Blackjack Card Counting

Andrew Chung
Stanford University
450 Jane Stanford Way, Stanford, CA 94305
awchung@stanford.edu

Pranav Sai Ravella
Stanford University
450 Jane Stanford Way, Stanford, CA 94305
pravella@stanford.edu

Abstract

What's the only way to beat a casino? Card counting. But it's extremely hard, as it requires mental tracking over multiple decks of cards and only guarantees around 1% edge against a casino. As a result, many avid casino players simply rely on pure chance, since the mental strain of this technique is not worth the potential payoff. However, we aim to detail an extremely precise method of counting cards with a video feed at a casino. This would take away from the casino players' need to remember the cards and only need to know the live counts. In this paper, we build the first step to tracking cards over a long period of time with card detection and recognition in an image. We achieve significantly better results compared to traditional methods that wrap both card detection and recognition in the same model, Yolov8 (5) and Grounding Dino. These approaches have been the commonly accepted standard for this type of problem, but separating the two components, along with a synthetic dataset and severe augmentation techniques, have yielded much better results.

1. Introduction

The objective of this paper is to accurately detect an identify playing cards within an image or video frame. This system aims to achieve near-perfect accuracy in determining the card value and suite (Ace, 2, 3, ..,) and (Hearts, Diamonds, Clubs, Spades) respectively. This system would then be used to recognize cards over a longer video setting. In order to ensure the robustness and reliability of the system in actual game play scenarios, this paper aims to address the various challenges of current methods of playing card detection in a scene.

1. Diverse card designs: Playing cards come in various designs, with differences in the size, position, and style of the value and suit symbols. The system must be able to handle this variability and accurately identify cards regardless of the specific design. The system designed in this paper utilizes multiple augmentation and scaling techniques to help the model not over-fit on the cards it was trained on, but rather apply to a generalized set of playing cards.
2. Cluttered environments: In a setting at a casino or poker table, there are many other objects that could interfere with detection, and even other cards that might cover up the cards in totality. The model presented in this paper can distinguish between backgrounds, playing cards, and other objects during card detection and recognition. The model also can detect and identify multiple cards within a single image or video frame, which is essential to keep track of all visible cards for card counting.
3. Motion blur: In video footage of card games, motion blur can occur due to the rapid movement of cards during shuffling or dealing. To mitigate the impact of motion blur, we incorporate motion blurring techniques as a preprocessing step, such as Wiener deconvolution.

This system utilizes two separate models. A card detection model that fine-tuned YoloV8 (5), and a card identification/recognition model that was built on top of Resnet50 (4) . The dataset utilized in this system accounted for various designs, orientations, lighting, perspective, partial occlusions, and noise. To measure the accuracy of our system, we compared our combined models to a fully-fine tuned YoloV8 model that was trained on all classes of cards and observed the situations where the card detction model was the bottleneck of identifying all cards in an image. We observed nearly a 15% higher mAP@50-95 using this system compared to the fine-tuned YoloV8 model. We also observed the qualitative predictions from the model, and its drawbacks with detection, especially when the cards are barley visible in the frame and small.

The proposed system for card detection and identification has the potential to be applied in various domains, including casino monitoring, game analysis, and even assisting visually impaired individuals in playing card games. By accurately detecting and counting cards in real-time, this system can provide valuable insights for both monitoring purposes and players' edge on casinos.

2. Related Works

A related paper was published from University of Canterbury, Christchurch, New Zealand by Chunhui (Brenda) Zheng and Dr Richard Green called ”Playing Card Recognition Using Rotational Invariant Template Matching.” (7) This paper took a simpler yet more computationally efficient method for card recognition. Rather than training a YOLOv8 model and a convolutional neural network to learn the underlying mechanisms behind card recognition, they took a more ”hard code” approach by creating templates for every suit and value. It works by scanning the image for matches to the template values and suits and finding the ones that match the best which is a lot more computationally efficient but can't handle variation as well as a neural network can and thus cannot generalize as well. They are similar in that they both handle rotational invariant though in different ways. Our model simply trains the model on significant data augmented sets that it learns to recognize rotation and properly categorize. For the related paper, they rotated the card pixels to match the rectangular shape/orientation that the template images were based on before doing the similarity matching.

