Learning Policies from Human Data for Skat

Anonymous
Address line

Abstract

Decision-making in large imperfect information games is difficult. Thanks to recent success in Poker, Counterfactual Regret Minimization (CFR) methods have been at the forefront of research in these games. However, most of the success in large games comes with the use of a forward model and powerful state abstractions. In trick-taking card games like Bridge or Skat, large information sets and an inability to advance the simulation without fully determinizing the state make forward search problematic. Furthermore, state abstractions can be especially difficult to construct because the precise holdings of each player directly impact move values.

In this paper we explore learning model-free policies for Skat from human game data using deep neural networks (DNN). We produce the new state-of-the-art bot for bidding and declaration by introducing methods to a) directly vary the aggressiveness of the bidder and b) declare games based on expected value while mitigating issues with rarely observed state-action pairs. Although cardplay policies learned through imitation are slightly weaker than the current best search based method, they run orders of magnitude faster. We also explore how these policies could be learned directly from experience in a Reinforcement Learning setting and discuss the value of incorporating human data for this task.

1 Introduction

Decision-making in large imperfect information games can be difficult. Techniques based on Counterfactual Regret Minimization (CFR) (Zinkevich et al. 2008) are currently considered state-of-the-art, but a forward model and expert abstractions are often required to scale these techniques to larger games. Some games are simply too large to solve with CFR methods on current hardware. For instance, in the popular 3-player card game of Skat there are 64,512,240 possible information sets for the first decision point right after the initial deal. Each of these information sets is comprised of 646,646 states. Overall, there are $\approx 4.4 \cdot 10^{19}$ terminal histories in the pre-cardplay portion alone and many more when taking cardplay into account.

The general approach for solving larger games with these methods is to first abstract the game into a smaller version of itself, solve that, and then map those strategies back to the original game. This process implies a game-specific tradeoff between abstraction size and how well the strategies computed on the abstraction translate to the real game. Recent advances in Poker (Moravčík et al. 2017; Brown and Sandholm 2017) highlight the effectiveness of this approach in some games. In Skat and Contract Bridge, however, the values of actions in an information set are highly dependent on the interactions between cards within a player's hand and the exact cards which each opponent possesses. This makes it difficult to construct abstractions that are small enough to use with CFR methods, but expressive enough to capture the per-card dependencies that are vital to success in the full game.

Good forward models are also difficult to produce in Skat due to the imperfect information nature of the game. In order to advance the model, the state must be "determinized" from the root information set. The current state-of-the-art in Skat uses a combination of open-handed simulation and a table-based state evaluator learned from human games (Buro et al. 2009). It relies on a forward model to perform the open-handed simulations and hand-based abstractions to build the state evaluator used for bidding. Open-handed simulations have been rightfully criticized across the literature (Frank and Basin 1998; Russell and Norvig 2016) because they assume that a strategy can take different actions in different states that are part of the same information set.

In this paper we focus on learning model-free policies for Skat from human-generated data. Abandoning the use of a forward model for Skat is complicated, but may be worthwhile not only because it alleviates many of the aforementioned problems, but also because it allows policies to be trained or improved directly through experience. In particular, techniques that alleviate some of the issues with learning from expert data are explored. We present a method for varying the aggressiveness of the bidder by viewing the output of the network as a distribution of the actions of the humans and selecting the action that maps to the percentile of bidder aggression we desire. Imitation policy performance is further improved by accounting for rarely-seen state-action pairs without generating new experience. Our contributions lead to a new state-of-the-art bidding system for Skat, and a reasonably strong cardplayer that performs orders of magnitude faster than search based methods. Finally, we explain how these policies could be learned directly from experience and discuss the value of incorporating human data into this process.

2 Background and Related Work

In this section we provide the reader with the necessary background related to the game of Skat. We also discuss previous work related to AI systems for Skat and similar domains.

Skat

Our domain of choice is a 3-player trick-taking card game called Skat. Originating in Germany in the 1800s, Skat is played competitively in clubs around the world. The following is a shortened explanation that includes the necessary information to understand the work presented here. For more in-depth explanation about the rules of Skat we refer interested readers to `https://www.pagat.com/schafk/skat.html`.

Skat is played using a 32-card deck which is built from a standard 52-card by removing 2,3,4,5,6 in each suit. A hand consists of each of the three players being dealt 10 cards with the remaining two kept face down in the so-called skat.

