Playing Set Online with YOLOv8-Based Image Detection Fine-Tuned on Synthetic Data

Owen Bluman
Department of Computer Science
Haverford College
Haverford, PA 19041
`obluman@haverford.edu`

1 Introduction

The game of Set has been a standard card game for many years, but has recently seen popularity in various online formats. The set up consists of a board of 12 cards, laid out in a 3x4 grid. Each card contains an image which can vary across four dimensions: color (red, green, or purple), quantity (one, two, or three), shape (diamond, oval, or tilde), and filling (striped, solid, or empty). Groups of three cards form a "set" if, across all four dimensions, either all the cards in a grouping have the same trait or all the cards have different traits. The object of each round of the game is to correctly identify and select a set before your opponents do. Found sets are replaced with cards until a deck of set cards is exhausted, at which point the player with the most found sets is the winner. The online version of the game found at `setwithfriends.com` offers an interface that allows accessible testing against other players, and was thus selected as the environment utilized for experimentation.

The primary machine learning technique we used in order to engineer our agent was a CNN-based image detection model, in order to quickly and accurately determine the state of the cards present on the board. This model was created by fine-tuning the pretrained YOLOv8 model using synthetically generated random Set boards to specifically detect Set cards. The model was then coupled with an automation script in Python in order to play the game online independently from our human interaction. We wish to explore how an autonomous agent would perform against a human player as well as against an alternative automated approach that uses template matching for set detection. We hypothesize that our model will be faster than both a human and the alternative approach at set detection, and will be able to defeat a human in a full Set game.

2 Related Work

The most direct precursor in the research area of game playing agents based on a machine learning framework, Mnih et al.[1], utilized Q-learning to create agents that played a variety of Atari video games. Their system was able to defeat a human player on several different games such as Breakout, Enduro and Pong. Our approach differs in that we use a CNN-based system to determine the game state, which is then passed to a brute force algorithm as a series of encoded attributes, in contrast to the Deepmind group's approach of using the raw pixel values directly as input to a Q-learning function.

Convolutional Neural Network's have been researched extensively for use in image detection such as by Chauhan et al.[2] and Hussain et al.[3] Relevantly, Chen and Yi [4] were able to build a CNN-based model for the N64 game Super Smash Bros that was competitive with prebuilt expert level AI. We have expanded the use of CNN's for the novel task of set card detection, with the ability to output card attributes and positions in order to determine actions.

The YOLO model specifically has been used as a basis for image detection in cases of limited available datasets such as the work done by Li et al.[5] Others such as Dewi et al. [6] have generated

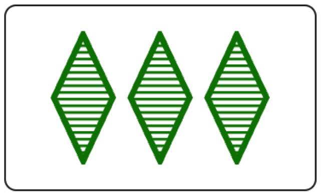

Figure 1: Example of a set card from the `setwithfriends.com` site, labeled with its manually annotated encoded attributes in the iconPics dataset as green_three_diamond_stripe.png

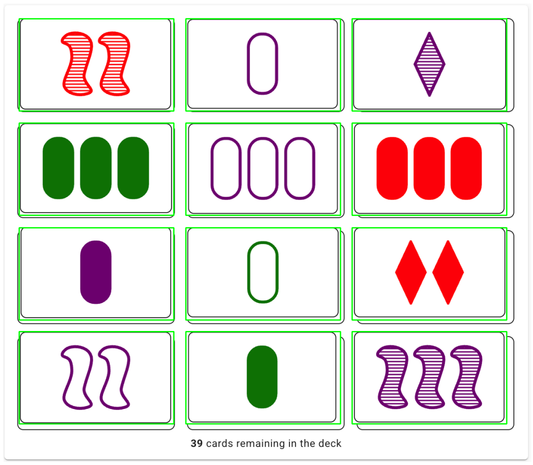

Figure 2: Green bounding box visualizations of the positional data from synthetic_set12_3x4_00000.txt overlaid upon the generated set board synthetic_set12_3x4_00000.png, show the programs ability to generate accurate data for use in the training process

synthetic datasets for use in training a YOLO model, however their method for data generation relied on a GAN framework while our images were constructed using repeated random sampling and arrangement of the initially collected card dataset.