Another related paper was published called ”Robust poker image recognition scheme in playing card machine using Hotelling transform, DCT and run-length techniques” (1) by Wen-Yuan Chen and Chin-Ho Chung. A significant difference was again how they handled rotational invariance. For the paper, they used Hotelling Transform to standardize the orientation while our model again just trained on a robust set of data. In essence, the model architecture of the related paper was more manual and thus required more precise preprocessing but was thus more computationally efficient. They used more traditional image processing methods like DWT and DCT and run-length encoding but these are a lot more sensitive and don't generalize and adapt as well as the neural network approach we took. I think we made a good choice using a neural network simply because of how large our synthetic dataset was so that we could learn very nuanced features and could rely on the advantage of not being limited by the dataset size to assume that the model would eventually learn all the important decisions made with more precise preprocessing.

An additional related paper that was published is called ”Poker card recognition with computer vision methods” (6) by Xuewen Hu, Tao Yu, Kai Wan, and Jie Yuan from Nanjing University. For this one, the primary difference was scalability. The related paper's model relied on Hu invariant moments, which are invariant to rotation and scale. These computations were compared to a known database of values corresponding to suits and values similar to template matching. Thus, this methodology was really good for handling a wide range of data augmented sets with low training volume but didn't scale in performance with increasing data size. On the other hand, the model we created relied heavily on the volume of training data and thus could capture more nuances as we trained it more extensively. The related paper's model was capped and constrained and thus might not be fit for many real life scenarios.

3. Dataset and Pre-processing

3.1. Card Identification Dataset

Finding a large card detection dataset with labeled boxes and suite classification proved to be an extremely hard ordeal. In order to combat this we created a synthetic dataset that superimposed real card images into fake ”scenes”. This approach allowed us to have complete control over the dataset's size, diversity in class sizes, annotation quality, and augmentation techniques. Using Geaxgx's notebook (3) as an initial step to develop this method, we were able to generate the dataset split as shown in Table 2. To generate the synthetic dataset, we followed these processes.

Dataset	Percentage	Number of Images
Train Set	88%	21,210
Validation Set	7%	1,697
Test Set	5%	1,211

Table 1. Distribution of images across the training, validation, and test sets for card detection.

1. We took a picture of each card in the 52 card deck in three different angles, to account for lighting, perspective, and card type.
2. We then took a mask to single out the card and remove any other features, such as background and other objects.
3. After each masked picture was created, we manually annotated the bounding box that surrounded the the suite and card value.
4. After combining the bounding box and attaching it to the card bounding box using openCV, we then took the cards at random and assembled them into random scenes with random jitters, rotations, perspectives, noise, and assembly. We also created ”fan-like” card draws to account for how these types of cards would be in a casino, and generated up to five cards per image at random. Since every card was at a uniform draw to be chosen for the image, we ensured that each card had equal representation within the dataset. The random scenes were from the Describable Textures Dataset (2).

Some examples of the cards and their bounding boxes are shown in Figure 1.

3.2. Card Recognition Dataset

After generating these synthetic scenes, within each of the training examples, the bounding box for each card in

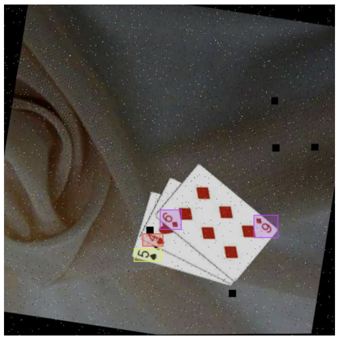
Figure 1. Example image from synthetic dataset.

the scene was used to crop out the suite and value of the card. For a given scene, multiple of the same suite and value image would have been cropped out due to superimposing some cards over the other corner of the card. Using this, we created another dataset that had the image of the suite and value of the card that was cropped, scaled to the center of 100x100 box and padded until it fit the box with blue color. As a result, each suite and value image was associated with the class that it belonged to and could fit into a small feature set that a model could process. Through this process, we generated 52,000 images of which 15% was validation and 15% was testing. An of this dataset is shown in Figure 2.

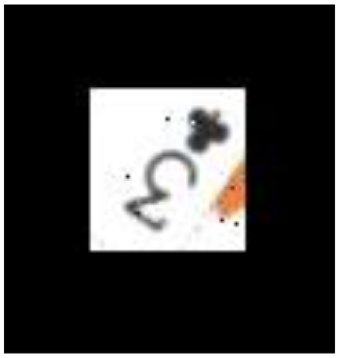
Figure 2. Example image from synthetic dataset.

