Weighting Information Sets with Siamese Neural Networks in Reconnaissance Blind Chess

Timo Bertram*†
*Dept. of Computer Science
Johannes-Kepler Universität
Linz, Austria
tbertram@faw.jku.at

Johannes Fürnkranz*†
†LIT Artificial Intelligence Lab
Johannes-Kepler Universität
Linz, Austria
juffi@faw.jku.at

Martin Müller
Dept. of Computing Science and Amii
University of Alberta
Edmonton, Canada
mmueller@ualberta.ca

***Abstract*—Research in Game Artificial Intelligence distinguishes between fully observable, perfect-information games and imperfect-information games, which hide part of the game's full information. In games with imperfect information, all possible game states that are consistent with a player's currently available information about the progress of the game are called the information set for that player. This information set can be used for multiple purposes such as determining the expected outcome of a certain move by evaluating it on all possible states in the information set. While in theory there is no way to distinguish states within an information set, players can use experience and other context information to estimate which states are the most likely. In this paper, we estimate a probability distribution over an information set from historic data such that we can assign a weight to each individual state. We achieve this by training a Siamese neural network with triplets of comparisons between different states in the information set given the context of the previously obtained information. A first evaluation in the game of Reconnaissance Blind Chess shows that we can learn to identify the one true game state in a large information set with high probability. In addition, when used within a naively constructed RBC agent, this approach shows promising gameplay performance. At the time of writing, a simple agent based on the Siamese neural network is ranked #6 of all agents on the public RBC leaderboard.**

***Index Terms*—Siamese Neural Networks, Imperfect Information Games, Artificial Intelligence, Reconnaissance Blind Chess**

I. Introduction

Game AI has achieved superhuman performance in many classic fully observable games such as chess [6], Go [17], and backgammon [20], and recently also showed similar results in the imperfect information game of Poker [5]. In imperfect information games, the game state is only partially observable to a player, which makes them an attractive area for artificial intelligence research. There, learning strong game-playing policies is especially difficult because of the uncertainty of the situation. Agents have to account for the lack of information, requiring the construction of probabilistic policies to avoid being predictable and therefore exploitable. In fully observable games, game tree search is a powerful tool, but it is much harder to use in imperfect-information settings [3]. Not even the root state of a search may be known, and defining optimal strategies is much more involved than simply backing up values from leaf nodes. To combat uncertainty and the much larger game tree, game theory for imperfect information games [22] clusters sets of states into *information sets*, sets of positions which one player cannot distinguish from each other given their information. However, being indistinguishable does not imply that all such states are equally likely in practice – good play by both agents will reach some states within an information set much more frequently than others. Furthermore, observing the opponent's past behaviour can give strong clues about the likelihood of their actions. In this work, we employ Siamese neural networks to learn a function from game data which maps an information set $\mathcal{I}$ to a probability distribution over the states in $\mathcal{I}$, which estimates the probability of each state being the true game state. This distribution is then used to inform play by building a simple agent that uses perfect information play on the most likely state with good empirical success.

In the remainder of the paper, we will, after a brief introduction to Reconnaissance Blind Chess (Section II), introduce our approach for using Siamese neural networks and relate it to the literature in that area (Section III). Data preparation and the training process are described in more detail in Sections IV and V, respectively. Finally, the quality of the obtained solution is evaluated in a predictive setting as well as in actual gameplay in a simple but powerful agent in Section VII.

II. Reconnaissance Blind Chess

Reconnaissance Blind Chess (RBC) [11] is an imperfect information variant of classical chess. While the game pieces start in the same arrangement as in classical chess, players do not receive full information about their opponent's moves and are thus generally unaware of the exact configuration of their opponent's pieces. A turn of Reconnaissance Blind Chess consists of four parts:

1) First, the player receives a limited amount of information about the move of the opponent. If the opponent has captured one of the player's pieces, the player is only notified about the square where the capture occurred. For non-capture moves, no information is obtained.

