A Data-Driven Approach for Gin Rummy Hand Evaluation

Sang T. Truong,[1] **Todd W. Neller,** [2]
[1] DePauw University
[2] Gettysburg College
sangtruong_2021@depauw.edu, tneller@gettysburg.edu

Abstract

We develop a data-driven approach for hand strength evaluation in the game of Gin Rummy. Employing Convolutional Neural Networks, Monte Carlo simulation, and Bayesian reasoning, we compute both offensive and defensive scores of a game state. After only one training cycle, the model was able to make sophisticated and human-like decisions with a $55.4\% \pm 0.8\%$ win rate (90% confidence level) against a Simple player.

Introduction

Although Gin Rummy was one of the most popular card games of the 1930's and 1940's (Parlett 2008, 2020) and remains one of the most popular standard-deck card games (Ranker Community 2020; BoardGameGeek.com 2020), it has received relatively little Artificial Intelligence (AI) research attention. Here, we develop initial steps towards hand strength evaluation in the game of Gin Rummy.

Gin Rummy is a 2-player imperfect information card games played with a standard (a.k.a. French) 52-card deck. Ranks run from aces low to kings high. The game's object is to be the first player to score 100 or more points accumulated through the scoring of individual rounds. We follow standard Gin Rummy rules (McLeod 2020) with North American 25-point bonuses for both gin and undercut.

A *meld* is either at least 3 cards of a rank or at least 3 cards of a suit with consecutive ranks. Cards not in melds are *deadwood*. Cards have associated point values with aces being 1 point, face cards being 10 points, and other number cards having points according to their number. *Deadwood points* are the sum of card points from all deadwood cards.

At the beginning of each *round*, each player is dealt a *hand* of 10 cards, and one card is dealt face-up to start the discard pile. The rest of the cards form a face-down draw pile. On each player's turn, the player takes two actions: (1) pick up a card (from either the face-up discard pile or the face-down draw pile) and (2) discard a card to the face-up discard pile. A player with total deadwood points less than or equal to 10 may also knock to end the round. Referring the game detailed rules (McLeod 2020), the player with less deadwood generally scores according to the deadwood difference.

Players need to make a series of non-trivial choices in Gin Rummy. This paper focuses on two play strategies that we call "offensive" and "defensive". Discarding offensively, the player focuses on developing their hand while potentially helping their opponent complete a meld. Discarding defensively, the player focuses on discarding cards that are unlikely to be used by their opponent at the cost of not improving their hand. An expert player often alternates between these strategies (Shankar 2015). Determining when to play which strategy requires metrics to assert the goodness of a hand (offensive score) and the safeness of each discardable card (defensive score).

Other than some typical suggestions to build a good hand (e.g. low deadwood points, high melds), developing patterns is an empirical rule for constructing an offensive hand. For example, the player should try to form a triangle of three cards (e.g. $4\heartsuit - 5\heartsuit - 5\spadesuit$) as this pattern becomes a meld if the player can draw a $3\heartsuit$, $6\heartsuit$, $5\diamondsuit$, or $5\clubsuit$. Other patterns, such as a square of four cards, i.e. two pairs in adjacent ranks, also have advantages (Kotsckowski 2020; Rubl.com 2020; Shankar 2015). Looking for all possible patterns is a daunting task, as there might be many good patterns to follow and bad ones to avoid. Playing offensively by balancing between reducing deadwood, increasing melds, and assembling patterns under time pressure is even more challenging. Since we do not know any system for quantitatively computing offensive scores, building a model to accomplish the above task is our first goal.

The existing defensive score system relies on the basic counting principle. For example, without any other information, since a 10 (denoted "T") can be used in six different ways by the opponent to form a meld of three cards (e.g. 89T, 9TJ, TJQ, and $3\times$TTT), it has a *discard risk score* of 6. Similarly, a K has a discard risk score of 4. Thus, without other information, it is safer to discard a K than to discard a T. The discard risk score can evolve as the game progresses. For example, if the player knows that the opponent does not have any 9 or J, their T's discard risk score is only 3. Hence, it is now relatively safer to discard the T than the K. This existing defensive scoring system gives an upper bound to the discard risk score of a card since it only considers the worst cases given the available information. Refining this scoring

system is our second goal. Beyond an upper bound, we aim to improve the estimation of each cards' discard risk score to help defensive players make a more informed decision.

With the two aforementioned goals, we present a data-driven approach for hand evaluation. We show that a Convolutional Neural Network (CNN) and a Monte Carlo (MC) simulation can be used to estimate the offensive score of a hand and that Bayesian reasoning can be used to estimate the discard risk score of each discardable card. With only a single training cycle, our player achieves $55.4\% \pm 0.8\%$ with a 90% confidence level (CL) winning rate against a Simple player (SP), the baseline of our research provided by (Neller 2020). The SP implements a simple strategy:

- Ignore opponent actions and cards that are not in play.
- Only draw a face-up to complete or improve a meld. Otherwise, draw face-down.
- Discard a highest-deadwood unmelded card, breaking ties randomly and without regard to breaking up potential meld patterns (e.g. pairs).
- Knock as soon as possible.

