Double Deep Reinforcement Learning

Josué Kiefer
Klaus Dorer
Institute for Machine Learning and Analytics
Offenburg University, Germany
jkiefer6@stud.hs-offenburg.de, klaus.dorer@hs-offenburg.de

***Abstract*—In many application areas, Deep Reinforcement Learning (DRL) has led to breakthroughs. In Curriculum Learning, the Machine Learning algorithm is not randomly presented with examples, but in a meaningful order of increasing difficulty. This has been used in many application areas to further improve the results of learning systems or to reduce their learning time. Such approaches range from learning plans created manually by domain experts to those created automatically. The automated creation of learning plans is one of the biggest challenges.**

In this work, we investigate an approach in which a trainer learns in parallel and analogously to the student to automatically create a learning plan for the student during this Double Deep Reinforcement Learning (DDRL). Three Reward functions, Friendly, Adversarial, and Dynamic based on the learner's reward are compared. The domain for evaluation is kicking with variable distance, direction and relative ball position in the SimSpark simulated soccer environment.

As a result, Statistic Curriculum Learning (SCL) performs better than a random curriculum with respect to training time and result quality. DDRL reaches a comparable quality as the baseline and outperforms it significantly in shorter trainings in the distance-direction subdomain reducing the number of required training cycles by almost 50%.

***Index Terms*—Deep Reinforcement Learning, Curriculum Learning, Proximal Policy Optimization (PPO), kick learning**

I. Introduction

In many application areas, the use of curriculum learning has succeeded in improving the performance of machine learning systems and reducing learning time [16]. In curriculum learning, the machine learning algorithm does not receive its examples in a random, but in a meaningful order where the difficulty is gradually increased [4]. Appropriate approaches range from learning plans created manually by domain experts to their automatic generation by a trainer learning with Reinforcement Learning (RL) [16]. One of the main challenges here is the automatic generation of the learning plan in an unknown domain and an unknown agent [3].

Curriculum learning has also achieved success in the SimSpark simulated soccer environment. For example, in [11] and [10] dribbling was learned with DRL in various forms. In both works, the importance of a manually created learning plan is emphasized as elementary for learning success.

In this work, an intelligent trainer automatically creates a learning plan for an agent learning to kick using DRL during the student's training. Two approaches are taken here. In Double Deep Reinforcement Learning (DDRL), the trainer learns analogously to its student using DRL. Statistical Curriculum Learning uses the last achieved goodness of the individual parameter combinations to determine probabilities for their selection by random sampling.

The remaining of the paper is organized as follows: Section II presents related work on intelligent trainers and automated curriculum generation. Section III explains the approach and setup used for this evaluation. Section IV provides background of the SimSpark domain used for learning. Section V presents in depth results to validate our approach. Finally, section VI concludes and shares ideas for future work.

II. Related Work

Learning behaviors end to end using DRL is a heavily researched area. One domain for this is the SimSpark simulated soccer environment[1] initially initiated by [12]. Successful attempts to dramatically improve the walking speed of the simulated NAO robots have been reported by [1], [2], [9].

The learning of static kicks and those in motion using DRL and the Proximal Policy Optimization algorithm (PPO) [13] was studied by Teixeira et al. in [15]. In the case of the static kick, an average distance of 8.53 m could be achieved. In less than 1 % of the cases the ball was not hit. The authors emphasize in particular the time improvement of the kick execution. This was on average 0.38 s and is a clear improvement to the previous kicks. The kicks in motion were divided into different subproblems, also here the results are very good especially regarding the temporal execution.

Spitznagel et al. [14] analyzed the learning of kicks using proximal policy optimization in the SimSpark simulated soccer environment. Compared to the aforementioned work, this work used the toe version of the NAO robot and learned kicks where the player does not almost always fall down after execution. Furthermore, the kicks are also multidirectional and tolerant to a wide range of relative ball positions. As a result, in addition to improving reliability, an expansion of the kickable area and the avoidance of falls, the overall level of play could also be significantly improved. Furthermore, the kick could be parameterized with the desired distance and direction. Which distance and direction to train has been randomly selected in this work.