979-8-3503-2277-4/23/$31.00 ©2023 IEEE

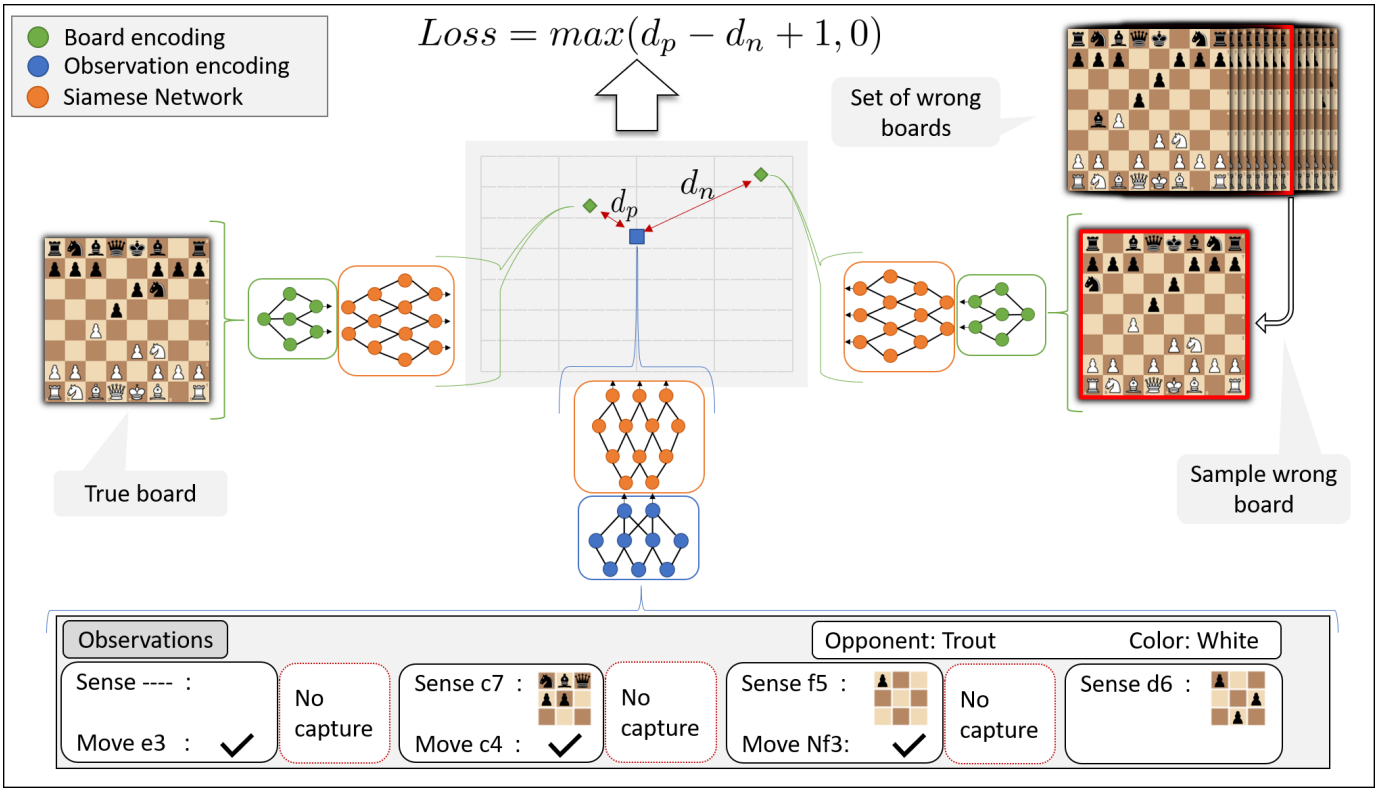

Fig. 1. Schematic overview of using a Siamese neural network for weighting RBC board states in an information set. The observation is preprocessed by an observation encoding network, while both the real board and a sampled incorrect board are preprocessed by a board encoding network. Next, all are input into the Siamese network. The distances of outputs in the embedding space model the probability of a board being true given the observations.

2) The player is allowed to *sense* parts of the board by choosing a 3x3 area, which reveals all pieces in that region. A player *never* receives any information about the opponent's sense.
3) With some minor differences, a player decides on a move as in conventional chess. Due to the missing information, players often cannot tell whether a move is legal.
4) Finally, the player receives information about whether their chosen move succeeded. Some illegal moves, such as when the path of a long-ranging piece is blocked, are truncated, and the player is notified about the true result of their move.

Thus, whenever we speak of one turn of RBC, this includes two separate actions, sensing and moving, which are chosen after another. Due to the imperfect-information nature, some rules of RBC differ from normal chess:

- Players can attempt illegal moves. "If a player tries to move a sliding piece through an opponent's piece, the opponent's piece is captured and the moved piece is stopped where the capture occurred" [21]. Other illegal moves are converted into a pass of the turn.
- Players are not notified if their king is in check. They can also move their king into check and castle through and out of checks. Due to this rule, stalemate, i.e., not being able to make a move without putting one's king into check, does also not apply in this game.
- Games end by capturing the opposing king.

These rules imply the following properties of RBC:

- A player receives enough information to always perfectly know the placement of their own pieces.
- Draws only occur if the maximum number of turns without capture is reached, which makes them rare in practice.
- By remembering all obtained information, it is always possible to compute the information set, i.e. the set of game states that are consistent with the received observations. However, the size of the information set can become large, especially when sensing is ineffective or players play erratically.
- Aggressive strategies that directly attack the opposing king, which would be self-destructive in classical chess, as well as speculative moves become a major factor. Such strategies can win quickly if the opponent does not recognise them.

III. Overview

In this section, we give a short overview of our learning architecture, summarised in Figure 1, and define the used terminology.