Our data-driven approach helps the player make sophisticated and human-like decisions to improve play performance against a SP for baseline comparison.

Related Work

As previously mentioned, there is little prior AI research on the game of Gin Rummy in particular. TD-Rummy and EVO-Rummy (Kotnik and Jugal Kalita 2003) perform reinforcement learning using an artificial neural network (ANN). That study seeks to compare traditional temporal-difference learning to an evolutionary learning of ANN weights, and thus primarily serves as a comparison of two learning methods. It does not fully implement the rules of Gin Rummy, e.g. there is no laying off of cards. The only baseline for comparison is a player with a randomly initialized ANN, so it is unclear how the players would perform against a baseline SP with a reasonable strategy.

More relevant is the literature on Poker AI. Like Gin Rummy, Poker is a stochastic, imperfect information card game, albeit with a significantly smaller number of information sets. DeepStack (Moravčík et al. 2017) and Libratus (Brown and Sandholm 2018) effectively solved Heads-up, no-limit Texas Hold 'Em Poker, but each require considerable computational resources for fewer information sets. For this study, we confine ourselves to techniques that require only commonly accessible computational resources, and we focus on the learning of deterministic models to inform better Gin Rummy play decisions.

Scoring Metrics

Offensive Score

We formulate computing offensive score of a hand as finding a mapping function f such that:

$$f : (x, o) \mapsto y \tag{1}$$

Variable $x \in \mathbb{Z}_2^{1\times4\times13}$ is binary matrix hand representation (see Figure 1 for an example) of the primary player (denoted "player"). Variable $o \in \mathbb{R}^{1\times4\times13}$ is opponent hand estimation, which will be explained in more details in Defensive Score section. Variable $y \in \mathbb{R}$ represents the goodness of a hand. At the end of each round (through which both players' hands change), the winner scores s. For a game of r rounds with index counting down from $r-1$ to 0, for a hand at round q with a discount factor γ:

	A	2	3	4	5	6	7	8	9	T	J	Q	K
♠	0	0	0	0	1	0	0	0	0	0	0	1	0
♡	0	0	1	0	0	1	0	0	0	0	0	1	0
♢	0	0	0	0	0	1	1	0	0	0	0	0	0
♣	1	0	1	0	0	0	0	0	0	0	1	0	0

Figure 1: [5♠, Q♠, 3♡, 6♡, Q♡, 6♢, 7♢, J♣, A♣, 3♣] 2-dimensional binary hand representation, in which each row is a suit and each column is a rank. 1 = having a card.

$$y = \begin{cases} s \times \gamma^q & \text{if player wins} \\ -s \times \gamma^q & \text{if player loses} \end{cases} \tag{2}$$

Having $\gamma = 1$ implies no discount. The score of the last hand is never discounted as its index q always equals 0.

We employ neural network $\hat{y} = \hat{f}$ to approximate $y = f$:

$$y = \hat{y} + \epsilon, \epsilon \sim \mathcal{N}(0,\, \sigma^2) \tag{3}$$

where ϵ is Gaussian error. Since $\hat{f}$ is a parametric function, finding $\hat{f}$ is equivalent to finding a set of weights that minimizes a given loss (e.g. mean square error). From this perspective, fitting a model through a dataset is summarizing information from that data into a set of parameters. Hence, the model trained from data collected from a player can only be as good as that player. For a model to yield superior play performance against a SP, we would want to train that model on a dataset that features play superior to that of the SP. One way to approach this is to explore possible lines of play with MC simulation.

We explore the state space to find superior lines of play to that of the SP by using MC simulation on a game in which a Random player (RP) plays against a SP. Occasionally, the RP finds a good sequence of decisions that beats the SP, which cumulatively contributes to a superior policy. We use CNN to summarize this superior policy from the dataset to form the generation 1 player. More details about our methods are in the Experimentation section.

Variables x and o are raw data of a game state, challenging the model to compress. Transforming x and o to extract essential features can support the learning process. We classify these hand features into two categories: macro-features and micro-features.

Hand Macro-features Hand macro-features vector m in this study includes deadwood points, number of sets, number of runs, and hit count. The *hit count* is the number of cards that, if drawn, will either complete or extend any meld.

Generally speaking, having many runs and sets is an advantage since it reduces deadwood points. Similarly, it is a disadvantage to have many deadwood points since it prevents knocking and allows the opponent to score more points. However, when the player needs to make a discard decision, removing the highest unmelded card is not always optimal, especially when it has an excellent chance to become a meld or when the opponent needs it. For an expert player, the number of deadwood points may vary non-monotonically across a hand (Kotsckowski 2020; Rubl.com 2020; Shankar 2015).

The hit count should be used with caution. Hit count often increases when patterns are assembled and decreases as patterns are converted into melds. While optimizing patterns can guide an offensive player, optimizing the hit count can mislead the decision-making process. For example, if a player tries to maximize the hit count, they mistakenly prefer a triangle (hit count = 4) over a meld (hit count = 1 or 2). Conversely, if the player tries to minimize hit count, they mistakenly prefer three separated cards (hit count = 0) over a triangle (hit count = 4). Thus, the hit count in isolation cannot inform a good strategy.

