Towards Embodied StarCraft II Winner Prediction

Vanessa Volz[1], Mike Preuss[2], and Mathias Kirk Bonde[3]

[1] TU Dortmund University, Germany
[2] Westfälische Wilhelms-Universität Münster, Germany
[3] University of Copenhagen, Denmark

Abstract. Realtime strategy games (and especially StarCraft II) are currently becoming the 'next big thing' in Game AI, as building human competitive bots for complex games is still not possible. However, the abundance of existing game data makes StarCraft II an ideal testbed for machine learning. We attempt to use this for establishing winner predictors that in strong contrast to existing methods rely on partial information available to one player only. Such predictors can be made available to human players as a supportive AI component, but they can more importantly be used as state evaluations in order to inform strategic planning for a bot. We show that it is actually possible to reach accuracies near to the ones reported for full information with relatively simple techniques. Next to performance, we also look at the interpretability of the models that may be valuable for supporting human players as well as bot creators.

Keywords: StarCraft II · outcome prediction · interpretability · embodied gamestate evaluation

1 Introduction

Being able to evaluate a game state in terms of imbalance towards (advantage for) one player is crucial to many types of generally applicable game-playing algorithms, such as Monte-Carlo tree search (MCTS)[7]. Heuristics based on expert knowledge are often used as an evaluation function (see survey in [20]), but depending on the game in question, such a measure is difficult to construct and verify. It is therefore beneficial to be able to learn game characteristics from replay data instead, as this approach avoids human bias and is also generalisable to other games, an important topic in current research [14]. One possible binary (component of an) evaluation function of a given game state is whether the player is likely to win or to lose. In this paper, we thus investigate embodied winner prediction, i.e. only using information available to a single player (or bot).

In fact, such information may also be helpful to players because the data driven heuristic may be more accurate than their own estimation of the current state of the game, or be accurate in cases that are difficult to decide for a human. We will address the differences between the (expert) human and machine learned judgment of the overall game state.

Being able to predict the winner of a game is also important in the context of dynamic difficulty adaptation [11] and in context of AI-assisted game design, e.g. when evaluating the decidedness or tension of a game [6]. Furthermore, the construction of such a predictor can improve the understanding of a game's characteristics and act as a basis for developing new strategies for human and AI players. The characteristics could even be tracked over time in order to obtain knowledge about the meta of the game.

The identification of game characteristics is especially important since AIs in commercial games typically rely on domain knowledge, because more general approaches would not fulfill resource or believability constraints.[4]. To do that, the predictor should be interpretable as well. This has the added benefit of facilitating the detection of overfitting issues.

In this paper, we thus investigate the feasibility of constructing a model for winner prediction for a complex AAA game that is both (1) embodied and (2) interpretable. To this end, we test different strategies, such as incorporating data from multiple ticks and vary feature sets and models. We choose to conduct our study on StarCraft II, as related AIs are a topic of active research [21] and we can assume that winner prediction is a challenging task. Additionally, as of recently, suitable data is available [22] in high quantity. The fact that the subject of winner prediction in StarCraft (II) has been approached previously (cf. related work Section 2.2) also demonstrates a high interest. However, to our knowledge, there are no winner prediction methods that do not assume full observability of all features and at the same time are interpretable and tested on human vs. human replays (cf. section 2.2). We aim to fill this gap with the work detailed in this paper. Due to the lack of a single consistent definition of interpretability, we instead anecdotally investigate it using the opinions and understanding of an expert StarCraft II player.

In the following section, we first briefly introduce the StarCraft series and then review the state-of-the-art of winner prediction on RTS games. Based on our literature overview, we formulate several hypotheses to test in this paper. To this end, we first describe our experimental setup in Section 3, including our data collection and preprocessing approach, as well as the prediction models we tested. We describe the results of our experiments and hypothesis tests in Section 4. Afterwards, we analyse the interpretability of the obtained models. We conclude our paper in Section 5 with a summary and possible directions of future work.

2 Background and Related Work

2.1 StarCraft Series

Game Description StarCraft II[5] is a popular real-time strategy (RTS) game with a science-fiction theme released by Blizzard in 2010, which was followed

[4] Duygu Cakmak: *The grand strategy approach to AI* at Emotech Meet AI 9 (London), March 22nd 2018

[5] https://starcraft2.com

up with further expansion packs in 2013, 2015, and 2016. It is the second game in the series, the first StarCraft game was published in 1998. StarCraft II was designed as an E-Sport[5] and has a massive following, regular tournaments (e.g. World Championship Series) and professional players.

StarCraft II features three playable races (Terran, Protoss, Zerg) and several game modes (1v1, 2v2, 3v3, 4v4 and campaign). Each player spawns with a town-hall building and a small number of workers at a predetermined location on a map, their base. The players can construct additional buildings, which can be used to produce more workers and military units. The properties of the available buildings and units are determined by the race played. There are additional upgrade mechanisms available to buildings as well as units. Buildings, units, and upgrades require different amounts of minerals and vespene gas, the two resources in the game. Both can be gathered by worker units. The supply value, which may be increased by additional buildings, poses a limit to the number of units that can be built by a player.

The player that successfully destroys all their opponent's buildings has won the game. The game also ends if a player concedes or if a stalemate is detected by the game.

StarCraft as Research Environment The first game version, and specifically its expansion pack *StarCraft: Brood War*, have been used in research as a benchmark and competition framework[6] for AI agents since 2009 [21]. In 2017, DeepMind and Blizzard published the *StarCraft II Learning Environment (SC2LE)* [22]. The SC2LE provides an interface for AI agents to interact with a multi-platform version of StarCraft II and supports the analysis of previously recorded games.

