Learning Macromanagement in StarCraft from Replays using Deep Learning

Niels Justesen (noju@itu.dk) & Sebastian Risi
IT University of Copenhagen

Introduction

The real-time strategy game StarCraft has proven to be a challenging environment for artificial intelligence techniques, and as a result, current state-of-the-art solutions consist of numerous hand-crafted modules.

Learning to play StarCraft end-to-end with deep learning, as it was done for Atari games, may be an infeasible approach. A simpler approach, which we have followed, is to apply deep learning to replace a specific function in a larger AI architecture. More specifically, we apply deep learning to predict human macro-management tasks in StarCraft, where after the learned network is integrated into an existing StarCraft bot.

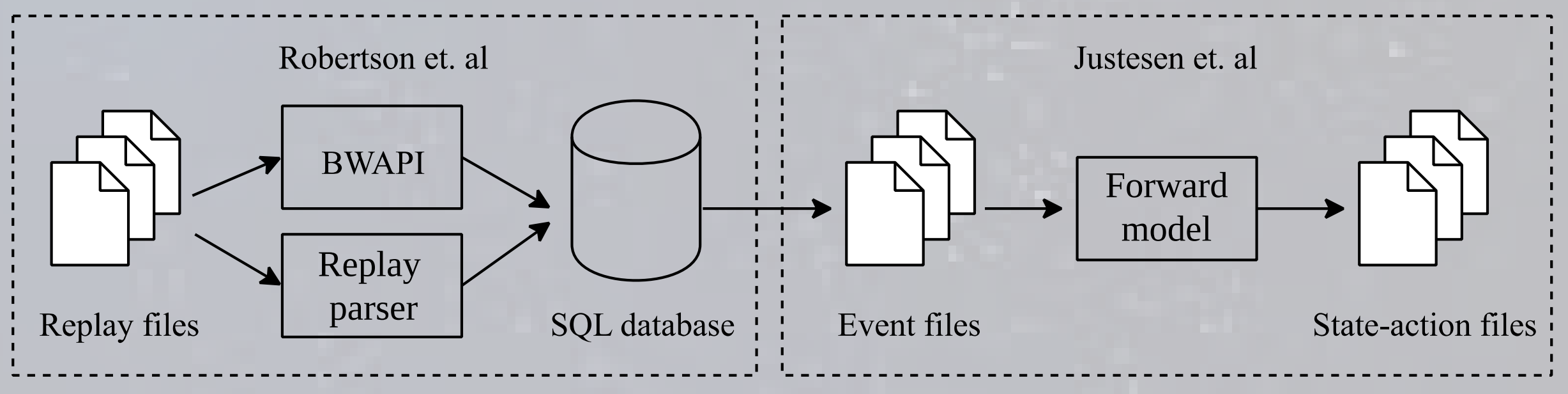
More details are available in our paper:

[1] N. Justesen and S. Risi. Learning macromanagement in StarCraft from replays using deep learning. In Computational Intelligence and Games, 2017. CIG 2017. IEEE Symposium on. IEEE, 2017