In an extensive-form imperfect information game, a game state e captures all information of the current situation, in-

cluding the private information of all players. An information set $\mathcal{I}$ for a player is a set of states $\{e_1, e_2, \ldots, e_{|\mathcal{I}|}\}$ which are indistinguishable from that player's perspective. In our case, this is the set of boards that are consistent with all the information a player has received throughout the game. Formally, the private observation history at player's turn t with $\mathcal{O}_t = (o_0, \ldots, o_t)$ implicitly defines their information set $\mathcal{I}_t$. In Figure 1, the observation history of the current player is depicted at the bottom, consisting of the current position of one's own pieces, the name of the opponent, and the history of previous senses and moves, including information about whether the own moves were successful, and whether pieces were captured by opponent's moves (see Table I). Of the boards in the information set, one is the correct board $p_t \in \mathcal{I}_t$, whereas all other boards $n_{t,i} \in \mathcal{I}_t \backslash \{p_t\}$ do not correctly reflect the true hidden game state. In Figure 1, the correct board p_t is shown on the left, whereas the boards $n_{t,i}$ are shown in the upper right corner.

We aim to estimate the probability of each state in the information set by learning a function $F : \mathcal{I} \to [0, 1]$ which maps each $e_i \in \mathcal{I}$ to the probability $p_i = F(e_i)$ that e_i is the *true state* in the current game. F is trained from past game data including each player's observations as well as the full true game state information at each move. More precisely, for each observation $\mathcal{O}_t$, we can derive $|\mathcal{I}_t| - 1$ *triplets* of the form $\langle \mathcal{O}_t, p_t, n_{t,i} \rangle$, which indicate that in the information set characterised by the observations $\mathcal{O}_t$, the positive example p_t was the true game state, while each of the negative examples $n_{t,i}$ was not. These triplets are used to train a Siamese neural network (shown in the middle of Figure 1), which uses the triplet loss for training. This should result in the positive example p_t being embedded in closer proximity to the observation history $\mathcal{O}_t$ than the negative examples $n_{t,i}$, as described in more detail in the next section.

A. Siamese Neural Networks for Imperfect Information Games

Traditionally, *Siamese neural networks* [4], [7] are used to compare the strength of a relationship of several *options* to an *anchor*. A famous application of this is one-shot learning in image recognition [12], [16]. Here, the network is trained to model that the *positive* image p is more similar to an *anchor* image a than a *negative* image n.

Given triplets $\langle a, p, n \rangle$, comparisons are constructed by feeding all three items $e \in \{a, p, n\}$ to a neural network F, resulting in high-dimensional output embedding vectors $F(e)$. The parameters of F are trained such that distances between these embeddings model the similarity of the inputs. For a triplet $\langle a, p, n \rangle$, given a distance metric d, we define $d_p = d(F(a), F(p))$ and $d_n = d(F(a), F(n))$, omitting the argument a for brevity. The goal is that $d_p < d_n$ as illustrated in the sketch in the centre of Figure 1. This is commonly achieved by minimising the *triplet loss*

$$L_{\text{triplet}}(a, p, n) = \max\left(d_p - d_n + m, 0\right). \tag{1}$$

The parameter m represents the desired margin between positive and negative images – it controls how much the distances must differ to achieve zero loss. As shown at the top of Figure 1, we set $m = 1$ in our experiments. For the distance metric, we chose the Euclidian distance.

As noted above, we apply a Siamese network to model the probability of each specific board state of an information set, using the observation history as the context. Thus, our training triplets are of the form $\langle \mathcal{O}_t, p_t, n_{t,i} \rangle$. Critically, in contrast to previously discussed tasks, in the imperfect information setting there is usually not a single "true answer". Which board actually occurs depends on the (generally unknown) move choice of the opponent. However, an agent can attempt to learn which board states are more likely from historical training data.

A key technical difference between the image recognition setting and our use case is that inputs to the Siamese network usually share a common representation. In our setting, the anchor, i.e., the observation history $\mathcal{O}_t$, encodes a different type of information than the two items that are compared, the positive and negative boards. Therefore, we have to add two small encoding networks, shown in blue and green in Figure 1, which transform the board states and the observation history respectively into matching latent encodings, which can then be used as the inputs for the Siamese network.

B. Related Work

Siamese architectures have been used for tasks other than image recognition. However, when used for game position evaluations, they are often based on comparisons without explicitly modelling the anchor. Tesauro used such pairwise Siamese neural networks to evaluate which of two Backgammon positions should be preferred over the other [19]. In DeepChess [10], chess positions are again compared using a twin Siamese neural network. Both of these applications compare fully observable board states with each other and aim to learn evaluations of the positions. In contrast, in our work, we add a player's observation history as a context, which is required to relate the preference to a specific situation, and fundamentally changes the way the neural network is trained. In our work, it does not make sense to compare arbitrary positions, as we can only decide between states in the same information set. We also do not aim to compare which of the two positions should be evaluated better, but rather we model the likelihood of them occurring in real gameplay.