Specifically, the SC2LE offers an interface through which a large set of game state observations[7] can be made available for every game tick in a replay or in real-time. The information that can be obtained includes raw data on features such as unit health and unit type in the form of heatmaps. At the same time, it also includes aggregated information that is usually displayed to game observers that can help to characterise a player's progress. Examples include the resource collection rate and the number of units destroyed represented as their value in resources. The SC2LE consists of multiple sub-projects, which include, among other things, a python wrapper library `pysc2`[8] used in this paper.

Even before releasing SC2LE, Blizzard has been allowing players to save their own StarCraft II games to a file using the `.S2Replay` format. These replays can then be watched using the StarCraft II software and even analysed using the *S2 Protocol* published by Blizzard. `s2protocol`[9] is a Python library that provides a standalone tool to read information from `.S2Replay` files. The files contain repositories with different information. The `metadata` repository, for example,

[6] http://bwapi.github.io/
[7] https://github.com/deepmind/pysc2/blob/master/docs/environment.md
[8] https://github.com/deepmind/pysc2
[9] https://github.com/Blizzard/s2protocol

contains general information on the game and the players, such as the result, the selected races, the game map, and the duration of the game as well as technical details such as the StarCraft II build number. More details on the recorded features, their interpretation, and how they are used in this paper can be found in Section 3.

2.2 Outcome Prediction

With the release of SC2LE, the authors of the corresponding paper [22] also presented several use cases for their research environment, among them predicting the outcome of a game given a game state. They test three different neural networks to predict game outcome based only on the observations a player makes at a given time. The two best performing models both contain two convolutional networks with 2 layers and 16 and 32 filters, respectively. To represent the game state, they use spatial features gathered from the minimap, as well as other measures observable by the player, such as the supply cap. With this setup, they are able to reach a prediction accuracy of about 60% after 8 minutes of game time and approach 70% after 20 minutes.

However, due to the complexity of the models, as well as the applications of learned spatial features, it is very difficult to interpret the resulting networks. While not absolutely necessary, interpretability is a nice feature both for understanding game characteristics, as well as biasing AIs with the discovered patterns. More importantly, more complex models always are more prone to overfitting. In this paper we therefore investigate how better interpretable and smaller models perform in comparison.

Additionally, the authors of [22] recognise that training a predictor not only on information from the current gametick, but also including previous ticks in the input, would likely increase prediction accuracy. We investigate this hypothesis in this work by integrating different amounts of historic data as well as different aggregation methods.

Lastly, the input that is used in the paper heavily depends on the exploratory behaviour of the players. It is unclear how different scouting behaviours and the resulting difference in information might influence the predictor. We thus chose to focus on directly interpretable measures that are not influenced by information gathering methods. However, it would be interesting to investigate this further and compare it to predictors trained on fully observable game states. In this case, we decided to investigate the baseline, i.e. no opponent information, instead.

Besides the paper described above, there are a number of related publications on outcome predictors for StarCraft: Brood War, some of which are also used within AI agents. In [15], for example, the authors predict the winner of StarCraft matches using several time-dependent and -independent features collected from human vs. human replays from the dataset described in [16]. The authors of [15] stress the fact that, in contrast to many other publications (such as [17, 18]), they work on data that was not artificially generated or collected from simplified StarCraft versions. Additionally, while similar work had previously only considered Protoss vs. Protoss match-ups using the same dataset [8], in

[15] the authors investigate how separate models for each race match-up can be combined. Both publications use features that express the difference between the two players at a given time, e.g. in terms of number of units, as well as a measure based on random rollouts similar to LDT2 (life time damage a unit can inflict) as proposed in [10]. In [8], the authors reach an average prediction accuracy of 58.72% for 5-10 minutes of observed gameplay and 72.59% after 15 minutes (Protoss vs. Protoss). In [15], the best accuracy reported for the separate model on Terran vs. Terran matches is 63.5%, but it was only trained on 298 matches.

In this work, we also use human vs. human replays which we processed via the SC2LE framework. We were however able to collect a larger dataset. Additionally, we avoid using rollout-based measures, so that the model is computationally efficient and can be used within a real-time SC2 agent to evaluate possible game states. For this purpose, in contrast to the existing literature, we also do not assume a fully observable game state. We however investigate how this limitation affects model performance.

A third, more recent paper using the dataset from [16] again, investigates how early winner predictions can be made [2]. It also assumes fully observable game states, but they abstain from using simulation-based measures and do not aggregate the obtained features. They are still able to achieve high prediction accuracies of over 80% after only 5 minutes with a KNN model. We therefore include a KNN in our comparison as well.

In addition, other types of predictions were tackled in the context of StarCraft and StarCraft II, such as army combat outcomes [19], the opponent's technology tree [1] and a player's league [3]. Winner prediction was tackled in other games as well, for example in MOBAs [23]. However, as we do not intend to develop new prediction techniques in this paper, we do not give a detailed overview of related work for other games.

2.3 Research Questions

Based on the limitations and future work directions described in the papers listed in the previous section, we formulate several research questions regarding (A) prediction accuracy and (B) interpretability of trained models.

A1 *Are specific models particularly (un-)suitable for this problem?*
Previous studies introduce several model types. How effective are they in our problem setting?

A2 *How important is opponent information?*
Studies such as [2] obtain high accuracies with full observability. Are these results transferable to our problem setting?

A3 *How important is information on game progress?*
[22] suggests it should improve accuracy. Is that true?

B1 *Are resulting models interpretable by (expert) human players?*
Can these models inform game designers and players?

B2 *Can (expert) human players identify indicators for outcome?*
Some features in related work [10] incorporate expert information. How does this effect performance? How best to integrate it?