Hand Micro-features A hand with many good patterns often has high offensive potential. For example, a pair (e.g. $5\heartsuit - 6\heartsuit$) is better than two separated cards (e.g. $5\heartsuit - 6\spadesuit$) because the likelihood of becoming a meld of $5\heartsuit$ and $6\heartsuit$ is much higher than that of $5\heartsuit$ and $6\spadesuit$. More abstractly, a hand with cards that strongly interact with their neighbors often has higher offensive potential. Both patterns and neighborhood interaction are considered to be hand micro-features in this study.

One method to extract hand micro-features is transforming binary vector hand representation $x \in \mathbb{Z}_2^{52}$ (obtained by flattening respective matrix representation) to selective interaction terms i that represent the interaction of cards with their neighbor. For example, since four neighbors of $A\heartsuit$ are $A\clubsuit$, $A\diamondsuit$, $A\spadesuit$, and $2\heartsuit$, interaction of $A\heartsuit$ with its neighbors is represented by $A\heartsuit \times A\clubsuit, A\heartsuit \times A\diamondsuit, A\heartsuit \times A\spadesuit$, and $A\heartsuit \times 2\heartsuit$ interaction terms.

Another method to extract hand micro-features is applying 2D convolution on binary matrix hand representation. If only 2×1 and a 1×2 filters with restricted kernels are used in convolution phrase, this is equivalent to the first approach using interaction terms.

2D convolution is a more general approach to extract hand micro-features than using interaction terms. However, it is more computationally expensive and requires a careful and complicated design process. In this study, we perform a comparative study of both.

Defensive Score

We can estimate the likelihood of the opponent holding a particular card given sufficient history of their play. For example, if the opponent picks up a face-up K$\clubsuit$, they are likely to have its neighbor (such as Q$\clubsuit$, J$\clubsuit$, or other K's). This observation can disincentivize the decision to discard a card neighboring K$\clubsuit$ even if it has a high rank. More generally, denoting A as the belief that the opponent holds card A and denoting B as the observation of the opponent's draw or discard of a particular card, using Bayes' theorem, we have:

$$\begin{aligned} P(A|B) &= P(B|A)P(A)/P(B) \\ &\propto P(B|A)P(A) = L(A|B) \end{aligned} \tag{4}$$

where $P(A|B)$ and $L(A|B)$ are the conditional probability and relative likelihood, respectively, of the opponent having card A given opponent behavior B. At the beginning of a game, as the player does not have any information of opponent's hand, $P(A)$ is uniformly distributed and has the value of $10/42$. As the game progresses, $P(A)$ is updated by $P(B|A)$, which can be estimated from historical data. For a given hand x with an opponent hand estimation o, the discard risk score of a card $a \in x$, $s(a)$, is defined as the sum of the joint relative likelihood of an opponent having card pairs that can form or extend a meld with a:

$$s(a) = \sum_{i \in \mathcal{N}(a)} L(b_i) \times L(c_i) \tag{5}$$

where $b_i, c_i \in o$ and $\mathcal{N}(a)$ is the index set of all card pairs that can form or extend a meld with card a.

Experimentation

The Data

We conduct our experiments on three datasets described in this section. Data is collected at the end of each round. Variables in each dataset are player hand, the opponent hand estimation, and the player's discounted score defined in equation 2. If game states are identical but score differently, the expected score is the average of all scores. Score distributions are in Figure 2. Due to Gin Rummy's high variance outcome, we use data from the hand's last play for training and validating by default.

The Simple dataset (1.6M observations) was collected from games between two SPs from (Neller 2020). Since pattern-building is not a trait of the SP, there are few patterns in the Simple dataset for the model to extract. Therefore, we explore the solution space to gather more potential meld pattern by using MC simulation with a RP. The only difference between the RP and the SP is that the RP randomly discards an unmelded card. We are aware that training the model on datasets from a RP is not ideal, but we did not have access to any other play strategy other than the SP. Using a RP is inspired by natural evolution, where random mutations play an important role. Similarly, by modifying SP to RP, we hope to diversify our player strategy, accumulate good moves over time, and transfer those good play strategies to our model.

We collect games between the RP and the SP to generate Random-16 and Random-256 datasets. We sample the discard actions in solution space 16 and 256 times for each initial pair of hands for the Random-16 (1.00M observations) and Random-256 (1.01M observations) datasets. We simulate the game until completion because Gin Rummy involves a chain of decisions with delayed rewards. Since we hypothesize that a higher hand score is a signal of the better pattern, only the best-scoring sample among 16 or 256 samples for each initial pair of hands is recorded in the dataset.