We may also view each triplet as a contextual preference, denoted as $(p_t \succ n_{t,i} \mid \mathcal{O}_t)$. In this context, our approach may also be seen as a version of *contextual preference ranking* (CPR), which uses Siamese neural networks for preference-based decision-making [1]. In particular, Bertram et al. used CPR to measure the synergy of cards in a collectable card game. There, the Siamese network modelled how well a selection of cards fits a set of previously chosen cards. Here, we extend this idea of relating decisions to the context in which they occurred to the case of observation histories under imperfect information. We apply this method to the game of Reconnaissance Blind Chess (RBC) and compute the probability of each state in an information set of states actually

occurring in play, whereas in [1], CPR is used to predict a player's card choice. Furthermore, while the anchors are rather unique in both applications, the number of different possible choices was limited to only 15 or fewer from a total set of 265 cards, while the size of information sets and possible boards in RBC is much larger.

IV. Data and preparation

A large amount of gameplay data for RBC is openly available [21]. We obtained $582\,450$ games as training data for our network. Each recorded game includes a turn-by-turn list of all observations received from each player's perspective, as well as other information such as the name of the opponent. From this information, we form the anchors $\mathcal{O}_t$, as shown at the bottom of Figure 1. From the game record, it is possible to fully reconstruct the information set for each player and each action. For some players, mostly early versions or malfunctioning ones without a reasonable sensing strategy, the information set can grow too large, so we limit its size to a maximum of $5\,000$ boards, which is rich enough for our purposes. In addition to the information set and the observations, we also extract which board represents the true underlying game state. Together, the *observations*, the *true (positive) board*, and one *wrong (negative) board* form the triplets used for training the neural network (Figure 1). For each game, we create one training sample for each decision point for each player, resulting in a total of 27 million samples. We split this data 90/10 into training and test data in order to compute an out-of-sample accuracy estimate for the final trained neural network. In training, we also take precautions to prevent oversampling decisions with large information sets (see Section V).

A. Representation of Boards and Observations

Each chess position is represented by a $12 \times 8 \times 8$ bit tensor which encodes the occurrences of the 12 different chess pieces. This representation omits some details, such as castling rights and turn numbers, but captures the vast majority of information.

We represent a truncated history of observations as follows: For each turn of the game, a $90 \times 8 \times 8$ bit tensor (Table I) encodes all information received from one player's point of view [2]. The majority of this encoding is taken up by specifying the last-requested move by the player [18]. We truncate the observation history to the 20 most recent turns, padding the input if fewer turns were played and discarding any turns further in the past. Finally, we one-hot encode the 50 most prominent opponent's names in 50 8×8 planes, since this information can have a large influence on the policy of an agent [8]. If the opponent's name is not in this list of players, all 50 planes are set to zero. Thus, the total observation representation is $20 \times 90 + 50 = 1850$ bitboards of size 8×8.

TABLE I
Encoding of the observations in one turn

Number of planes	Information represented
1	Square where the opponent captured one of our pieces
73	Last move taken, encoded as in AlphaZero [18]
1	Square where agent captured opponent's piece
1	1 if the last move was illegal
6	Position of own pieces (One plane per piece type)
1	Last sense taken
6	Result of last sense (One plane per piece type)
1	Color

B. Neural Network Architecture

To translate the two different representations of boards and observation histories into a common input to the Siamese network, we use two small convolutional neural networks that differ in input but share the same output format. In training, the boards and observations are transformed using their respective encoding network (blue and green in Figure 1) before reaching the common Siamese neural network (orange).

1) Encoding Networks: The two encoding networks are basic convolutional neural networks with 5 layers, 64 filters per layer, and the ELU activation function [9]. They only differ in the shape of the input, but share the same inner structure and output. Each encoding network translates its input into a feature tensor of shape of $128 \times 8 \times 8$.

2) Siamese Network: The Siamese neural network is another convolutional neural network. It is larger than the encoding networks with 10 convolutional layers and 128 filters each, also uses ELU activations, but additionally utilises skip connections. The output block of the network uses two more convolutional layers but decreases the filter size from 3×3 to 1×1 to combine the different feature planes into one final output. A single fully-connected $\tanh$ layer forms the final output of the network. We tested several architectures with more fully-connected layers, but predominantly using convolutions worked much better. The dimensionality of the resulting embedding space, i.e., the number of output neurons of the final fully-connected layer, was set to 512 in order to capture the complex relationships between the samples in the triplets.

V. Training process

The neural network is trained on the dataset described in Section IV-A using mini-batches of 1024 triplets with a learning rate of 0.0001. Larger learning rates significantly harmed the training process. For gradient updates, we use the AdamW algorithm [13]. The combined network, consisting of both encoding networks and the Siamese network was trained end-to-end, driving the encoding networks towards latent representations that are most useful for the Siamese network.