4. Methods

4.1. Card Detection

We also utilized several augmentation techniques that were applied to synthethic images to further improve the dataset.

- **Geometric Transformations:** Random rotations, translations, scaling, and shearing were applied to simulate different viewing angles and distances.
- **Color Jittering:** Adjustments to brightness, contrast, saturation, and hue were made to account for various lighting conditions.
- **Random Cropping:** Parts of the images were randomly cropped to mimic different framing conditions.
- **Noise Addition:** Gaussian noise was added to simulate various image qualities.
- **MixUp and CutMix:** Techniques combining multiple images to create new training samples were used to help the model learn more robust features.

After this data augmentation, we fine-tuned YoloV8 with respect to a single class of being that called a ”card”. YoloV8 follows a single-stage detection architecture, which predicts bounding boxes and class probabilities directly from full images in one evaluation. This is different to two-stage detectors, like Faster R-CNN. Faster R-CNN first generates regional proposals and then classifies them. The backbone of YolovV8 is a convolutional neural network (CNN). The backbone of the network is then processed through a Feature Pyramid Network to enhance the feature maps at various different scales. The YoloV8 head predicts the bounding boxes, objectness scores, and class probabilities and uses anchor boxes to predict bounding boxes at multiple scales.

YOLOv8 predicts bounding boxes using anchor boxes. Each bounding box is parameterized by four values: the center coordinates (b_x, b_y), width b_w, and height b_h. These are predicted relative to the anchor box dimensions and the grid cell location.

$$b_x = \sigma(t_x) + c_x \tag{1}$$
$$b_y = \sigma(t_y) + c_y \tag{2}$$
$$b_w = p_w e^{t_w} \tag{3}$$
$$b_h = p_h e^{t_h} \tag{4}$$

where:

- σ is the sigmoid function.
- t_x, t_y, t_w, t_h are the predicted offsets.
- c_x, c_y are the coordinates of the top-left corner of the grid cell.
- p_w, p_h are the dimensions of the anchor box.

The objectness score P_o represents the confidence that an object is present in the predicted bounding box with a sigmoid function over the predicted objectness score. The class probabilities P_c for each class are also predicted using the sigmoid function over the predicted class score.

$$L_{\text{total}} = \lambda_{\text{loc}} L_{\text{loc}} + \lambda_{\text{conf}} L_{\text{conf}} + \lambda_{\text{cls}} L_{\text{cls}}$$

The localization loss measures the error between the predicted bounding box and the ground truth bounding box using mean squared error (MSE). The confidence loss measures the error in the objectness score prediction. The classification loss measures the class probability error. These both use binary cross-entropy loss.

$$L_{\text{loc}} = \sum_{i=1}^{N} \left((b_x - \hat{b}_x)^2 + (b_y - \hat{b}_y)^2 + (b_w - \hat{b}_w)^2 + (b_h - \hat{b}_h)^2 \right)$$

$$L_{\text{conf}} = \sum_{i=1}^{N} \left(P_o \log(\hat{P}_o) + (1 - P_o) \log(1 - \hat{P}_o) \right)$$

$$L_{\text{cls}} = \sum_{i=1}^{N} \sum_{c=1}^{C} \left(P_c \log(\hat{P}_c) + (1 - P_c) \log(1 - \hat{P}_c) \right)$$

Through the curated dataset aforementioned, we fine-tuned the YoloV8 model and predicted the accuracy of the model using mAP scores. To calculate mAP we first measure the Intersection over Union (IoU):

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Average Precision (AP) is calculated for each class by measure the area under the precision-recall curve, which is obtained by plotting precision against recall at different confidence thresholds. Precision is the ratio of true positive detections to the total number of positive detections (both true positives and false positives) and recall is the ratio of true positive detections to the total number of ground truth objects.

1. Sort the predicted bounding boxes by their confidence scores in descending order.
2. Calculate precision and recall at each detection threshold.
3. Plot the precision-recall curve.
4. Compute the area under the precision-recall curve

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^{C} \text{AP}_c$$

In our proposed system, we compare YoloV8 trained on all 52 classes to identify and recognize all cards in the picture with one single prediction. However, we quickly realized, that the the recognition portion of the detector often confused suites together and numbers. As a result, we separated the two to have a card detection and card recognition model. For every prediction the card detection model made, we would scale and pad the bounding boxes that included the cards' suite and value, and pass this into the card recognition model that was trained on size 100x100 images to identify the card.