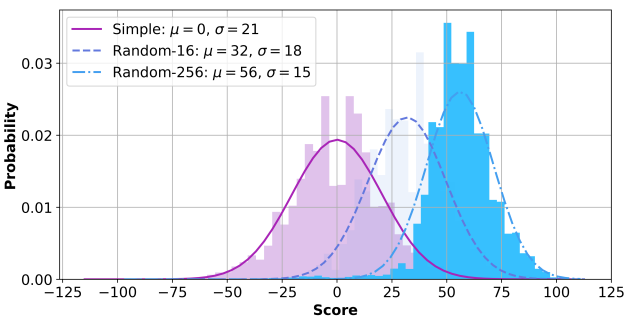


Figure 2: Scores in all datasets are approximately normally distributed. As the sample frequency increases, the mean increases, the standard deviation decreases, and the distribution skews upward.

Offensive Score

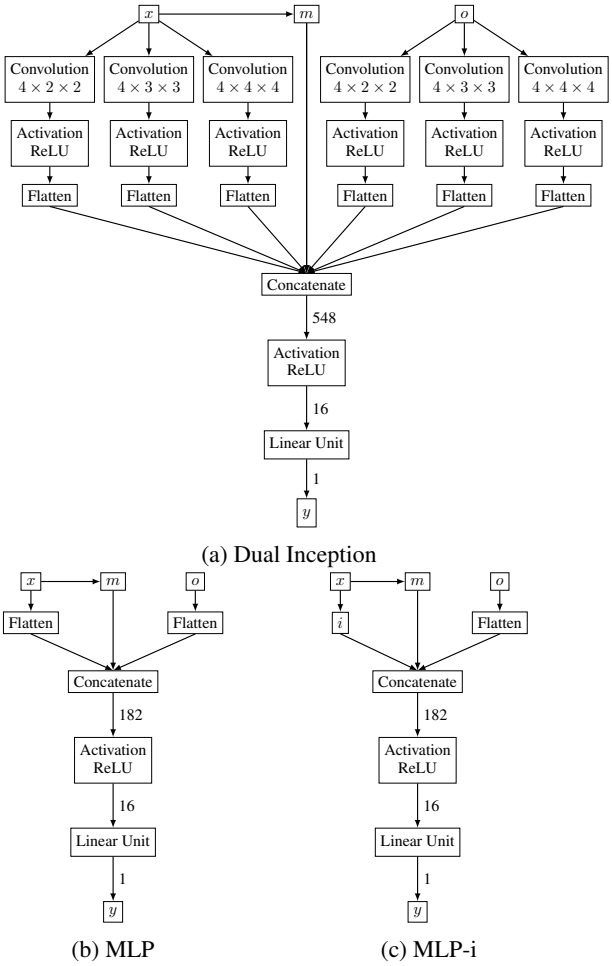
Our CNN' architecture is in Figure 3a. We apply 2D convolution to both player hand data and opponent hand estimation. Each convolution layer consists of four filters for each of three square sizes ($2 \times 2, 3 \times 3$, and 4×4) over adjacent ranks and suits in a binary matrix hand representation with no padding followed by a rectified linear unit (ReLU). We use multiple filter sizes simultaneously to minimize the total number of parameters without sacrificing model performance. For example, since the 2×2 filter can capture a pair of cards in consecutive ranks, larger filter size is not necessary for that purpose. Since this approach was inspired by Inception v1 (Szegedy et al. 2015) and since our model has dual inputs, we name our network *Dual Inception*. To show the positive impact of using 2D convolution in pattern recognition, we compare the performance of Dual Inception with a Multi-Layer Perceptron (MLP) (Figure 3b) and MLP with interaction terms (MLP-i) (Figure 3c).

We implemented all neural networks using the Python Tensorflow Keras 2.3.0 library with a batch size of 64. The training-validation ratio was 80:20. The initial learning rate was 10^{-3} and was scheduled to decay by a factor of 0.1 every 32 epochs. All models were trained over 96 epochs. Other training parameters were set as default. More implementation details can be found at https://github.com/sangttruong/ginrummy. The trained network, $\hat{f}$, was used solely to decide which card to discard. Specifically, among 11 cards, we discard the one that leaves a hand with the highest offensive score. We present a detailed result of this experience in Table 1.

Defensive Score

We record history of 10000 games between 2 SPs to estimate relative likelihood $L(B|A)$ in equation 4 by classifying 52 cards into buckets based on their 6 properties:

- Rank of the opponent hand card considered, the discarded card, and the face-up card (three properties).
- Whether or not the card is held when the player discards a card that is suited, i.e. sharing the same suit, with the currently face-up card.