Our training procedure differs significantly from training Siamese networks for image recognition. When training with triplets of images, it is possible to choose arbitrary combinations of images, as long as one can be considered more similar to the anchor than the other. For our task, triplets are

more restricted. The preference of the positive board over all the negative boards in its information set is only valid for the specific history in which this decision occurred. In our data, each anchor is associated with exactly one information set, including one true board and numerous negatives. In order to avoid oversampling of triplets from large information sets, which would often occur with weaker agents that did not choose senses that kept the information set small, we define one epoch of training as follows:

- One *epoch* consists of seeing each anchor and positive example exactly once.
- For each of those pairs, one single negative board is sampled from all possible options. While it is possible to use uniform sampling, this can easily lead to generating uninformative triplets with unrealistic negative boards. Instead, we choose triplets in a way that is related to semi-hard triplets [16]: x negatives are randomly sampled from the set and their distances to the anchor are computed. For the computation of the loss and backpropagation, only the negative in closest proximity to the anchor, i.e. the one regarded most likely, is used. At the start of training, x is set to 3 and it is increased after epochs where improvements stall.

The Siamese neural network, evaluated in Section VII, was trained until the evaluation loss stalled for 3 consecutive epochs. In total, this process trained for 22 epochs, which took 61 hours on a single Nvidia A100 GPU. Our code is publicly available.[1]

VI. A Minimal Siamese RBC Agent

While we can evaluate the quality of the trained neural network with conventional machine learning metrics (and we will do so in Section VII-A), it is not clear how much a real agent that plays Reconnaissance Blind Chess can profit from improved information set probability estimation. For this, we implemented and evaluated a minimalist RBC agent that strongly utilises the Siamese neural network to aid sense and move selection.

Building an RBC agent requires three main components: (i) handling information received throughout the game, (ii) choosing a sensing action, and (iii) selecting the move to be played. Our agent is built around the idea of tracking the information set of board states, starting with the initial piece configuration and adding and removing states based on the received information. The trained Siamese network is heavily used for both sensing and moving. When sensing, the agent aims to minimise a weighted measure of the information set size, which is achieved by computing the weighted number of board conflicts per sensing square (see Section VI-B). To choose a move, the agent computes the most likely board with the Siamese network and selects a move on that specific board using the strong open-source classical chess program Stockfish[2] (see Section VI-C). Thus, the agent naively plays under the assumption that the Siamese network can always identify the true board state.

[1]https://github.com/timobertram/Weighting-Information-Sets-with-Siamese-Neural-Networks-in-Reconnaissance-Blind-Chess

[2]https://stockfishchess.org/

Algorithm 1: Compute sense score of each square

```
Data: boards, weights
Result: scores
scores ← list();
end ← min(100, len(boards));
for square ← 1 to 64 do
    diffResults ← list();
    for i ← 0 to end do
        res ← senseResult(boards[i],square);
        if res ∈ diffResults then
            diffResults[res] ←
              diffResults[res] + weights[i];
        else
            diffResults[res] ← weights[i];
        end
    end
    scores[square] ← 0;
    s ← sum(diffResults)
    for res ∈ diffResults do
        scores[square] ←
          scores[square] + (res/s) · (s − res);
    end
end
```

A. Handling Information Sets

The majority of strong RBC agents are based on tracking all possible board states, i.e. the information set $\mathcal{I}$ [15]. For computing its probabilities, our agent also needs to track $\mathcal{I}$. After each opponent's turn, the set of possible states is replaced by all possible states that could follow each of the boards in the previous set. If the opponent captured a piece, this set typically stalls or decreases in size, as only a limited number of previous states allow capturing pieces, while non-capture moves greatly increase its size. Whenever the agent itself senses or moves, it removes boards that are inconsistent with the new observations. Sensing and moving never directly increase the size of the set, and most of the time significantly decrease it. Nevertheless, if sensing does not remove enough states, the information set can grow very large over time.

B. Choosing a Sensing Action

Most strong RBC agents sense in a way that aims to mainly reduce the information set size [15], and our agent follows this practice. To sense, first, a probability distribution over the whole information set is estimated from the distances in the embedding space created by the Siamese network. The distances of the embedded boards to the embedded history are computed, inverted and normalised to sum to one. This converts them into a probability distribution where boards in closer proximity to the history have a higher weight. Based on this, the agent computes a score for each of the 64 possible

sensing locations.[3] The score estimates the expected number of board state eliminations based on conflicts of possible sensing results, weighted by the probability of each of the top 100 likeliest boards (see Algorithm 1). Finally, the square with the highest score is chosen as the centre of the sense.

C. Choosing a Move

While the Siamese network influences the agent's sensing through board weighting, its move selection is influenced even further. After receiving new information from the previous sense, the agent appends this observation to its history, which shifts the anchor in the embedding space. Then, the agent again computes a probability distribution over the remaining boards in the information set. Next, in the simplest and most optimistic way, the agent chooses the most likely board and queries Stockfish for the best move on that board.[4]

The success of this policy greatly depends on the accuracy of the Siamese network. If it does not provide accurate predictions, the agent will sense poorly as well as make inaccurate moves that only work well on the wrong board. However, whenever it correctly identifies the board, the agent plays as if it had perfect information. We make two observations about this agent's optimistic behaviour:

1) The agent ignores potentially dangerous boards, that could lead to immediate defeat if they are not regarded as the most likely one.
2) In contrast to most other agents, our agent has no concept of "cautious" moves that perform adequately on many boards. Instead, it takes gambles based on guessing the correct board.

