Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Directed Curriculum Generation

Author:
Thomas Edlich

Supervisor:
Aldo Faisal

Submitted in partial fulfillment of the requirements for the MSc degree in Computing (Machine Learning) of Imperial College London

September 2018

Abstract

Learning complex tasks, such as navigation or object manipulation from sparse rewards, is a great challenge for reinforcement learning as the agent needs to randomly reach the desired goal in order to receive a useful feedback signal. Curriculum learning (CL) is one approach which aims to alleviate the challenge of learning such tasks by breaking it down into smaller more manageable subtasks, which are learned sequentially. By doing this, the agent is able to build upwards from smaller successful attempts and apply this to more difficult tasks. Curriculum learning has often been applied in machine learning and especially reinforcement learning and has usually required a human expert to define a sequence of tasks. Such manual curriculum design can however be tedious and therefore an automatic curriculum generation is highly desirable and recently a few such methods have been proposed. However, these methods progress from one task to another solely based on difficulty and performance but neglect the practical usefulness of a task and therefore generate an undirected curriculum.

In this thesis, we are examining approaches to automatically generate a curriculum which is directed towards a pre-defined target region. We propose two approaches, Directed GoalGAN and Directed Hindsight Experience Replay, which build on two existing curriculum generation approaches, GoalGAN and Hindsight Experience Replay, and extend them with a process that samples the goal states for the next iteration based on their usefulness. The measure of usefulness is hereby learned using a discriminator network, inspired by Generative Adversarial Networks, which is trained to distinguish between states from the target region and other states.

We tested our first approach, Directed GoalGAN, on two continuous grid world problems. Our results show that using our directed extension indeed helps the algorithm to reach the desired goal region slightly faster than the original GoalGAN method. Unfortunately, we detected that the GoalGAN method is not suitable for high dimensional goal regions due to the inaccuracy of GANs and the sample-inefficiency of reinforcement learning algorithms in high-dimensional tasks with sparse rewards. However, HER is perfectly suitable for high-dimensional goal spaces. Therefore, we tested our Directed HER approach on both grid world and two robotics tasks, hand movement and in-hand block manipulation. We observed that our Directed HER approach is for some problems capable of learning faster than normal HER.

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Aldo Faisal, for his guidance and support during this project. This thesis would not have been possible without his critical feedback and his advice.
Furthermore, I want to express my love and gratitude for my parents who always supported me and made my studies at Imperial possible.
Last but not least, I want to extend my thanks to anyone, who has supported me during this project, either through technical discussions or emotional support.

Contents

List of Figures

Chapter 1

Introduction

When humans learn how to perform a certain task, they usually do so by breaking the task down into smaller subtasks and then first learn how to master easier tasks prior to progressing to more difficult ones. Such stage-wise, progressive learning is often called curriculum learning (CL) and has also shown to be a promising approach in machine learning in order to accelerate the learning of complex tasks (see e.g. (Bengio, Louradour, Collobert & Weston, 2009; Sanger, 1994; Zaremba & Sutskever, 2014)).

Curricula in human learning are often provided by a higher entity such as teachers or coaches. Similar to this, first applications of curriculum learning in machine learning required a pre-specified set of tasks which should be trained on sequentially (see e.g. (Karpathy & Van De Panne, 2012)). However, curricula in human learning often also arise intrinsically without directives from the outside, e.g. when first learning to walk. Recently, there have been several approaches in reinforcement learning which aim to automate the curriculum generation for the agent in order to get rid of the tedious manual design of curricula (see (Molchanov, Hausman, Birchfield & Sukhatme, 2018; Florensa, Held, Wulfmeier, Zhang & Abbeel, 2017; Sukhbaatar et al., 2017; Held, Geng, Florensa & Abbeel, 2018)).

Such approaches usually start by learning tasks which are easy to achieve for an agent without any knowledge of the environment such as transitioning to nearby states. Once the performance of the agent improves, these algorithms then generate slightly more complex tasks building on the easier tasks learned in the previous iteration.

However, these methods only consider the difficulty of tasks but not how useful the new tasks would be in order to achieve a final goal. This is a strong contrast to human learning, where curricula usually first learn the basics but then sooner or later specialise in only some directions rather than progressing in all possible directions. In this work, we propose a way of generating a curriculum for reinforcement learning agents which is directed towards a given target region. We achieve this

by extending two existing curriculum generation methods, namely GoalGAN (Held et al., 2018) and Hindsight Experience Replay (HER) (Andrychowicz et al., 2017), with a sampling process which selects from a set of possible goal states the ones which are most similar to the given states from the target region. As pre-defined similarity measures like the ℓ_2-norm are likely to fail in high-dimensional problems, we learn the similarity measure by training a discriminator, inspired by Generative Adversarial Networks, to distinguish between target states and all other states.

Our experiments show that our approaches indeed generate a curriculum which expands towards the target region. Furthermore, our Directed GoalGAN technique is able to learn how to reach the target region faster than the undirected method on two continuous gridworld problems. However, we had to observe that GoalGAN methods are unsuitable for high-dimensional goal spaces due to the inaccuracy of GANs and the sample-inefficiency of RL algorithms in high-dimensional spaces. Our proposed Directed HER approach also outperforms the original HER method on a grid-world task and on a complex in-hand manipulation task. However, on a hand movement task, we observed slightly worse performance for Directed HER compared to the standard HER algorithm.

This report is structured as follows. After this introduction, we progress to Chapter 2 to review some necessary prerequisites of reinforcement learning and generative adversarial networks. Herein, we furthermore present related approaches regarding curriculum generation in reinforcement learning. Afterwards, we provide details of our contributed approaches in Chapter 3. Experimental results are then shown in Chapter 4 before giving a final conclusion and possible directions of future work in Chapter 5.

Chapter 2

Background and Related Work

Our problem falls into the domain of reinforcement learning. In the following sections we will therefore present the fundamental definitions of reinforcement learning as well as some of the most popular reinforcement learning algorithms and techniques. As an integral part of our proposed approaches was inspired by the training process of generative adversarial networks (GANs), we will furthermore present the theory behind them and present some applications of GANs in reinforcement learning. We will also distinguish our approach from existing work in the field of curriculum generation and give detailed explanations of the two main curriculum learning approaches we are building on, namely GoalGAN and Hindsight Experience Replay.

2.1 Reinforcement Learning

Reinforcement learning is concerned with teaching an agent a specific task. Key to reinforcement learning is the agent's interaction with the environment whereby the agent executes some action in order to modify the state of the environment and then receives the updated state of the environment and additionally a numerical reward which acts as a feedback signal. Using this feedback signal, the agent is then able to learn how to act in order to maximize the expected reward for his actions. Figure 2.1 visualises this interaction cycle of agent and environment.

More formally, reinforcement learning problems are generally formulated as a *Markov Decision Process* (MDP). A MDP $\mathcal{M}$ is defined as the tuple $(\mathcal{S}, \mathcal{A}, T, \gamma, S_0, R)$ where $\mathcal{S}$ stands for the state space of the environment; $\mathcal{A}$ for the action space from which the agent can choose a single action a_t at time t to execute; T are the transition probabilities denoting the probability $T(s_{t+1}|s_t, a_t)$ of transitioning from state s_t to state s_{t+1} if action a_t is executed; $\gamma \in [0, 1]$ is a so called discount factor which determines the preference of short-term over long-term returns. For $\gamma \to 0$ the agent only considers

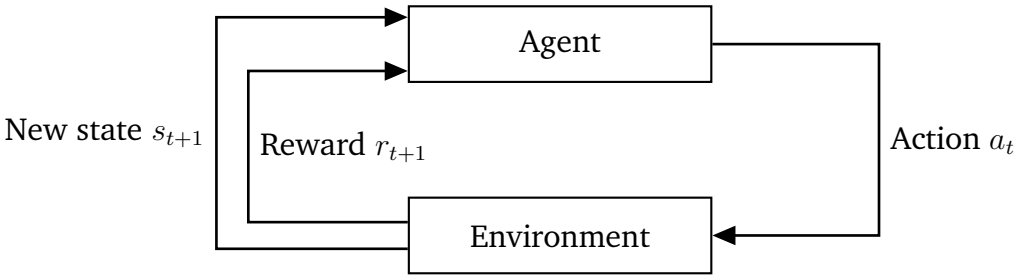


Figure 2.1: Agent-Environment Interaction Cycle.

immediate returns while putting less preference on rewards which might be higher but would be received after a longer period of time, on the other hand for $\gamma \to 1$, the agent prefers higher rewards even though they might be received at a later time; furthermore, S_0 denotes the distribution over start states $s_{t=0}$ and $R : \mathcal{S} \to \mathbb{R}$ specifies a reward function which maps an environment state to a numerical reward.