(a) Dual Inception

(b) MLP

(c) MLP-i

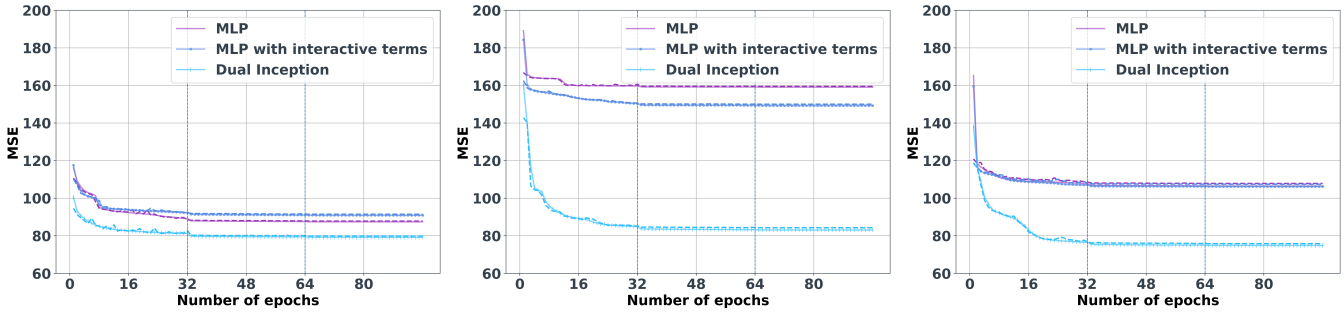
Figure 3: Architecture of (a) Dual Inception, (b) MLP, and (c) MLP-i. x, o, and y are the player's hand, the opponent hand estimation, and the player's score, respectively. m is the macro-features vector generated from x, and i is the micro-features interaction term vector. Dual Inception, MLP, and MLP-i have 9057, 1761, and 2945 total parameters, respectively, all of which are trainable.

- Whether or not the card is not suited with either the current face-up card or discarded card, suited with the face-up card, or suited with the discarded card.
- Whether or not the card is held when the player draws face-up.

For each bucket, recorded instances are averaged and transformed according to the following Inverse Square Root Unit function to form relative likelihoods of holding a given card:

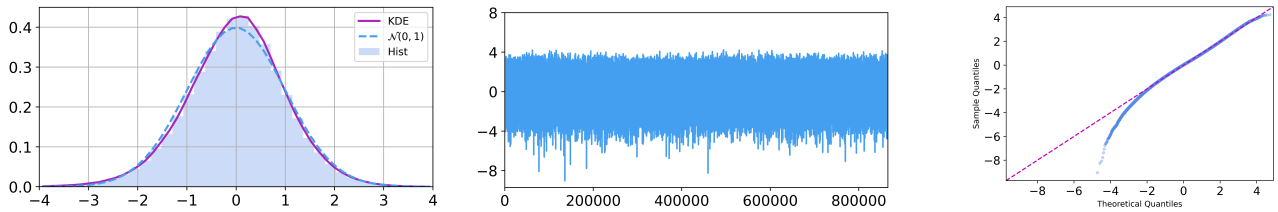
$$g(o) = \frac{o}{\sqrt{\alpha + o^2}} \tag{6}$$

This transformation is chosen both to bound relative likelihoods between 0 and 1 and to allow adjustment of the convergence rate via hyperparameter α. We select $\alpha = 1$ for this study as it leaves the initial uniformly distributed relative likelihood $(10/42)$ little-changed.



(a) (b) (c)

Figure 4: Training loss (solid line) and validation loss (dashed line) of MLP, MLP-i, and Dual Inception on Simple (a), Random-16 (b), and Random-256 (c) dataset. The vertical dashed line at 32 epochs and 64 epochs indicated a decay learning rate from 10^{-3} to 10^{-4} and 10^{-5}, respectively. Since MLP does not have micro-feature extraction, it often gives the worst performance. MLP-i uses a naive method for pattern recognition, hence it has worse performance than Dual Inception.



(a) Normalized residual distribution (b) Normalized residual plot (c) Quantile-quantile plot

Figure 5: Residual $(y - \hat{y})$ diagnostic for performance of Dual Inception on Random-256 dataset. Residual distribution is approximately Gaussian, which agrees with the assumption of ϵ on equation 3.

Results and Discussion

Our results are summarized in Table 1 and Figure 4. All models converged after the 96 epochs without symptoms of overfitting. A low MSE often correlates with a high winning rate. On average, each game lasts 500 ms (1 core, no GPU accelerator). Obtaining a computationally inexpensive hand evaluation is an important criterion for our research.

MLP and MLP-i have insignificant differences in performance among all datasets, indicating that interaction terms did not effectively extract hand micro-features. The superior performance of Dual Inception indicates that 2D convolution better extracts micro-features. Variants of Dual Inception with architectures that are deeper, wider, and having more filters were experimented with, but none showed significant improvement. Going from the Simple dataset to the Random-16 and Random-256 datasets, all models' winning rate increases, indicating that the abundant amount of information from exploring a more diverse sample of the state space helps models learn a better policy.

We conduct residual diagnostics on Dual Inception trained on the Random-256 dataset in seeking model defects (Figure 5). Kernel density estimation (KDE) using Scott's rule for bandwidth selection of the normalized residual approximates a Gaussian curve, indicating that the residual distribution is relatively normal as assumed in equation 3. The residual and quantile-quantile plots show that the model tends to over-predict a low score, which can be explained by gin or undercut bonuses. Since we never explicitly introduced these two concepts to the model, the model prediction in these cases is far from expected performance. Overall, residual diagnostics shows that Dual Inception is a good approximator $\hat{f}$ as defined in equation 3.