A similar approach to DDRL is taken by Dong et al. in [5]. The authors extend one RL instance with a second RL instance

979-8-3503-0121-2/23/$31.00 ©2023 IEEE

[1]https://gitlab.com/robocup-sim/SimSpark

as an intelligent trainer. They call this approach reinforcement on reinforcement (RoR). The first instance is a model-based DRL system. From the model-based approach, there is a significant manual effort to improve the hyperparameters for optimizing the joint control policy, learning the system dynamics, and sampling data from two sources. Therefore and different to this work, the authors use the second instance as an intelligent trainer that learns the optimal hyperparameter configuration for the first instance. Various RL algorithms can be used for the trainer in this regard. The architecture was applied to several OpenAI-Gym problems. As a result, up to 56 % of the sampling cost could be saved.

Automatic learning plan generation for curriculum learning has been the subject of several works. For example, in [6], Graves et al. present a concept for automatically generating a learning plan using a measure of the amount a network learns from each sample as a reward for a non-stationary multi-armed bandit algorithm. In several experiments, the approach has been shown to speed up learning to a satisfactory quality of result, and in some cases to halve it. Zhao et al. [17] propose a framework for data selection based on the Actor-Critic model, which aims to learn a curriculum to improve a machine translation neural network (NMT). The Critic predicts the expected model performance based on a given sample. The Actor learns to select the best sample from a random set of samples. In comparison, Kumar et al. in [7] use a Deep Q-Network (DQN) to automatically generate a curriculum for solving an NMT task. Both approaches improved the performance of the NMT and outperformed several baseline methods.

Matiisen et al. [8] propose a Teacher-Student Curriculum Learning Framework (TSCL) for performing automatic curriculum learning, which can be used for both supervised learning and RL tasks. They formalize the TSCL as a partially observable Markov decision process (POMDP). Although POMDPs are usually solved with RL algorithms, the authors resort to simpler heuristics. The reason is that they wanted to train the student in a single training episode. They describe a set of algorithms that preferentially learn the tasks on which the student makes the most progress. The performance of the framework is evaluated on two tasks with a discrete parameterization. The framework was able to outperform a hand-crafted learning plan in both tasks. Furthermore, in one task, it was able to solve a problem that could not be solved with a learning plan created by hand. Additionally, learning was also significantly faster in both cases.

III. Approach

In contrast to the aforementioned works, this work uses the DDRL to combine two DRL instances. An intelligent trainer is used to train a kick learning DRL student in the SimSpark simulated soccer environment. Here, the trainer does not learn the optimal hyperparameter configuration of its student, but automatically creates a learning plan for the student during its own training. No heuristics are used in this process. The trainer is trained analogously to its student with the same DRL algorithm. Its training also consists of several episodes. The trainer's reward is directly based on that of the student. The trainer's reward functions are not aimed at the amount the network learns from the samples or the greatest possible learning progress of the subtasks, but at the absolute goodness of the subtasks achieved so far. With statistical curriculum learning, a statistical method for the generation of an automatic learning plan is additionally investigated. Here, the probability that a parameter combination is trained depends on the previously achieved goodness of the learner for this combination.

In DDRL, the trainer is trained analogously and in parallel to its student using DRL. While training the student, it learns which training parameter combinations add the most value to the student's training progress. Figure 1 shows that the trainer receives state information from the environment about the student's learning progress. This information is used as input for the neural network, which determines the next training parameters based on this information. These training parameters define the next environment setup/ training task for the student. This is followed by the next training step of the student. In addition to the student, the trainer also receives feedback for the quality of its action. The trainer's reward is based on that of the student. By rewarding the choice of training parameters that the student is either not yet good at or already good at, a learning plan is created for the student during its own training.

