | # of Parameters |
|---|---|---|
| Single Model | 78% | X |
| Two Individual Models | 99% | 2X |
| Single Model with Dual Outputs | 99% | X |

Table 1: Model Comparison Table

once widely used. However, deep learning, particularly Convolutional Neural Networks (CNNs), has significantly improved performance on various vision tasks. ResNet [2] introduced residual connections to address the vanishing gradient problem, achieving state-of-the-art results in image recognition. EfficientNet [3] further advanced the field by proposing a compound scaling method to balance network depth, width, and resolution, achieving better accuracy with fewer parameters.

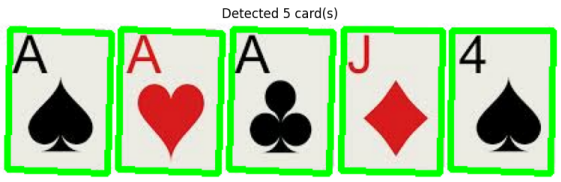



Figure 4: open-CV module example

For image captioning and multi-task outputs, encoder-decoder architectures like Show and Tell [4] demonstrated the effectiveness of combining CNNs for feature extraction with recurrent neural networks (RNNs) for sequential prediction tasks. This idea inspired multi-headed designs where different outputs (e.g., rank and suit) are predicted separately but share a common feature backbone, which improves both accuracy and efficiency.

Preprocessing techniques remain crucial for real-world applications. OpenCV [1] provides robust tools for contour detection, perspective transformations, and image preprocessing. In the specific context of card recognition, preprocessing plays an essential role in isolating individual cards from complex backgrounds or overlapping scenarios, which is often a major challenge in practice.

Our work integrates these ideas: leveraging OpenCV for preprocessing, using a multi-headed CNN model inspired by encoder-decoder structures, and applying modern lightweight architectures like EfficientNet for efficient real-time card classification.

## 3. Methodology

The model for card recognition uses a 2-Head architecture to predict both the rank (13 classes) and suit (4 classes) of a playing card. The choice of a 2-Head structure is due to the nature of the problem: combining both tasks into a single classification (52 classes) increases complexity and reduces accuracy since ranks and suits are independent features. By separating them, the model can better capture specific patterns relevant to each attribute.

The backbone of the model is EfficientNet-B3, chosen for its ability to balance accuracy and computational efficiency. Compared to heavier architectures like ResNet-50, EfficientNet achieves higher accuracy with fewer parameters due to its compound scaling method. This makes it suitable for applications requiring both high precision and speed, especially important when processing multiple cards in real-time. The final layers include two independent heads that output the predicted rank and suit, allowing more accurate and robust classification.

We defined the overall loss function as

$$\mathcal{L}_{\text{total}} = 0.67\,\mathcal{L}_{\text{CE}}(\hat{y}_{\text{rank}}, y_{\text{rank}}) + 0.33\,\mathcal{L}_{\text{CE}}(\hat{y}_{\text{suit}}, y_{\text{suit}})$$

where $\mathcal{L}_{\text{CE}}$ denotes the standard Cross-Entropy loss function.

We assign a larger weight (0.67) to the loss from rank prediction because rank prediction has more categories than suit prediction. A misclassification in rank prediction could happen more often than suit prediction. Also a misclassification in rank has greater impact on the poker hands evaluation. By adjusting the loss weights, we could obtained a more balanced classification model.

### 3.1. Data Augmentation

To simulate real-world conditions where playing cards are captured via mobile cameras, we applied strong data augmentation techniques during training. These augmentations include random rotations, brightness and contrast adjustments, partial occlusions, perspective transformations, and slight blurring. Such strategies are crucial to improve the model's robustness against common visual distortions, such as tilted angles, uneven lighting, and incomplete card views. By introducing significant variability into the training data, we aim to ensure that the model can generalize well to diverse and challenging environments.

### 3.2. OpenCV Module

In this project, we utilized a classical OpenCV-based pipeline to detect and extract poker cards from images. The processing sequence includes grayscale conversion, automatic binarization using Otsu's thresholding, morphological closing and dilation to enhance card contours, and contour extraction to identify potential card regions. We used polygonal approximation to make sure we only take quadrilateral shapes. We also performed a perspective transformation based on finding its width and height, as the model is train by only up-right images. Laplacian edge detection was then added to filter out noisy or blank regions by measuring the edge strength. This multi-stage process ensures that the final extracted cards are geometrically corrected, consistently sized, and cleanly separated so that it can be used as the good input for the downstream module which is prediction.

