Proceedings of 2020 IEEE 10th International Conference on Intelligent Systems

Forward Model Learning for Motion Control Tasks

Alexander Dockhorn
School of Electrical and Computer Engineering
Queen Mary University London
United Kingdom
Email: a.dockhorn@qmul.ac.uk

Rudolf Kruse
Faculty of Computer Science
Otto von Guericke University Magdeburg
Germany
Email: rudolf.kruse@ovgu.de

***Abstract*—In this work, we study the capabilities and limitations of forward model learning agents and their applications to motion-control tasks. Forward model learning agents learn to approximate the environment dynamics to apply planning algorithms for action-selection. While previous work has shown that forward model learning agents can efficiently learn to play simple video games, we extend their applicability to domains with continuous state and action spaces. Our experiments show that such agents are quickly able to learn an approximate model of their environment, which suffices to solve several simple motion-control tasks. Comparisons with deep reinforcement learning further highlight the sample efficiency of forward model learning agents.**

***Keywords*—Forward Model Learning; Deep Reinforcement Learning; Motion Control Tasks**

I. Introduction

Previous works have shown that several motion control tasks with known environment dynamics can effectively be solved by planning agents [1]. However, those cannot be applied in case the dynamics are unknown. In contrast, reinforcement learning agents have been able to learn even complex behavior through continuous interactions with such environments [2]. While model-free reinforcement learning agents solely focus on the expected return of an action, model-based reinforcement learning models also approximate the environment's dynamics. Benchmarks have shown that model-based reinforcement learning often results in a higher sampling efficiency during the agent's training [2]. Similarly, forward model learning agents construct a model of their environment by observation. In contrast to model-based reinforcement learning, forward model learning agents determine the value of an action at run-time. For this purpose, the approximate model of the environment is used for simulating the result of its actions, and therefore, allows the application of planning algorithms.

A benchmark on the game Sokoban has shown that state-of-the-art model-based reinforcement learning agents need to observe about 10^8 time-steps until the first levels can be solved [3]. Similar experiments on forward model learning agents have shown a rapid increase in performance after observing a few thousand game-steps [4]. While this comparison highlights the possible sampling efficiency of forward model learning agents, the presented experiments were limited to categorical state and action spaces [4]–[6]. Building upon these results, we introduce forward model learning agents that can handle continuous state and action spaces using simple models of their environment.

The key contributions of this paper are:

- **Forward model learning for continuous control tasks:** We propose the use of various regression models for predicting state transitions in real-valued state spaces and discuss which action-selection methods seem suitable for continuous action spaces.
- **Comparing the sample efficiency:** Using motion control tasks of varying complexity we compare the proposed forward model learning agent to several deep reinforcement learning agents in terms of sampling efficiency and the agents' performance.

The remainder of this paper is structured as follows. In Section 2 we review recent works on forward model learning and introduce the required notation for the model's design process. Section 3 introduces a decomposed differential forward model able to model continuous state spaces. Furthermore, we review and compare planning algorithms for continuous action spaces. In Section 4 we compare the final model to several deep reinforcement learning algorithms followed by a short discussion in Section 5. The paper concludes in Section 6.

II. Forward Model Learning and Comparable Approaches

Forward model learning agents consist of two components, one for learning a model of the environment and a second one for action-selection. The learning component's goal is to create an approximation of the environment model. In reinforcement learning, the environment model is considered to be a stochastic process that maps the sequence of previous actions $a \in \mathcal{A}$ and states $s \in \mathcal{S}$ to a probability of observing the next state and reward $r \in \mathbb{R}$.

$$p(s_t, r_t \mid s_{t-1}, a_{t-1}, \ldots, s_1, a_1)$$

We consider each state $s_t \in \mathcal{S} \subseteq \mathbb{R}^n$ to be a vector of real-valued observations $s_t = (s_t^1, s_i^2, \ldots, s_t^n), s_t^i \in \mathbb{R}$ at time t. The same notation applies to available actions and the action space: $a_t \in \mathcal{A} \subseteq \mathbb{R}^n$.