The game of Set has been explored mathematically in sources such as McMahon et al. [7] with a focus on the combinatorial aspects of set discovery. While the problem of finding sets is NP-complete, we believe that we can exceed a human benchmark through efficient and accurate image detection and an adaptable automated script.

3 Dataset

To collect our initial dataset (iconPics), we manually captured each of the 81 possible set cards from the `setwithfriends.com` site, and then annotated each card images filename in the format "color_quantity_shape_filling.png" (Figure 1). This gave us a labeled dataset of each Set card and its corresponding attributes. In order to fine-tune the YOLO model, we created a second dataset (synthetic_dataset) of 500 synthetic full Set boards alongside an accompanying label file listing the cards found in the image and their normalized horizontal and vertical positions. This dataset was programmatically compiled (generate_synthetic_data.py) by first taking an image of a Set board to use as a background and then calculating the fixed pixel positions where the 12 Set cards should go. Next, for each of the 12 slots, a Set card was randomly sampled from the original dataset of possible cards, and then resized if necessary and placed in its designated position. Concurrently, the cards label (alphabetical place in the dataset), and positional attributes (x and y center norms and width and height norms) were stored in an associated text file. The position coordinates can be verified with a bounding box visualization (Figure 2).

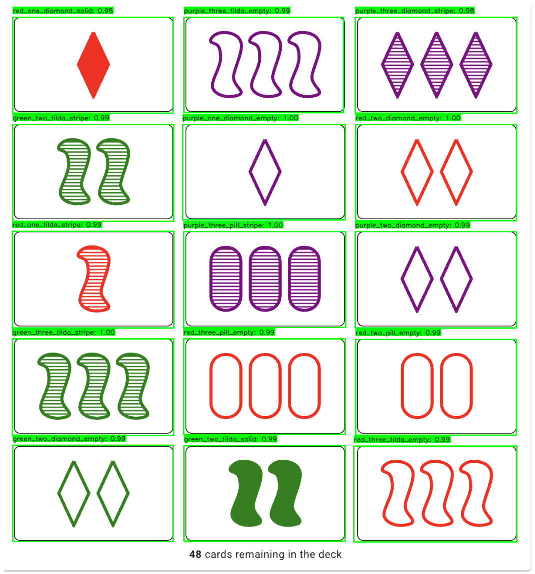
red_one_diamond_solid: 0.98
purple_three_tilda_empty: 0.99
purple_three_diamond_stripe: 0.98
green_two_tilda_stripe: 0.99
purple_one_diamond_empty: 1.00
red_two_diamond_empty: 1.00
red_one_tilda_stripe: 0.99
purple_three_pill_stripe: 1.00
purple_two_diamond_empty: 0.99
green_three_tilda_stripe: 1.00
red_three_pill_empty: 0.99
red_two_pill_empty: 0.99
green_two_diamond_empty: 0.99
green_two_tilda_solid: 0.99
red_three_tilda_empty: 0.99
48 cards remaining in the deck

Figure 3: After being passed a screenshot of a 3x5 Set borad, the YOLO model fine-tuned on synthetic data output the following predicted classes and locations of set cards with high confidence

4 Agent Creation

The dataset of synthetic boards/label files as well as the name and quantity of predicted cards were configured into a .yaml file in order to allow for the training of the YOLOv8 model, specifically "yolov8n.pt" the nano sized pre-trained version. The training took place in a Colab notebook environment, making use of the T4 GPU runtime type in order to speed up training time. The model was trained for 100 epochs with a batch size of 16, taking in board images, and outputting the label and location attributes of detected cards.

Ultimately,the optimal weights found during training were saved in the best.pt file. While the model was solely trained on 3x4 board grids, as is usually the board setup for the game of Set, the `setwithfriends.com` implementation occasionally provides a 3x5 board set up in the case that the initial set up of four rows does not contain a set. Thus we tested the trained model on an image of a board containing this "bonus row" and found that the model had adequately generalized its image detection to account for cards in previously unseen locations, as it detected all 5 rows of cards with each confidence percentage at or above 98% (Figure 3).