Games start with the bidding phase. The winner of this phase plays as the soloist against the team formed by the other players during the cardplay phase of the game. Upon winning the bidding, the soloist decides whether or not to pickup the skat followed by discarding two cards face down, and then declares what type of game will be played during cardplay. The game type declaration determines both the rules of the cardplay phase and also the score for each player depending on the outcome of the cardplay phase. Players typically play a sequence of 36 of such hands and keep a tally of the score over all hands to determine the overall winner.

The game value, which is the number of points the soloist can win, is the product of a base value (determined by the game type, see Table 1) and a multiplier. The multiplier is determined by the soloist having certain configurations of Jacks and other high-valued trumps in their hand and possibly many game type modifiers explained in Table 2. An additional multiplier is applied to the game value for every modifier.

After dealing cards, the player to the right of the dealer starts bidding by declaring a value that must be less than or equal to the value of the game he intends to play — or simply passing. If the soloist declares a game whose value ends up lower than the highest bid, the game is lost automatically. Next, the player to the dealer's left decides whether to accept the bid or pass. If the player accepts the bid, the initial bidder must proceed by either passing or bidding a higher value than before. This continues until one of the player's decides to pass. Finally, the dealer repeats this process by bidding to the player who has not passed. Once two players have passed, the remaining player has won the bidding phase and becomes the soloist. At this point, the soloist decides whether or not to pick up the skat and replace up to two of the cards in his hand and finally declares a game type.

Cardplay consists of 10 tricks in which the trick leader (either the player who won the previous trick or the player to the left of the dealer in the first trick) plays the first card.

Table 1: Game Type Description

Type	Base Value	Trumps	Soloist Win Condition
Diamonds	9	Jacks and Diamonds	$\geq$ 61 card points
Hearts	10	Jacks and Hearts	$\geq$ 61 card points
Spades	11	Jacks and Spades	$\geq$ 61 card points
Clubs	12	Jacks and Clubs	$\geq$ 61 card points
Grand	24	Jacks	$\geq$ 61 card points
Null	23	No trump	losing all tricks

Table 2: Game Type Modifiers

Modifier	Description
Schneider	$\geq$90 card points for soloist
Schwarz	soloist wins all tricks
Schneider Announced	soloist loses if card points $<$ 90
Schwarz Announced	soloist loses if opponents win a trick
Hand	soloist does not pick up the skat
Ouvert	soloist plays with hand exposed

Play continues clockwise around the table until each player has played. Passing is not permitted and players must play a card of the same suit as the leader if they have one — otherwise any card can be played. The winner of the trick is the player who played the highest card in the led suit or the highest trump card.

In suit and grand games, both parties collect tricks which contain point cards (Jack:2, Queen:3,King:4,Ten:10,Ace:11) and non-point cards (7,8,9). Unless certain modifiers apply, the soloist must get 61 points or more out of the possible 120 card points in the cardplay phase to win the game. In null games the soloist wins if he loses all tricks.

Previous Work

Previous work on Skat AI has applied separate solutions for decision-making in the pre-cardplay and cardplay phases. The cardplay phase has received the most attention — probably due to its similarity to cardplay in other trick-taking card games.

Despite its shortcomings, Perfect Information Monte-Carlo (PIMC) Search (Levy 1989) continues be the state-of-the-art cardplay method for Skat and other trick-taking card games like Bridge (Ginsberg 2001) and Hearts (Sturtevant 2008). Later, Imperfect Information Monte-Carlo Search (Furtak and Buro 2013) and Information Set Monte Carlo Tree Search (Cowling, Powley, and Whitehouse 2012) sought to address some of the issues inherent in PIMC while still relying on the use of state determinization and a forward model.

The current state-of-the-art for the pre-cardplay phase (Buro et al. 2009) uses forward search — evaluating leaf nodes after the discard phase using the GLEM framework (Buro 1998). The evaluation function is based on a generalized linear model over table-based features indexed by abstracted state properties. These tables are computed using human game play data. Evaluations take the player's hand, the game type, the skat, and the player's choice of discard into account to predict the player's winning probability. The maximum over all player choices of discard and game type is taken and then averaged over all possible skats. Finally, the program bids if the such estimated winning probability is higher than some constant threshold.

3 Learning Bidding Policies from Human Data

In this section we train pre-cardplay policies for Skat using human data. First, we present a simple policy learned through direct imitation of human play. Next, we study the issue of trying to imitate an action from supervised data when intent is not visible. This is a problem when learning policies for Skat's bidding phase because the data doesn't show how high the player was willing to bid with their hand. Finally, we explore using a value network in conjunction with a policy for the declaration/pickup phases.