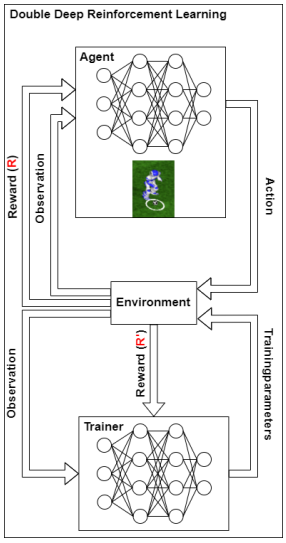

Fig. 1. Principle of DDRL.

In statistical curriculum learning, by comparison, no neural network is trained. The training parameters are determined in each episode by random sampling. The selection probability of a parameter combination depends on the last achieved trainer reward of this combination and increases with a higher reward.

A. Observation Space

The trainer's observation space is inspired by Matiisen's et al. simple formalization of the partially observable Markov

Decision Process [8]. The resulting observation contains the last achieved rewards of the student on all possible training parameter combinations.

$$o_{zt} = \frac{\sum_{i=1}^{n} \hat{R}_{izt_x}}{n} \tag{1}$$

o_{zt} is the observation entry of the training parameter combination $z = a_z$ in the current episode t. a_z corresponds to the action representing the parameter combination. The training of the student is executed with n threads in parallel, so the value is the average of n student rewards ($\hat{R}$) of the combination z from the episode t_x in which it was last trained. Figure 3 gives an idea on the dimensionality of the observation space.

B. Action Space

The dimension and concrete entries of the trainer's action space depend on the variable training parameters. The action contains the training parameter combination that the student should train next. For kicking in the SimSpark simulated soccer environment, four variable training parameters (desired distance, direction, x- and y-coordinate of the player position relative to the ball) result in a four-dimensional action space. However, for a better representation of the results, two separate training sessions are performed, each with only two variable parameters. The action contains the desired distance and direction to be trained in one training and the x and y coordinate for the player position relative to the ball in another.

C. Student Reward Function

Since kicking is a relatively quick behavior (18 cycles = 360 ms), the reward function did not contain continuous reward, but only episode reward. The episode reward for the multi-directional kicking experiments (student) is a mixture of the relative distance to the desired kick position and a penalty for falling: $reward = a_1 * (d_0 - d)/d_0 - a_2 * (1 - s/88)$, where d_0 is the desired kick distance and d the distance of the (estimated) ball end position to the desired kick position and a_1, a_2 are parameters balancing both penalties tuned to 100 and 25 respectively in earlier learning runs. To save time, an episode is stopped if the agent falls or if after 88 cycles (approx. 1.5s) after triggering the kick it is highly likely that the agent is stable. In both cases, the achieved kick position is estimated as the 8s future ball position.

D. Friendly Reward Function

The Friendly-Reward function of the trainer is the higher the better the result of the student is. This is to give preference to parameter combinations that already work well. The trainer's reward is therefore equal to the student's reward. Since the student is trained with several threads at the same time, an average value is used.

$$R_F = \frac{\sum_{i=1}^{n} \hat{R}_i}{n} \tag{2}$$

Here, $\hat{R}_i$ is the reward of a student thread after an episode with the current parameters with n the total number of threads.

E. Adversarial Reward Function

The Adversarial strategy is used to preferentially train weak points of the student. The Adversarial Reward Function therefore evaluates the trainer better, the worse the result of the student is on a subtask. For this reason, the previously described friendly reward function is extended by a factor -1.

$$R_A = \frac{\sum_{i=1}^{n} (-1)\hat{R}_i}{n} \tag{3}$$

F. Dynamic Reward Function

The Dynamic-Reward function mixes the Friendly and Adversarial strategies. It starts with Friendly and switches to Adversarial when a variable threshold (b_1) of total completed episodes (b) is reached.

$$R_D = \begin{cases} R_F, & for\ b < b_1 \\ R_A, & for\ b \geq b_1 \end{cases} \tag{4}$$

G. Random Episodes

The trainer's observation of an episode contains the last achieved rewards of the student for each possible parameter combination (subtask). It therefore does not represent the current state of the student, but is merely a memory. However, if parameter combinations are no longer trained because they had a significantly lower benefit for the trainer at a point in time, the student unlearns them. The trainer, however, does not notice this unlearning, because the entry in the observation is only updated if the parameter combination is trained. The authors in [8] call a similar behavior "problem of forgetting". To mitigate this problem and to update the observation from time to time, the training parameters are randomly chosen after every 100 episodes.

