Department of Computer Science
Texas State University

YOLO Poker Hand Detection

Angel Sanchez *ijj11@txstate.edu*

Kayla Vincent *jqg8@txstate.edu*

Abstract

Identifying poker hands in a fast-paced environment is error-prone when up to individual interpretation. In order to quickly and accurately identify the hand, we created a Poker hand classifier. The identifier first detects on-screen cards, then assesses the hand. The final product takes video input from the webcam and detects playing cards. If it detects five cards, it classifies the cards as a Poker hand if it fulfills the requirements. If five cards are not detected, the identification of the card and confidence score still display on the screen, shown as bounding boxes around the corners of the cards.

1 Introduction

Throughout the course of this class, we have used static images and pre-recorded videos to learn about different methods of object detection. Many of these methods required multiple passes over the same image in order to ultimately detect the object. These methods are incredibly inefficient and make real-time object detection either extremely computationally expensive or almost impossible. To manage this problem, we wanted to research ways to detect objects in real time using Convolutional Neural Networks and Machine Learning principles.

A group of objects we felt would be easy to classify in real-time – both because they are easily identifiable and because there are many instances of them – was playing cards. Going further, we felt that it would be an interesting challenge to identify Poker hands in real time based on the detected cards. The main problem behind real-time detection is the lack of accuracy that comes with only doing a single sweep of the on-screen image. Because of this, we would want to set the threshold relatively high when checking for each card's label.

1.1 Goals

The main goal of this project was to quickly and accurately detect cards from real-time video input. Then, we wanted to identify Poker hands if five cards were detected that fit the requirements of a hand. One of our main goals with expanding our knowledge was learning how to use real-time detection with a ML model, as well as learn more about other types of CNNs.

2 Methodology

Our project implements a real-time poker hand classifier using a two-shot object detector. Initially, we used the YOLO11 model on the "Cards Image Dataset-Classification" Kaggle dataset, training it with 50 epochs. The Classification dataset used thousands of cropped images of cards with a corresponding class label. However, this dataset contained no background imagery, so the model was only able to detect cards if they took up the entire screen. To fix this problem, we switched to using a detection-oriented dataset. The dataset is specifically the "Playing Cards Object Detection Dataset" from Kaggle. This dataset contains images with multiple cards per image and corresponding YOLO-format bounding box annotations, which essentially enables the model to learn card detection in multi-card scenarios. After changing the dataset, the model was retrained for 100 epochs to achieve better accuracy. The retrained YOLO model was then integrated into a webcam-based detection system that captures live video frames, detects individual cards using the YOLO model with a confidence threshold of 0.5, and then extracts card information, rank, and suits from the detection results. When exactly five cards are detected, the detector applies a poker hand classification algorithm that analyzes card patterns through a single-pass computation, checking for flushes, straights, and rank to classify hands according to Poker rankings: Royal Flush, Straight Flush, Four of a Kind, Full House, Flush, Straight, Three of a kind, Two pairs, One Pair, and High card. The detector has real-time visual feedback by displaying bounding boxes around the detected cards, showing the number of the detected cards, and displaying the classified poker hand type when 5 cards are shown.

3 Experiments & Results

3.1 Model Training

There were two distinct phases for training the model in this project. We trained using a Classification dataset and then a Detection dataset. As a part of the YOLO model training code, it automatically generates evaluation metrics for each epoch of the model. It would not be fair to directly compare the results of these models, as they used completely different datasets and had different epochs. However, we can compare the results of using the Detection dataset with 50 epochs and then with 100 epochs. Figure 1 shows the evaluation metrics with 50 epochs, and Figure 2 shows the evaluation metrics using 100 epochs.