The goal of reinforcement learning is to learn a specific behaviour. This behaviour is formally denoted as a policy π, where $\pi : \mathcal{S} \to \mathcal{A}$ is usually a function which computes the corresponding action a_t or a distribution over actions $P(a_t|s_t)$ conditioned on the current state. In order to compare and evaluate policies, the value function $V^\pi(s_0) : \mathcal{S} \to \mathbb{R}$ is used, which is defined as the expected discounted sum of rewards which are received when executing actions according to π:

$$V^\pi(s_0) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] = R(s_0) + \gamma \sum_{s' \in \mathcal{S}} T(s, \pi(s), s') V^\pi(s'), \tag{2.1}$$

where $V^\pi(s_t)$ is the value of state s_t when following policy π. This value is defined as the expected return of following π starting from s_t:

$$V^\pi(s) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k+1})|\pi] \tag{2.2}$$

The value of a state essentially measures how good it would be to encounter it. In order to choose what action to take in a given state, we additionally need a quality measure over states and actions. This measure is called Q-function with $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$:

$$Q^\pi(s, a) = R(s) + \gamma \mathbb{E}_{s' \sim T(s,a,s')}[V^\pi(s')] = R(s) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s') V^\pi(s') \tag{2.3}$$

The Q-function tells us, how beneficial it is to take a certain action in a state. For given Q-values, the optimal policy π^* can then simply be found by taking the action a with the highest Q-value for an encountered state s:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q^\pi(s, a) \tag{2.4}$$

2.1.1 Q-Learning

However, the Q-values are typically unknown and therefore have to be learned. For this, (Watkins & Dayan, 1992) introduced the Q-Learning algorithm which iteratively computes the Q-values using the following update rule:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[R(s_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)], \tag{2.5}$$

where $\alpha > 0$ is the learning rate. This algorithm is able to approximate the optimal Q-values regardless of the executed policy, making Q-Learning a so called *off-policy* algorithm. For exploration purposes, it is helpful not to execute the current deterministic policy or a completely random policy but to have a mixture between random actions (exploration) and greedy actions (exploitation) which act optimally to the current Q-values. Therefore, the executed policy for Q-Learning is often an ϵ-greedy policy, which with probability ϵ executes a random action and with probability $1 - \epsilon$ the optimal action. In many cases, the value of ϵ is decreased as training progresses in order to focus more on exploitation.

In order to apply Q-Learning to large, continuous state spaces, a variant of Q-Learning called Deep Q-Learning can be used. Herein, the Q-function is approximated by a neural network (*Deep Q-Network (DQN)*). This technique was introduced by (Mnih et al., 2015; Mnih et al., 2013) and has achieved great success, for example by beating human performance on Atari games. The key difference to the original Q-Learning algorithm is the use of a experience replay buffer in which encountered experiences (s_t, a_t, r_t, s_{t+1}) are stored. For training the neural network, a minibatch of transitions is sampled from the replay buffer. Using this buffer annihilates the temporal correlation between transitions from one episode and furthermore enables us to use a transition more than once for training purposes which improves the sample efficiency.

2.1.2 Policy Gradients

In contrast to Q-Learning, policy gradient methods use a parametrized stochastic policy $\pi_\theta(a|s) = P(a_t = a|s_t = s, \theta)$ to determine which action to execute instead of deriving the action based on Q-values. An optimal policy π^* maximizes the expected discounted return. Hence, the optimal parameters θ^* can be found by:

$$\theta^* = \operatorname*{argmax}_\theta \mathbb{E}[\sum_{t \geq 0} \gamma^t r_t | \pi_\theta] = \operatorname*{argmax}_\theta J(\theta), \tag{2.6}$$

which can be solved by using gradient ascent on the parameters θ resulting in the following update rule:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta), \tag{2.7}$$

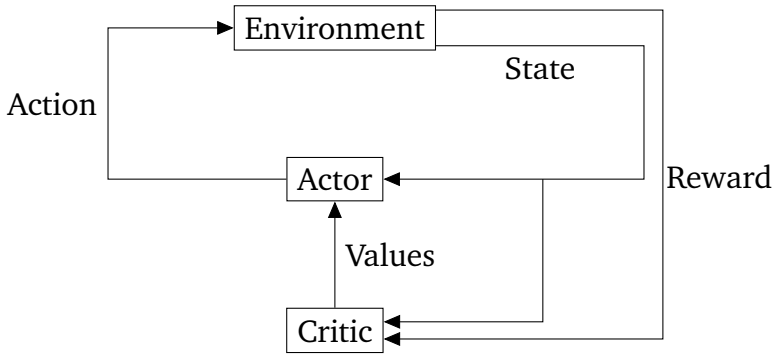


Figure 2.2: Actor-Critic Architecture.

where $\alpha \in (0, 1)$ denotes the learning rate. The expression ∇_θ is called *policy gradient*. Therefore, methods which follow this general principle are called policy gradient methods. However, the actual gradient $\nabla_\theta J(\theta)$ is difficult to compute and finding a good approximation is the core of every policy gradient method. One well-known policy gradient algorithm is *REINFORCE* (Williams, 1992) which estimates the policy gradient by (Peters & Schaal, 2006):

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log p_\theta(\tau) r(\tau)], \tag{2.8}$$

where τ denotes a rollout of policy π_θ and $r(\tau) = \sum_{t=0}^{T} \gamma^t R(s_t)$ denotes the discounted return of τ.

Policy gradient methods often suffer from high variance resulting in unstable policy updates (Wang et al., 2016). To overcome this issue, methods like Trust Region Policy Optimisation (TRPO) ((Schulman, Levine, Abbeel, Jordan & Moritz, 2015)) were introduced which ensure that the updated policy does not differ too significantly from the previous policy. However, TRPO relies on complicated theory making implementations difficult. Proximal Policy Optimisation (PPO) was therefore introduced in order to simplify the implementation, while still ensuring only stable policy updates (Schulman, Wolski, Dhariwal, Radford & Klimov, 2017).

2.1.3 Actor-Critic Methods

The actor-critic framework (Barto, Sutton & Anderson, 1983) aims to combine the best of critic-only approaches such as Q-Learning and actor-only methods such as policy gradients. Hereby, a the value function (the critic) and a policy (the actor) are learned simultaneously. More formally, the critic computes the Q-function $Q(s, a)$ while the actor uses these values to update the actor network using policy gradients. This general approach can also be seen in Figure 2.2.

Deep Deterministic Policy Gradients (DDPG) (Lillicrap et al., 2015) is one such actor-critic algorithm and has proven to be very effective in solving continuous control

problems (see (Gu, Holly, Lillicrap & Levine, 2017; Kumar, Paul & Omkar, 2018; Cabi et al., 2017)) which are unsolvable by Q-Learning approaches due to their continuous action space. Both the actor and the critic in DDPG are implemented as neural networks. The critic is hereby trained like a DQN as it is supposed to compute the optimal Q-values. The actor is trained using the deterministic policy gradient theorem introduced by (Silver et al., 2014). The DDPG policy network π computes a single deterministic action which should be executed in a certain state. As DQN, DDPG is an off-policy RL algorithm meaning that it uses a different policy during training than testing. During training, noise is added to $\pi(s)$ in order to explore the state space.