3 Experimental setup

In Section 4, we will present the results of several experiments designed to investigate the previously stated research questions. In the following, we describe our experimental setup. All experiments rely on training machine learning classifiers using supervised learning. We describe the models used in Section 3.4, as well as how we collect and preprocess the data required for training (see Sections 3.1 and 3.2, respectively). In Section 3.3, we characterise the obtained data in more detail, specifically by discussing the obtained features, to facilitate interpretation of the results.

The interpretation and subsumption of results is done by an expert StarCraft II Terran player. He competed in the European Grandmaster league (16/17) and analyses the success of different play styles in his function as game coach.[10]. Whenever we refer to an expert opinion in the following, we describe the opinion of this particular player.

3.1 Data collection

The data utilized in our experiments was extracted from replays randomly selected from the dataset published by Blizzard together with the SC2LE [22]. In order to streamline our analysis, we chose to focus on Terran vs. Terran matchups exclusively (TvT in the following). This has the added benefit of being our expert's main area of expertise.

For each selected replay, we recorded its metadata using the s2protocol library (see 2.1). In order to obtain information on the progression of the game, we then added details on the players' progress with a modified replay.py script from the pysc2 library. The script outputs all available observations (cf. Section 3.3) every 224 gameticks, which translates to about 10 seconds of gameplay.

Using this method, we were able to collect information on 4899 TvT games. We then processed the collected information with `R`, thus generating datasets characterising the players' progress for each recorded gametick, i.e. every 10 seconds starting from 10 seconds. The longest game we recorded ended after 1:53:30h. Additionally, we also compiled a dataset containing information of all games at their respective last gametick.

3.2 Preprocessing

A detailed look at the collected data made the need for preprocessing obvious. Since the dataset is comprised of real-world data, it contains a considerable amount of instances where players left and thus lost due to reasons not reflected in their game progress. For example, after watching some replays, we observed examples where players were idle when the game started and others, where players lost on purpose by destroying their own buildings. Since this behaviour is arguably not explainable by the collected data and would bias the trained

[10] `https://www.gamersensei.com/senseis/mperorm`

models, we sought to remove the corresponding replays from the datasets. Based on discussion with the expert player, we thus remove games

1. in which at least one player performed 0 actions per minute,
2. that lasted 30 seconds or less,
3. where at least one player spent less than 50 minerals and already destroyed one of their own buildings.

Condition (1) removes games where at least one player was not active. (2) removes games where one player was not active, returned late to their PC and conceded before enemy contact was possible. (3) removes games where a player self-destructs intentionally, as friendly fire is not a sensible strategy until later in the game. The latter behaviour might be observed if players intentionally want to lower their player rating.

Within this preprocessing step, we removed 870 games, leaving us with 4029. Of the remaining games, only 17 were a tie, i.e. less than 0.5%. We therefore also removed all tied games, as they are highly uncharacteristic and our dataset does not contain enough examples to train a predictor. After preprocessing, our dataset contains data on both players of 4012 TvT replays.

3.3 Data Characterisation

From the recorded observations, we excluded features that contain no information due to game logic-based reasons (i.e. information on upgrades, shields, energy and warp gates, but also race) as well as such features that would not be available during a game, such as player performance statistics (`apm` and `mmr`) and game duration. Due to our choice of models, we also removed all discrete features (`map name`). Furthermore, we removed features that did not vary (`base build`, `score type`). The remaining features are listed in the following, presented as Cartesian products, i.e. $\{A\}x\{B\} = \{(a, b)|a \in A, b \in B\}$.

```
gameloop, army count, score
{∅, collected, collection rate, spent} x {minerals, vespene}
{friendly fire, lost, killed, used, total used} x {minerals, vespene}
   x {none, army, economy, technology}
{total value, killed value} x {structures, units}
{total} x {damage dealt, damage taken, healed} x {life}
{food} x {workers, used army, used economy, used}
{idle} x {production time, worker time, worker count}
```

They mainly contain information on resource collection, usage and destruction, as well as units and structures, food supply, idle times, and damage. The features are similar to those presented in the game observation screen and represent a player's progress in different areas of the game (e.g. army vs. economy). Other features are intended to reflect a player's efficiency (e.g. information on idleness). The features also include some information on the consequences of the opponent's actions (e.g. damage taken, resources lost). They do not, however, include spatial data, nor any reflection on observations made by scouting. As such, they are all easily available to a player or bot within a game.

3.4 Prediction methods

Basically, we treat winner prediction as a binary classification problem, and there are of course many available methods for this, as it is one of the two fundamental activities in machine learning (the other one is regression). However, we restrict ourselves to four well-known algorithms here. One could, e.g., add Support Vector Machines or other methods, but we do not expect very different results. We have performed some manual parameter testing to configure the algorithms. However, performance differences due to parametrisation seem to be small in all cases.

Decision Trees Recursive partitioning [4] produces (usually binary) decision trees that use simple rules for splitting the remaining data set into two parts by means of a one variable condition at every node. Interpreted graphically, this means that every decision represents a horizontal or vertical split through the decision space, which is quite restrictive concerning the obtainable partitions. However, the advantage of this method is that it can produce interpretable models, even for people with limited knowledge of machine learning. We employ the `R` package `rpart` with the following control parameters:

```
minsplit=20, cp=0.001, maxcompete=4, maxsurrogate=10,
usesurrogate=2, xval=30, surrogatestyle=1, maxdepth=4
```

KNN (k-)Nearest neighbour methods work by utilising similarities between existing data points. In principle, k may be any positive number, this is the number of data points nearest to a new point that is used to decide how to classify it. In its simplest case (1nn classification), we are looking only for the nearest neighbour according to a distance function defined over the decision space, and the new point then gets the same label as this neighbour. For $k > 1$, the decision can be taken by majority vote or any other decision rule.

We utilize the `knn` method from the `R` package `class`, with k set to 3 and l to 0. Note that these algorithms do not possess a training step, they are computed directly.