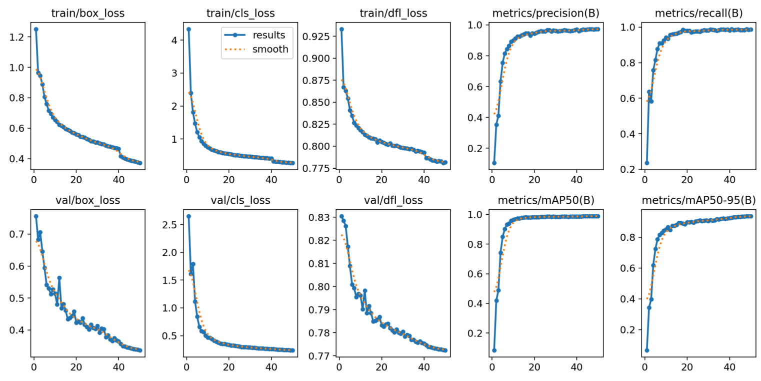

Figure 1: Detection Model Results - 50 Epochs

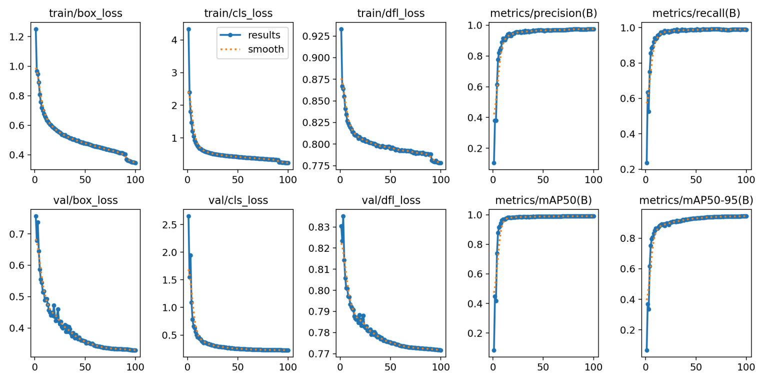

Figure 2: Detection Model Results - 100 Epochs

These figures give us important information about the value of increasing epochs when training models. Although the general shape of the results are the same in both figures, the results from Figure 2 are more smoothed out. This is especially noticeable in the val/dfl_loss section of the results, where the peaks get smoothed out and are shown to more consistently decrease closer to the 100th epoch. These results show that, by using 100 epochs, we were able to decrease the DFL loss for the group of val images over time. DFL, or Distribution Focal Loss, is used for refining bounding box precision on images. This is incredibly helpful in our case because the cards will not always be in the same spot or orientation, so some robustness to this benefits the model's ability to predict the bounding boxes.

3.2 Experimentation

Most of the experimentation was done using cards that we already knew classified as a Poker hand before testing. For example, we would regularly test for a Royal Flush because that is one of the most recognizable Poker hands, so it was easy to analyze results based on that. Figure 3 shows a positive case of the Poker hand detector identifying the hand.

Also, although not intentional with this Figure, we can see that the detector is not as robust to changes in angle or scale for certain cards. You can tell this from the fact that the K and 7 cards in the background are completely missed.

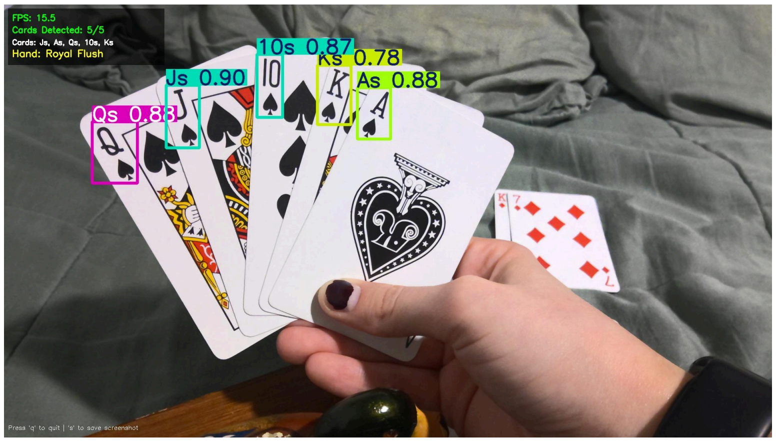

Figure 3: Royal Flush Detection

It was easier to experiment with the project this way because we already knew what result to expect. For the most part, the hand identification was very accurate, identifying hands in the order of highest ranking first. However, there were many cases where the detector either identified too many cards (counting the same card twice) or did not detect enough cards to classify a hand. Figure 4 shows an example of a detected duplicate leading to a misidentification of the Poker hand. These results indicate that the card is being detected based on their corners, and not the entire bounding surface they take up.