H. Training Setup

The setup of the training is shown in Figure 2. The yellow boxes extend the existing student architecture with the intelligent trainer. The Python component is a framework implementation for launching and configuring the stable baselines[2] PPO implementation PPO2, which is used for both the student and the trainer. A TCP interface is provided for communication with the component.

For the student, the multiprocess functionality of the PPO2 implementation is used. For this, an environment instance is created for each thread on Java side and on Python side. All these threads train the same network model. In the Java part, n threads are created for training the student. Each runner independently executes its episodes like kicks in the specific case. Before each episode, the learning parameters such as the desired distance and direction of the kick are prepared by the trainer. For the DDRL, the trainer PPO2 Python component is

[2]https://github.com/hill-a/stable-baselines

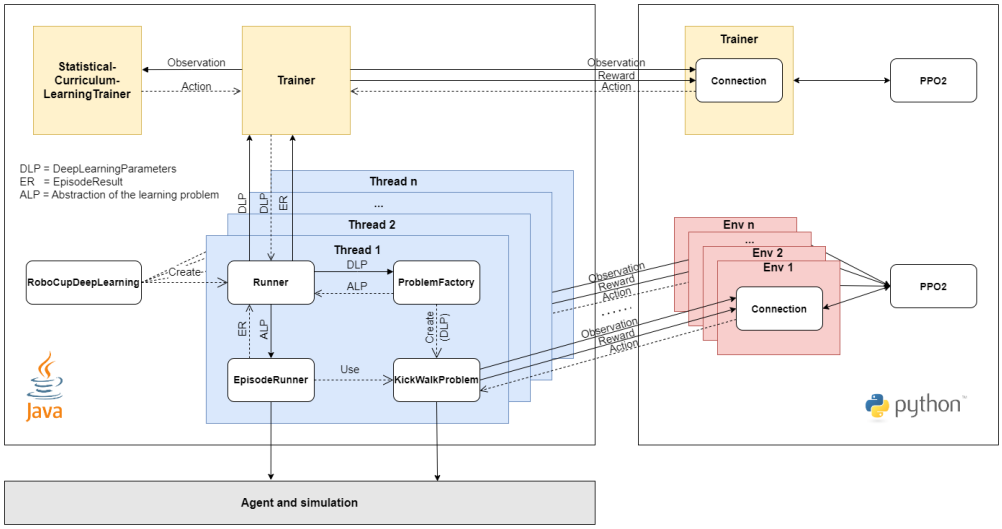

Fig. 2. Training Setup.

used. For Statistical Curriculum Learning, the corresponding Java component is used. In each episode, a Java representation of the concrete learning problem (KickWalkProblem) is generated. This abstraction of the learning problem (ALP) takes into account the learning parameters prepared by the trainer and performs appropriate initializations based on them.

Each episode consists of several steps. This means, for example, that a kick is not executed with only one action, but consists of a sequence of actions of the agent. In each step, the ALP determines the observation and the reward of the last action and transmits both to the Python component. The Python component then responds with the next action for the player. The episode reward is also transmitted to the Python component by the ALP. The Runner transmits the result of the student to the trainer component, which determines the trainer reward and the new observation of the trainer. The Runner then starts a new episode, if necessary.