For simplification, we consider the environment model to be a Markov decision process of m-th order, whereas m is the length of the state-action sequence, that the agent considers in its model:

$$p(s_t, r_t \mid s_{t-1}, a_{t-1}, \ldots, s_{t-m}, a_{t-m})$$

We further split the environment model into a state transition model and a reward model:

$$\begin{aligned} \text{State Transition Model:} \quad & p(s_t \mid s_{t-1}, a_{t-1}, \ldots, s_{t-m}, a_{t-m}) \\ \text{Reward Model:} \quad & q(r_t \mid s_{t-1}, a_{t-1}, \ldots, s_{t-m}, a_{t-m}) \end{aligned}$$

To fit such a model the agent needs to frequently interact with its environment while storing a data set of made observations. Given such a data set a supervised learning method can be used to predict upcoming states. Depending on the structure of the state-space a classifier (discrete and nominal state-spaces) or a regressor (real-valued state spaces) can be used to predict upcoming states.

In recent works of model-based reinforcement learning, the state transition model has been represented in the form of deep neural networks. The *World Models* agent learns a latent vector representation using LSTMs [7]. This allows the agent to keep track of previous events and further condense its action policy. Following up on this idea, the imagination-based *I2A* agent [3] adds a rollout phase for improving the agent's estimates of an action's expected return for long-time horizons. Combining these two ideas, the *Dreamer* agent [8] has shown that latent state models can learn motion control behaviors of varying complexity based on high-dimensional sensory inputs.

In contrast to model-based reinforcement learning, forward model learning agents do not model the return of an action (accumulated reward) but estimate an action's value based on its simulated outcome. State-of-the-art planners such as Monte Carlo Tree Search [9] and Rolling Horizon Algorithms [10] have recently shown applications in game AI [10], [11] and motion control [12]. Since the accuracy of the planning process is dependent on the accuracy of the agent's prediction, the agent's goal is to improve the forward model's accuracy and not the agent's policy. The latter is determined at run-time as a result of the planning process.

III. Forward Model Representation

Learning a model of the environment has been actively studied in model-based reinforcement learning. While many approaches rely on the generality of deep neural networks, they have also shown to not be very sample efficient. Therefore, the model requires many iterations do be trained until a reliable prediction of the upcoming state can be made. Finding a suitable state representation can drastically reduce the required training time as well as improve the model's final prediction accuracy.

For this reason, the focus of our previous work has been to find suitable state representations and model types for improving the efficiency of the model. For graph-like structures this has resulted in the development of the local forward model, which assume a measurable distance of observable sensors. Exploiting a sensor value's independence of sensor values outside its neighborhood yields a considerable reduction in the model's complexity.

Such dependencies among observable sensor variables may also exist in motion-control tasks, but the underlying structure may be unknown. Therefore, we propose to relax our assumption of independence. In case of a decomposed forward model we assume that the next state's sensor values are dependent on the previous state (and possibly its predecessors), but independent of each other. This results in a decomposability of the state transition model into several sub-models, whereas each sub-model predicts the updated sensor value or respectively its changes until the next time step:

$$\begin{aligned} \text{i-th Component Model:} \quad & p(s_t^i \mid s_{t-1}, a_{t-1}, \ldots, s_1, a_1) \\ \text{i-th Differential Model:} \quad & p(s_{t-1}^i - s_t^i \mid s_{t-1}, a_{t-1}, \ldots, s_1, a_1) \end{aligned}$$

Aggregating the predictions of all sensor values and the reward, the agent can predict the outcome of an action. Similarly, a hierarchical structure can be built, in which the environment is first split into several units and the future state values of each unit are predicted independently [6], [13]. Since the result is another state observation, the agent can predict whole action sequences by repeatably applying the learned forward model. Using this, actions can be determined using forward planning methods or a policy can be learned using reinforcement learning on the simulated environment [14].