2.1.4 Imitation Learning

Imitation learning (IL)[1] is a subfield of reinforcement learning in which no access to a reward function is given. Instead, the agent receives demonstrations from a human expert executing the desired task and is then supposed to imitate the expert's behaviour. There are two common approaches to imitation learning: trying to approximate the expert's state-action distribution or approximating the expert's internal reward function and then using traditional reinforcement learning techniques to find an optimal policy with respect to these rewards. The former can either be done by using supervised learning techniques such as decision trees (see e.g. (Sammut, Hurst, Kedzier & Michie, 1992)) or neural networks (e.g. (Bogdanovic, Markovikj, Denil & De Freitas, 2015)). The latter is also known under the name inverse reinforcement learning (IRL) (Abbeel & Ng, 2004). IRL tries to find a reward function which best explains the given demonstrations, usually assuming optimal or near-optimal behaviour by the expert. Examples of IRL techniques include (Abbeel & Ng, 2004), (A. Y. Ng, Russell et al., 2000), (Ziebart, Maas, Bagnell & Dey, 2008), (Vroman, 2014). In this work we will demonstrate a technique to learn how to reach a given target region. This region is usually defined through a number of example states which are within this region. Imitation learning and our approach differ in the way that IL assumes that demonstrations are given as a sequence of state-action pairs, while we only assume access to some final demonstrated goal states without considering any actions which achieve these goal states or any intermediate states.

2.1.5 Reward Shaping

Reinforcement learning problems with sparse rewards are inherently difficult to solve due to the fact that the agent would have to reach the desired goal several times while executing random actions. One common technique which is used to make such sparse easier is *reward shaping* whereby the reward function is re-designed to guide

[1] A literature review of imitation learning techniques was done during my Individual Study Option module in the spring term.

the agent towards the final goal (Andrew Y Ng, Harada & Russell, 1999). Manual reward shaping is however a very tedious and complicated task which requires significant domain knowledge. In contrast to reward shaping techniques, our work does not modify the reward function in order to guide the agent towards a goal but instead proposes a series of goal states in separate episodes where the goals slowly wander towards a given target region. The reward functions for these goals is still sparse and does not need to be shaped, making our approach more automatic and therefore less tedious than reward shaping.

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) were introduced by (Goodfellow et al., 2014) and have recently gained large popularity. The goal of generative models is to approximate a given data distribution $p(X)$. While GANs are certainly not the only generative techniques available (e.g. Restricted Boltzmann Machines (Nie, Wang & Ji, 2017; Ackley, Hinton & Sejnowski, 1985), Auto-Encoders (Bengio, Yao, Alain & Vincent, 2013), Stochastic Neural Networks (Tang & Salakhutdinov, 2013)), they have proven to be very effective even in high-dimensional spaces and have for example been used for image generation (see (Goodfellow et al., 2014; Mirza & Osindero, 2014)) and text-to-image synthesis (Reed et al., 2016; Zhang et al., 2017).

As a major part of our contribution was inspired by the training process of GANs and as one of the curriculum generation methods we build on uses a generative adversarial network in order to generate the goals to train on in each iteration, we will now review the theory behind GANs. Furthermore, we will present some applications of GANs in reinforcement learning problems and will explain how these relate to our approach.

2.2.1 GAN

Intuitively, GANs let two neural networks compete against each other in an adversarial two-player zero-sum game ((Goodfellow et al., 2014)). The first network, called the *discriminator* D is trained to distinguish between real samples from $p(X)$ and generated fake samples. Thus, the discriminator is trained to solve a binary classification problem. The second network, called the *generator* G on the other hand aims to generate data points of such good quality that it fools the discriminator. This setting can be formally represented as the following value function of a two-player minimax game:

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))], \tag{2.9}$$

where z denotes a noise vector drawn from an arbitrary probability distribution. However, in general Gaussian noise is used.

In order to solve equation (2.9), (Goodfellow et al., 2014) propose to alternate between updating the discriminator using mini-batches sampled from $p(X)$ and mini-batches sampled from the generator and training the generator by updating its parameters in order to maximize the discriminator output for generated samples.

2.2.2 LSGAN

Standard GANs as introduced by (Goodfellow et al., 2014) have proven to be complicated to train as the discriminator and generator need to be kept in balance. There have been many proposed solutions for making GAN training more stable, e.g. Wasserstein GANS (Arjovsky, Chintala & Bottou, 2017) and Conditional GANs (Mirza & Osindero, 2014)). One such approach is Least Squares GAN (LSGAN) (Mao et al., 2017). Standard GANs as introduced by (Goodfellow et al., 2014) use a sigmoid cross entropy loss in order to train the discriminator network. However, (Mao et al., 2017) found that this loss function can lead to training instabilities due to the fact that samples which fool the discriminator cannot be penalized even though they deviate from the true data distribution.

To overcome this issue, the authors propose to use a least squares loss function for the discriminator training instead. Therefore, they drop the sigmoid activation function of the output unit, making the possible output range of the discriminator unbounded. This enables us to penalize samples which are far away from the decision boundary, regardless of which side of the boundary they are on. However, we loose the interpretability of the discriminator output as it is not bounded to $(0, 1)$ anymore. Replacing the sigmoid cross-entropy loss function with a least squares loss results in the following updated loss function for training LSGAN:

$$\begin{matrix} \min_D \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[(D(\boldsymbol{x})-b)^2] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[(D(G(\boldsymbol{z}))-a)^2] \\ \min_G \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[(D(G(\boldsymbol{z}))-c)^2] \end{matrix}, \tag{2.10}$$

where a is a constant which the discriminator should assign for generated samples (e.g. 0), b denotes which output is desired for real samples (e.g. 1) and c determines which value the generator should try to achieve from the discriminator output. However $(a = 0, b = 1, c = 1)$ are not the only possible choices and changing these values might result in more stable training (Mao et al., 2017).

2.2.3 GAN Applications in Reinforcement Learning

Both reinforcement learning and generative models, especially GANs, have seen a massive increase of research focus in recent years. Unsurprisingly, there have also been a few approaches which aim to combine these two fields. As our work also falls into this category, we will now review several of these approaches.

Most of these methods use the discriminator network as a reward function, therefore training the agent to maximize the discriminator output. Our approach is motivated by this idea. However, we do not modify the reward function but instead use a discriminator network to judge the usefulness of states and use the most useful states as initial goal states for policy rollouts.

2.2.3.1 Generative Adversarial Imitation Learning

Generative adversarial networks have proven to be a powerful technique to approximate the distribution of a given dataset, i.e. to create new data points which look very similar but not identical to provided data points. While their most common application is image generation, (Ho & Ermon, 2016) proposed Generative Adversarial Imitation Learning (GAIL)[2] which uses a GAN-like structure in order to approximate a state-action distribution in the context of imitation learning. As described in Section 2.1.4, imitation learning attempts to learn a reinforcement learning policy from demonstration by a human expert. To achieve this, the authors propose to train a discriminator network $D : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and a policy network $\pi : \mathcal{S} \rightarrow \mathcal{A}$, where the policy network can be seen as a generator network which is fed states instead of noise. Training the discriminator is very similar to standard GAN training, in this case the discriminator is optimized to distinguish between state-action pairs drawn from the collected expert trajectories τ_E and collected trajectories τ_i which were collected by executing π in iteration i. After updating the discriminator, the policy is being updated such that it better fools the discriminator. Training the policy is done using policy gradients, or more specifically TRPO (Schulman et al., 2015), where the reward $R(s, a)$ for executing action a in state s is defined as the discriminator's output $D(s, a)$. After updating both the discriminator and the policy network, new trajectories are sampled from the policy network and the process repeats. This approach has for example successfully been applied to imitate driver behaviour (Kuefler, Morton, Wheeler & Kochenderfer, 2017).

[2] A literature review of imitation learning techniques, including GAIL, was done during my Individual Study Option module in the spring term.