The pre-cardplay phase has 5 decision points: max bid for the Bid/Answer Phase, max bid for the Continue/Answer Phase, the decision whether to pickup the skat or declare a hand game, the declaration and the discard. The bidding phases display sequential bids between two players, but further bids can only be made if the other player has not already passed. This allows a player to effectively predetermine what their max bid will be. This applies to both the Bid/Answer and Continue/Answer phases. However, in the Continue/Answer phase remaining players must consider which bid caused a player to pass in the first bidding phase. The Declaration and Discard phases happen simultaneously and could be modelled as a single decision point, but for simplicity's sake we separate them.

For each decision point, a separate DNN was trained using human data from a popular Skat server (DOSKV 2018). For discard, separate networks were trained for each game type except for Null and Null Ouvert. These were combined because of their similarity and the low frequency of Ouvert games in the dataset.

The features for each network are one hot encoded. The features and the number of bits for each are listed in Table 3. The Bid/Answer network uses the Player Hand, and Player Position features. The Continue/Answer network uses the same features as Bid/Answer, plus the Bid/Answer Pass Bid, which is the pass bid from the Bid/Answer phase. The Hand/Pickup network uses the same as Continue/Answer, plus the Winning Bid. The Declare network uses the Player Hand + Skat feature in place of the Player Hand feature, as the skat is part of their hand at this point. The Discard networks use the same features as the Declare network.

Note that the game type is not included in the feature set of the Discard networks because they are split into different networks based on that context. Assuming no skip bids (not normally seen in Skat) these features represent the minimum information needed to reconstruct the game state as observed from the player (except for the ambiguity between Null and Null Ouvert in the Discard phase). Thus abstraction and feature engineering in our approach is limited.

The outputs for each network correspond to any of the possible actions in the game at that phase. The legality of the actions depend on the state, however, this is only enforced during actual gameplay and not during training. Table 4 lists the actions that correspond to the outputs of each network, accompanied by the number of possible actions.

Table 3: Network input features

Features	Width
Player Hand	32
Player Position	3
Bid/Answer Pass Bid	67
Winning Bid	67
Player Hand + Skat	32

Table 4: Corresponding actions to Network Outputs

Phase	Action	Width
Bid/Answer	MaxBid	67
Continue/Answer	MaxBid	67
Hand/Pickup	Game Type or Pickup	8
Declare	Game Type	7
Discard	Pair of Cards	496

Table 5: Pre-cardplay training set sizes and imitation accuracies

Phase	Train Size (millions)	Train Acc.%	Test Acc.%
Bid/Answer	23.678	83.5	83.3
Continue/Answer	23.678	80.0	79.8
Pickup/Hand	23.647	97.4	97.3
Declare	22.112	85.6	85.6
Discard Diamonds	2.375	75.6	75.2
Discard Hearts	3.048	75.8	75.6
Discard Spades	3.801	75.9	75.4
Discard Clubs	4.984	76.1	75.9
Discard Grand	5.621	71.2	70.5
Discard Null	1.424	83.4	82.9

The networks are all identical, except for the input and output layers. Each network has 5 fully connected hidden layers with RELU (Nair and Hinton 2010) activation gates. The network structure can be seen in Figure 1. Tensorflow (Abadi et al. 2016) was used for the entire training pipeline. Networks are trained using the ADAM optimizer (Kingma and Ba 2014) to optimize cross-entropy loss with a constant learning rate set to 10^{-4}. The middle 3 hidden layers incorporate Dropout (Srivastava et al. 2014), with keep probabilities set to 0.6. Each network was trained with early stopping (Prechelt 1998) for at most 10 epochs. The size of the training sets, and accuracies after the final epoch are listed for each network in Table 5. These accuracies appear to be quite reasonable, given the number of options available at each decision point. The test dataset sizes were set to 100,000, with the exception of the Null Discard network which was limited to 50,000 samples.

The goal of these networks is to imitate the human players. One issue is that while the exact actions during the bidding phase are captured, the intent of how high the player would have bid is not. The intent is based largely on the strength of the hand, but how high the player bids is dependent at what point the other player passed. For example, if a player decided their maximum bid was 48 but both other players passed at 18, the maximum bid reached is 18. For this reason, the max bid was trained on the pass bids of the players. We know what the max bid for these players is because they either passed at that bid (if they are the player to