4.2. Card Recognition

As previously said in the Data Processing Section, the dataset entering the card detection stage were 100x100 images of the corner pieces of cards (including number and suit) and were separated into training (75%), validation (15%), and testing (15%) sets. As highlighted in the flow model architecture shown in Figure 3, the execution included three primary steps: ResNet, convolution layers, and linear layer. For the ResNet section, we decided to opt for ResNet 50 for one primary reason: given that the accuracy heavily relied on distinguishing between similar values like 6/9 and face cards and also would need to distinguish similar looking suits and colors, the model would require deeper networks to accurately capture these details. In order to tailor the model to our specific task, we removed the final pooling and linear layer from the ResNet to add our own trainable convolutions, while freezing previous layers. Entering the convolution layer, our spatial dimensions were 4x4 which was initially a concern, but given that the images were already zoomed into the area of importance and majority of the critical features took a large portion of the image, we projected that the results would have been indifferent to this change. In Figure 4, you can see the visualized intermediate feature maps that are outputted by the ResNet, which though don't give very understandable information in sheer form, clearly highlight that 2048 of these feature maps would be able to capture an enlarged number and suit that each take a near-quadrant of the image. After the ResNet, we have two convolution layers each followed by a batch normalization and ReLU. Each of these convolutions have a kernel size of 3, stride of 1, and padding of 1 in order to not reduce the spatial dimensions any further and remain a 4x4 throughout. The reason for a larger kernel size is again to capture larger features given that finer details do not matter as much. The adaptive average pooling layer was simply in order to reduce the number of learned weights in the fully connected layer and also under the assumption that the individual 4x4 squares at this point would not portray any significant unique information. The final layer was a fully connected layer to simply bring it back to the 52 card classes.

In terms of the hyperparameters, the main reason behind choosing Adam was that the model had so many parameters to learn that having a separate learning rate per parameter would become very useful in learning a broad range of features. The learning rate of 0.001 was empirically very successful and was chosen simply because it was a standard value that wouldn't cause any drastic gradients and also doesn't affect Adam as much. The decaying learning rate was in order to stabilize learning and converge to a solid model consistently given that we had a lot of data per epoch and thus wouldn't expect that many large changes towards the later epochs.

Initially, we were also considering using data augmentation again at this stage before the card recognition portion, in addition to the initial round during card detection for further randomization. Though data augmentation would improve generalization and better hierarchical understanding of the features, we realized that there was no need to for this given how large our training data set was. We did not need to artificially increase the size of our dataset and already got enough variation from the training dataset itself and the first round of augmentation, so data augmentation would add unnecessary complexity. We also tried this and found that it was empirically worse.

We were also considering alternatively using a fully connected network following the ResNet rather than the CNN for simplicity sake under the assumption that the ResNet itself would be enough to capture the spatial features. But, as supported from our empirical results, we found out that it greatly worsened the performance. This was for two reasons: flattening the data lost the spatial relationships previously gained and thus we would need to postpone it as late as possible, and the number of learned parameters exponentially increased especially since the ResNet output was so large when flattened that it was difficult to scale the network or learn properly.

Another alternative approach we considered prior was just simply using a simplified convolutional neural network to run directly on the input pixels. What we found was that this lead to really good results on training data but failed to perform at all on testing data sets. This was simply because it was very difficult to effectively train deeper convolutional networks to not overfit given the number of parameters and also the lack of layer-skipping. ResNet was an important switch because it allowed us to incorporate a much deeper network that could effectively learn the hierarchical features, without necessarily overfitting because of its skip connections.