2.2.3.2 SPIRAL

Another approach combining GANs and reinforcement learning is SPIRAL (Synthesizing Programs for Images using Reinforced Adversarial Learning) (Ganin, Kulkarni, Babuschkin, Eslami & Vinyals, 2018). The motivation behind SPIRAL was that Generative Adversarial Networks often struggle with small details of images like for example straight, natural looking lines. To overcome this issue, the authors propose to train a reinforcement learning agent to use a graphic program to create images instead, hoping that issuing commands to a high-level program generates more realistic looking images than generating it from random noise like GANs. SPIRAL frames this image generation problem in the reinforcement learning setting by using a discriminator whose output is used as a reward function for the reinforcement learning agent. The agent therefore tries to generate images which receive a high score from the discriminator and the discriminator is trained to distinguish between given images and images generated by the agent.

2.2.3.3 SeqGAN

Similarly to SPIRAL, SeqGAN (Sequence GAN) (Yu, Zhang, Wang & Yu, 2017) also uses a discriminator network as a reward function. SeqGAN was designed for generating sequences of discrete tokens. GANs are a effective way of generating real-valued data, however they struggle with discrete data due to the fact that it is difficult to update the generator network using gradient descent when the output has to be discrete. To overcome this issue, SeqGAN models the generator network as a stochastic policy whose actions are the available discrete tokens. Sequence generation can therefore be modelled as a sequential discrete decision making process. After generating a whole sequence, the reinforcement learning agent then receives the reward, i.e. the discriminator's output for the whole sequence. The policy is then updated using policy gradients and the discriminator is trained using example sequences and generated sequences as in standard GAN training.

2.3 Curriculum Generation in Reinforcement Learning

Curriculum learning is an interesting technique in both machine learning and reinforcement learning as it alleviates learning of complex tasks by first learning easier subtasks before progressing to more difficult tasks. The manual design of a curriculum can however be difficult even for human experts, as it is often not intuitively clear into how many stages a task can be broken down and how long the agent will need to learn a subtask. Hence, the automatic generation of a curriculum for a task is highly desirable and recently some approaches have been proposed in this area.

Generating a curriculum automatically is usually not done pre-training but rather during training time as the agent's learning progress needs to be monitored in order to know when to progress to more difficult tasks. In the following, we will present the main approaches in the area of automatic curriculum generation.

2.3.1 Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play

(Sukhbaatar et al., 2017) proposed a method for automatic curriculum generation which is based on self-play. Hereby, two policies acting within the same environment are competing against each other. The first policy, called Alice, proposes a task (i.e. a sequence of actions) to the second policy, called Bob, which then has to either repeat or to reverse the presented task. Where repeating means that he needs to reach the final state of Alice's proposed rollout and reversing means that he needs to reach Alice's start state after starting from Alice's end state. The two agents then receive the following internal reward signals:

$$R_{bob} = -\sigma t_{bob} \tag{2.11}$$

$$R_{alice} = \sigma \max(0, t_{bob} - t_{alice}), \tag{2.12}$$

where σ is a scaling factor, t_{bob} is the time Bob needed to complete the task (or $t_{bob} = t_{MAX}$ if he failed to reach the goal within a given time limit t_{MAX}) and t_{alice} is the length of the proposed rollout by Alice. This way, Bob is rewarded for completing a task as quickly as possible and Alice is rewarded for proposing tasks for which Bob needed more steps than Alice ($t_{bob} > t_{alice}$). Therefore, Alice will learn to propose more and more complicated tasks and Bob will learn how to perform these proposed tasks. Hence, this asymmetric self-play method generates a curriculum of increasingly difficult tasks.

2.3.2 Hindsight Experience Replay

Hindsight Experience Replay (HER) is a recent invention introduced by (Andrychowicz et al., 2017) and addresses the issue of learning in environments with a sparse reward function. Sparse reward functions assign each non-goal state the same reward (often -1 or 0) and the goal states a higher reward (0 or 1 respectively). This is an obstacle for traditional reinforcement learning techniques since the agent only experiences a useful reward signal upon reaching a goal state. However, reaching the goal state by chance is highly unlikely especially in high-dimensional, continuous state and action spaces which can typically be found in robotics. HER addresses this issue that the agent is not able to learn anything unless he reaches a desired goal state. This is achieved by pretending in hindsight that one or several of the states the agent reached within an episode had been the actual goal, i.e. in hindsight

the agent might not have reached the initially set goal, but he has reached other states and should learn from these experiences. Traditional reinforcement learning algorithms would not be able to learn anything from these failed trajectories. The idea behind HER is now to assume in hindsight that certain states would have been successfully reached if the objective would have been to reach them. This essentially converts failures on one task to successes on a different task.

More formally, HER assumes that an initial goal g is sampled at the beginning of each episode. Therefore the policy $\pi(s|g)$ is executed during this episode and experiences $(s_t, a_t, r_t, s_{t+1}, g)$ are added to the replay buffer. Furthermore, we assume access to a mapping $m : \mathcal{S} \to \mathcal{G}$ which transforms an encountered state into an *achieved goal*, where $\mathcal{G}$ denotes the goal space. In hindsight, we can then replace the original goal g in collected experiences tuples with an achieved goal ag and add the new experience tuple $(s_t, a_t, r_t, s_{t+1}, ag)$ to the replay buffer as well. This way, the replay buffer contains a trajectory which maybe fails or succeeds to reach goal g but it definitely contains a trajectory which successfully reaches ag. This hindsight replacement of goals is only possible for off-policy reinforcement learning algorithms like DDPG as the policy $\pi(s, ag)$ has actually never been executed.

The authors of HER propose several strategies for selecting how many and which achieved goals should be used for hindsight experience replay. These strategies are:

- *final*:
  The *final* replay strategy chooses the last state of an episode as a replay goal, i.e. $g' = m(s_{T-1})$.
- *future*:
  For each transition sampled from the replay buffer, the *future* replay strategy selects k states which were achieved in the same episode but after the sampled transition for replay.
- *episode*:
  For each transition sampled from the replay buffer, the *episode* replay strategy selects k states which were achieved in the same episode either before or after the sampled transition for replay.
- *random*:
  The *random* replay strategy choose k random states which have been encountered in any episode so far for replay.

Experimental results from (Andrychowicz et al., 2017) show that the *future* replay strategy is most effective for all tasks in their experiments. We therefore adopt this replay strategy for our own experiments. The formal HER Algorithm is shown in Algorithm 1.

The initial goal for an episode is usually randomly selected and HER can therefore not be seen as an algorithm which follows an explicit curriculum as for this the initial

goals would need to progress from easier tasks to more difficult ones. However, the authors argue that HER can still be seen as an *implicit* curriculum learning algorithm, as the agent first learns easy tasks, i.e. randomly encountered and therefore most likely close to the start state, and then harder and harder tasks as the policy improves (Andrychowicz et al., 2017). While the authors do not propose a strategy to select the initial goals, our Directed HER approach will replace this initial goal distribution by a sampling process which selects goal states which have been encountered in the last episode and are most similar to given target goals. This will change the executed policy for the agent and therefore guide exploration towards these goals and hence finally towards the target region.

Algorithm 1 Hindsight Experience Replay (Andrychowicz et al., 2017)

Require:

- off-policy RL algorithm $\mathbb{A}$
- goal sampling strategy $\mathbb{S}$
- sparse reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{R}$

for $i = 1, \ldots$ **do**
  sample goal g and initial state s_0
  for $t = 0, \ldots, T-1$ **do**
    $a_t \leftarrow \pi(s|g)$ according to $\mathbb{A}$
    Execute a_t and observe s_{t+1}
  end for
  for $t = 0, \ldots, T-1$ **do**
    $r_t = r(s_t, a_t, g)$
    Store $(s_t, a_t, r_t, s_{t+1}, g)$ in replay buffer
    Sample set of additional goals $G = \mathbb{S}(episode)$
    for $g' \in G$ **do**
      $r'_t = r(s_t, a_t, g')$
      store $(s_t, a_t, r'_t, s_{t+1}, g')$ in replay buffer
    end for
  end for
  for $i = 1, \ldots, N$ **do**
    sample minibatch from replay buffer
    train policy using minibatch
  end for
end for

2.3.3 Reverse Curriculum Generation

Reverse curriculum generation (Florensa et al., 2017) deals with problems in which the agent has to reach one specific goal state from any other state in the state space. Learning this task by randomly choosing a start state is basically impossible due to the sparsity of the reward function and potentially huge state spaces. The key insight of reverse curriculum generation is to first learn how to reach the goal region from start states close to the desired goal as the chance of reaching the goal state by executing a random policy is comparably high. After mastering the task from these close-by states, the algorithm then generates new start states by executing short rollouts of randomly selected, small actions. The start states from the previous iteration are therefore perturbed using noise from the action space rather than the state space as random perturbation in the state space might yield states quite far from the original state measured in the number of actions that need to be taken to transition from one state to another or it might even yield infeasible states.

