

Knowledge representation in AlphaStar

Seminar Knowledge Representation and Processing
Friedrich-Alexander Universität Erlangen-Nürnberg (FAU)

Benjamin Gorny

August 31, 2020

Contents

1	Introduction	3
2	Background	3
2.1	StarCraft II	3
2.2	Artificial neural networks	3
2.3	Supervised learning	4
2.4	Reinforcement learning	5
3	Learning in AlphaStar	5
4	Evaluation	6
5	Conclusion	7
5.1	Comparison to AlphaZero	7

1 Introduction

Creating systems that can play games on the same level as or better than humans has been a part of artificial intelligence research for a long time, so it does not come as a surprise that DeepMind’s research extends into this area, too. With the development of AlphaGo in 2015, which managed to beat the number one Go champion at the time in 2017, and AlphaZero, which not only beat AlphaGo at Go, but was also able to play chess and shogi at a superhuman level, they introduced two very successful game-playing agents [2].

Their latest development in 2019, AlphaStar [7], as well as the underlying technology which is used to represent knowledge in this system, will be showcased in this report. First, some background information about the game of StarCraft II (for simplicity called “StarCraft” in the following) as well as artificial neural networks and the two main techniques for learning, which are used in AlphaStar, will be given. After that, I will cover the training process of the agent. Finally, AlphaStar will be compared to DeepMind’s earlier system of AlphaZero.

2 Background

Real-time strategy games have been of interest to AI researchers for some time, but they often only studied sub-problems such as base building or unit micromanagement [7]. StarCraft has become the consensus for a suitable domain to explore the challenge of all these aspects combined [7]. Before the introduction of AlphaStar, no agent has been able to match top StarCraft players, despite playing the game on a simplified level [7].

2.1 StarCraft II

StarCraft II is a science-fiction real-time strategy game developed at and released by Blizzard Entertainment in 2010. Its gameplay aspects are typical for an RTS game, ranging from resource gathering and base construction to the creation and micromanagement of units. The player can choose from three different races (Terran, Zerg and Protoss), which largely vary in terms of mechanics, available units and buildings and strategies [4].

Some units in StarCraft have abilities which have to be controlled manually by the player, too, allowing for complex interactions with the environment or other units [4]. In addition to that, the game contains a fog of war - meaning that the player can only see parts of the game environment which are revealed by their own units or buildings. This incomplete information requires the player to explore the environment to be able to counter their opponent’s strategy.

StarCraft II is focused on one-on-one multiplayer and is played professionally on an international level, with an especially large esports scene in South Korea [4].

Challenges for AI agents StarCraft is particularly challenging for AI agents because of its partially observable world, which requires an agent to “remember” parts of the environment it previously saw, perhaps even blindly predicting some of their opponent actions. Additionally, all decisions must be made in real time and in parallel to other players’ actions. At each time step, there are approximately 10^{26} possible choices to make [7], whereas there are only around 150 – 250 at each turn in Go, and ca. 37 in chess [3].

2.2 Artificial neural networks

Artificial neural networks are inspired by the human brain and the connections between its neurons, resulting in human intelligence [6]. They are able to “learn” by example, without

requiring domain-specific knowledge, which makes them applicable for a variety of tasks [1].

Artificial neural networks consist of nodes, called **units** and **links** between them, which propagate activations $a_{i,j}$ from unit i to j . Each link has a weight $w_{i,j}$, representing the strength of the connection between the units, and each unit has a dummy input $a_0 = 1$ with an associated weight $w_{0,j}$. [6]

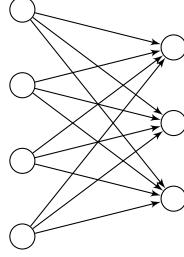


Figure 1: Exemplary layout of an artificial neural network with one layer

Each unit first computes the weighted sum of its inputs

$$in_i = \sum_{j=0}^n w_{i,j} a_j$$

and then feeds it into its activation function g , calculating the unit's output value:

$$out_i = g(in_i) = g\left(\sum_{j=0}^n w_{i,j} a_j\right)$$

[6]

An activation function can either represent a hard threshold or use a logistic function to calculate the output [6].

A basic **feed-forward network** just propagates information forward through its units; its links are organized as a directed acyclic graph. This kind of network is unable to store information, which is why **recurrent networks** are more prevalent. They feed the outputs of their units back into the inputs, resulting in a form of short-term memory [6].

Usage in AlphaStar AlphaStar utilizes neural networks for a variety of tasks, ranging from the representation of domain-specific knowledge to the actual decision making procedure, which uses a policy network to select its next action [7].