While this policy is naive, our evaluations show that this basic agent, which leaves vast room for improvement, already achieves top-tier playing strength in RBC (see Section VII-B).

VII. Evaluation

In order to evaluate the proposed approach, we test the performance of the Siamese network in two ways: by its accuracy of predictions on a held-out test set, and by the playing strength of the simple Siamese RBC agent outlined in Section VI.

A. Board Ranking Accuracy of the Siamese Network

The experiments in this section evaluate the trained Siamese network and its predictions against the ground truth data. This evaluation provides a fine-grained test of how well the network selects boards from a set. Together with the game-play evaluations in Section VII-B, these results provide a strong validation of our approach.

For each game position in the test set, we embed all states in the information set as well as the observation history with the Siamese network (see Figure 1). In the embedding space, we rank all states based on their distances to the observations and check the position of the true board in the ranking.

We report three metrics:

- top-k accuracy: how often is the true board ranked among the top k boards?
- top-k-percentage accuracy: how often is the true board ranked in the top k-percent boards (rounded up)? This is similar to top-k accuracy but also takes into account the variable size of the information set.
- pick-distance: the position of the true board in the ranking of all boards in the information set.

All metrics are averaged across a large number of samples from the unseen test set. As such, they should provide a strong estimate of the generalisation performance of the neural network.

We compare our results to three baseline rankings: random ranking, ranking by the evaluations of the boards given by Stockfish, and ranking by using the internal evaluations of StrangeFish2. Ranking positions with Stockfish, or other classical chess engines, is used in many RBC agents [11], [15], and assumes that the opponent is more likely to play moves that have a higher evaluation. While evaluations from normal chess cannot directly translate to RBC evaluations and often give vastly differing results, most of the current strong RBC players use a chess engine, making this comparison more relevant than it may seem at first glance. One of the strongest agents, StrangeFish2 [14] is one of such players that bases its evaluations on Stockfish, but makes several RBC-specific adaptations.

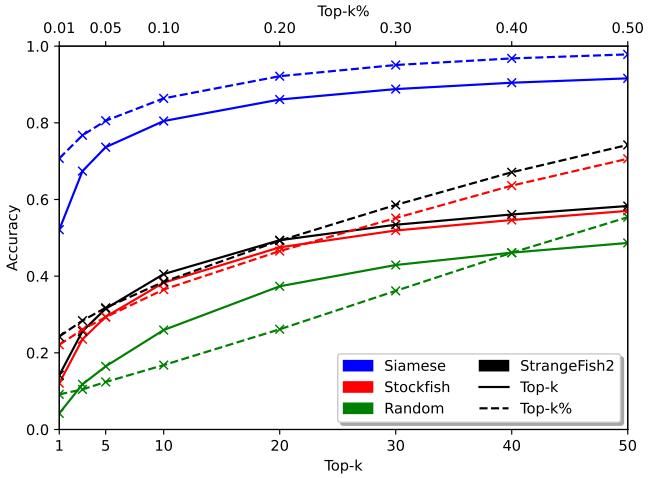

Fig. 2. Comparison of top-k and top-k-percent accuracy of the Siamese network to random, Stockfish, and StrangeFish2 ranking. The average size of the information set is 1100. The Siamese network vastly outperforms the three baselines, achieving much higher accuracies than all other metrics.

Figure 2 shows the top-k (solid) and top-k-percent (dashed) accuracy of the four methods. Ranking boards based on their Stockfish evaluations achieves higher accuracies than random ranking, with the additional adaptions of StrangeFish2 leading to slight improvements over Stockfish. However, our Siamese

[3] In practice, the 28 border squares are ignored because the same (and more) information can be obtained by sensing an adjacent interior square.

[4] We make some RBC-specific adaptions such as capturing the opponent's king if possible.

network beats the three baselines by a large margin. In more than half of the samples (52%), the true board is ranked first, so the network is able to correctly identify the correct choice more often than not. In addition, with a top-50-percent accuracy of 98%, the true board is ranked in the bottom half of options only in a few exceptional cases. Also note that under the assumption that Stockfish is able to select the best move, this move will not only be played when the true board has been correctly identified by our network, but may also be the best move on an incorrectly selected board. In our experiments, there were an additional 16% of such cases.