Random Forests This is an ensemble method based on decision trees, where different subsets of data and features are used to learn a large number (typically 500) of decision trees which are then simultaneously used for classifying a new data set by means of a voting scheme. The graphical interpretation of random forests is similar to the one of decision trees, but much more fine-grained. Much more complex surfaces can be approximated. For example, a line that is not axis-parallel can be produced by simply putting many small lines together that are shifted only by one pixel. On the other hand, it is of course much more difficult to explain how a specific random forest actually makes decisions. One can of course measure how important single features are, but this does not provide any information on how they interact. More complex approaches are needed here to identify meaningful features and their attribution[13].

In our experiments, we use the default parametrization (500 trees) of the `R` method `randomForest` in a package of the same name.

Artificial Neural Networks These are the predecessors of the currently popular deep neural networks, being somewhat simpler (and usually smaller) in structure. They also lack special layer types for processing graphical data, known as convolutional layers. However, the basic working principle is the same. The network consists of nodes that can have multiple inputs and outputs, and each of these edges carries a weight. Within a node, an activation function (and maybe a bias) is used to compute the output according to the weighted sum of the input values. The weights are adapted during training in order to minimize the error, usually by means of a method named backpropagation. It is a greedy optimization procedure (following the local gradient according to the partial derivative). In principle, ANNs with at least 3 layers can learn all types of existing functions. However, for complex functions, they may require a lot of training and a large number of nodes, which in turn requires sophisticated methods to enable interpretation [13].

In our experiments, we only use a very small ANN with a single hidden layer with 7 nodes trained for 1000 iterations, implemented using the `R` package `nnet`. The parameters were determined using a simple grid search as well as based on observed convergence behaviour.

3.5 Training

All experiments are executed using the same settings for model training / computation. The available data is split, so that 90% of it is used for training and 10% for validation, as is done in [22]. Like in [8], the data is split per game, to ensure a fair evaluation. This way, we prevent instances, where data from the same game, e.g. from different players or gameticks, is contained in both training and test set.

We test our models for playtimes of 5-20 minutes, as these are characteristic game lengths and this timeframe is comparable with related work, such as [2, 8]. In addition, we also investigate the final stages of play, as in [15]. The prediction accuracies are computed using 10% hold-outs in 30 independent runs on a HPC cluster (LIDO2). A single evaluation of a model for all gameticks took between a few minutes and 4 hours, depending on the model and the settings.

4 Experimental Analysis

In the following, we describe the experiments and results for each of the research questions in Section 2.3. Most results are visualised in Figure 1. The graphs show the prediction accuracies of different models, where the experimental settings are encoded as linetype (legend on top-left corners) and the model type as colour (legend in caption). The values displayed are the mean accuracies as well as mean accuracy ± 1.5 standard deviation in a slightly lighter colour. Plot 1 (b)

shows a parallel plot of different experiments, while all others show behaviour in relation to the amount of observed gametime.

The different experiments are set up by varying different parameters that slightly modify the dataset used for training and testing. The parameters we are using are listed in the following, with default values in bold.

fs `feature selection:` {**all**, `expert`, `expert2`}
both `both players? :` {`true`, **false**}
tw `time window:` $\mathbb{N}_0$ **(0)**
da `data aggregation:` {**NA**, `none`, `trend`, `combined`}

Feature selection `fs` is usually set to `all`, meaning all features described in Section 3.3. We have also created additional feature selection methods that only select a subset of the available features based on which ones our expert player deemed to be good indicators of the outcome. A detailed description of selected features can be found in the following (paragraph *B2*). With parameter `both`, data from the second player can be included or excluded. It is important to note that if `both=t`, data from the players is included separately. Therefore, there is no obvious connection between the two players in a game, and the resulting model is thus still embodied. However, in order to avoid any correlations between samples, we usually only include the data collected for one player.

The `time window` specifies how much historic data is available to the model. For example, if `tw=2` and we are training models after 10 minutes of gameplay, the data from 9:40mins and 9:50mins would be available as well. By default, however, only the gametick in question is available. The additional historic data is added after being transformed by a data aggregation method `da`. More details on the data aggregation methods tested can be found in the following (paragraph *A3*).

We also want to note that the implementation of the ANN model only allows for a limited amount of features. For this reason, this model failed to run for a few of the settings discussed in the following. In these cases, the corresponding runs are excluded from the results and do not appear in the figures. The KNN model computation timed out (8 hours) for one experiment setting and is thus not included in the corresponding plot.

In the following, we interpret our experimental results in the light of the research questions formulated in section 2.3 (A1, A2, A3, B1, B2).

A1: Model Comparison In most of the work referenced in Section 2, several prediction models were tested. In our study, we are therefore looking to apply the best performing model types, i.e. KNNs in [2] and ANNs in [22] to our problem setting. We focus on simple and small models as well as ones that are likely interpretable, such as decision trees. For more details on the models and their implementation in our analysis, see Section 3.4.