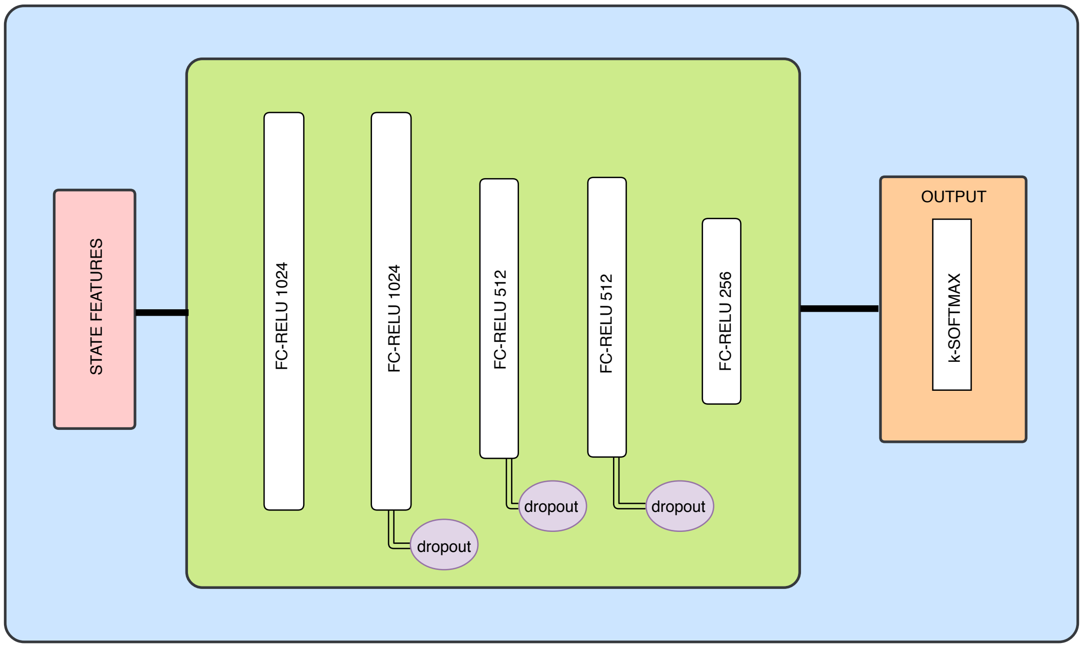

Figure 1: Network architecture used across all game types for both soloist and defenders.

bid) or at the next bid (if they are the player to answer).

The output of this network corresponds to the probability the player should pass at a certain level. The argmax would provide the most likely bid, but it would not utilize the sequential structure of the bids. What we propose is to take the bid that corresponds to a given percentile, termed A. In this way, the distribution of players aggressiveness in the human population can be utilized directly to alter the aggressiveness of the player. Let B be the ordered set of possible bids, with b_i being the i^{th} bid, with b_0 corresponding to passing without bidding. The maxbid is determined using

$$\text{maxbid}(s, A) = \min(b_i) \text{ s.t. } \Sigma_{j=0}^{i} p(b_j; \theta|I) \geq A \quad (1)$$

Given the returned maxbid is b_i and the current highest bid in the game is b_{curr}, the policy for the bidding player is

$$\pi_{bid}(b_i, b_{curr}) = \begin{cases} b_{i+1} & \text{if } b_i > b_{current} \\ \text{pass} & \text{otherwise} \end{cases} \quad (2)$$

while the policy for the answer player is

$$\pi_{ans}(b_i, b_{curr}) = \begin{cases} \text{yes} & \text{if } b_i \geq b_{current} \\ \text{pass} & \text{otherwise} \end{cases} \quad (3)$$

Another limiting factor in the strength of direct imitation is that the policy is trained to best copy humans, regardless of the strength of the move. While the rational player would always play on expectation, it appears there is a tendency for risk aversion in the human data set. For example, the average human seems to play far fewer Grands than Kermit. Since Grands are high risk / high reward, they are a good indication of how aggressive a player is. To improve the pre-cardplay policy in the Hand/Pickup and Declare phases, we can instead select actions based on learned values for each possible game type. Formally, this policy is

$$\pi_{MV}(I, a; \theta) = argmax(v(I, a; \theta)) \quad (4)$$

where v is the value of the game for the given Information Set I, action a, pair, and parameters θ in the trained network.

We trained two additional networks, one for the Hand/Pickup phase and one for the Declare phase. These networks were identical to the previous ones, except linear activation units are used for the outputs. The network was trained to approximate the value of the actions. The value labels are simply the endgame value of the game to the soloist. The loss was the mean squared error of prediction on the actual action taken. For the Hand/Pickup network, the train and test loss were 993 and 1004 respectively. For the Declare network, the values were 855 and 860. These values seem quite large, but with the high variance and large scores in Skat, they are in the reasonable range.