IV. Learning Domain

Learning is performed in the RoboCup 3D soccer simulation environment which is based on SimSpark. The simulator is able to run 22 simulated NAOs in real-time on reasonable CPUs. It is used as competition platform for the RoboCup 3D soccer simulation league[3]. For training, only a single robot is on the field.

The NAO robots used here have 24 degrees of freedom. Two motors control the two toe joints, one on each leg. The action space of the student DRL directly controls these motors (for kicking only the 14 leg motors). The student's observation space is a mixture of raw sensor information from joint encoders, IMU, foot force sensors and processed information like ball position or the player's up vector.

[3]https://www.robocup.org/leagues/23

V. Results

In order to evaluate the described approach, extensive simulation runs of more than 7000 server hours (Intel Xeon E5-2630 v4 @ 2.2 GHz, 10 Cores) were conducted and are documented in this section.

A. Quality Metrics

The quality of a learning run has to be measured over the entire task-space of the subdomain under investigation. For the direction-distance subdomain, for example, this means to define a quality measure over all 18 x 14 directions / distances. Although long kicks and straight kicks are somewhat more important in this domain, we decided to use the unweighted average reward over all combinations of subtasks. The quality of a subtask is the average student reward of 100 repetitions of the subtask after finishing learning. Figure 3 shows the single subtask rewards for the direction-distance subdomain using as a baseline a random subtask selection. The overall quality of the baseline is 85.8.

It is important to stress, that the learning curves of the student training are deceiving with respect to measuring the learning success. This is illustrated with Figure 4. The students advised with a friendly trainer strategy collected significantly more reward after 3500 episodes compared to the student under adversarial trainer strategy. However, with respect to the overall task of kicking into various directions and distances, the adversarial student performed significantly better. This is because the friendly trainer strategy prefers those subtasks for which the student gets high rewards, while the adversarial trainer strategy focuses on those subtasks that do not yet gain high rewards.

B. Statistic Curriculum Learning

Table I shows the results of the friendly, adversarial and dynamic strategy when using SCL over an increasing amount of episodes (relative to a full training length of two days).

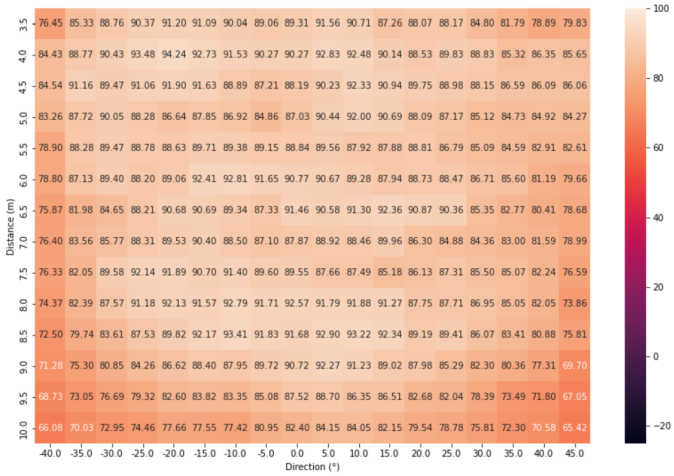

Fig. 3. Average-100 reward for kicking into the desired direction (x-axis) and distance (y-axis) of a random trainer strategy (baseline).

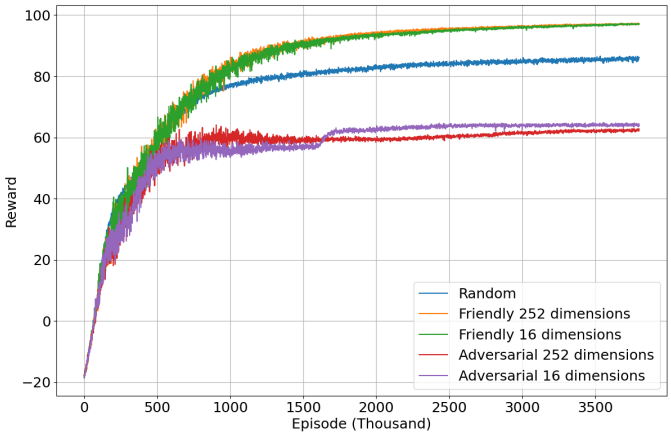

Fig. 4. Student learning curves for students advised with a friendly, adversarial and random trainer.