The fine-tuned model was then inserted into an automated pipeline which would repeatedly take a screenshot of the region of the screen containing the board, pass it to the image detection model to output the found cards and their locations. Then these card labels were converted to the direct card attributes so that a brute force search algorithm could quickly determine which collection of three cards would form a set. Then using the positional data of the cards and the pyautogui automation library, the program would click the positions of the cards forming a Set. This positional data had to be altered to account for the location of the board within the overall screen as well as the way that Mac displays are rendered with double pixel scaling. This pipeline was repeatedly looped to find all Sets until the program is stopped.

5 Methodology

To determine the performance of our Set card detection model and overall Set agent, we performed a speed and accuracy comparison against a human player, as well as an alternative agentic approach that used the locate function from the pyautogui library, which performs template matching, as the basis for its set detection. This alternative approach, created previously by the author, uses an almost identical pipeline for attribute decoding and determining which cards form a set, allowing the bulk of the comparison to be focused on each models method for detecting the Set cards from the online interface.

For each entity, we facilitated 10 trials on an individual game of Set via the `setwithfriends.com` site, and recorded the total time to complete the full game, as well as the average set time and overall accuracy. The automated scripts were encoded to directly output the timing results for their performance while the human trials used the games built in timer. All set misses that factored into the final accuracy calculation (where a user clicks on three cards that are not a set), were tracked manually as the site displays a text box indicated an unsuccessful attempt. Next we performed a head to head matchup consisting of 5 games of our Set agent against the human player, and 5 games of the alternative approach against the human, recording the score and winner of each game. These two experiment types allow us to determine our model's ability to detect Sets independently, as well against a human player in a real game scenario.

6 Results

In the individual trials, our novel model performed noticably better than both the template matching based alternative approach, and the human player, as the novel model had an overall average set time over three times as fast as the alternative model, and over four times as fast as the novel model (Table 1). Relatedly, the average game time was over a minute shorter for our model then either of the other entities. In terms of overall set accuracy, our model had the lowest, but still was over 97% accurate in terms of its set predictions. The alternative model had the highest accuracy, never incorrectly guessing a set in all trials.

Table 1: Results from the first experiment comparing the speed and accuracy of each entity in an individual Set game with average time results listed in seconds and the best results are listed in bold

Entity	Average Game Time	Overall Average Set Time	Overall Set Accuracy
Novel Model	**23.71**	**0.929**	97.05%
Alternative Model	91.54	3.694	**100%**
Human Player	112.32	4.27	98.56%

In both head to head matchups, the human player was unable to defeat either approach, with the closest they were able to come being Game 4 vs the alternative approach, where they lost by a score of 3. Notably, the human player only scored a single point (guessed a set faster than the opponent) on the novel approach, in Game 3. Our novel approach had a more than double average margin of victory in comparison to the alternative approach.

Table 2: Results from the second experiment putting the different agentic approaches against a human benchmark, with the game scores being listed in the form agentic points - human points and the margin of victory being listed in points, where a point is a correct set guess input to the site before an opponent

Matchup	Game 1	Game 2	Game 3	Game 4	Game 5	Average Margin of Victory
Novel vs Human	26-0	25-0	25-1	27-0	25-0	**25.4**
Alternative vs Human	19-7	22-4	18-9	14-11	21-5	11.6

7 Discussion

From the results of the independent trials in Section 6, it is apparent that our fine-tuned YOLO model has a clear speed advantage over the other entities, with a relatively small sacrifice of accuracy in terms of guessing sets of 2.95% compared to the alternative approach. The template matching approach was able to achieve 100% accuracy due to its high confidence parameter value of 0.97, however this came with a noticeable drawback in speed, in that it was winning far closer games against the human benchmark and its average set time was less than a 0.6 seconds faster that the human player. This speed for accuracy tradeoff is acceptable in the context of an online game where there is no penalty for a missed guess, however in another game or perhaps a real world situation where the cost of a mistake may be far higher, this tradeoff might not be beneficial.