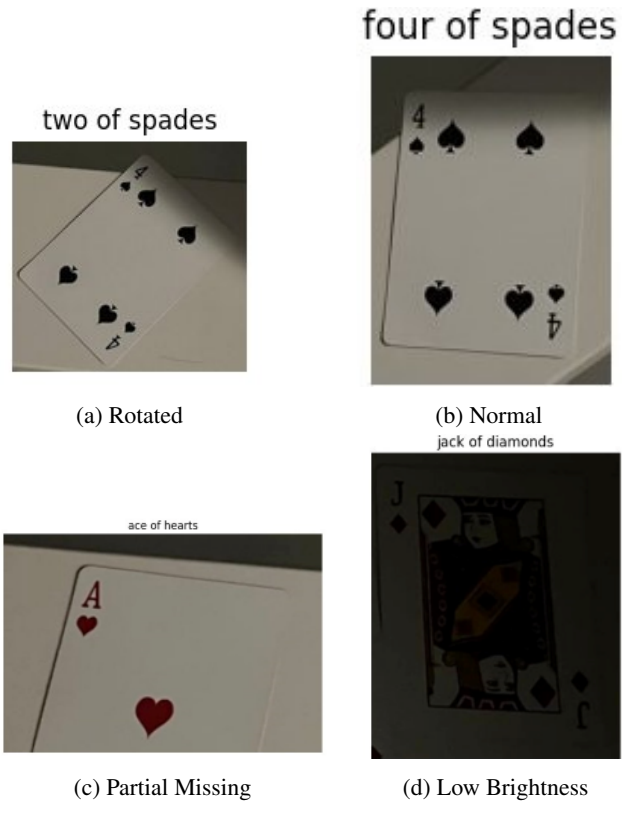
### 3.3. Integration with Large Language Models

To enhance user interaction and provide richer textual responses, we connected our card recognition system to OpenAI's GPT-4o large language model through API integration. After the visual model predicts the rank and suit of each card, the recognized results are formatted and sent as prompts to GPT-4o. The language model then generates natural language descriptions or answers user queries related to the identified hand.

This hybrid setup allows our system not only to perform visual recognition but also to serve as an intelligent chatbot that can explain poker hands, suggest strategies, or engage users in more dynamic conversations. By leveraging the powerful language understanding capabilities of GPT-4o, we significantly improve the user experience and broaden the application's potential beyond simple classification tasks.

## 4. Experiment

We have tested many images under different environment conditions, including brightness, missing part of the card and weird angle and so on. Here are some of examples of testing extreme conditions.




(a) Rotated (b) Normal

(c) Partial Missing (d) Low Brightness

Figure 5: Four types of card input conditions

The first example gives an incorrect prediction because the model are fed with data that not rotated, we decided to make this be repaired in the open-CV module which processes a perspective transformation to each card. Besides that, the model could handle other cases well. More example on evaluating an image with more than one cards indicating the hand type.

Our system still had a big problem facing overlapping images. The reason is that our open-CV module so far is based on detect square like objects by evaluating the contour. When the cards are overlapped to some extent, the system would not be able to divide the cards. We will focus on how to deal with this case in the later project and how to

make it efficient in real-time cases.


Figure 6: Overlapped example

## 5. Results

### 5.1. Training and Validation Accuracy

Figure 7 shows that the card classification model, based on a pre-trained EfficientNet-B3 backbone, converged rapidly and achieved high accuracy with minimal overfitting. Early stopping occurred around epoch 4, where validation accuracies for both rank and suit predictions exceeded 98%. These strong and stable visual recognition capabilities provide a reliable foundation for subsequent natural language interaction tasks.

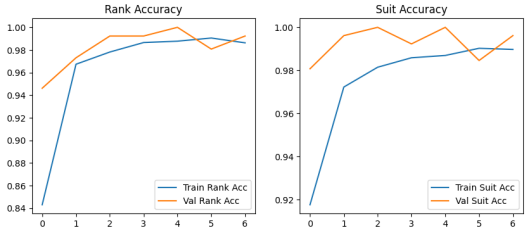



Figure 7: Training and validation accuracies for rank (left) and suit (right) classification over epochs.