5. Results

5.1. Card Detection

On the card detection task for single card image we achieved an mAP@50-95 of 82.92% over seven epochs and five hours. Since mAP contextualizes both the localization of the bounding box and classification accuracy, mAP scores with a threshold between 50 and 95 were much lower for the YoloV8 model trained on all classes However, when qualitatively looking at the predictions made by this YoloV8 model, we realized that the model was good at identifying the corner suite and values, but had an extremely hard time identifying the card itself. Eventually for the YoloV8 model trained on all classes, the loss and accuracy plateaued as it could not distinguish between suit and number values any further. This is evident in Figure 5. Switching to the YoloV8

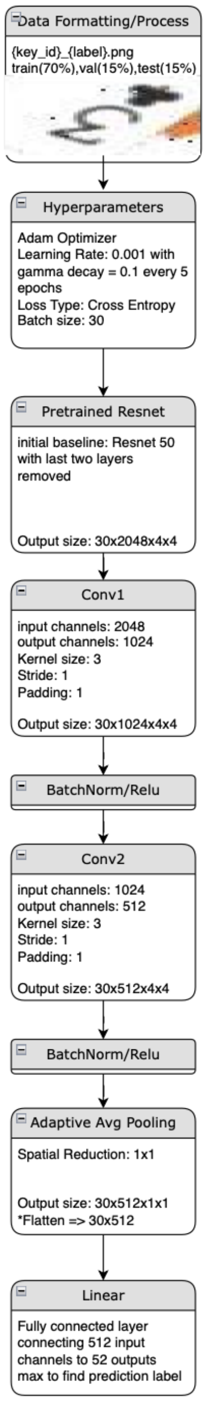

Figure 3. Card Recognition Model Architecture

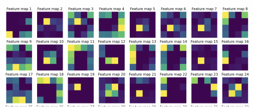

Figure 4. Feature Maps of ResNet Outputs

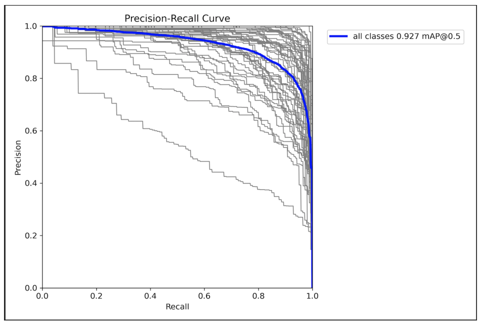

Figure 5. Precision v. Recall Curve for YoloV8 on 52 Card Classes

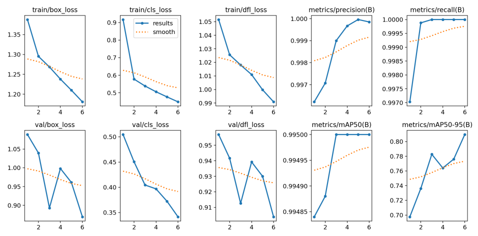

Figure 6. Training Curves for YoloV8 on Single "Card" Class

model trained only on detecting the card itself resulted in not as much over-fitting on the training dataset, as evident in the test mAP, compared to the YoloV8 model trained on all classes Figure 6. The training curve for the single-class detection is still very jumpy, but even for training with longer periods before GPU time-out the model was still actively learning and had a much smoother loss curve compared to the 52-class detector, which intuitively makes sense.

The model trained only detecting cards in the scene did fail in some scenarios. When the cards were extremely small or blurred in the background, as in Figure 7, the model failed to detect the extremely small bounding box surrounding the suite and value. A better approach to this system would be to segment the cards out individually with a model and then have the card recognition model use the entirety of the segmented image. An image segmentation model would be able to segment out the entire white-space of the card and suite values, but recognizing very small objects with computer vision models is still a prevalent issue.

5.2. Card Recognition

Given the enlarged 100x100 images from card detection, we trained 10 epochs with a batch size of 30 and measured the training loss and validation accuracy after each epoch. We ultimately reached a training loss of 0.01752118092591223 and a validation accuracy of 98.84969325153374% after the last epoch and the exact trajectory is shown in Figure 8 and Figure 9. The final test accuracy was 99.02862985685071%.