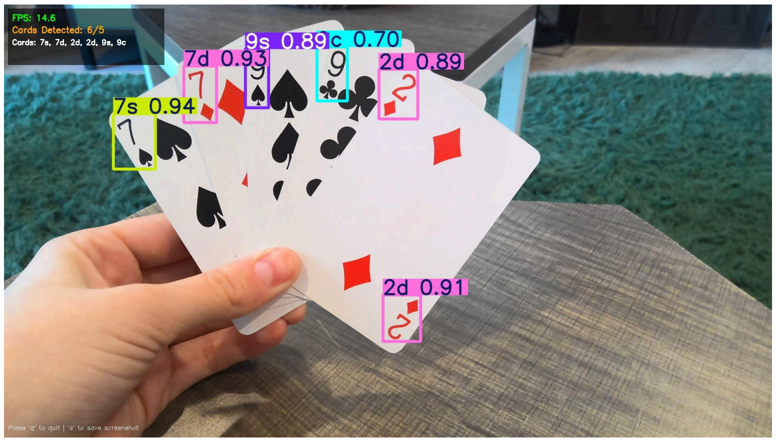
FPS: 14.6
Cards Detected: 6/5
Cards: 7s, 7d, 2d, 2d, 9s, 9c
7s 0.94
7d 0.93
9s 0.89
c 0.70
2d 0.89
2d 0.91
Press 'q' to quit | 's' to save screenshot

Figure 4: Duplicate Card Detected

Some of the more notable failure cases were a result of different thresholds used to identify the cards. Also, there were some interesting issues with the model identifying cards with the values 3 or 8. In individual cases, the model would very easily identify them with high confidence, but when grouped together in a hand, the confidence would be very low. Figures 5 and 6 show the results of different thresholds used on the same hand of cards. With our detector application, you can automatically increase or decrease the threshold while the webcam is open. This ensures that the lighting and other setting conditions of these two images were almost identical when taken. These results could indicate some issue with the model's ability to distinguish certain numbers when they are held together like this.