### 5.2. Evaluation results

To evaluate our system's real-world performance, we conducted extensive tests using images containing seven playing cards per frame. A total of 100 distinct images were captured under varied conditions, including different lighting, angles, and slight occlusions.

The key quantitative results are summarized as follows:

| Metric | Result |
|---|---|
| Single-Card Recognition Accuracy | 99.42% |
| Seven-Card Hand Recognition Accuracy | 96% |
| Hand Type Recognition Accuracy | 96% |
| Average Chatbot Inference Time | 3 sec |

Table 2: Quantitative evaluation results on seven-card images.

## 6. Conclusion

In this project, we developed a system to detect playing cards from real-world images and evaluate poker hands using a multi-task CNN with OpenCV-based detection. The model accurately predicts both rank and suit, and a large language model (OpenAI GPT-4o) was integrated to enable natural language interaction.

While the visual recognition component is robust under varied conditions, the language model sometimes struggles with complex poker reasoning, leading to occasional inconsistencies. Future work could explore fine-tuning the model with poker-specific data or employing Retrieval-Augmented Generation (RAG) techniques using curated poker rule documents.

Handling overlapping cards remains a key challenge. Future improvements may involve object detection approaches for better segmentation, along with enhancing real-time performance and system robustness.

**Acknowledgments.** We used OpenAI's ChatGPT-4o to assist with code formatting, language refinement, and minor implementation details. All core ideas, system design, and experimental results were developed independently by our team. We affirm that the report reflects our original work and has not involved any form of academic dishonesty.

## References

[1] OpenCV Library. `https://opencv.org/`

[2] K. He, X. Zhang, S. Ren, and J. Sun. D̈eep Residual Learning for Image Recognition.Ïn *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] M. Tan and Q. Le. ËfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.Ïn *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[4] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. S̈how and Tell: A Neural Image Caption Generator.Ïn *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

## A. Extra Figures

Figure 8: Extra example 1

Hand Type: ('Straight', (('jack', 'spades'), ('ten', 'hearts'), ('queen', 'hearts'), ('king', 'hearts'), ('ace', 'spades')))

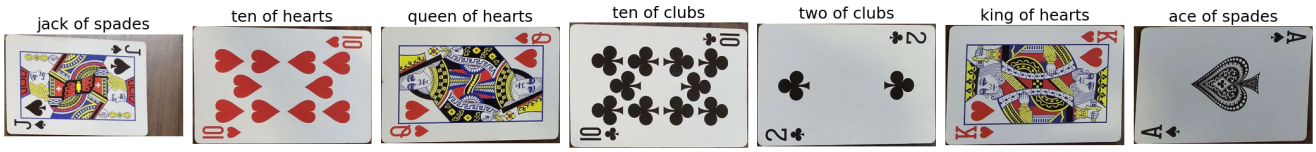



Figure 9: Extra example 1: Recognition results

Figure 10: Extra example 2

Hand Type: ('Two Pair', (('queen', 'clubs'), ('king', 'hearts'), ('eight', 'spades'), ('eight', 'hearts'), ('queen', 'spades')))

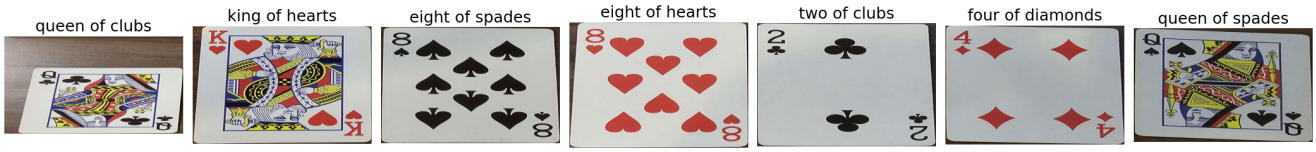



Figure 11: Extra example 2: Recognition results

Figure 12: Extra example 3

Hand Type: ('Full House', (('king', 'spades'), ('queen', 'clubs'), ('queen', 'spades'), ('queen', 'hearts'), ('king', 'hearts')))


Figure 13: Extra example 3: Recognition results