TABLE II
TOP-1 BOARD PREDICTION ACCURACY FOR DIFFERENT OPPONENTS. OPPONENTS SORTED BY CURRENT ELO

Opponent	Top-1 accuracy	Opponent	Top-1 accuracy
StrangeFish2	0.39	Stockenstein	0.29
Fianchetto	0.56	Testudo	0.38
JKU-CODA	0.41	genetic	0.25
Châteaux	0.45	Marmot	0.34
Kevin	0.54	Dyn. Entropy	0.35
ROOKie	0.44	trout	0.67
StrangeFish	0.46	attacker	0.93
Oracle	0.50	random	0.28
penumbra	0.43	No player	0.39

Table II breaks up the overall top-1 accuracy according to opponents. This is important because the opponent is encoded in the observation history, and therefore influences the embedding. As a result, different boards may be selected for the same information sets if opponents differ. Here, we can see that the accuracy of the network is not dominated by single individual players. Nevertheless, the predictability of move choices varies between different players. Intuitively, *random* is one of the least predictable opponents, but the Siamese network is able to learn that this player often makes illegal moves, still allowing for some modelling of how it will act. On the other end, boards in games against the strongly scripted *attacker* are easiest to identify. Altogether, the Siamese network can model all opponents to some degree, including those with the highest Elo ratings, *Strangefish2*, *Fianchetto*, *JKU-CODA*, and *Châteaux*. This is especially reassuring given that the games of these players result from dozens of different versions of different strengths that all play under the same name. Finally, we observe that the left column with higher-rated opponents seems to be more consistent in predictability than the right column with lower-strength players.

Figure 3 provides more insights into the performance of the network by summarising the individual pick-distances for a large number of samples. The Siamese ranking is clearly better at modelling player behaviour than the three baselines. In the majority of cases, the pick distance is below 10, and only about 5% of samples have a distance over 100. For the method that many other agents use, namely ranking or weighting by Stockfish, many samples have much higher distances, and

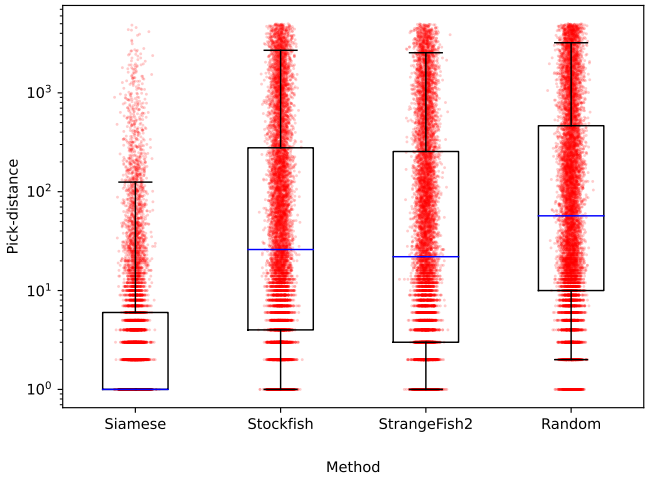

Fig. 3. Comparison of pick-distance of the true board, using the Siamese network, Stockfish, StrangeFish2, and random ranking. Individual samples per method have small uniform noise on the x-axis added for better visualisation. The Siamese network has some amount of outliers due to the stochastic nature of the task, but is generally able to achieve a very high ranking of the correct choice and a median rank of 1.

StrangeFish2 only improves this slightly.

B. Playing performance

In this section, we evaluate the agent described in Section VI on the publicly available RBC leaderboard [21], which allows for comparison of its strength against a variety of players, including many of the strongest known agents.

TABLE III
ABLATION STUDY OF COMPONENTS OF PROPOSED RBC AGENT. ELO IS OBTAINED FROM THE RBC LEADERBOARD.

Sense weighting			Board selection			ELO
Uniform	Stockfish	Siamese	Random	Stockfish	Siamese	
✔			✔			1197
		✔	✔			1253
	✔			✔		1204
		✔		✔		1419
		✔			✔	1534

The RBC leaderboard is a freely available service which provides automated testing against all other currently active agents. While active players change, a small selection of agents is always connected, whose strength varies from rather basic to state-of-the-art. This ensures that players can be assigned a representative Elo rating, albeit with some variance.

Our test on the leaderboard involves two parts; evaluating the full *Siamese Optimist* agent, and ablation studies where the sense weighting or board selection is replaced by a baseline method to separately test their effect on the player rating. The results of this are shown in Table III. In all versions, sensing uses Algorithm 1, only differing in how the weights are computed. For *Uniform*, all weights are equal, for

Stockfish, all boards are evaluated by Stockfish and weights are generated proportionally to how high that evaluation is from the opponent's perspective, and for *Siamese*, boards are weighted in inverse proportion to the distance of the board to the observation history $\mathcal{O}_t$ in the embedding space. For playing, all agents use Stockfish to compute the best move but differ in the board selection strategy: *Random* chooses a board randomly, *Stockfish* chooses the board with the highest Stockfish evaluation from the opponent's view and *Siamese* uses the learned embedding to choose the board that is closest to the observation history.