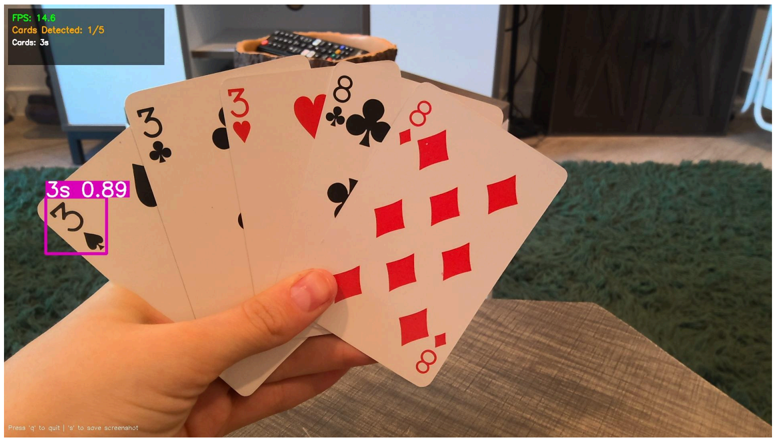

Figure 5: High Threshold

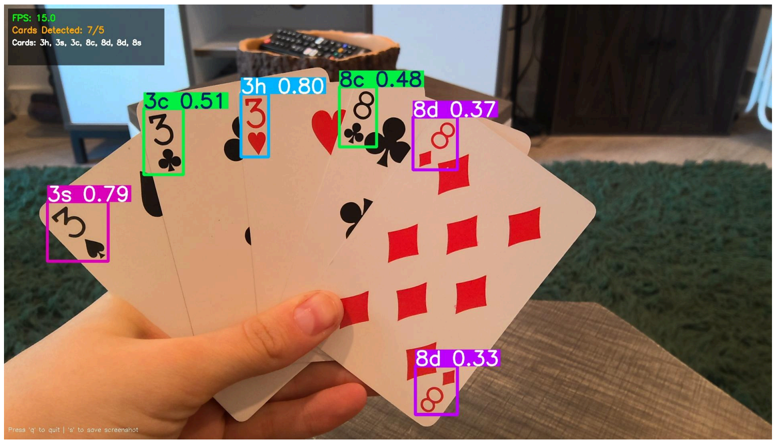
FPS: 15.0
Cards Detected: 7/5
Cards: 3h, 3s, 3c, 8c, 8d, 8d, 8s
3c 0.51
3h 0.80
8c 0.48
8d 0.37
3s 0.79
8d 0.33
Press 'q' to quit | 's' to save screenshot

Figure 6: Low Threshold

3.3 Analysis

With the YOLO11 model trained for 100 epochs, the real-time detector achieved about 15-16 FPS on CPU during live webcam testing. Due to lighting conditions, camera angles, and the usage of AI-generated images in the training dataset, single card recognition proved to be somewhat unreliable, despite the poker hand classification algorithms operating with high accuracy when all cards were recognized. The detector accurately classified hands across all poker levels and consistently identified all five cards when they were clearly visible. Optimizing detection for various cards was made easier by the interactive confidence threshold adjustment. All things considered, the detector proved useful for analyzing poker hands in real time.

4 Discussion & Challenges

This project effectively demonstrated a YOLO-based object detection approach for real-time poker hand analysis. Even though the poker hand classification logic was very accurate when every card was found, there were occasionally a number of issues with detecting the individual cards.

One of the difficulties we encountered was that, in the early stages of development, we trained the dataset on a CPU, which proved to be incredibly slow for experimentation. Training runs took

hours, making it challenging to effectively analyze results or iterate on different parameters. Later, we moved to an NVIDIA GPU, which resolved the performance problem and significantly shortened the training runs, enabling us to retrain the model and test changes considerably faster. However the transition was not seamless. We had to reinstall certain libraries, reorganize our development environment, and make sure the PyTorch and CUDA drivers were compatible. GPU training was essential for finishing the project.
Another challenge we faced was duplicate card readings. Because the YOLO model was trained to identify cards primarily from their corner markings rank and suits, it sometimes detected both the top left and bottom-right corner of the same cards as separate cards. This led to the same physical card being counted twice in the results which disrupted the poker hand logic.

5 Conclusion & Future Work

The results of our project show that, although YOLO detection works extremely well for real-time object detection, its accuracy for our use-case is not entirely reliable. The shortcomings we experienced, such as missed detections or misidentifications, could very likely stem from issues with the dataset we used. This indicates the importance of having a good dataset when training a model. From this project, we learned more about the process of training a model and using that model in a computer vision application.

A few ideas for future implementations include integrating a Poker hand classifier, including bounding boxes around entire cards, and potentially training the dataset using more epochs. The purpose of the Poker hand classifier would be to identify hands even more quickly, because currently the project has some difficulties in detecting hands. However, that issue might stem more from shortcomings in the model training. To deal with this, the easiest thing would be to train the model using more epochs, and seeing if that has any substantial difference in the results. Another way to manage the shortcomings could be to add bounding boxes around individual cards to decrease the likelihood that the same card gets counted twice.

References

[1] Andy8744. (2022) Playing Card Object Detection Dataset. Kaggle. Retrieved from https://www.kaggle.com/datasets/andy8744/playing-cards-object-detection-.

[2] Gerry. Cards Image Dataset - Classification. Retrieved from https://www.kaggle.com/datasets/gpiosenka/cards-image-datasetclassification/code