In addition to the above datasets, we trained all models on all stages of the Simple dataset, hoping that they can learn additional behaviors under limited information (Table 1b). As earlier in the game, the data is noisier, presenting a significant learning challenge. All all-stage model performances are worse than their performance on the last-stage dataset. On all stages of the Simple dataset and among tested models, we observe that the simple model achieves better performance, which can be explained by the fact that a simple model generalizes better when there is limited information.

Since artificial suit order in matrix hand representation seems to be unnecessary for the learning process, we introduced data augmentation to remove this effect (Table 1c): for every matrix hand representation given to a model, we also gave the model 4! = 24 row-permutations of the input. Augmentation introduces obstacles to the learning process since it increases the input size 24^2 times. Increasing the number of filters in each convolution layer from 4 to 16 was necessary to have a comparable performance with the model trained on un-augmented datasets.

We examined players' behavior to understand better the policy learned from different datasets (Figure 6). The SP

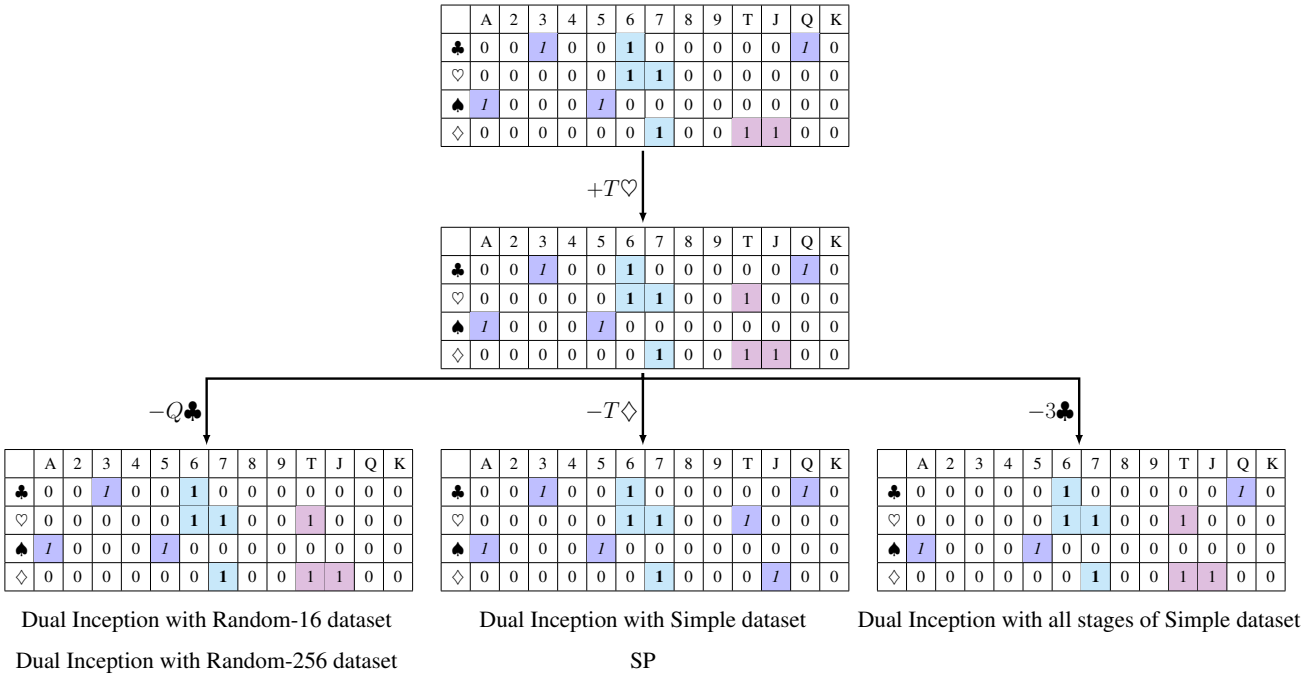


Figure 6: Behavior of SP and Dual Inception on Random-16, Random-256, Simple (all-stage and last-stage) datasets. Cyan (bold) cards form a cluster (double triangle), pink (non-bold, non-italic) cards form other clusters (which was a pair, then becomes a triangle after T♡ is drawn), and violet (italic) cards are isolated.

randomly discards one of the highest-rank unmelded cards (either T♡, T♢, J♢ or Q♣). Since $3/4$ of these cards are in a triangle, it is likely that the SP will break this pattern, and indeed it did. Dual Inception trained on the last-stage Simple dataset has the same behavior. Dual Inception trained on all-stages Simple dataset also tries to preserve the patterns in hand by keeping the triangles, but it fails to discard the highest unmelded card that is not in any pattern: it discards a 3♣. At the first stage of the game, this is considered to be a poor decision. Both Dual Inception on Random-16 and Random-256 discarded the highest unmelded card that is not in a pattern (Q♣), reducing deadwood points and preserving patterns at the same time. At the very first stage of the game, this is considered to be a great decision.