We have executed all experiments described above for a ● decision tree, ● ANN, ● random forest, and ● KNN model. The achieved accuracies are displayed in the respective graphs in Figure 1. We observe that random forests

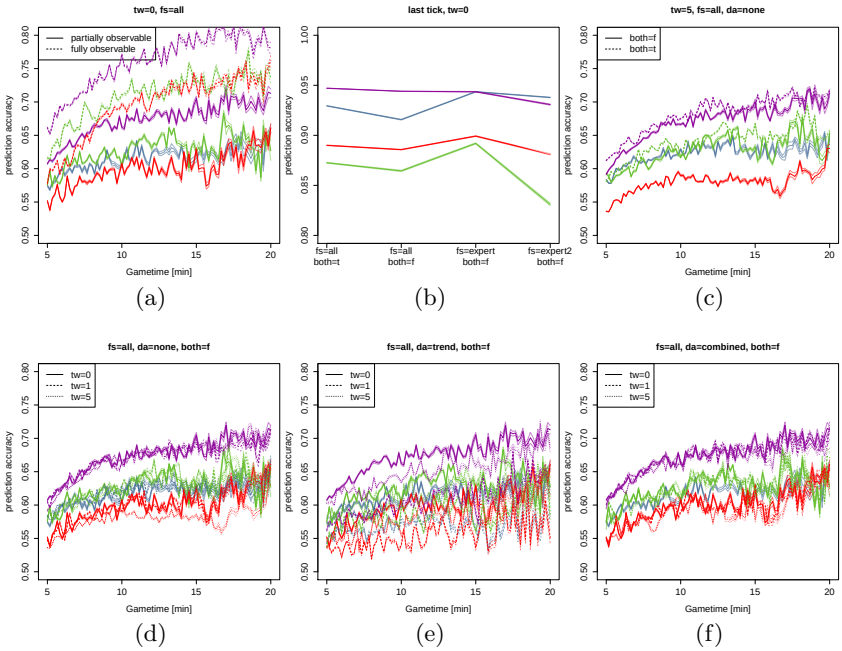


Fig. 1. Prediction accuracy with different methods (a), at the end of the game with different features (b), with and w/o both players datasets (c), and with different types of historic information (d-f), over the most interesting game period (not (b)).
Prediction methods: ● Decision tree, ● ANN, ● Random Forest, ● KNN.

perform well in general and achieve the best performance in most settings. Except for the experiment visualised in Figure 1 (b), the decision tree and ANN have similar performance. In contrast to the expectations resulting from its stellar performance in [2], our KNN model has the weakest performance in most of our tests. However, the dataset the models in [2] were trained on are significantly smaller than ours. It is also important to note that the random forest and ANN models in particular could benefit from an even larger dataset.

Except for in the experiment on the last gametick displayed in Figure 1 (b), all models seem to react similarly to additional or modified data. This means the performance improves and decreases for the same experimental settings by roughly the same amount. The algorithms also display very similar scaling behaviour with an increasing length of observed gametime. Therefore, different approaches should be tested in the future. Additionally, the fact that the most complex model, the random forest in our case, performs best in general, might suggest that there are patterns in the data that are not picked up by a simple model.

The only experimental setting where this relative behaviour of the models is not displayed are the ones visualised in Figure 1 (b). For the last tick, ANNs are in some cases competitive with random forests, while the decision tree and

KNN models obtain similar, but lower accuracies. A possible explanation for this behaviour is that at the last tick, there certainly must be very good indicators of the game outcome that should be easy to model. This is of course assuming that the data was successfully preprocessed so that games with external causes of conceit are removed. Seeing that the models reach accuracies of up to 95%, our preprocessing method described in 3.2 seems to have been successful in this regard. Additionally, all models have only a small standard deviation in terms of prediction accuracy. This is a further indication that our preprocessing method was successful in removing uncharacteristic outliers from the dataset.

Furthermore, we can observe that using different feature selection methods to introduce expert information barely affects the performance of the random forest models, but can improve the performance of all others. This could indicate that the parametrisation of the models is sub-optimal and the predictor e.g. needed more iterations to identify the best features. Alternatively, data-driven feature selection methods could be used to boost algorithm performance.

Overall, we achieve competitive performance to the related work described in Section 2.2. We were able to improve the results presented in [22], the most comparable experimental setup, in terms of early prediction accuracy by around 5% at the 8 minute mark. We achieved similar accuracies after 20 minutes of gametime (around 70%), even with our considerably less complex modelling approach. Improvements in terms of earlier prediction can also be observed when compared to [8]. We have also achieved better results than the TvT win predictor reported on in [15], but we had access to more data. We are, however, not able to reproduce the extremely high prediction accuracies obtained in [2], even when assuming complete observability (see below).

A2: Observability Much of the work described in Section 2.2 allows fully observable game states instead of our embodied perspective. In the following, we therefore investigate how much the observability of the opponent's behaviour affects prediction accuracy. In figure 1 (a), we visualise the performance improvement of our models when information on the opponent is added. To do this, we train the models on a modified dataset where each sample contains all features 3.3 for both players at the same time. All of the trained models show a significant improvement of between 2-10%, depending on model and gametick.

This shows that the results from papers with the assumption of fully observable game states are definitely not directly transferable to our situation and work on embodied outcome prediction is therefore still necessary. This result also seems to suggest that a suitable use of the scouting mechanics available to players in StarCraft II will improve a player's capacity for assessing their own chances of winning. Furthermore, the fact that information on a single player's progress is not sufficient to explain the outcome, strongly suggests that no strategy exists that dominates all others. Otherwise, the winner of a game would always be the player that best executed the dominating strategy. Instead, the players seem to have multiple viable strategies available. This is even more remarkable since we only analyse Terran vs Terran games. This was probably intended by the game designers [5].

A3: Historic Data The authors of [22] presume that prediction accuracy can be significantly improved if it was based on more than just the observations at a given time. This sentiment was echoed by our expert player. We therefore investigate different ways to include information on previous gameticks in the data. We compare the case where no historic data is used (`tw=0`), only the previous observation is used (`tw=1`) and one where we look as far back as a minute (`tw=5`). For each of these different settings of `tw`, we also experiment with different aggregation methods of the data `da`. If the time window is set to 0, this parameter is obviously not meaningful and is thus usually set to NA. If `tw` $\neq 0$, however, historic data is either inserted as independent samples (`none`), combined into a single sample (`combined`) or the observed trend (`trend`) is computed (as the mean of consecutive differences of the observed values per feature). While we do not encode any domain knowledge with `none`, we provide a bit more information to the models. With `combined` we suggest a potential correlation of the values and with `trend`, we specifically force the model to focus on trend information.