2.3 Supervised learning

The task of supervised learning is to learn a hidden function based on input and output values. Given a function f and a **training set** of input-output pairs

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$$

where $y_i = f(x_i)$, a function h should be discovered which approximates f . The goal is to find a function h (called **hypothesis**) which performs well not only on the training set, but also on other data. This performance can be measured by applying it to a **test set** of data, distinct from the training set [6].

A hypothesis is said to **generalize well** if it correctly predicts y for new examples. It is called **consistent** when it fits with all data. Russel and Norveig (2010) mention a tradeoff between complex hypotheses which are consistent and simpler hypotheses that might generalize better [6].

2.4 Reinforcement learning

One problem in supervised learning is that an agent needs to be told each “good move” for every situation it could be in. If such feedback is not available, the agent can learn a transition model for its moves, but will have no notion of the *quality* of these moves, based on which it could make a decision [6].

Such feedback for the quality of an action is called **reinforcement** or **reward**. In many games, this reinforcement is only received at the end, i.e. “you win” or “you lose”. To be able to evaluate the quality of actions before the game ends, the agent learns an evaluation function, which estimates the probability of winning in its current situation [6].

3 Learning in AlphaStar

AlphaStar’s observation of the game state is designed to be similar to how human players play. Therefore, it receives an overview map and a list of units it sees. Actions are output in a manner of **what**, **who**, **where** and **when**, i.e. the agent first specifies which kind of action it wants to perform, then which unit or building is used to carry out the action, the location of the action and finally when the next action will be issued [7].

In addition to that, actions are subjected to network delays and the game’s processing time, just like they are for humans. AlphaStar’s monitoring layer sets an action limit, too, to prevent the agent from performing too many actions at once. To deal with the partial observable nature of the game, AlphaStar employs a long short-term memory system [7].

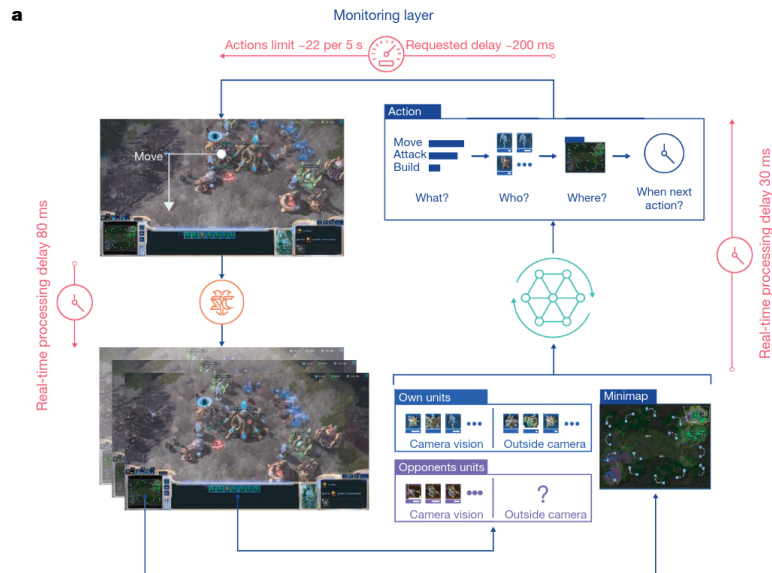


Figure 2: AlphaStar’s perception of the game and decision procedure [7]

At the heart of AlphaStar’s decision procedure lies its policy network with parameters θ , which represents the policy $\pi_\theta(a_t, s_t, z) = \mathbb{P}[a_t|s_t, z]$, which takes all observations since the start of the game, as well as a statistic z , sampled from human data, into account and outputs actions [7].

Its parameters are initialized from *supervised learning*, based on a publicly available, anonymized collection of human replays, optionally also conditioned on z . After the initial training, the parameters were subjected to a *reinforcement learning* algorithm which maximized the win rate of the agent. When the agent is conditioned on a strategy z , it receives rewards for following it and penalties for deviating from it [7].