The π_{MV} seen in Equation 4 is problematic. The reason for this that a lot of actions, while legal, are never seen in the training data within a given context. This leads to action values that are meaningless, which can be higher than the other meaningful values. For example, Null Ouvert is rarely played, has high game value and is most often won. Thus the network will predict a high value for the Null Ouvert action in unfamiliar situations which in turn are not appropriate situations to play Null Ouvert. This results in an over optimistic player in the face of uncertainty, which can be catastrophic. This is demonstrated in the results section.

To remedy this issue, we decided to use the supervised policy network in tandem with the value network. The probability of an action from the policy network is indicative of how often the move is expected to be played given the observation. The higher this probability is, the more likely we have seen a sufficient number of ”relevant” situations in which the action was taken. With a large enough dataset, we assume that probabilities above a threshold indicate that we have enough representative data to be confident in the predicted action value. We chose 0.1 as the threshold. There is no theoretical reason for this exact threshold, other than it is low enough that it guarantees that there will always be a value we are confident in. Furthermore, the probability used is normalized after excluding all illegal actions.

The policy for the Hand/Pickup and Declare phases using the method described above is

$$\pi_{MLV}(I, a; \theta) = argmax(v_L(I, a; \theta)) \tag{5}$$

where

$$v_L(I, a) = \begin{cases} v(I, a; \theta) & \text{if } p_{legal}(a; \theta | I) \geq \lambda \\ -\infty & \text{otherwise} \end{cases} \tag{6}$$

in which p_{legal} is the probability normalized over all legal actions and λ is a constant set to 0.1 in our case.

4 Bidding Experiments

The current strongest Skat bot is Kermit which plays at human expert strength (Buro et al. 2009). Kermit's bidding is based on estimating winning probabilities during bidding and thresholding the estimates to decide to increase the bid, to accept a bid, or to pass. As this algorithm is the state-of-the-art for pre-cardplay phase of Skat, it is used as the baseline for the rest of this paper. Since the network-based player learned off of human data, it is assumed that defeating Kermit is indicative of the overall strength of the method, and not based on exploiting it.

Because Skat is a 3-player game, each match in the tournament is broken into six games. In a match, all bot configurations are considered, with the exception of of all three being the same bot, resulting in six games (see Table 6). In each game, once the pre-cardplay phase is finished, the rest of the game is played out using Kermit cardplay, an expert level player based on PIMC search which samples 160 worlds — a typical setting. The results for each bot is the resultant average over all the games played. Each tournament was ran for 3,000 matches. All tournaments featured the same identical deals in order to decrease variance.

Different variations of the pre-cardplay policies were tested against the Kermit baseline. Unless otherwise stated, the policies use the aggressiveness transformation discussed in the previous section, with the A value following the policies prefix. The variations are:

- Direct Imitation **(DI)**: selects the most probable action from the imitation networks
- Aggressive Bidding **(AB)**: like DI, but uses the aggressiveness transformation in bidding
- Maximum Value **(MV)**: like AB, but selects the maximum value action in the Hand/Pickup and Declare phases
- Maximum Likely Value **(MLV)**: like AB but uses the maximum likely value policy, π_{MLV}, in the Hand/Pickup and Declare phases

While the intuition behind the aggressiveness transformation is rooted in increasing the aggressiveness of the bidder, the choice for A is not obvious. MLV and AB were investigated with A values of 0.85, 0.89, 0.925, and 0.95. Through limited trial and error, these values were chosen to

Table 6: Player configurations in a single match consisting of six hands (K=Kermit, NW=Network Player)

Game Number	Seat1	Seat2	Seat3
1	K	K	NW
2	K	NW	K
3	K	NW	NW
4	NW	NW	K
5	NW	K	NW
6	NW	K	K

approximately result in the player being slightly less aggressive, similarly aggressive, more aggressive, and much more aggressive than Kermit's bidding, as measured by share of soloist games played in the tournament setting. MV was only tested with A of 0.925 since it was clear that the issues of overoptimism were catastrophic.

An overview of the game type selection breakdown is presented in Table 7, while an overview on the performance is presented in Table 8. To measure game playing performance we use the Fabian-Seeger tournament point (TP) scoring system which awards the soloist (50 + game value) points in case he wins. In case of a loss, the soloist loses (50 + 2· game value) points and the defenders are awarded 40 points. All tournament points per game **(TP/G)** difference values reported were found to be significant.

Clearly, naively selecting the max value (MV) in the Hand/Pickup and Declare phases cause the bot to perform very poorly as demonstrated by it performing 54 TP/G worse than the Kermit baseline. It plays 25.4 Null Ouvert games as soloist per 100 games player, which is extremely high to the point of absurdity. The reason for this overoptimism was discussed already in the previous section, and these results bear this out.