2.3.4 Automatic Goal Generation

(Held, Geng, Florensa & Abbeel, 2017) propose an approach called GoalGAN which automatically generates *goals of intermediate difficulty* (GOID) for the agent. The set of GOID goals is formally defined as:

$$GOID = \{g : R_{min} \leq \mathbb{E}[R^g(\pi)] \leq R_{max}\}, \tag{2.13}$$

where $\mathbb{E}[R^g(\pi)]$ denotes the expected reward for reaching goal g and $R_{min}, R_{max} \in (0, 1)$ are hyperparameters with the following meaning: all states with $\mathbb{E}[R^g(\pi)] \in [0, R_{min})$ are considered too hard for training and all states with $\mathbb{E}[R^g(\pi)] \in (R_{max}, 1]$ are assumed to be mastered and therefore too easy for further training. The authors propose that such GOID goals are ideal for training as they are not too hard but also not too easy for the current policy and therefore provide a stable learning feedback signal. The goal generation is implemented as a generative adversarial network, more specifically a LSGAN (see Section 2.2.2).

In each iteration, the algorithm starts by sampling n_{goals} goals, Which consist of n_{new} goals sampled from the generator and n_{old} goals sampled from the goal replay buffer in which goals which have already been mastered are stored. The generated goals are for progressing the policy as they are expected to not have been mastered yet and the replayed goals are used in order to prevent catastrophic forgetting of previously learned tasks. After sampling the goals, the policy is trained using these goals. Each goal is used n_{trials} times for training, such that each iteration consists of $n_{trials} \cdot n_{goals}$ episodes. Afterwards, the updated policy is used in order to label the sampled goals

based on their expected return, where the label is defined as:

$$label(g) = \begin{cases} 1 & \text{, if } g \in GOID \\ 0 & \text{, otherwise.} \end{cases} \tag{2.14}$$

These labels can then be used in order to update the GoalGAN. An interesting insight of the authors is to use positive and negative examples for training the GAN discriminator. Usually, GAN discriminators are trained to distinguish between positive examples from a given dataset and negative examples sampled from the generator. However, in this case, there exists no given dataset of positive examples but only generated goals such that the discriminator is trained only on generated examples. The last step of each episode then consists of adding the goal states which have already been mastered, i.e. $R^g(\pi) > R_{max}$, to the replay buffer.

One critical point of this algorithm is the labelling of goal states. In order to estimate the average reward for all goals in an iteration, several evaluation rollouts for all goals need to be performed. Using these rollouts only as evaluation but not for training makes the GoalGAN algorithm terribly inefficient. However, using them for training makes the computed labels incorrect as the policy is updated after a goal has been labelled. The authors however show that this only causes slight inaccuracies in the label computations with insignificant influence on the training performance. Hence, the actual labels are computed while training. We show the formal algorithm of GoalGAN in Algorithm 2.

Algorithm 2 GoalGAN: Automatic Goal Generation For RL(Held, Geng, Florensa & Abbeel, 2018)

$G, D \leftarrow pretrain\_GAN()$
$goal\_replaybuffer \leftarrow \emptyset$
for $i = 1, \ldots$ **do**
    $goals \leftarrow G(z) \cup sample(goal\_replaybuffer)$
    $\pi_i \leftarrow update\_policy(goals, \pi_{i-1})$
    $labels \leftarrow label\_goals(goals, \pi_i)$
    $G, D \leftarrow train\_GAN(goals, labels)$
    $goal\_replaybuffer \leftarrow store\_goals(goals, goal\_replaybuffer)$
end for

2.3.5 Region-Growing Curriculum Generation

While reverse curriculum generation (Section 2.3.3) and GoalGAN (Section 2.3.4) deal with problem settings where either the start or the goal state is fixed, region-growing curriculum generation (Molchanov et al., 2018) addresses problems where any state should be reached from any start state. This is achieved by defining a

region in which the agent's current policy is able to reach any state from any other state within the region. This region is then expanded by performing random action rollouts from states within the region and adds the final state from this short rollout to the region. Then the policy is updated to learn how to reach all states within the region, including the newly added states. A critical point for this algorithm is the choice of variance used for exploration as the authors show in their experiments that different fixed variances can yield vastly different performances. The authors therefore propose a method which dynamically adapts the variance based on the average reward from the previous iterations. A low average reward means that the current region has not been mastered yet, therefore exploration should be slowed down. However, high average rewards indicate that the learner has already mastered the current region and therefore new states should be added to avoid a slow down in learning.

2.4 Summary

In this chapter, we have reviewed fundamental definitions and techniques of reinforcement learning and generative adversarial networks. We have furthermore reviewed related concepts such as imitation learning and reward shaping and have distinguished our approach from these areas. Lastly, we presented related work in the area of automatic curriculum generation in reinforcement learning which is the main category in which our topic falls. Next, we will progress to motivating the general idea of our contribution and presenting our approaches.

Chapter 3

Directed Curriculum Generation

In this chapter, we will first motivate this project and the expected advantages of directed curricula over undirected curricula. Furthermore, we will concisely state which problems this project concerns. Finally, we will present our approaches to directed curriculum generation.

3.1 Problem Statement and Motivation

Learning in stages of increasing difficulty, i.e. following a curriculum, has proven to be a very effective way of learning complex tasks in machine learning. Furthermore, it is also the way humans learn many of their most complex tasks, e.g. in high-school and university. Curriculum learning essentially breaks down a complex task into a sequence of learning tasks where each learning task builds on previous tasks and therefore is harder to achieve than previous tasks. This structure helps a reinforcement learning agent to overcome the need for significant exploration in order to achieve a complex goal task.

Curricula generating approaches (see Section 2.3) generally expand the current set of tasks towards tasks with higher difficulty once the agent has achieved a desired performance on the previous tasks and therefore guides the agent from easier tasks to more difficult tasks. Using only a measure of difficulty results in an undirected curriculum in the way that the usefulness of tasks is not considered. This is in strong contrast to human curriculum learning where curricula usually are designed to first teach fundamental things but then specialise on some specific areas instead of progressing into all possible directions as this would not be feasible. Humans hereby normally take both the difficulty and the usefulness of tasks into account when choosing the next stage of learning. One example for this is the selection of electives at university, where students choose courses based on their interest or based on the necessity to learn about a certain area because they later want to work

within this area. On the other hand, humans do not select to learn tasks which they deem non-interesting or unnecessary for their future goals.

The goal of this thesis is to apply this idea of expanding a curriculum based on both difficulty and usefulness to reinforcement learning problems. We assume that we are given a set of final goals which we will call targets from now on. These targets essentially constitute the complex task which should be learned and all intermediate goals generated by the curriculum should be directed towards this goal region in order to expand the curriculum towards the target region and not into regions of less interest.

In the following sections, we will first explain what our general idea is in order to bias the curriculum and exploration towards a given target region and then we will present our two contributions, Directed GoalGAN and Directed Hindsight Experience Replay.

3.2 General Principle of Directed Curriculum Generation

The task we consider for this project is to expand a curriculum based on difficulty and usefulness of tasks instead of just difficulty as current curriculum generation approaches do. To do so, we need to find an appropriate measure for the usefulness of states. A trivial choice for such a similarity function would be a distance measure, such as the ℓ_2-norm. However, such measures compute a numerical value for a fixed path from one state to another. This fixed path is unfortunately often not a got indication of actual distance since it computes the distance in the state space and not the action space. Two states might be located very close to each other, however it is possible that many actions would need to be taken to transition from one to the other due to obstacles in the environment or due to other restrictions, such as kinematic constraints.