Although Dual Inception is a suitable approximator as indicated by MSE and residual diagnostics, its performance is rather modest: it only wins $55.4 \pm 0.8\%$ (90% CL) against the SP. To understand why a good estimator does not necessarily have greater performance, we examine the hand of Dual Inception player on Random-16 dataset at the last round, where information about the opponent is most abundant (Figure 7). Based on opponent hand estimation, we can compute the defensive score of each unmelded card using equation 5 (Figure 7). Dual Inception decides that discarding 9♡ is the best offensive decision. However, the model does not realize that discarding 9♡ is also highly risky, based on the discard risk score. It is also unlikely that 9♡ can be laid off except when the opponent has 7♡ and 8♡. As the player cannot simultaneously optimize two criteria (i.e. defensive and offensive), it discards 9♡, allowing the opponent to complete another meld and win the game with 30 points. Note that the model is aware of its disadvantaged game states as all predicted offensive scores are near -30. Discards based solely on discard risk score do not yield a superior player to the SP. Discarding is a multi-criteria decision, and it is up to the player to select the criteria. Our model can provide some insight for decision-making for each criterion but cannot choose which criterion to use.

A high winning rate does not necessarily imply a good play strategy. For example, to exploit the knowledge that SP knocks as early as possible, we can knock late to gain an undercut bonus and obtain a winning rate of $70.1\% \pm 0.75\%$ (90% CL). Nonetheless, this strategy is not advisable because it allows the opponent to complete more melds or go Gin (Shankar 2015). This player would be far from optimal and be exploitable itself. We reiterate that the goal of this research is not to optimize the winning rate against the SP but rather to find a policy that makes sophisticated, high-quality decisions. Cumulatively making good decisions leads to a better winning rate, as shown in the case of the Dual Inception player trained on the Random-16 and Random-256 datasets.

Conclusions and Future Work

We present a data-driven approach for hand evaluation. Dual Inception is the most competitive model among all tested architectures and datasets. Its success results from micro-features extraction using CNN, exploring the state space using MC simulation, and opponent hand estimation using Bayes' theorem. Data augmentation and all-stage datasets

Dataset	Model	Train loss	Valid loss	Win rate (%)
Simple	MLP	87.4	87.7	42.9 ± 0.8
	MLP-i	90.8	91.4	43.6 ± 0.8
	Dual Inception	79.1	79.8	45.8 ± 0.8
Random -16	MLP	159.0	159.4	44.0 ± 0.8
	MLP-i	149.2	150.0	42.3 ± 0.8
	Dual Inception	83.0	83.0	50.1 ± 0.8
Random -256	MLP	106.1	107.4	48.1 ± 0.8
	MLP-i	106.4	107.8	45.8 ± 0.7
	Dual Inception	74.8	75.7	55.4 ± 0.8

(a) Experiment 1: Training MLP, MLP-i, and Dual Inception on Simple, Random-16, and Random-256 datasets.

Simple (All stages)	MLP	153.7	154.5	34.3 ± 0.7
	MLP-i	153.5	155.7	29.3 ± 0.6
	Dual Inception	144.1	146.5	19.7 ± 0.6

(b) Experiment 2: Training MLP, MLP-i, and Dual Inception on all stage of Simple dataset.

Random -16-A	Dual Inception	181.7	181.4	42.8 ± 0.8
	Dual Inception 16 filters	132.6	132.0	51.2 ± 0.8

(c) Experiment 3: Training Dual Inception on augmented Random-16 dataset.

Table 1: Performance of Dual Inception, MLP, MLP-i on the Simple, Random-16, and Random-256 datasets. We record the winning rate in games where these models served as the primary discard-decision-making tools against SP. We compute Wilson confidence intervals with a 90% CL on 10000 games. Data augmentation can more generally represent the population but also introduces additional difficulty for the learning process. Increasing the number of filters for each convolution from 4 to 16 (35937 parameters) is necessary to keep the model's performance at the same level as the one that was trained on un-augmented data.

	A	2	3	4	5	6	7	8	9	T	J	Q	K
♣	0.10	0.09	F	0.18	0.29	T	0.55	0.61	0.49	0.33	0.44	F	0.39
♡	0.10	0.08	0.10	0.12	0.13	F	F	0.33	F	F	0.43	F	0.48
♠	F	F	0.07	0.09	F	0.33	0.23	0.24	0.32	F	0.33	0.30	F
♢	0.10	0.08	0.09	0.11	F	0.40	F	0.31	0.46	F	F	0.14	F

(a) Opponent hand estimation and player's hand (cyan).

	A	2	3	4	5	6	7	8	9	T	J	Q	K
♣	0	0	*1*	0	0	0	0	0	0	0	0	0	0
♡	0	0	0	0	0	**1**	**1**	0	*1*	***1***	0	0	0
♠	*1*	0	0	0	1	0	0	0	0	***1***	0	0	0
♢	0	0	0	0	1	0	**1**	0	0	***1***	0	0	0

(b) Real opponent hand. Violet (italic) are isolated cards. Orange (italic and bold) are cards in meld. Other cards are in patterns.