For motion control, the underlying modeling task is considered a supervised learning problem in which a regressor is trained to predict upcoming states. Several attributes of the model need to be considered upon model selection:

- **model accuracy:** the trained model needs to be accurate for previous observations and future time steps. This is especially relevant for consecutive predictions, since the error will propagate over multiple predictions.
- **model speed:** the trained model needs to be applied very often during the search process. Studies on MCTS have shown that increasing the number or the quality of rollouts can improve the agent's performance [15].
- **model size:** the model's size (in terms of parameter count) can impact the training time and the number of observations required for optimizing the model's parameters. While regression models such as linear regression are the most simple to train, their applicability is quite limited. In contrast, deep neural networks have shown to be flexible, but can require large amounts of training data.
- **model interpretability and reliability:** a characteristic that is often neglected in deep learning approaches is the model's interpretability. While deep reinforcement learning has shown great performance, the black-box nature of deep neural networks may not allow human interpretation of its results. This complicates to measure the reliability or risk of a trained model. In contrast, planning based approaches can transparently summarize the search path and its predicted states. This can be especially important in risk critical applications and allow the computation of confidence bounds.

TABLE I
CHARACTERISTICS OF STUDIED MOTION CONTROL TASKS.

OpenAI Gym Environment	State Space	Action Space Dim	Action Space Type	Rewards
Cartpole-v1	$\mathbb{R}^4$	1	discrete	discrete
Acrobot-v1	$\mathbb{R}^6$	1	discrete	cont.
LunarLander-v2	$\mathbb{R}^8$	1	discrete	cont.
Pendulum-v0	$\mathbb{R}^3$	1	cont.	cont.
Swimmer-v2	$\mathbb{R}^6$	2	cont.	cont.

TABLE II
AGENT PERFORMANCE AFTER 100.000 TRAINING STEPS MEASURES AS THE AGENTS' AVERAGE PERFORMANCE OF THE LAST 10 TRAINING TRIALS.

	CartPole	Acrobot	Lunar Lander	Pendulum	Swimmer
DFM	**376.32**	-272.53	-76.93	**-297.84**	**30.40**
Sarsa	269.49	-983.63	**76.66**	—	—
DQN	325.30	**-131.08**	37.39	—	—
CFM	47.14	-665.60	-237.58	—	—
NAF	—	—	—	-587.93.55	0.34

IV. EXPERIMENTS ON MOTION-CONTROL TASKS

A. Creating a Decomposed Differential Forward Model Agent

We apply the decomposed differential forward model to several motion control tasks. In consideration of the model's requirements, we chose to use decision tree regression to learn a model of the environment dynamics during interaction with it. This regression method has shown to adapt well to a large number of environments while keeping the computation time low and allowing for high parallelization during the prediction phase. An apparent drawback is that the training process cannot be continued after new observations have been registered. We compensate for this by waiting for 100 time-steps until the training process is repeated, which in turn, increases the response time to unforeseen results. We apply the Rolling Horizon Evolutionary Algorithm [10], [16] for action-selection. Its main benefits are its constant computation time (given a fixed computational budget) and its parallelizability of rollouts.

We evaluate the proposed approach using five well-known motion control tasks, which are made available through the OpenAI Gym framework [17]. The proposed decomposed differential forward model (DDFM) will be compared to several popular reinforcement learning algorithms. In case of a discrete action space we chose to use Sarsa [18], DQN [19], and CEM [20]. Whereas for environments with continuous action spaces, we compared our approach with the NAF algorithm [21]. The hyperparameters of each algorithm were tuned by a simple grid-search. The source code of our experiments can be accessed at https://github.com/ADockhorn/Forward-Model-Learning-for-Motion-Control-Tasks.

B. Environments

In the following, we shortly review the environments used in our evaluation. They were selected to represent various amounts of in- and outputs to the agent as well as continuous and discrete action spaces. A comparison of all environments based on these attributes is provided in Table I. Additionally, each environment is illustrated in Figure 1.

The *cart-pole* problem [22] requires the agent to balance a pole by moving a cart back and forth (see Figure 1a). The pole is attached to a cart by an un-actuated joint, which moves along a frictionless track. The agent can apply a discrete force of $+1$ or -1 to the cart. The pole is starting in an upright position and needs to be kept in the range of $[-15, +15]$ degree from vertical. If the pole is falling below that threshold or the cart moves more than 2.4 units from the center, the episode ends. A maximal reward is achieved after balancing the pole for a maximum of 500 time-steps.