Direct Imitation (DI) performed much better, but still performed slightly worse than the baseline by 1.35 TP/G. The issue with being overly conservative is borne out with the player being soloist approximately half as often as Kermit.

The direct imitation with the aggressiveness transformation all performed better than Kermit. The best value for A was 0.85 (AB.85) which leads to +2.86 TP/G against Kermit. The advantage decrease with increasing A values. At the 0.85 A value, the player is soloist a fewer of 1.81 times per 100 games played, indicating it is a less aggressive bidder than Kermit.

The players selecting the max value declarations within a confidence threshold (MLV) performed the best overall, outperforming the AB players at each A value level. The best overall player against Kermit is the MLV.85 player. It outperforms Kermit by 3.65 TP/G, 0.79 TP/G more than the best AB player.

The actual breakdown of games is quite interesting, as it shows that the AB and MLV players are drastically different in their declarations. Across the board, AB is more conservative as it plays more Suit games and less Grand games (worth more and typically more risky) than the corresponding MLV player. Kermit falls somewhere in between. One other trend is that as the A values increase, the share of soloist games increases, but the majority of the extra games are Suit. There is a diminishing returns in the number of

Table 7: Game type breakdown by percentage for each player, over their 3,000 match tournament. First player soloist games are broken down into types. Defense games (Def) and games that were skipped due to all players passing (Pass) are also included. The K vs X entries list breakdowns of Kermit playing against player(s) X with identical bidding behavior.

Match	Grand	Suit	Null	NO	Def	Pass
DI vs K	5.9	13.2	0.5	0.7	66.7	13.0
MV.925 vs K	7.0	1.8	0.1	25.4	64.0	1.6
AB.85 vs K	8.4	20.8	1.2	1.0	65.6	3.0
AB.89 vs K	8.7	21.8	1.3	1.1	64.9	2.2
AB.925 vs K	9.0	22.9	1.3	1.2	64.0	1.6
AB.95 vs K	9.5	23.8	1.5	1.3	62.9	1.1
MLV.85 vs K	11.5	17.5	1.2	1.2	65.6	3.0
MLV.89 vs K	11.9	18.4	1.3	1.2	64.9	2.2
MLV.925 vs K	12.2	19.5	1.4	1.3	64.0	1.6
MLV.95 vs K	12.7	20.4	1.5	1.4	62.9	1.1
K vs DI	10.8	21.7	3.8	2.0	50.5	11.1
K vs *.85	11.0	17.8	2.6	1.8	63.4	3.4
K vs *.89	11.0	17.0	2.3	1.7	65.2	2.7
K vs *.925	11.0	16.2	2.1	1.7	66.9	2.0
K vs *.95	11.0	15.2	1.8	1.7	68.7	1.5

Table 8: Tournament results over 3,000 matches between learned cardplay polices and the baseline player (Kermit). All players use Kermit's cardplay. Rows are sorted by score difference (TP/G=tournament points per game, S=soloist percentage)

Player (P)	TP/G(P)	TP/G(K)	diff.	S(P)	S(K)	diff.
MV.925	-21.95	32.06	-54.01	34.4	31.1	3.34
DI	20.53	21.88	-1.35	20.3	38.4	-18.09
AB.95	23.36	21.86	1.50	36.1	29.7	6.35
AB.925	23.83	22.03	1.80	34.4	31.1	3.34
MLV.95	24.12	21.81	2.31	36.1	29.7	6.35
MLV.925	24.55	22.19	2.36	34.4	31.1	3.34
AB.89	24.20	21.72	2.48	32.9	32.1	0.78
AB.85	24.35	21.48	2.86	31.4	33.2	-1.81
MLV.89	24.84	21.84	3.01	32.9	32.1	0.78
MLV.85	25.14	21.49	**3.65**	31.4	33.2	-1.81

high value Grand games.

One additional tournament was run between the best AB and MLV players, which are AB.85 and MLV.85. In this 3,000 match tournament, MLV.85 outperformed AB.85 by 0.60 TP/G

5 Learning Cardplay Policies

With drastically improved pre-cardplay policies, the next step was to create a cardplay policy based off the human data. To do this, a collection of networks were trained to imitate human play using the same network architecture used for the pre-cardplay imitation networks. Six networks were trained in all; defender and soloist versions of Grand, Suit, and Null.