$5\spadesuit : 0.16$ $A\spadesuit : 0.01$ $6\heartsuit : 0.90$ $5\diamondsuit : 0.09$

$7\heartsuit : 0.13$ $3\clubsuit : 0.10$ $7\diamondsuit : 0.40$ $9\heartsuit : 0.53$

(c) Defensive scores of player's hand.

$5\spadesuit : -29.9$ $A\spadesuit : -32.4$ $6\heartsuit : -29.2$ $5\diamondsuit : -29.4$

$7\heartsuit : -29.0$ $3\clubsuit : -29.4$ $7\diamondsuit : -29.0$ $9\heartsuit : -29.7$

(d) Offensive scores of player's hand.

Figure 7: Offensive-defensive dilemma: offensive optimum is defensive suboptimum.

are shown to introduce extra obstacles to the learning process without any significant benefit.

There are various directions to extend this project. Firstly, message passing (Fey and Lenssen 2019) should be considered an alternative to 2D convolution to avoid the artificial suit-ordering introduced by a matrix hand representation. Secondly, other state space exploration methods (e.g. flat MC tree search) should be investigated as an alternative for MC simulation in this study. Both message passing and advanced state space exploration methods are much more computationally expensive and require many complex and non-trivial design decisions. Thirdly, a player that can jointly optimize both offensive and defensive scores should be developed. Last but not least, reinforcement learning should be investigated for Gin Rummy. Specifically, the training cycle with CNN and MC simulation (described in the offensive score section) could be repeated: we could recollect data from a generation $n-1$ player to re-train the model and obtain a generation n player that has marginally better performance than generation $n-1$. Theoretically, with a carefully designed network and $(n \to +\infty)$ generation, this approach can produce a superhuman player as it was done similarly in other games such as No-Limit (Moravčík et al. 2017) and Limit (Heinrich and Silver 2016) Texas Hold'em Poker. Obtaining such a player is beyond the scope of this paper. With limited time and computational resources, we only demonstrate that CNN and MC simulation is a promising approach to develop an excellent player. Although reinforcement learning has achieved some successes in other games such as Poker, the only prior work that applies reinforcement learning to Gin Rummy is (Kotnik and Jugal Kalita 2003).

Gin Rummy is a complex, imperfect information game that offers an excellent sandbox for research in decision-making. With limited computational resources and time, we show that CNN, MC simulation, and Bayes' theorem can formulate a player that can make complex decisions. While this is a modest improvement, we establish a foundation that brings the research community one step closer to achieving superhuman Gin Rummy performance.

Acknowledgments

We would like to express our very great appreciation to Seoul Robotics Co., Ltd. for providing Sang T. Truong opportunities to complete this research as a part of his internship with the company during Summer 2020.

References

BoardGameGeek.com. 2020. Most popular standard deck card games. https://tinyurl.com/bggstdcardgames. Accessed: 2020-09-28.

Brown, N.; and Sandholm, T. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 359(6374): 418–424. ISSN 0036-8075. doi: 10.1126/science.aao1733. URL https://science.sciencemag.org/content/359/6374/418.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Heinrich, J.; and Silver, D. 2016. Deep Reinforcement Learning from Self-Play in Imperfect-Information Games. In *Deep Reinforcement Learning Workshop at Conference on Neural Information Processing Systems*.

Kotnik, C.; and Jugal Kalita. 2003. The Significance of Temporal-Difference Learning in Self-Play Training: TD-Rummy versus EVO-rummy. In *the Twentieth Conference on Machine Learning held in Washington, DC*, 369–375.

Kotsckowski, M. 2020. RummyTalk: The Most Comprehensive Gin Rummy Guide Online. http://rummytalk.com/. Accessed: 2020-09-28.

McLeod, J. 2020. Gin Rummy - Card Game Rules. https://www.pagat.com/rummy/ginrummy.html. Accessed: 2020-09-28.

Moravčík, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356(6337): 508–513. ISSN 0036-8075. doi:10.1126/science.aam6960. URL https://science.sciencemag.org/content/356/6337/508.

Neller, T. W. 2020. Gin Rummy EAAI Undergraduate Research Challenge. URL http://cs.gettysburg.edu/~tneller/games/ginrummy/eaai/.

Parlett, D. 2008. *The Penguin Book of Card Games*. Penguin. ISBN 9780141037875.

Parlett, D. 2020. GIN RUMMY "The game of the stars". https://www.parlettgames.uk/histocs/ginrummy.html. Accessed: 2020-09-28.

Ranker Community. 2020. The Most Popular Fun Card Games. https://www.ranker.com/crowdranked-list/most-fun-card-games. Accessed: 2020-09-28.

Rubl.com. 2020. Gin-Rummy Playing Tips. http://www.rubl.com/rules/gin-rummy-tips.html. Accessed: 2020-09-28.

Shankar, P. 2015. *How To Win At Gin Rummy: Playing for Fun and Profit*. Echo Point Books & Media. ISBN 978-1626541979.

Szegedy, C.; Wei Liu; Yangqing Jia; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9.