The *pendulum problem* is a classic problem in the control literature. In this version of the problem, the pendulum starts in a random position, and the goal is to swing it up so it stays upright. The reward penalizes deviations from the equilibrium and the magnitude of the agent's applied actions.

The *acrobot problem* requires the agent to swing up a pendulum with two links. Similar to the pendulum problem, the agent can apply a discrete rotational force to the joint of the first link. Initially, both links hang downwards, and the goal is to swing the end of the lower link up to a given height indicated by the target line. The reward is the height of the tip of the pendulum.

A more complex environment is represented in the *Lunar Lander* environment in which the agent's task is to land a drone at a landing pad. At each time-step, the agent can fire one of three engines to adjust the drone's position and speed. A reward is gained for successfully landing the drone. Crashing into the surface with too much speed, spending fuel, and landing outside the landing pad decreases the agent's reward.

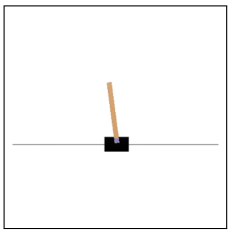
The *Swimmer* environment lets the agent take control of a worm-like robot in a viscous fluid. Its goal is to swim forward by controlling its joints while using minimal control inputs.

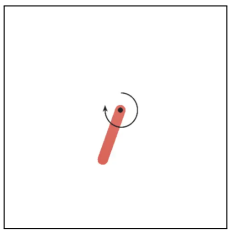
C. Training and Evaluation Setup

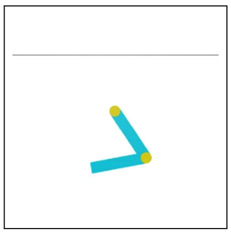
For each environment, we continuously train our agents for a total of 100000 time-steps. During training, we record the agents' reward per episode. We repeat the training process 10 times to get more stable results. The final performance of each model is measured by the average return of the last 10 training episodes.

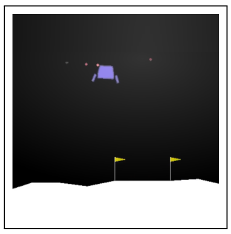
D. Results

Figure 2 shows the agents' performance in each of the five training environments. The results indicate that simple problems, such as the Cart Pole and the Pendulum environment, can efficiently be solved after just a few training steps. In these, the agent outperforms deep-reinforcement learning approaches in terms of sampling efficiency. Nevertheless, the proposed model performs worse environments that require long planning horizons such as Lunar Lander and Acrobot. Due to the simplicity of many environments, the learning curve is very steep and converges quickly. The convergence level seems to


(a) Cart Pole


(b) Pendulum


(c) Acrobot


(d) Lunar Lander

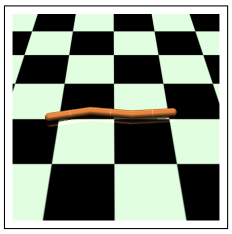
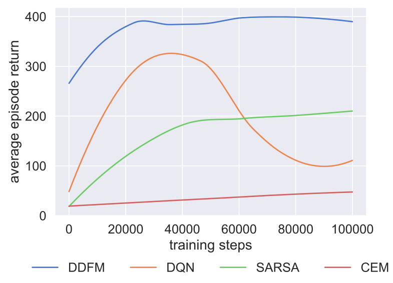
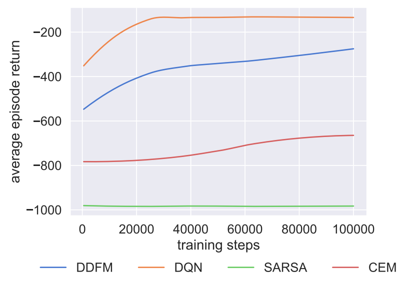

(e) Swimmer

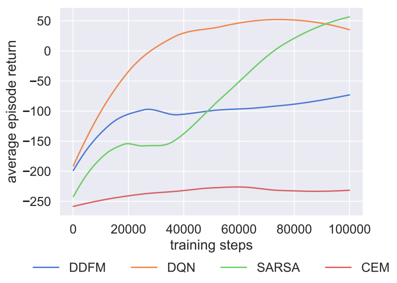
Fig. 1. Six motion-control environments and their representation in the OpenAI Gym framework.