To capture the intricacies of the cardplay phase, we use handcrafted features — listed in Table 9. Player Hand represents all the cards in the players hand. Hand Value is the sum of the point values of all cards in a hand (scaled to the max possible value). Lead cards represents all the cards the player lead (first card in the trick). Sloughed cards indicate all the non-Trump cards that the player played that did not follow the suit. Void suits indicate the suits which a player cannot have based on past moves. Trick Value provides the point value of all cards in the trick (scaled to the max possible value). Max Bid Type indicates the suit bid multipliers that match the max bid of the opponents. For example, a max bid of 36 matches with both the Diamond multiplier, 9, and the Clubs multiplier, 12. The special soloist declarations are encoded in Hand Game, Ouvert Game, Schneider Announced, and Schwartz announced. Skat encodes the cards placed in the skat by the soloist, and Needs Schneider indicates whether the extra multiplier is needed to win the game. Both of these are specific to the soloist networks. Specific to the defenders are the Winning Current Trick and the Declarer Ouvert features. Winning Current Trick encodes whether the current highest card in the trick was played by the defense partner. Declarer Ouvert represents the soloist's hand if the soloist declared Ouvert. Trump

Table 9: Network input features

Common Features	Width
Player Hand	32
Hand Value	1
Played Cards (Player, Opponent 1&2)	32*3
Lead Cards (Opponent 1&2)	32*2
Sloughed Cards (Opponent 1&2)	32*2
Void Suits (Opponent 1&2)	5*2
Current Trick	32
Trick Value	1
Max Bid Type (Opponent 1&2)	6*2
Soloist Points	1
Defender Points	1
Hand Game	1
Ouvert Game	1
Schneider Announced	1
Schwarz Announced	1

Soloist Only Features	Width
Skat	32
Needs Schneider	1

Defender Only Features	Width
Winning Current Trick	1
Declarer Position	2
Declarer Ouvert	32

Suit/Grand Features	Width
Trump Remaining	32

Suit Only Features	Width
Suit Declaration	4

Table 10: Cardplay train/test set sizes

	Train Size	Train	Test
Phase	(thousands)	Acc.%	Acc.%
Grand Soloist	5,990	79.9	78.0
Grand Defender	10,210	83.3	82.3
Suit Soloist	14,680	78.8	77.6
Suit Defender	26,650	82.9	81.5
Null Soloist	560	82.7	80.4
Null Defender	950	69.2	65.6

Table 11: Cardplay tournament results over 3,000 matches between bots using pre-cardplay policies from the previous section and the learned cardplay policies. All variants were played against the baseline, Kermit (TP/G=tournament points per game, S=soloist percentage)

Player (P)	TP/G(P)	TP/G(K)	diff	S(P)	S(K)	diff
K+C	21.55	24.59	-3.04	31.6	31.6	0.00
AB.85+C	23.36	24.73	-1.37	31.4	33.2	-1.81
AB.925+C	23.39	24.70	-1.31	34.4	31.1	3.34
MLV.85+C	23.86	24.86	-1.01	31.4	33.2	-1.81
MLV.925+C	24.07	24.66	**-0.59**	34.4	31.1	3.34

remaining encodes all the trump cards that the soloist does not possess and have not been played, and is used in the Suit and Grand networks. Suit Declaration indicates which suit is trump based on the soloist's declaration, and is only used in the Suit networks. These features are one-hot encoded, except for Trick Value, Hand Value, Soloist and Defender points, which are floats scaled between 0 and 1. The network has 32 outputs — each corresponding to a given card.

The cardplay network was trained on a subset of the available data. The top 90 players in the dataset (as determined by average TP/G over a minimum of 20,000 games played) were determined, and only data for those players were used. Further experimentation is needed to determine whether this was a beneficial decision. The resultant data set sizes, and accuracies after the final epoch are listed in Table 10. The test set had a size of 10,000 for all networks. The accuracies are quite high, however, this doesn't mean much in isolation as actions can be forced, and the number of reasonable actions is often low in the later tricks.

6 Cardplay Results

Bidding policies from the previous section, as well as Kermit bidding, were used in conjunction with the learned cardplay networks. The cardplay policy (C) takes the legal argmax of the game specific network's output, and plays the corresponding card. The best bidding policies, AB.85 and MLV.85 were tested. AB.925 and MLV.925 were also tested to see the effects of more aggressive bidding within the new context. Each played against Kermit in the same tournament setup from the previous section. Again, all TP/G difference reported were found to be significant. Results are reported in Table 11.