Dynamic switched from friendly to adversarial after 75% of the specified episodes as found to work best in earlier experiments.

As can be seen, an adversarial SCL strategy reaches the quality of a random baseline learning with 10% less episodes and improves the overall result by 1%. Figure 5 shows the detailed comparison of the performance for this subdomain. Blue fields mark kick distances/directions where SCL is better, red, where the baseline is better. From Figure 6 we can see that the adversarial strategy focused on areas in which the agent performed below average on the baseline.

TABLE I
SCL AND DDRL RESULTS FOR THE DISTANCE-DIRECTION SUBDOMAIN.

Epi-sodes	Base-line	Friendly		Adversarial		Dynamic	
		SCL	DDRL	SCL	DDRL	SCL	DDRL
100%	85.80	84.34	86.03	86.81	84.42	86.06	86.15
90%	84.04	83.23	85.61	86.00	83.78	84.80	85.44
75%	83.44	82.73	85.31	85.03	83.73	84.11	85.20
50%	81.94	81.67	85.38	82.82	82.65	82.75	83.56

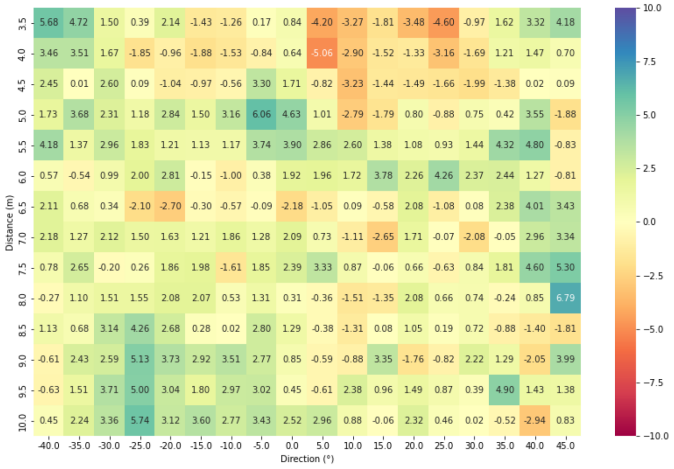

Fig. 5. SCL results in the distance-direction domain as a delta to the baseline.

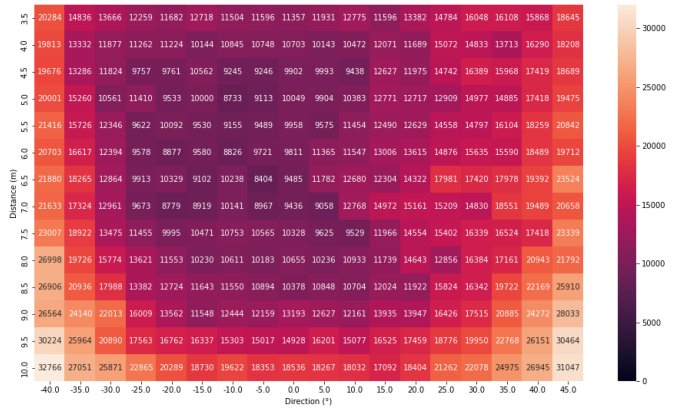

Fig. 6. Number of explorations of the adversarial SCL in the distance-direction subdomain.

C. Double Deep Reinforcement Learning

The results of having the trainer also use DRL to learn a trainer strategy are shown in Table I. Dynamic, again, switched from friendly to adversarial after 75% of the specified episodes. The most successful strategy is a dynamic strategy when running the full two days training. However, the friendly strategy almost reached the results of the baseline with only 50% of the cycles and still performing slightly better than the baseline after 100% of all episodes.