The results are displayed in Figures 1 (d)-(f). Surprisingly, for da=none and da=combined, we cannot observe a performance improvement when adding historic data. In fact, the performance even seems to decrease for da=trend. The latter might be a result of the specific modelling of the trend we chose in this case, but the results are still unexpected. They could potentially be explained by the fact that we have only recorded observations every 10 seconds of the game. This limitation might not have been differentiated enough, as units are usually killed and reproduced within very short time spans. However, this is generally not true for (important) structures. Another potential reason could be that the complexity of the relationship between historic and current data could not be represented by our simple models and given a relatively small amount of data. Given more data, deep learning approaches (especially those intended to capture trends in data) might be better suited for this experiment and have a potential to improve the results.

B1: Interpretability In the following, we anecdotally investigate the interpretability of our decision tree models, as they are the most suitable approach for this analysis. To do this, we have generated decision trees for various scenarios, namely

- Last tick prediction with expert features
- Prediction after 10 minutes with expert features
- Prediction after 10 minutes, with expert features, but using historic data (trend aggregations)
- Prediction after 10 minutes, with all features, but using historic data (trend aggregations)

The resulting models are depicted in Figure 2. See also paragraph *B2* for more details on the feature sets. They were able to obtain prediction accuracies of approximately 0.898, 0.67, 0.556, 0.595, respectively. The last three models depict

the data observed at the same point in time and were generated from the same split of training and test data to facilitate a comparison.

It is important to note, that studies suggest that non-experts can only simultaneously grasp 5-9 different variables with 3-15 values [12]. Therefore, to guarantee interpretability, the features would need to be discretised. Additionally, the decision should be parameterised accordingly to prevent generating too many branches. Interestingly, the decision tree trained on data containing all features (i.e. `fs=all`), instead of just a subset, comes closest to the mentioned constraints of interpretability.

Despite this caveat, we asked our human expert player to interpret the models, as they still contain interpretable information when focusing on different branches separately. According to him, the first model seems very reasonable in context of the game. It asks if the player has any army left. If not, then it tries to assess with damage taken and dealt, whether the opponent has any army left either. A human player would probably look at similar statistics.

The second model appears to be quite similar. What stands out as unexpected, is that both models use the number of idle workers. One possible interpretation is that players have many idle workers if there is an imminent and critical threat, which forces the player to pull their workers away from mining. A large number of idle workers also indicates a non-efficient management of resources and build order.

The third model seems much closer to the approach a human would take to evaluate their chances of winning, i.e. interpret the trends in game data. If the amount of damage done increases and the amount of damage taken does not increase significantly in a time frame of 10 seconds (`tw=1`), this indicates the player won a battle. Of course, a won battle generates a strong advantage over the other player at that point in time. If the collection-rate increases compared to the last observed game state, the player was able to expand their economy without being punished for it. This is another indicator of success and can be interpreted as a sign that winning chances are good.

Tree number 4 seems reasonable as well. However, resource production, which is used as the main feature here, is very dependent on the strategy and thus not an ideal predictor, especially with partial info. Furthermore, while resource production is an important part, it is more telling how efficiently they are used for military and economy (`resources_spent`). This is because gathering resources is relatively easy, the more difficult skill is developing and executing a good and adaptive build order. Further down the tree, however, the distinctions are comparable to the previous trees.

Overall, according to the expert, it is hard to understand why model 3 has a comparably weak prediction rate, as it definitely is the most human-like approach. However, it is also surprising that predictions on the game state can be so good with so little information. Concluding, while there are caveats to the interpretability from a formal standpoint, some models seem to be able to capture the thought process of (expert) human players. Others seem to find different patterns in the data. This suggests that after applying suitable complexity

constraints and a careful evaluation and selection of the specific model to use, decision trees have potential to be used to inform players.

B2: Domain Knowledge Finally, we investigate ways to integrate domain knowledge from expert players into the models. This is an approach that is commonly used in industry, but also in research (see Section 2.2), in order to bias the machine learning model suitably. In our case, we chose to test restricting the search space for model training, hoping that this will improve the convergence properties of the models. At the same time, it should reduce the risk of the models reflecting non-interpretable data artifacts instead of underlying game characteristics.

From a human perspective, there is an extraordinary amount of features that can be recorded by the SC2 research framework to describe a game state. This includes all sorts of information from features readily available to the player such as supply (known in the API as food) to information that no player would ever have a chance of keeping track of, such as the total amount of damage they have dealt in a match.

For many of these features, players would assume that they carry little to no information related to the outcome of a match. Our expert player has thus chosen two sets of features called `expert` and `expert2` from the available ones described in Section 3.3, the second set being even smaller than the first one. The selected features were deemed to be the most important ones for predicting the winner of the match. They are listed in the following:

Expert Feature Set:
`gameloop`
`{∅, collection rate} x {minerals}`
`{killed, lost} x {minerals} x {economy, army}`
`{total value, killed value} x {units}`
`killed value structures`
`{total} x {damage dealt, damage taken} x {life}`
`{food} x {used army, used economy}`

Expert2 Feature Set:
`gameloop`
`{killed, lost} x {minerals} x {economy, army}`
`{total value, killed value} x {units}`
`{food} x {army, workers}`

`Food_army`, the amount of supply currently spent on army, was chosen as it directly correlates with a player's army size. If a player has little to no army, especially in later stages of the game, this is a strong indicator that they are losing. `Food_workers` can also be a strong indicator of whether a player is winning or losing. If the player has few workers left, they will not be able to rebuild an army, and thus risk losing if their army is destroyed. `Killed_minerals_army`, `killed_minerals_economy`, `lost_minerals_army`, and `lost_minerals_econ-`

`omy` were chosen as they can be used to gauge which player won the battles. This is because these features allow some information on the success of the opposing player. If a player has killed significantly more minerals than they have lost, they are certainly in a favorable position.
`Killed_value_structures` was chosen as a clear indicator of a critical situation in the game. Players generally do not lose many structures.