Instead, we propose to learn this similarity measure using a neural network such that this network outputs a value in $(0, 1)$ where high values indicate high usefulness or similarity to the targets. This idea is inspired by GAN training, where the discriminator essentially learns a function which distinguishes between real and fake data. Hence, we use a discriminator network with one sigmoid output unit in order to compute a measure of usefulness. The discriminator is trained to assign a label of 1 for the target states and 0 for all other states. Similarly to GANs these negative examples will be generated states. However, we do not use a generator network but consider all states generated by policy rollouts as negative examples. As the policy improves and therefore learns to reach a higher variety of states, the discriminator will also get better.

Knowing how we will compute the similarity measure however does not explain how a curriculum can be expanded based on both difficulty *and* usefulness. The general idea of our approaches is to use an existing curriculum generation algorithm which creates a difficulty-based curriculum and then to select from goals generated by this curriculum building algorithm the goals which are most interesting. Interesting goals are hereby considered to be goals for which the discriminator outputs higher values than for other goals. We therefore essentially subsample goals based on their discriminator rating from a set of goal states provided by a difficulty-based curriculum generator.

In the following two sections we will now show, how we can combine our abstract idea of using a discriminator network to rate the importance of states with two curriculum building algorithms, namely GoalGAN and Hindsight Experience Replay.

3.3 Directed GoalGAN

GoalGAN (see (Held et al., 2018) and Section 2.3.4) is a recent approach which uses a generative adversarial network to generate goal states which provide a useful feedback signal for the reinforcement learning agent. The authors assume that states which are not too hard but also not too easy for the current policy provide the most useful learning signal. To denote goals with a good learning signal, the authors define *goals of intermediate difficulty* (GOID), which are generated using a GAN. In each iteration, the current policy is trained on goals generated by the GAN, the goals are labelled using the success rates of reaching them from this iteration and the GAN is trained using those labels. As the policy should have improved in the last iteration, the GOID region has been moved to more difficult states and hence the curriculum progresses to more difficult tasks. Using this training process explores and learns the state space in a principled way, starting from easy to reach goals and then progressing to more difficult tasks.

However, this curriculum expands in all possible directions as it only considers which goals are appropriate for training in each iteration but does not take any other attributes of states into account. As we aim to generate a curriculum which expands towards a goal region, we need to modify the goal generation process of GoalGAN. In each iteration, GoalGAN generates new goals which are most likely of appropriate difficulty. In order to bias the goal generation process towards a target goal region, we train a second discriminator network to distinguish between target goals and other states. More specifically, in each GoalGAN iteration, we train the discriminator to output label 1 for our given target goals and label 0 for all goals which were generated by GoalGAN and used for training the policy.

In the next iteration, we then generate k goals using GoalGAN and rate all these generated goals by passing them through the discriminator network. Hence, we obtain labels $q(g_i) = D(g_i)$ for each generated goal g_i. Assuming that we always

train on n goals in each episode we then select the $\epsilon \cdot n$ goals with the highest ratings from the k generated goals, where $\epsilon \in [0, 1]$ is a parameter determining how strongly we want to focus on regions of high interest. Furthermore, we uniformly sample additional $(1 - \epsilon) \cdot n$ goals from the k generated goals. These randomly selected goals ensure that the curriculum does not progress too quickly towards regions of high interest as it is not guaranteed that states with high interest are indeed helpful for reaching the target region.

We call this modified GoalGAN algorithm Directed GoalGAN. A formal notation of this algorithm can be seen in Algorithm 3. Additions made to the original GoalGAN algorithm are shown in bold. The pre-training of the target discriminator was hereby performed by randomly collecting some states close through the start state via short random rollouts and training the target discriminator using these collected states and the given target states.

Algorithm 3 Directed GoalGAN

Require: set of target goals: $targets$, ϵ
1: $\mathbf{target\_D} \leftarrow \mathbf{pretrain\_target\_discriminator}()$
2: $G, D \leftarrow pretrain\_GAN()$
3: $goal\_replaybuffer \leftarrow \emptyset$
4: **for** $i = 1, \ldots$ **do**
5: $\quad goals \leftarrow G(z) \cup sample(goal\_replaybuffer)$
6: $\quad \mathbf{sampled\_goals} \leftarrow \mathbf{sample\_by\_rating}(\mathbf{goals}, \epsilon)$
7: $\quad \pi_i \leftarrow update\_policy(\mathbf{sampled\_goals}, \pi_{i-1})$
8: $\quad labels \leftarrow label\_goals(\mathbf{sampled\_goals}, \pi_i)$
9: $\quad G, D \leftarrow train\_GAN(\mathbf{sampled\_goals}, labels)$
10: $\quad \mathbf{target\_D} \leftarrow \mathbf{train\_target\_discriminator}(\mathbf{sampled\_goals}, \mathbf{targets})$
11: $\quad goal\_replaybuffer \leftarrow store\_goals(\mathbf{sampled\_goals}, goal\_replaybuffer)$
12: **end for**

An intuition of the directed GoalGAN sampling process is visualised in Figure 3.1. The figure shows an artificial example. We assume the points in the left image to be a set of goals generated by GoalGAN. The middle image then shows how a discriminator network could rate these goals, with higher ratings on one side and lower ratings on the other. The final picture on the right then shows which goals we select for our Directed GoalGAN approach, namely the goals with the highest discriminator ratings.

3.4 Directed Hindsight Experience Replay

As described in Section 2.3.2, Hindsight Experience Replay improves the sample efficiency of RL algorithms by also learning from failed examples. As GoalGAN aims to

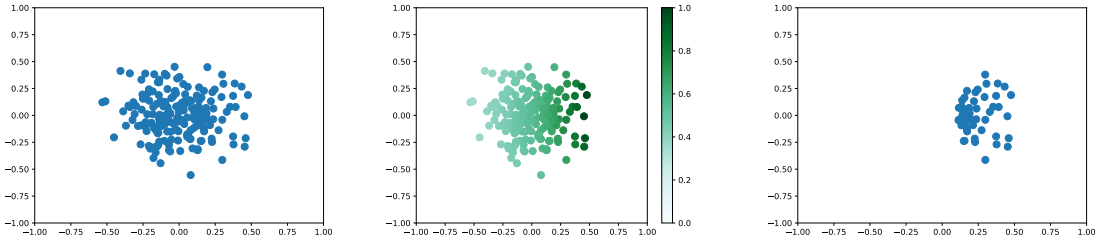


Figure 3.1: Visualisation of the sampling process on an artificial example. Left: GOID goals as generated by GoalGAN, Middle: GOID goals labelled by discriminator, Right: sampled goals based on discriminator ratings.

train on goals which can sometimes be reached by the current policy but not always, the training process includes many trajectories which fail to reach the goal state. Using HER, these failed trajectories would still provide helpful learning signals as it guarantees that something is learned. However, combining GoalGAN and Hindsight Experience Replay is non-trivial. GoalGAN generates goals by strictly tracking which goals are mastered, feasible or infeasible. Unfortunately, Hindsight Experience Replay would make this skill tracking impossible, as the agent might learn how to reach a completely different region of states than he is actually trained on. This essentially constitutes a clash between the strict explicit curriculum which GoalGAN generates based on the agent's performance and the implicit curriculum inherently created by HER. We therefore did not attempt to combine GoalGAN and HER into one algorithm and then modifying it for directed curriculum generation, but instead we aim to direct the implicit curriculum generation of HER towards a target region.

The main benefit of HER is that it learns even from failed attempts, that means even if the initial goal of an episode was not reached, the agent still learns something about how to reach the states he actually encountered (Andrychowicz et al., 2017). HER proved to be an efficient approach to multi-goal reinforcement learning in continuous control which makes it an interesting approach for our problem (Plappert et al., 2018). The original HER algorithm always samples an initial goal for an episode from the goal space, which in our case would be a state from the given target states. These states are initially however very unlikely to be reachable such that the agent only learns in each episode how to reach states he has randomly encountered. As the policy improves, the agent will also extend its range of states he can reach. Due to the generalisation capabilities of neural networks, this implicit curriculum will already expand towards the goal region, however slowly as it requires the agent to randomly encounter similar states to the target states first.