As seen in Figure 8, the training loss graph was a standard logarithmic curve. There were many aspects of the

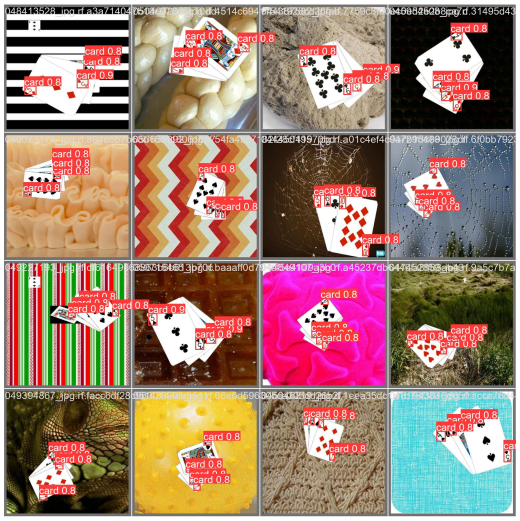
Figure 7. Test Images Prediction on 1-class Card Detector

Results	Value
Training Loss	.0175211809259122
Validation Accuracy	98.84969325153374%
Test Accuracy	99.02862985685071%

Table 2. Summary of Card Recognition Results

curve that led to higher confidence in the model's intended learning. First, the mode significantly decreased in training loss between the first two epochs and then gradually reduced the loss in smaller magnitudes after that which shows steady convergence and a good adaptive learning rate. The loss ultimately stabilized and did not continue to keep decreasing after reaching a certain state which was also good because it led to belief that the model was not necessarily overfitting.

As seen in Figure 9, the validation accuracy showed a steady step-wise increase over epochs. The rapid increase in validation in the beginning shows that training is learning the hierarchical features well as it is generalizing very well. The lack of any significant decline also shows that the model is not likely overfitting. The graph also stabilizes towards the last few epochs which is promising as it seems like the model learned the true underlying mechanisms and plateaued at optimal performance without overfitting thus would likely do well in generalization.

To identify common failure modes, I identified the most common cases of mislabeling in the test dataset in order to see in what way it was mislabeled and also what in the input image caused this mislabeling. The most common reasoning was because of partial occlusion. In order to ensure that our model could handle real-life scenarios where part of the image could be obscured, we included images with parts of the image covered. We found that a majority of the issues arose from cases where the blocked out por-

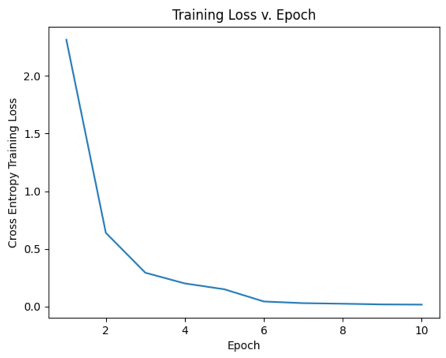

Figure 8. Training Loss Graph

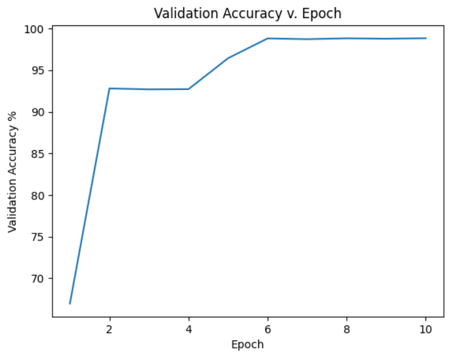

Figure 9. Validation Accuracy Graph

tion was critical in identifying the value. The values had similar curves and differed most significantly in the blacked out portion. The second most common reason was coloring. As part of the data augmentation process, the pixel values sometimes became gray scaled. The primary issue with this is that spades and clubs are black and diamonds and hearts are red which would be a significant identifying factor. What we found was that diamonds were often identified as spades when turned to grayscale, but we deemed this not that important of a problem given that real-life scenarios wouldn't really have this problem and would be likely full color. Actually, we found that we never ran into issues where hearts/diamonds were identified as spades/clubs for full color images, which highlights how important the color values were for suit identification. Given that these were majority of our 1% error rate, we were pretty confident our model performed well on the tasks as the misclassified images were intuitively difficult and likely not a problem in real-life circumstances.