The head to head matchups revealed some interesting strategy choices by the human player, in an attempt to beat the agents. Against our novel approach where the average set time was less then a second, the player quickly defaulted to randomly guessing as there was almost no chance of processing the board fast enough to make a decision in time to beat the agent. Since they would need to guess the singular correct set of three cards out of a board of 12 cards (in the standard case where one set is present on the board), there are C(12,3) = 120 possible combinations, meaning the odds of a random guess being correct is less than half a percent. Thus, it logically plausible that of the 129 sets found among all five games against our novel approach, the human player was only able to score a single point.

Against the alternative approach, the human player performed as usual, describing their play style during the games as being no different then usual. This was likely due to the relatively small difference in average set time between the alternative approach and the human player, meaning that the player would have a relatively standard amount of time to formulate a set guess, in comparison to simply playing another strong human player.

Both initial hypothesis were confirmed as our model was much faster than both the human benchmark and the alternative approach at set detection, and the full agentic pipeline was able to defeat a human in a full Set game over multiple trials by a 25.4 point average margin of victory.

8 Limitations and Future Work

A notable limitation of the overall agentic pipeline is that it is specifically designed for use by a M2 Macbook Air with a 2560×1664 resolution. As noted previously when creating the agentic pipeline in Section 4, the Mac machine operated by the authors uses a "retina" display that scales every pixel into a 2x2 square of subpixels in order to increase image quality. This means that when being used on another machine where pixel values are represented in a standard fashion, the agent will click in the wrong location as it is assuming a different pixel coordinate.

Additionally, the image detection model itself is specifically fine-tuned for use on the `setwithfriends.com` version of online set. This means that when other online Set websites use Set cards with slightly different icons or coloring, the model may not be able to fully generalize its detection abilities for a new set of cards.

Future work built upon the results of this paper could seek to improve its findings in several ways. There is potential to build a more generalized agent that could adapt to different displays or cards styles, as discussed in the above limitations. Additionally, the YOLO model could be fine-tuned with a more exhaustive dataset covering every possible board permutation, or trained for a greater number of epochs, both of which may increase accuracy. A final possibility is using the same general framework of image detection and automation to build agents capable of defeating humans in other, more complex online games, such as poker or chess.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[2] Rahul Chauhan, Kamal Kumar Ghanshala, and R. C. Joshi. Convolutional neural network (cnn) for image detection and recognition. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 278–282. IEEE, 2018.

[3] Mahbub Hussain, Jordan J. Bird, and Diego R. Faria. A study on cnn transfer learning for image classification. In Ahmad Lotfi, Hamid Bouchachia, Alexander Gegov, Caroline Langensiepen, and Martin McGinnity, editors, *Advances in Computational Intelligence Systems*, pages 191–202, Cham, 2019. Springer International Publishing. ISBN 978-3-319-97982-3.

[4] Zhao Chen and Darvin Yi. The game imitation: Deep supervised convolutional networks for quick video game ai, 2017. URL `https://arxiv.org/abs/1702.05663`.

[5] Guanqing Li, Zhiyong Song, and Qiang Fu. A new method of image detection for small datasets under the framework of yolo network. In *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 1031–1035, 2018. doi: 10.1109/IAEAC.2018.8577214.

[6] Christine Dewi, Rung-Ching Chen, Yan-Ting Liu, Xiaoyi Jiang, and Kristoko Dwi Hartomo. Yolo v4 for advanced traffic sign recognition with synthetic training data generated by various gan. *IEEE Access*, 9:97228–97242, 2021. doi: 10.1109/ACCESS.2021.3094201.

[7] Liz McMahon, Gary Gordon, Hannah Gordon, and Rebecca Gordon. *The Joy of SET: The Many Mathematical Dimensions of a Seemingly Simple Card Game*. Princeton University Press, Princeton, 2017. ISBN 9781400884483. doi: 10.1515/9781400884483. URL `https://doi.org/10.1515/9781400884483`.