Similarly to Directed GoalGAN (see Section 3.3), we want to bias this expansion towards the target region in order to learn faster how to reach this target region and we again propose to do this by modifying the selection of the initial episode goal which determines the policy being executed by the agent. If the initial goal is too complicated for the agent, this policy will essentially result in random actions being taken. It therefore makes sense, to choose the initial goals in a way that the

agent knows at least the approximate direction of the goal, as this will guide the exploration process towards this goal.

Unlike GoalGAN, HER does not have a goal generation process for each iteration. However, we again want to rate a set of states using a discriminator network and choose the most promising states as initial episode goals. One possible way would be to replace the set of generated goals from GoalGAN by the set of all states which have been achieved in the episodes during the last iteration. Choosing this set of states has the advantage that all contained states are definitely feasible which is not the case for GoalGAN which might generate infeasible states which can slow down the training progress as training episodes are wasted trying to reach infeasible goals. However, in contrast to the set of goals generated by GoalGAN, the set of achieved goals is highly correlated since the states were visited successively. Selecting the goal states with the highest rating would therefore most likely result in a high concentration of goal states. Instead, we sample the best state from the achieved goals from each episodes as this removes the correlation between sampled goals but maintains that primarily interesting states are selected.

As in our Directed GoalGAN approach, this rating is intuitively motivated that states with higher ratings are probably more similar to the target region and are therefore worth exploring. Furthermore, sampling from all achieved states is intuitively motivated by the way humans learn: Whenever we attempt to learn a new task and we achieve something interesting, for example a movement which feels better or performs better, we aim to achieve this goal again. We transfer this motivation into our Directed HER approach by essentially asking the agent to repeat tasks which he has achieved in the previous episode and that are most interesting.

Additionally to the selected goals, we also want to train on actual target goals from time to time, as this can help to advance the curriculum by discovering new states. This helps the expansion of the curriculum as executing the policy for these goals results in noisy but more and more target-directed trajectories. We therefore define a procedure $\texttt{sample\_initial\_goal}(\texttt{goals}, \texttt{targets}, \epsilon)$ which selects the initial goal from the sampled goal with probability $1 - \epsilon$ or from the given target set with probability ϵ. The Directed HER procedure is formally shown in Algorithm 4. In the first episode, we sample goals from the set of target goals as we do not have any achieved states collected yet, however we could also sample randomly from the state space as the policy in the beginning is completely random and the first initial goals will most likely not be reached anyway.

Selecting the best states from each episode may still result in selected goals which are not useful for training as an episode can fail to reach any state of interest at all. We therefore also introduce *prioritised goal selection* (Directed HER Prio) which modifies `sample_initial_goal` such that again with probability $1 - \epsilon$ we select a random goal from the goal set. Hereby, the probability of a goal being sampled is not uniform anymore but instead proportional to its rating. Hence, goals with higher ratings are being trained on more frequently and goals with very low ratings have

Algorithm 4 Directed HER

Require: Target States T, n_{goals}, ϵ
$D \leftarrow init\_discriminator$
$goals = random\_sample(T, n_{goals})$
for $i = 1, \ldots$ **do**
    $new\_goals = \emptyset$
    for $j = 1, \ldots n_{goals}$ **do**
        $initial\_goal = sample\_initial\_goal(goals, targets, \epsilon)$
        $achieved\_goals_j = HER(goals_j)$
        $scores_j = D.rate(achieved\_goals_j)$
        $new\_goal \leftarrow sample\_best(achieved\_goals_j, scores_j)$
        $new\_goals = new\_goals \cup new\_goal$
    end for
    $goals = new\_goals$
    $D = train\_discriminator(achieved\_goals_i, T)$
end for

little chance of being selected. The probability of sampling a goal state g from a goal set G is hereby defined as:

$$p(g) = \frac{D(g)}{\sum_{g' \in G} D(g')}, \tag{3.1}$$

where $D(g)$ denotes the output of the discriminator. Again, with probability ϵ we select a target goal as the initial goal. We will investigate the effect of these strategies on training in Chapter 4.

3.5 Summary

In this chapter, we have introduced the two contributions of this project. Directed GoalGAN extends the GoalGAN curriculum generation algorithm with an additional discriminator network which is used to rate the goals generated by GoalGAN. These ratings are used to select the best goals, which are then used for training. Directed HER is a strategy for choosing the initial goal for HER. We again train a discriminator network. This time we rate all states which were reached in the previous episode and again use the discriminator's ratings to choose which goals to use for training in the next iteration.

Chapter 4

Experiments

After presenting our general approach in the last chapter, we will now provide experimental results which examine whether our approach is able to bias curricula towards a goal region and whether this goal bias achieves an improved performance in comparison to other approaches.

We start by examining our directed GoalGAN approach on small continuous two-dimensional gridworld problems which help to visualise how our approach compares to the original GoalGAN method and how it affects the expansion of the reachable goal state and the coverage in general. These experiments show that our directed approach is indeed able to reach a defined target region faster than GoalGAN as it explores mainly in the direction of this target region.

Afterwards, we tested our approach on more complex problems with both larger state and action space. The first problem we used for testing, are hand movement where a robotic hand has to reach a specific position. Our target region was represented as a set of examples where the hand is in a thumbs-up position. The second problem concerned the in-hand manipulation of a cube, where the specific task was to rotate the cube by 90 degrees around the x-axis. Unfortunately, we observed that GoalGAN methods are not suitable for high-dimensional goal spaces and we therefore only compare Directed HER and the original HER method on these robotics tasks. The results show that Directed HER does not yield an increase performance on the hand movement task but instead even performs slightly worse than the original HER method. However, on the cube task Directed HER is able to outperform original HER if we use prioritised goal sampling. Non-prioritised goal sampling unfortunately failed the task completely due to repeatedly sampling trivial goals.

4.1 Directed GoalGAN

This project's initial goal was to modify the GoalGAN algorithm in order to direct the generated curriculum towards a target region. We therefore start presenting our achieved results with the examination of GoalGAN and our modification, Directed GoalGAN.

4.1.1 Environments

Our first set of experiments was performed on two continuous gridworld environments. Such toy problems are perfectly suited for quick comparison of reinforcement learning algorithms. Furthermore, the two-dimensional goal space is perfect for visualising the progression of the curriculum learning approaches we are testing. A visualisation of our two gridworld environments is shown in Figure 4.1. The first environment, we call it CrossGrid, is a plus-shaped grid with a central start point. The target region is located at the right of the cross and is formally defined as:

$$\mathbb{G} = \{(x, y) | 0.9 \leq x \leq 1.0, -0.25 \leq y \leq 0.25\} \tag{4.1}$$

The shape of this environment was chosen because it presents a great way of testing a directed curriculum in comparison with undirected curricula. Undirected curricula are expected to expand in a circular way from the start point into all directions as they only consider the difficulty of the task which in this task is strongly correlated with the distance to the starting point. Our directed curriculum is however supposed to expand mainly towards the direction of the target region and should not explore too many goals within the other three branches of the cross. Secondly, we designed an environment called MazeGrid, which was inspired by the MazeGrid used in (Held et al., 2018) but was slightly modified by adding an additional vertical corridor. In this environment, the agent needs to find his way to the bottom right corner, which constitutes the target region. At the beginning of each episode the agent is reset to the start state which is located at the origin. The target region is formally defined as:

$$\mathbb{G} = \{(x, y) | 0.75 \leq x \leq 1.0, -1.0 \leq y \leq -0.75\} \tag{4.2}$$

The state space for both environments is $\mathbb{R}^8$: The x-position, the y-position, the velocity in x- direction x_{vel}, the velocity in y-direction y_{vel} and the distance to an obstacle or wall in positive x-direction (x^+_{dist}), negative x-direction (x^-_{dist}), positive y-direction (y^+_{dist}) and negative y direction (y^-_{dist}). The goal space for both environments is $[-1, 1]^2$. The reward function is given by:

$$r(s, g) = \mathbb{1}[\|s - g\|_2^2 < \epsilon] - 1, \tag{4.3}$$

where ϵ is set to 0.02 for both environments. The maximum speed for the agent is set to 0.01 for the CrossGrid environment and 0.02 for the MazeGrid environment.