Figure 7 shows the detailed comparison of the DDRL performance for this subdomain. Blue fields mark kick distances/directions where DDRL is better, red, where the baseline is better. For the short and medium range kicks, the training worked better with DDRL. It is still under investigation, why especially the long straight kicks dropped in performance in this setup. One possible reason is that PPO creates many border values (-1 and 1) in the action space of the trainer especially in the early phase when the student's rewards are almost equally bad in the whole task domain (see Figure 8). This probably prevents even better results.

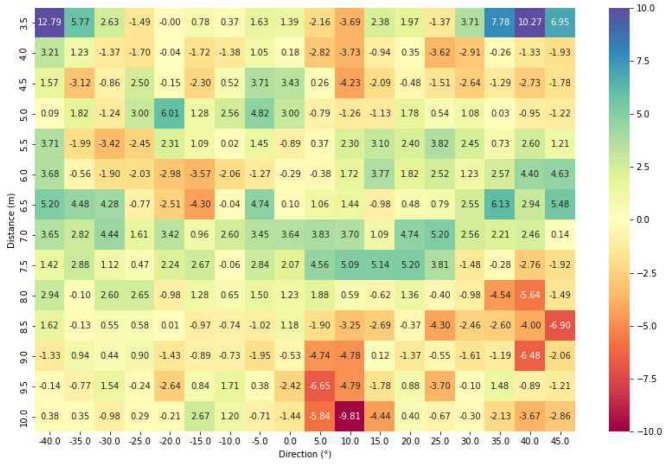

Fig. 7. DDRL results as a delta to the baseline using dynamic strategy.

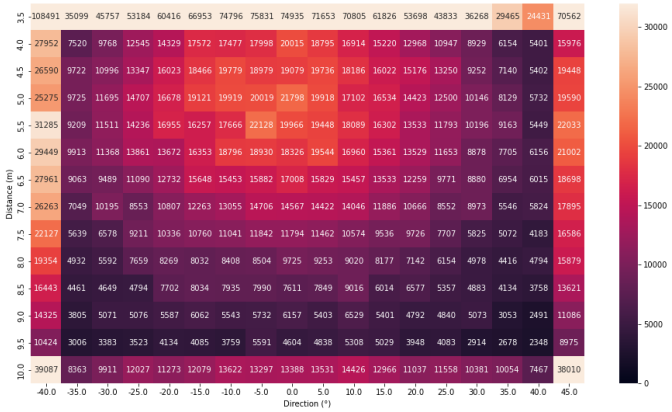

Fig. 8. Number of explorations of the dynamic DDRL.

VI. Conclusion and Future Work

In this paper we have shown that DRL can be used by a trainer to automatically create learning curriculums for DRL student tasks. This Double Deep Reinforcement Learning with trainer and student using DRL lead to significantly faster training in the distance direction kicking subdomain. Two important insights are that the learning curves alone are deceiving with respect to the quality of the overall learning task and that some randomness in the curriculum is still necessary to counteract the degrading of quality for one subtask while others are learned.

One problem that is still under investigation is, that PPO creates many border values (-1 and 1) in the action space of the trainer especially in the early phase when the student's rewards are almost equally bad in the whole task domain. Future work should investigate in using alternative sampling functions than those used in PPO to have a more evenly sampled action space. Alternatively, other on-policy or off-policy algorithms could be used instead of PPO.

Acknowledgment

We thank the magmaOffenburg team for the implementation of the stable baselines environment wrapper used in this work.

References

[1] M. Abreu, N. Lau, A. Sousa and L. P. Reis (2019). Learning low level skills from scratch for humanoid robot soccer using deep reinforcement learning. In Proceedings of 2019 International Conference on Autonomous Robot Systems and Competitions (ICARSC). IEEE.

[2] M. Abreu, L. P. Reis, N. Lau (2019). Learning to Run Faster in a Humanoid Robot Soccer Environment Through Reinforcement Learning. In RoboCup 2019: Robot World Cup XXIII (pp. 3–15). Springer International Publishing.