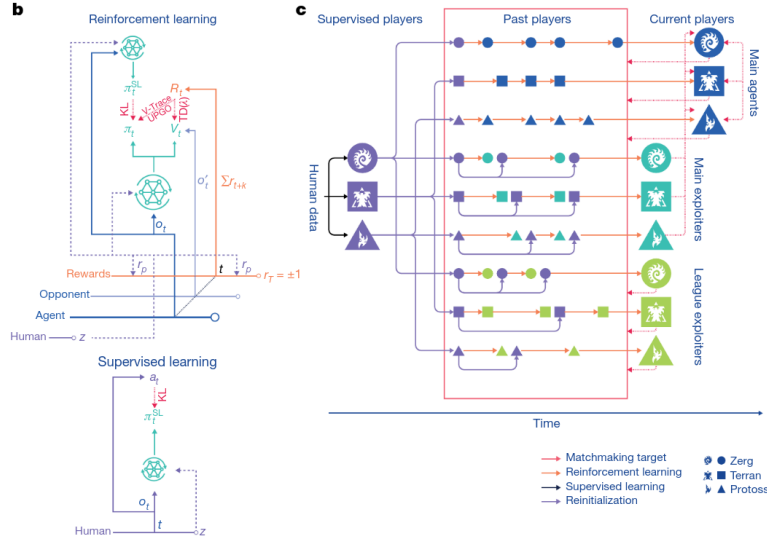


Figure 3: AlphaStar’s training setup [7]

During training, agents were split into three pools, one for each playable race in the game. At specific points in the training process, frozen copies of agents were added to leagues, and the main agents trained against those league players, as well as themselves. This, in combination with the use of **exploiter agents** (which target the weak points in an agent’s strategy to force it to adapt), avoided cycles in learning progress, which often occur when utilizing naive self play [7].

4 Evaluation

AlphaStar’s final agents were evaluated against humans on Blizzard’s battle.net matchmaking service – starting on fresh, unranked accounts – and reached Grandmaster level, which places them above 99.8% of ranked human players. In addition to that, a preliminary version of AlphaStar won all games in two five-game series against StarCraft professional players Grzegorz ‘MaNa’ Komincz and Dario ‘TLO’ Wunsch in December 2018 [7].

The agents played the matches on the matchmaking services without being provided information about their opponent, and themselves played under anonymous accounts, making it a double blind scenario. It has to be noted that this evaluation did not take the possibility of exploitation under repeated play into account, i.e. if human players would perform better against AlphaStar

after playing several matches against it, allowing them to recognize and counter its strategies better [7].

The fact that AlphaStar’s *supervised* agents ranked at above 84% of human players shows that, while supervised learning in itself is pretty effective, adding reinforcement learning on top of it resulted in a significant increase in success. Additionally, it could be observed that the unit compositions, that the agents employed during the evaluation games, varied, meaning that they did not in fact find an ultimate strategy that most human players were unable to beat, but instead utilized diverse strategies in successful ways [7].

Finally, professional player Dario ’TLO’ Wunsch stated that the limitations that were put on AlphaStar do not make it feel superhuman, in spite of its excellent and precise control, and playing against it felt “very fair, like it is playing a ’real’ game of StarCraft”. He also mentions that AlphaStar does not play “on a level that a human couldn’t theoretically achieve” [7].

5 Conclusion

Similar to StarCraft, many real-world domains both require real-time decisions and have imperfect information. As AlphaStar’s success shows, machine learning algorithms based on neural networks can deal with both of these aspects relatively well, suggesting possible uses for these techniques in real-world scenarios.

5.1 Comparison to AlphaZero

Comparing AlphaStar to one of DeepMind’s earlier agents, AlphaZero, which is capable of learning and playing Go, chess and shogi, the main similarities of these systems are the use of self-play for learning, as well as the forgoing of domain-specific knowledge in favor of artificial neural networks, although one must note that AlphaStar’s league play is fundamentally different from the naive self-play which AlphaZero employs [7][5].

Aside from that, training times are vastly different, as AlphaStar’s final agents took 44 days to train, where AlphaZero’s agent was fully trained after 72 hours. In addition, the problem of incomplete information is not present in any of the games AlphaZero plays, and the vastness of StarCrafts action space makes search algorithms infeasible for AlphaStar, while AlphaZero utilizes a Monte Carlo tree search guided by a policy network [7][5].

References

- [1] Artificial neural network (wikipedia). https://en.wikipedia.org/wiki/Artificial_neural_network, 2020.
- [2] Deepmind (wikipedia). <https://en.wikipedia.org/wiki/DeepMind>, 2020.
- [3] Go (game) (wikipedia). [https://en.wikipedia.org/wiki/Go_\(game\)](https://en.wikipedia.org/wiki/Go_(game)), 2020.
- [4] Starcraft ii: Wings of liberty (wikipedia). https://en.wikipedia.org/wiki/StarCraft_II:_Wings_of_Liberty, 2020.
- [5] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.
- [6] Peter Norveig Stuart J. Russel. *Artificial Intelligence: A Modern Approach, Third Edition*. 2010.
- [7] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P Agapiou, Max Jaderberg, Alexander S Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.