Dataset

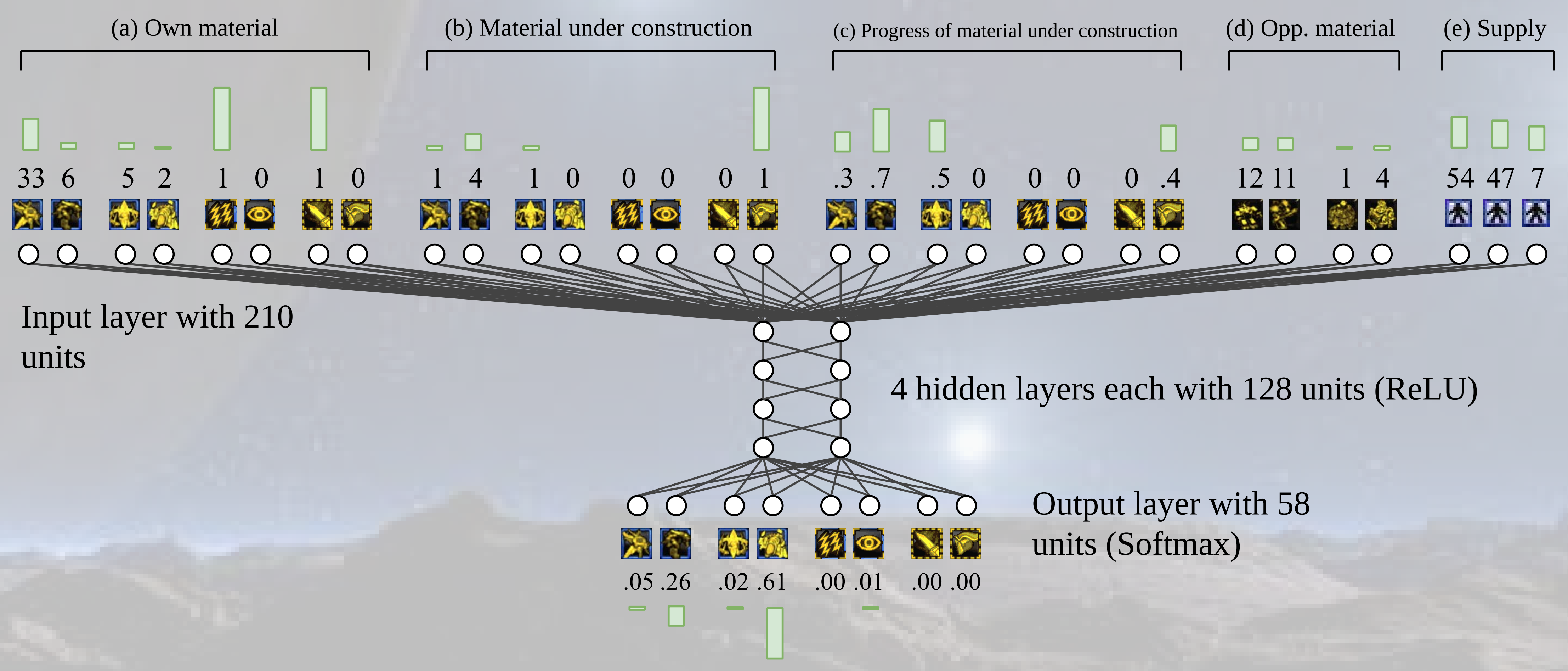
Our dataset is extracted from 7,649 replays of highly skilled players (collected by Synnaeve et al. [2]), and was done in three steps:

1. Actions and state changes are extracted from replays and stored in a database (Robertson et al. [3]).
2. Events related to material changes are saved in .json format.
3. Abstract StarCraft games are simulated using our build order forward model, and state-action pairs are saved. States describe the number of each unit, building, technology, and upgrade present and under construction, as well as enemy material observed. Actions describe what the players decides to produce next.



[2] G. Synnaeve and P. Bessiere. A dataset for starcraft ai & an example of armies clustering. arXiv preprint arXiv:1211.4552, 2012.
[3] G. Robertson and I. D. Watson. An improved dataset and extraction process for starcraft ai. In FLAIRS Conference, 2014.

Network Architecture


(a) Own material
(b) Material under construction
(c) Progress of material under construction
(d) Opp. material
(e) Supply
33 6 5 2 1 0 1 0
1 4 1 0 0 0 0 1
.3 .7 .5 0 0 0 0 .4
12 11 1 4
54 47 7
Input layer with 210 units
4 hidden layers each with 128 units (ReLU)
Output layer with 58 units (Softmax)
.05 .26 .02 .61 .00 .01 .00 .00

Training

The dataset of 789,571 state-action pairs is split into a training set of 631,657 pairs (80%) and a test set of 157,914 pairs (20%). Xavier initialization is used for all weights in the hidden layers and biases are initialized to zero.