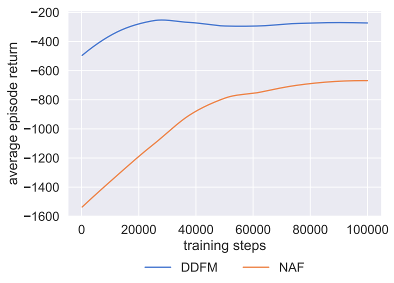
(a) Cart Pole



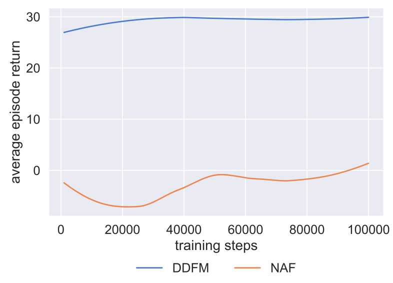
(b) Acrobot



(c) Lunar Lander



(d) Pendulum



(e) Swimmer

Fig. 2. Training comparison of forward model learning and deep reinforcement learning agents. Graphs are showing the average episode return per training steps smoothed using the local regression (loess). Each agent has been trained 10 times for 100000 steps each.

be very much dependent on the applied planning algorithm, e.g. degrading the Cart Pole environment to an episode return of 200 in case of shorter planning horizons.

V. Discussion

Overall, the decomposed differential forward model has shown great learning performance in simple environments. In the following, we want to highlight some problems of the proposed approach and how they may be overcome in the future:

a) Planning horizon dilemma: A common problem of planning algorithms is the choice of the planning horizon. While increasing the search can improve the accuracy of a planning agent's reward estimation, it also exponentially increases the number of possible states to be analyzed. In the case of forward model learning agents, the planning horizon is further restricted by the accuracy of the learned forward model. In contrast to using the true environment dynamics, predictions of a learned forward model become less accurate with increasing search depth. For choosing a suitable rollout length for a trained model we analyzed the model's prediction error over the length of multiple predicted action sequences. This can hint a suitable parameter combination but has shown to be too costly to repeat throughout the model learning process. Dynamic measurements of the model's confidence in its prediction might help in improving the agent's performance. Similarly, algorithms like MCTS that do not use a fixed planning horizon may be beneficial.

b) Exploration vs. Exploitation: During training the agent needs to optimize the learned model as good as possible but also needs to focus on beneficial actions. Therefore, the agent needs to focus on actions that maximize our chances of succeeding, while adding new data to the forward model for improving its accuracy. In the present study, we let the agent solely decide based on its predicted reward. However, it may show beneficial to include exploring actions during training. This may not just improve the model's accuracy, but on the long run, also make the agent more robust to unlikely or new situations.

VI. Conclusion and Future Work

In this paper, we proposed the decomposed differential forward model learning agent. The agent consists of a forward model learning component, for which the decision tree regressor was used, and a planning component, for which we used a rolling horizon evolutionary search. We evaluated the approach based on five motion-control tasks, of which the simple scenarios could be solved faster than with deep-reinforcement learning approaches. The decomposed nature of the prediction allows for efficient parallelization. Similarly, the evaluation of multiple rollouts can be parallelized without much additional effort. Both assure a high number of samples to be tested before deciding for an action. However, the prediction, as well as the search component of the agent, are prone to problems regarding the environment. In case the learned forward model is not able to fit the environment with high accuracy, the result of predicted rollouts can be misleading and cause inefficient actions to be selected more frequently. Overall, the approach has not shown to scale well with higher dimensional state- and action-spaces which may be overcome by more sophisticated model learning and planning components.