While performing 0.59 TP/G worse than Kermit, the strongest full network player was MLV.925+C. MLV.85 was stronger than MLV.925 in the bidding tournament, so its not immediately clear how these learned policies for pre-cardplay and cardplay interact with each other. Again though, MLV+C was stronger than AB+C at both values of A. Kermit's search based cardplay is quite strong, and it is clearly stronger than the imitation cardplay. One advantage the imitation network cardplay has is that its much faster, taking turn at a constant rate of around 2.5 ms, as compared to Kermit which takes multiple seconds on the first trick, and an average time of around 650 ms (both using a single thread on consumer level CPU).

7 Conclusion

In this paper we have demonstrated that pre-cardplay policies for Skat can be learned from human game data and that it performs much better than Kermit's pre-cardplay — the prior state-of-the-art. Naively imitating all aspects of the pre-cardplay (DI) resulted in a bidding policy that performed an average of 1.35 TP/G worse than the Kermit baseline. Using the novel method to increase the aggressiveness of the bidder led to it performing 2.86 TP/G better than the baseline, with A set to 0.85 (AB.85). Using this in conjunction with game declaration based on the predicted values and probabilities of actions (MLV.85), resulted in the best overall pre-cardplay policy, beating the baseline by 3.65 TP/G. This is a substantial increase in playing strength. Also, the time for pre-cardplay decisions are much faster, as it does not rely on search.

The direct imitation cardplay policy decreases the strength of the overall player, performing 3.04 TP/G worse than the Kermit player when utilizing the Kermit bidder. The best overall full network based player was MLV.925+C, which only performed 0.59 TP/G worse than Kermit. We expect the policy can be tuned to perform better than this, and possibly to be better than Kermit overall. While the learned cardplay is weaker, it is orders of magnitude faster than Kermit's search based methods and doesn't rely on a forward model.

Future Work

Now that we have established some degree of success training model-free policies from human data in Skat, the next logical step is to improve these policies directly through experience similar to the process shown in the original AlphaGo (Silver et al. 2016). In (Srinivasan et al. 2018) regret minimization techniques often used to solve imperfect information games are related to model-free multi-agent reinforcement learning. The resulting actor-critic style agent showed fast convergence to approximate Nash equilibria during self-play in small variants of Poker. Applying the same approach may be difficult because of Skat's size and the fact that it is not zero-sum, but starting with learning a best response to the full imitation player discussed in this work should be feasible and may yield a new state-of-the-art player for all phases of the game. However, a fully-fledged self-play regime for Skat remains as the end goal of this work.

Learning policies though self-play has shown to yield strategies that are "qualitatively different to human play" in other games (Silver et al. 2017). This could be problematic because Skat involves cooperation on defense during the cardplay phase. Human players use conventions and signals to coordinate and give themselves the best chance of defeating the soloist. In order to play well with humans, policies need to account for these signals from their partner and send their own. We plan to explore how continually incorporating labelled data from human games helps alleviate this problem.

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, 265–283.

Brown, N., and Sandholm, T. 2017. Superhuman AI for heads-up no-limit Poker: Libratus beats top professionals. *Science* eaao1733.

Buro, M.; Long, J. R.; Furtak, T.; and Sturtevant, N. R. 2009. Improving state evaluation, inference, and search in trick-based card games. In *IJCAI*, 1407–1413.

Buro, M. 1998. From simple features to sophisticated evaluation functions. In *International Conference on Computers and Games*, 126–145. Springer.

Cowling, P. I.; Powley, E. J.; and Whitehouse, D. 2012. Information set Monte Carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games* 4(2):120–143.

DOSKV. 2018. DOSKV: Deutcher Online Skatverband. `https://www.doskv.de/skat-spielen.htm`.

Frank, I., and Basin, D. 1998. Search in games with incomplete information: A case study using Bridge card play. *Artificial Intelligence* 100(1-2):87–123.

Furtak, T., and Buro, M. 2013. Recursive Monte Carlo search for imperfect information games. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, 1–8. IEEE.

Ginsberg, M. L. 2001. GIB: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research* 14:303–358.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Levy, D. N. 1989. The million pound Bridge program. *Heuristic Programming in Artificial Intelligence*.

Moravčík, M.; Schmid, M.; Burch, N.; Lisỳ, 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.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

Prechelt, L. 1998. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks* 11(4):761–767.

Russell, S. J., and Norvig, P. 2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529(7587):484.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of Go without human knowledge. *Nature* 550(7676):354.

Srinivasan, S.; Lanctot, M.; Zambaldi, V.; Pérolat, J.; Tuyls, K.; Munos, R.; and Bowling, M. 2018. Actor-critic policy optimization in partially observable multiagent environments. *arXiv preprint arXiv:1810.09026*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Sturtevant, N. 2008. An analysis of UCT in multi-player games. *ICGA Journal* 31(4):195–208.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, 1729–1736.