[3] A. Bassich and D. Kudenko (2019). Continuous curriculum learning for reinforcement learning. In Proceedings of the 2nd Scaling-Up Reinforcement Learning (SURL) Workshop IJCAI (2019).

[4] Y. Bengio, J. Louradour, R. Collobert und J. Weston (2009). Curriculum Learning. In Proceedings of the 26th Annual International Conference on Machine Learning, Ser. ICML '09, Montreal, Quebec, Canada: Association for Computing Machinery, 2009, S. 41–48, ISBN: 9781605585161. DOI: 10.1145/1553374.1553380.

[5] L. Dong, Y. Li, X. Zhou, Y. Wen and K. Guan (2021). Intelligent Trainer for DynaStyle Model-Based Deep Reinforcement Learning. IEEE Transactions on Neural Networks and Learning Systems, Jg. 32, Nr. 6, S. 2758–2771, 2021. DOI: 10.1109/TNNLS.2020.3008249.

[6] A. Graves, M. G. Bellemare, J. Menick, R. Munos and K. Kavukcuoglu (2017). Automated Curriculum Learning for Neural Networks. arXiv:1704.03003. DOI: 10.48550/ARXIV.1704.03003.

[7] G. Kumar, G. Foster, C. Cherry and M. Krikun (2019). Reinforcement Learning based Curriculum Optimization for Neural Machine Translation. arXiv:1903.00041. DOI: 10.48550/ARXIV.

[8] T. Matiisen, A. Oliver, T. Cohen and J. Schulman (2017). Teacher-Student Curriculum Learning. arXiv:1707.00183. DOI: 10.48550/ARXIV.1707.00183.

[9] L. Melo and M. Maximo (2019). Learning Humanoid Robot Running Skills through Proximal Policy Optimization. arXiv:1910.10620.

[10] A. F. V. Muzio, M. R. O. A. Maximo und T. Yoneyama (2022). Deep Reinforcement Learning for Humanoid Robot Behaviors. Journal of Intelligent & Robotic Systems, Jg. 105, Nr. 1, S. 12, Apr. 2022, ISSN: 1573-0409. DOI: 10.1007/s1 0846-022-01619-y.

[11] A. F. V. Muzio, M. R. A. Maximo und T. Yoneyama (2020). Deep Reinforcement Learning for Humanoid Robot Dribbling. In 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), 2020, S. 1–6. DOI: 10 . 1109 / LARS/SBR/WRE51543.2020.9307084.

[12] O. Obst and M. Rollmann (2005). SPARK - A Generic Simulator for Physical Multiagent Simulations. Computer Systems Science and Engineering, 20(5).

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov (2017). Proximal Policy Optimization Algorithms. arXiv:1707.06347.

[14] M. Spitznagel, D. Weiler and K. Dorer (2021). Deep Reinforcement Multi-Directional Kick-Learning of a Simulated Robot with Toes. In 2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2021, S. 104–110. DOI: 10.1109/ICARSC52212.2021.9429811.

[15] H. Teixeira, T. Silva, M. Abreu and L. P. Reis (2020). Humanoid Robot Kick in Motion Ability for Playing Robotic Soccer. 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Ponta Delgada, Portugal, 2020, pp. 34-39, doi: 10.1109/ICARSC49921.2020.9096073

[16] X. Wang, Y. Chen and W. Zhu (2020). A Survey on Curriculum Learning. arXiv:2010.13166. DOI: 10.48550/ARXIV.2010.13166.

[17] M. Zhao, H. Wu, D. Niu and X. Wang (2020). Reinforced Curriculum Learning on Pre-Trained Neural Machine Translation Models. Proceedings of the AAAI Conference on Artificial Intelligence, Jg. 34, Nr. 05, S. 9652–9659, Apr. 2020. DOI: 10.1609/aaai.v34i05.6513.