It is important to note that the selected features do not necessarily coincide with the features identified as most important by our models (cf. decision tree 4 in Figure 2). A larger scale study on the usage of features by the different models would be needed to allow general statements on feature selection. However, we can observe the influence of restricting the feature sets on the models. The corresponding plot can be found in Figure 1 (b).

The effect of this approach to integrating domain knowledge seems to depend heavily on the model type and is not necessarily positive. This finding demonstrates the difficulties of suitably biasing general machine learning approaches to improve their performance. In our experiments, restricting the search space did definitely decrease the computation time, however. From our data, we cannot conclude whether a strictly data-based approach with more complex models and larger amounts of data is more promising than identifying better ways to integrate domain knowledge. Both appear to offer potential for improving the prediction models.

In the following, we thus take a different approach to assessing domain knowledge. Our expert player has attempted to go through his own replays to find scenarios, where either predicting the winner was easy or he failed to do so. Based on this information, we outline how human players approach the subject of estimating the likelihood of winning independent of our models. In one instance, the outcome was reportedly easy to tell based on information gathered from contacts with the opposing player. It is however unclear how this information is reflected in the features we collected in our study. Deep learning approaches trained on raw information collected from the minimap as done in [22] might be able to capture this. Our data would only reflect the trades made for buildings and military.

In scenarios where these trades are obvious, a predictor is likely able to pick up on them. However, since units and structures are encoded via their resource value in the recorded data, this can only be interpreted in a meaningful way if the resource values accurately reflect the value of the unit or building in the game. This is unlikely, because the value of certain units depends on the strategy of the opponent. Instead, this issue could be approached by counting the number of different units and structures individually, so that the tradeoffs can be identified via data analysis. Different units have been encoded separately in [2] (mostly by type, e.g. air and ground units). But this approach increases the number of features, which reduces the interpretability of the resulting models, which is why it was avoided in this study.

An interesting avenue to characterising the difference between our trained model predictors and human experience would be to identify enough replays

that are similar to scenarios where winner prediction is easy or hard for human players. The models could then be used to evaluate the corresponding replays to assess whether data-driven approaches could be used as information tools for pro-players in these specific cases.

5 Conclusion and Outlook

In this paper, we have approached embodied winner prediction in StarCraft II as a machine learning problem using replays collected and published by Blizzard and DeepMind [22]. We were able to achieve competitive results, even with

- simple, interpretable models
- with simple features
- on human vs. human data.

While there is still potential for improvement of the respective models, finding the best predictor was not the intent of this paper. Instead, we focus on investigating which directions of future research suggested in previous work seem most promising. To this end, we identify several research questions, mostly targeted towards the influence of observability and other information constraints on achievable prediction accuracy. Furthermore, we give a detailed description of our preprocessing method as well as a baseline accuracy for embodied winner prediction in StarCraft II.

We demonstrate that while adding information on the opposing player can facilitate the prediction of the winner, this is not necessarily true when adding information on previous game states instead. This was a surprising result, but it is possible that different methods of integration or a larger amount of data might produce different results. Furthermore, we find that biasing trained models with domain knowledge does also not necessarily translate into improved performance. Finally, while we are able to produce models that contain information comprehensible to an (expert) human player, more work is required to guarantee and investigate interpretability.

Therefore, in the future, more research is required concerning different ways of integrating different types of information into data-driven prediction models. This is true for additional information and interpretative measures based on domain knowledge (such as LTD2 [10] and the expression of strategy in [15]), as well as for information gathered from previously made observations in the same game. More work is also required towards formal definitions on the interpretability of trained models for complex games. Frameworks that allow the visual exploration of different features and their attribution [13] are a promising approach to make even more complex models interpretable. Alternatively, the feature space a more complex model learns can also be used as a proxy task to train simpler models on. This way, knowledge can be distilled into a less complex model [9].

In terms of improvement of the achieved prediction accuracies, increasing the complexity of the models as well as the amount and complexity of training

data (by adding more replays, but also spatial data) appears to be a promising approach with hopefully interesting newly discovered patterns in the data. Another way of enhancing the data is to include more historic data. A thusly enhanced data set might even allow the use of deep learning models that are potentially able to uncover more complexities. We ran preliminary tests with deep learning models on the existing data, which did not improve the obtained prediction accuracies. But, given more or enhanced data, these methods definitely have potential.

We, however, want to concentrate our future efforts on transferring our findings on winner prediction to approaches on game state evaluation. While this would certainly be an interesting topic for future StarCraft II AI research, we want to focus more on its application to AI-assisted game design to express aspects such as drama or decisiveness [6].

References

[1] H. Alburg et al. "Making and Acting on Predictions in StarCraft: Brood War". University of Gothenburg, 2014.

[2] A. Álvarez-Caballero et al. "Early Prediction of the Winner in StarCraft Matches". In: *International Joint Conference on Computational Intelligence*. 2017.

[3] T. Avontuur, P. Spronck, and M. van Zaanen. "Player Skill Modeling in Starcraft II". In: *Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Boston, MA, USA: AAAI Press, 2014, pp. 2–8.

[4] L. Breiman et al. *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[5] D. Browder. "The Game Design of STARCRAFT II: Designing an E-Sport". In: *Game Developers Conference (GDC)*. `http://www.gdcvault.com/play/1014488/The-Game-Design-of-STARCRAFT`. 2011.