The resulting Elo ratings of each version (Table III) match the observations from Section VII-A. In both parts of the player, replacing the Siamese network with other options results in much worse performance, especially when changing the board selection. Selecting a board randomly results in very weak players for all weighting methods. Selecting based on Stockfish's evaluations only performs well when using the Siamese sense-weighting, but the fully Siamese player is still more than 100 ELO points stronger than that version. At the time of writing, the fully Siamese version *SiameseOptimist* is ranked #6 on the public leaderboard.

VIII. Conclusion and future work

We trained a Siamese network to estimate situation-specific occurrence probabilities of states in an information set. Our experiments confirmed that the learned network is able to perfectly identify the true state in a large number of cases, even when the information set is large. In addition, we show that this allows for a simple perfect-information approach to the imperfect-information game of Reconnaissance Blind Chess. The agent naively assumes that the network can always identify the correct state in the information set, resulting in only a single game position to consider, and uses the chess engine Stockfish to generate a strong move for that position. While limited, this makes the game much easier to play for an agent and shows surprisingly strong performance, such that our agent is currently ranked among the strongest on the public leaderboard. The surprising strength of perfect information gameplay in imperfect information games was recently also observed in [3]. Integrating our weighting of positions in the information set with their Monte-Carlo search for evaluating these positions seems to be an obvious and promising direction for future work.

Despite this, we consider the fact that moves are still selected using Stockfish to be the main weakness of our current approach. In particular, RBC-specific policies, such as strategies using speculative attacks on the opponent's king, can not be generated in this setting. Thus, changing the final move selection, for example, to an RBC-specific neural network, or conducting a search with our learned probability distributions, may further improve our agent.

Although only tested in one domain so far, our approach is general, allowing its future use in other imperfect-information decision-making tasks, which we also intend to explore.

Acknowledgments

Müller acknowledges funding from the Canada CIFAR AI Chair program and NSERC, the Natural Sciences and Engineering Research Council of Canada.

References

[1] T. Bertram, J. Fürnkranz, and M. Müller, "Predicting human card selection in Magic: The Gathering with contextual preference ranking," in *2021 IEEE Conference on Games (CoG)*. IEEE, 2021, pp. 1–8.

[2] ——, "Supervised and reinforcement learning from observations in reconnaissance blind chess," in *2022 IEEE Conference on Games (CoG)*. IEEE, 2022, pp. 608–611.

[3] J. Blüml, J. Czech, and K. Kersting, "AlphaZe**: AlphaZero-like baselines for imperfect information games are surprisingly strong," *Frontiers in Artificial Intelligence*, vol. 6, no. 1014561, 2023.

[4] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "Siamese" time delay neural network," *Advances in Neural Information Processing Systems*, vol. 6, 1993.

[5] N. Brown and T. Sandholm, "Superhuman AI for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, 2019.

[6] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu, "Deep blue," *Artificial intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.

[7] D. Chicco, "Siamese neural networks: An overview," *Artificial neural networks*, pp. 73–94, 2021.

[8] G. Clark, "Deep synoptic Monte-Carlo planning in reconnaissance blind chess," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4106–4119, 2021.

[9] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *Proc. 4th International Conference on Learning Representations (ICLR)*, 2016.

[10] O. E. David, N. S. Netanyahu, and L. Wolf, "Deepchess: End-to-end deep neural network for automatic learning in chess," in *Proceedings of the 25th International Conference on Artificial Neural Networks and Machine Learning (ICANN), Part II*. Springer, 2016, pp. 88–96.

[11] R. W. Gardner, C. Lowman, C. Richardson, A. J. Llorens, J. Markowitz, N. Drenkow, A. Newman, G. Clark, G. Perrotta *et al.*, "The first international competition in machine reconnaissance blind chess," in *NeurIPS 2019 Competition and Demonstration Track*. PMLR, 2020, pp. 121–130.

[12] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *Proceedings of the ICML Deep Learning Workshop*, Lille, France, 2015.

[13] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019.

[14] G. Perrotta, "StrangeFish2," Aug. 2022. [Online]. Available: https://github.com/ginoperrotta/reconchess-strangefish2

[15] G. Perrotta, R. W. Gardner, C. Lowman, M. Taufeeque, N. Tongia, S. Kalyanakrishnan, G. Clark, K. Wang, E. Rothberg, B. P. Garrison *et al.*, "The second NeurIPS tournament of reconnaissance blind chess," in *NeurIPS 2021 Competitions and Demonstrations Track*. PMLR, 2022, pp. 53–65.

[16] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[17] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[18] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[19] G. Tesauro, "Connectionist learning of expert preferences by comparison training," *Advances in Neural Information Processing Systems*, vol. 1, 1988.

[20] ——, "TD-Gammon, a self-teaching backgammon program, achieves master-level play," *Neural Computation*, vol. 6, no. 2, pp. 215–219, 1994.

[21] The Johns Hopkins University Applied Physics Laboratory LLC. (2018) Reconnaissance blind chess. [Online]. Available: https://rbc.jhuapl.edu

[22] J. Von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, 2nd ed. Princeton University Press, 1947.