A learning rate of 0.0001 were used with the Adam optimization algorithm, with a batch size of 100 and the cross entropy loss function.

The problem is thus treated as a classification problem, in which the network tries to predict the next build action given a description of the game state.

Results: Prediction

Method	Top-1 error	Top-3 error	Top-10 error
Our approach	**54.60% ± 0.12%**	**22.92% ± 0.09%**	**4.03% ± 0.14%**
Probe	73.90% ± 0.00%	73.90% ± 0.00%	73.90% ± 0.00%
Random	98.28% ± 0.04%	94.87% ± 0.05%	82.73% ± 0.08%

This table shows the top-1, top-3 and top-10 error rates on the test set of our trained network (averaged over five runs) and two baselines predictors.

Probe always predicts the next build to be a probe and *Random* predicts randomly with uniform probabilities.

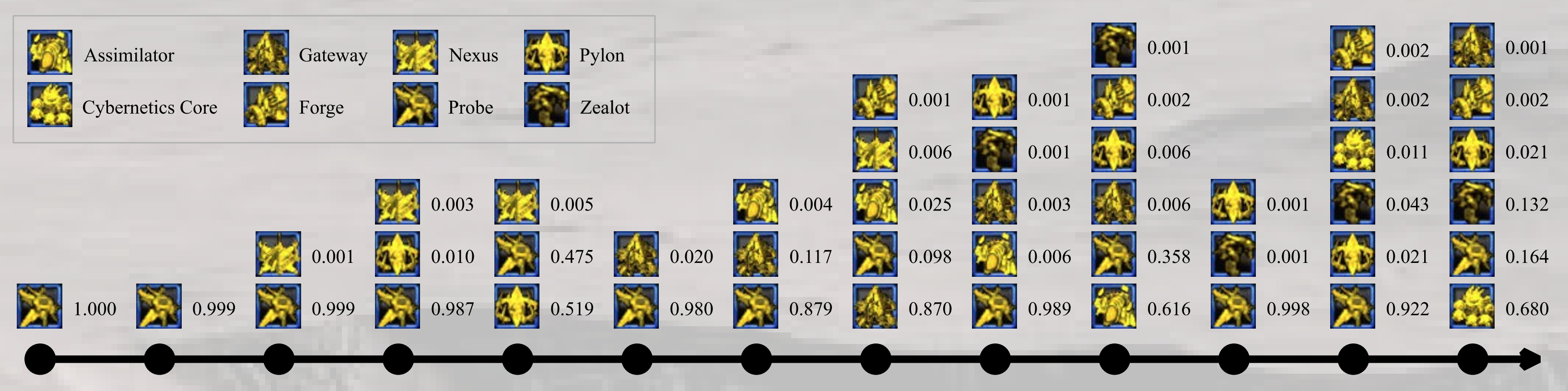
Results: StarCraft

Method	Win rate
Our approach	**68%**
Random	0%
Scripted dragoon rush	100%

This table shows the win percentage of UAlbertaBot with the trained neural network as a production manager against the built-in Terran bot. Actions were selected with probabilities equal to the outputs of the network.

We also tested the probabilistic approach against UAlbertaBot with the original production manager configured to follow a fixed marine rush strategy and our approach won 45% of 100 games.

Learned Opening



This figure shows the opening build order learned by our neural network with action probabilities shown at each step. In this case, the action selection method always picks the build with highest probability.

Future Work

Our results demonstrate how deep learning can be used to learn macromanegement tasks, and we believe this is an important step towards human-level AI for StarCraft.

To reach this goal, we want to improve our work in these three directions:

1. Fix faulty behaviors in UAlbertaBot when controlling large armies and multiple bases, or develop a new bot suitable for our approach.
2. Apply reinforcement learning to improve bahaviors to beat other bots, which is an important step for other game-playing systems relying on deep learning.
3. Extend our model with convolutional layers to consider positional information, which we currently ignore.

Web site: www.njustesen.com, Twitter:@nojustesen