[6] C. Browne and F. Maire. "Evolutionary Game Design". In: *IEEE Transactions on Computational Intelligence and AI in Games* 2.1 (2010), pp. 1–16.

[7] C. B. Browne et al. "A Survey of Monte Carlo Tree Search Methods". In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.1 (2012), pp. 1–43.

[8] G. Erickson and M. Buro. "Global State Evaluation in StarCraft". In: *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 2014, pp. 112–118.

[9] G. E. Hinton, O. Vinyals, and J. Dean. "Distilling the Knowledge in a Neural Network". In: *NIPS Deep Learning Workshop*. `https://arxiv.org/abs/1503.02531`. 2014.

[10] A. Kovarsky and M. Buro. "Heuristic Search Applied to Abstract Combat Games". In: *Advances in Artificial Intelligence*. Ed. by B. Kégl and G. Lapalme. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 66–78.

[11] R. Lopes and R. Bidarra. "Adaptivity Challenges in Games and Simulations: A Survey". In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.2 (2011), pp. 85–99.

[12] G. Miller. "The magical number seven, plus or minus two: Some limits on our capacity for processing information". In: *The Psychological Review* 63 (1956), pp. 81–97.

[13] C. Olah et al. "The Building Blocks of Interpretability". In: *Distill* (2018). https://distill.pub/2018/building-blocks.

[14] D. Perez-Liebana et al. "General Video Game AI: Competition, Challenges and Opportunities". In: *AAAI Conference on Artificial Intelligence*. 2016, pp. 4335–4337.

[15] Y. N. Ravari, S. Bakkes, and P. Spronck. "StarCraft Winner Prediction". In: *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 2016.

[16] G. Robertson and I. D. Watson. "An Improved Dataset and Extraction Process for Starcraft AI". In: *FLAIRS Conference*. 2014, pp. 255–260.

[17] A. A. Sánchez-Ruiz-Granados. "Predicting the Winner in Two Player StarCraft Games". In: *CoSECivi*. 2015, pp. 24–35.

[18] M. Stanescu et al. "Evaluating Real-Time Strategy Game States Using Convolutional Neural Networks". In: *IEEE Conference on Computational Intelligence and Games*. 2016.

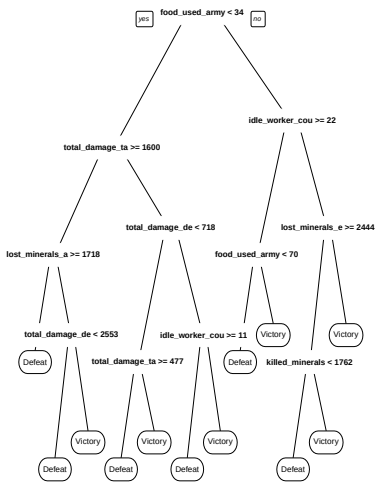
[19] M. Stanescu et al. "Predicting Army Combat Outcomes in StarCraft". In: *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 2013, pp. 86–92.

[20] A. Summerville et al. "Understanding Mario: An Evaluation of Design metrics for Platformers". In: *Foundations of Digital Games*. ACM New York, NY, 2017.

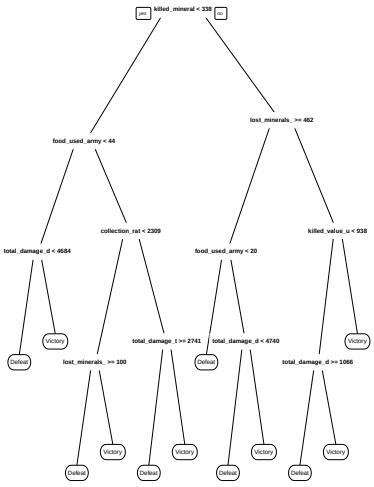
[21] M. Čertický and D. Churchill. "The Current State of StarCraft AI Competitions and Bots". In: *Artificial Intelligence and Interactive Digital Entertainment Conference*. 2017.

[22] O. Vinyals et al. "StarCraft II: A New Challenge for Reinforcement Learning". In: *CoRR* abs/1708.04782 (2017). arXiv: `1708.04782`.

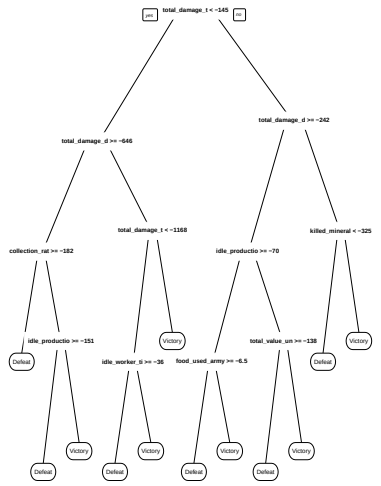
[23] P. Yang, B. E. Harrison, and D. L. Roberts. "Identifying patterns in combat that are predictive of success in MOBA games". In: *Foundations of Digital Games*. 2014.



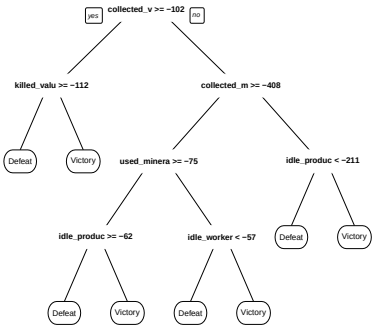
last tick, fs=expert, both=f
(a)



10mins, tw=0, fs=expert, both=f
(b)



10mins, tw=1, fs=expert, both=f
(c)



10mins, tw=1, fs=all, both=f
(d)

Fig. 2. Decision trees learned for the last tick (a) and for the game state after 10 minutes (b-d) as samples for checking differences between machine learning based and expert human reasoning.