Even if reinforcement-learning has shown to be able to solve many motion-control tasks, these methods still require large amounts of training data. Since the proposed planning based approaches have shown great learning performance during the first steps of acting in a new environment, the next step will be to combine these two algorithmic schemes in a single agent. Such an agent may dynamically choose to either trust the reinforcement learner's value model or to rely on a learned forward model for using a planning-based approach. Similarly, the planner could benefit from the reinforcement-learner's policy during the search phase. Since curiosity-driven learning and other intrinsic reward schemes have already shown promising results for improving reinforcement-learning agents, similar improvements could be achieved when being applied to forward model learning agents.

References

[1] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[2] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking Model-Based Reinforcement Learning," pp. 1–25, 2019. [Online]. Available: http://arxiv.org/abs/1907.02057

[3] S. Racanière, T. Weber, D. P. Reichert, L. Buesing, A. Guez, D. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, R. Pascanu, P. Battaglia, D. Hassabis, D. Silver, and D. Wierstra, "Imagination-augmented agents for deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 5691–5702, 2017.

[5] S. M. Lucas, A. Dockhorn, V. Volz, C. Bamford, R. D. Gaina, I. Bravi, D. Perez-Liebana, S. Mostaghim, and R. Kruse, "A Local Approach to Forward Model Learning: Results on the Game of Life Game," in *2019 IEEE Conference on Games (CoG)*. IEEE, aug 2019, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/8848002/

[4] A. Dockhorn, S. M. Lucas, V. Volz, I. Bravi, R. D. Gaina, and D. Perez-Liebana, "Learning Local Forward Models on Unforgiving Games," in *2019 IEEE Conference on Games (CoG)*. London: IEEE, aug 2019, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/8848044/

[6] A. Dockhorn and D. Apeldoorn, "Forward Model Approximation for General Video Game Learning," in *Proceedings of the 2018 IEEE Conference on Computational Intelligence and Games (CIG'18)*. IEEE, aug 2018, pp. 425–432. [Online]. Available: https://ieeexplore.ieee.org/document/8490411/

[7] D. Ha and J. Schmidhuber, "World Models," 2018. [Online]. Available: http://arxiv.org/abs/1803.10122

[8] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to Control: Learning Behaviors by Latent Imagination," pp. 1–20, 2019. [Online]. Available: http://arxiv.org/abs/1912.01603

[9] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, mar 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6145622/

[10] R. D. Gaina, J. Liu, S. M. Lucas, and D. Pérez-Liébana, "Analysis of Vanilla Rolling Horizon Evolution Parameters in General Video Game Playing," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017, vol. 10199 LNCS, pp. 418–434. [Online]. Available: http://link.springer.com/10.1007/978-3-319-55849-3{_}28

[11] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, jan 2016. [Online]. Available: http://www.nature.com/articles/nature16961

[12] P. Clary, P. Morais, A. Fern, and J. Hurst, "Monte-Carlo planning for agile legged locomotion," *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, vol. 2018-June, no. Icaps, pp. 446–450, 2018.

[13] A. Dearden and Y. Demiris, "Learning forward models for robots," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1440–1445, 2005.

[14] D. Nguyen-Tuong and J. Peters, *Model learning for robot control: A survey*, 2011, vol. 12, no. 4.

[15] A. Dockhorn, C. Doell, M. Hewelt, and R. Kruse, "A decision heuristic for Monte Carlo tree search doppelkopf agents," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, nov 2017, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/8285181/

[16] R. D. Gaina, S. M. Lucas, and D. Perez-Liebana, "Rolling horizon evolution enhancements in general video game playing," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, aug 2017, pp. 88–95.

[17] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning*, 2nd ed. Cambridge: The MIT Press, 2018.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. a. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, feb 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[20] I. Szita and A. Lorincz, "Learning tetris using the noisy cross-entropy method," *Neural Computation*, vol. 18, no. 12, pp. 2936–2941, 2006.

[21] S. Gu, T. Lillicrap, U. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," *33rd International Conference on Machine Learning, ICML 2016*, vol. 6, pp. 4135–4148, 2016.

[22] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, 1983.