5.3. Combined System

The YoloV8 model trained on all 52 card classes was trained for five epochs over 2 hours, but after 3 we cut training off after noticing the model's learning tended to plateau. Using the test images, we compared the the mAP@50-95 and mAP@50 scores of this YoloV8 model compared to the combined system in Table 3. As noted, our combined system resulted in much less over-fitting and more confident predictions in the cards detected and recognized. One of

Model	Metric	Score
YOLOv8 (52 classes)	Validation mAP@50	0.927
	Validation mAP@50-95	0.637
	Test mAP@50	0.898
	Test mAP@50-95	0.558
Our System	Validation mAP@50	0.991
	Validation mAP@50-95	0.801
	Test mAP@50	0.989
	Test mAP@50-95	0.785

Table 3. Comparison of mAP scores for YOLOv8 model and combined YOLOv8 Detector + Card Recognition system.

the key advantages of the combined system is its ability to mitigate overfitting. The YOLOv8 model, when trained in isolation, showed signs of overfitting, as evidenced by the divergence between training and validation mAP scores. In contrast, the combined system, which separates the detection and recognition tasks, demonstrated more stable learning curves and higher confidence in its predictions. This is likely due to the specialized training of each component, allowing for better generalization to unseen data.

However, in both situations, the systems are bottle necked by the ability to detect the cards in the image. Despite the improvements in recognition accuracy, both systems are fundamentally limited by their ability to detect cards in the image. The detection stage is crucial, as any missed or inaccurately localized cards directly impact the subsequent recognition stage. Enhancing the detection accuracy is therefore essential for further improving the overall performance of the system.

The combined system performed extremely well when presented with normal images and some augmentation. However, when cards were rotated or partially occluded, as shown in Figure 10, the model failed to either detect or recognize the card properly. We chose an arbitrary 100x100 input size for the card recognition model, however, sizing this down to 50x50 would have resulted in less memory being used, higher batch size, and better ability to recognize these augmentations. Furthermore, 98% of the images were 75% of the area of 100x100 feature, and though the model learned to ignored the padding, a smaller input feature could have allowed deeper training.

6. Conclusion

We developed and evaluated a combined system of card detection and a custom card recognition model to classify and detect cards in images with sever augmentations.

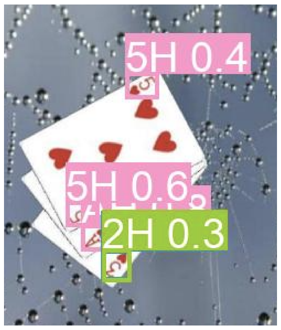

Figure 10. Error in prediction using combined system

The approach of separating out the two models in a two-stage system demonstrated significant improvement in performance compared to a standalone YoloV8 model trained to perform both detection and recognition tasks simultaneously. The YOLOv8 model trained on all 52 card classes showed signs of over-fitting and struggled to distinguish between similar card values and suits. By separating the detection and recognition tasks, our combined system achieved more stable learning curves and higher confidence in its predictions.

However, both systems were fundamentally limited by their ability to detect cards in the image. The detection stage is crucial, as any missed or inaccurately localized cards directly impact the subsequent recognition stage. Enhancing the detection accuracy is therefore essential for further improving the overall performance of the system. Future work will focus on improving the detection capabilities of the YOLOv8 model, exploring advanced data augmentation techniques, and integrating object tracking algorithms to maintain a consistent count of detected cards over time. Furthermore, with more allocated time, a custom image segmentation model to mask out the cards would have been another tested approach to try, in order to give the high-performing card recognition model more of a priority rather than being ”bounded” by small bounding boxes in the image.

Additionally, we observed that the combined system performed well under normal conditions but faced challenges when cards were rotated or partially occluded. Furthermore, our current method of the card recognition model was trained separately from the card detection model, but having a custom card detector that is fine-tuned synchronously to the recognition model with a fused layer between the two, could result in better predictions than a custom scaling and padding when passing the predictions of the detector into the recognition model.

References

[1] W.-Y. Chen and C.-H. Chung. Robust poker image recognition scheme in playing card machine using hotelling transform, dct and run-length techniques. `https://www.sciencedirect.com/science/article/pii/S105120040900178X`, 2010.

[2] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[3] geaxgx. playing-card-detection. `https://github.com/geaxgx/playing-card-detection?tab=readme-ov-file#readme`, 2019.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[5] Ultralytics. YOLOv8: A state-of-the-art real-time object detection system. `https://docs.ultralytics.com`, 2021.

[6] K. W. Xuewen Hu, Tao Yu and J. Yuan. Poker card recognition with computer vision methods. `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9563607`, 2021.

[7] C. B. Zheng and D. R. Green. Playing card recognition using rotational invariant template matching. `https://digital.liby.waikato.ac.nz/conferences/ivcnz07/papers/ivcnz07-paper51.pdf`, 2007.