At each timestep the agent chooses an acceleration action a, updating the current velocities by:

$$\begin{bmatrix} x_{vel} \\ y_{vel} \end{bmatrix} \leftarrow clip\left(\begin{bmatrix} x_{vel} \\ y_{vel} \end{bmatrix} + a * max\_acceleration, -max\_speed, max\_speed \right), \quad (4.4)$$

where the maximum acceleration was set to $\frac{max\_speed}{25}$ for the CrossGrid and to $\frac{max\_speed}{5}$ for the MazeGrid environment. Both max_speed and $max\_acceleration$ were chosen to such that both environments have a similar time frame. This maximum horizon is set to 150 for both environments. A summary of the environment parameters are given in Table 4.1.

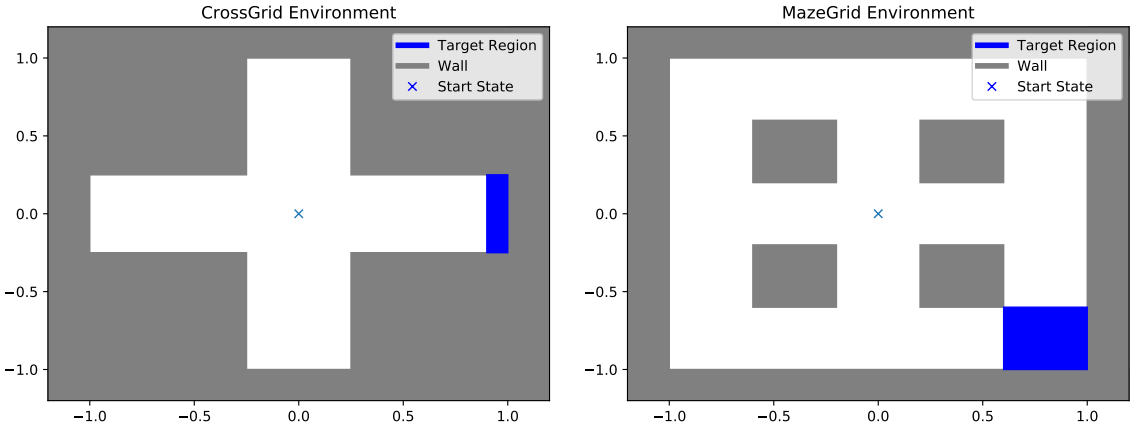


Figure 4.1: Visual layout of our two gridworld environments. White color denotes the space in which the agent can move, grey indicates walls and obstacles and blue indicates the desired target region. The start state is indicated by "X".

	CrossGrid	MazeGrid
State Space	$\mathbb{R}^8$: $x, y, x_{vel}, y_{vel}, x^+_{dist}, x^-_{dist}, y^+_{dist}, y^-_{dist}$	$\mathbb{R}^8$: $x, y, x_{vel}, y_{vel}, x^+_{dist}, x^-_{dist}, y^+_{dist}, y^-_{dist}$
Goal Space	$[-1, 1]^2$	$[-1, 1]^2$
Action Space	$[-1, 1]^2$	$[-1, 1]^2$
Target Region	$\{(x, y) \mid 0.9 \leq x \leq 1.0, -0.25 \leq y \leq 0.25\}$	$\{(x, y) \mid 0.75 \leq x \leq 1.0, -1.0 \leq y \leq -0.75\}$
Maximum Speed	0.01	0.02
Maximum Acceleration	$0.01/5$	$0.02/25$
Distance Threshold ϵ	0.02	0.02
Maximum steps T	150	150

Table 4.1: Summary of environment specifications for our two gridworld environments, CrossGrid and MazeGrid.

4.1.2 Implementation Details

While (Held et al., 2018) used TRPO for training the policy, we replaced it with PPO, as it easier to implement and shows similar or even better performance on many tasks (Schulman et al., 2017). For the PPO implementation we used OpenAI Baselines (Dhariwal et al., 2017). We used the same set of parameters for both environments. The policy is defined as a neural network with two layers with 64 hidden units each. We train GoalGAN for a total of 100 iterations. Each iteration consists of sampling 150 goals, with a ratio of $70 : 30$ between new goals and old goals from the replay buffer. Then the policy is trained 5 times for each goal. Finally, the GAN is updated for 3000 minibatches of size 32. For Directed GoalGAN, we used an ϵ-value of 0.75, meaning that we sample the best $0.75 \cdot n_{goals}$ goals from all generated goals and additionally $0.75 \cdot n_{goals}$ randomly selected goals.

4.1.3 Results

We compare the results of our comparison of GoalGAN on both quantitative and qualitative measures. The quantitative measurements are obtained by evaluating two criteria: the coverage of the whole state space and the coverage of states from the target region. Both criteria are computed by uniformly sampling a number of goals from the respective state (100 for the whole state space, 50 for the target states). As we aim to learn how to reach a certain target region rather than how to reach an arbitrary goal from the state space, the goal coverage metric is our main evaluation measure for comparison. We start by presenting qualitative results which visualise the expansion of the curriculum based on what goals are trained on in each iteration and how the skill region expands. Afterwards, we present the numerical results.

4.1.3.1 Qualitative Results

In this section we will qualitatively compare GoalGAN and our modification to it. This is done by visually inspecting which goals the algorithm selects in which episode and therefore how the curriculum expand. This will be a first indication whether our Directed GoalGAN approach is able to extend the curriculum primarily towards a given target region.

GoalGAN is a curriculum generation algorithm which is supposed to progress from goals that are easy to reach to more difficult goals based on the agent's current performance. In our two-dimensional gridworld scenarios, the difficulty of goals is highly correlated with the ℓ_2-distance from the start state. We therefore expect GoalGAN, to ideally expand its curriculum in a circular way by increasing the radius of this circle in each iteration. Furthermore, we expect GoalGAN to adapt the

curriculum based on the agent's performance. Hence, the generated goals should contain a mixture of states which are mastered, within the GOID region or infeasible. Our findings are visualised in Figure 4.2. It can be seen that while the proposed goals do not expand in a perfectly circular shape, the area does continuously expand and eventually reaches all states in the state space. It is however also visible, why a directed curriculum in this case would be helpful, as GoalGAN expands into all or random directions but should expand primarily towards the target region which is located at the end of the right finger of the CrossGrid. The green trail of goals can be explained by the goal replay from the replay buffer.

In comparison, we show the curriculum development for our Directed GoalGAN algorithm in Figure 4.3. The strong contrast in the way how the curricula expand is clearly visible. While GoalGAN explores into all possible directions, Directed GoalGAN finds its way quickly towards the target region and then only after reaching the target region expands into other directions. However, as we only care about the target region, Directed GoalGAN could have been terminated after the target region has successfully been reached. This example clearly motivates the need of a directed curriculum for tasks where we need to reach a specific target region and not the whole state space. The green trail of goals can again be explained by the goal replay from the replay buffer.

Goal development results for the MazeGrid environment are shown in Appendix C. These confirm the results presented in this section for the CrossGrid environment and it is clearly visible how the directed curriculum created by our approach only explores necessary paths towards the target region.

4.1.3.2 Quantitative Results

After showing the qualitative evaluation of our experimental comparison between GoalGAN and Directed GoalGAN, we now aim to show numerical values which confirm the qualitative findings from the previous section. For evaluation purposes we uniformly sampled 100 states from within the reachable state space in order to measure the achieved coverage of the whole state space and 50 states from the target region to measure the target coverage. Every third iteration we then empirically measured the fraction of states that are reachable by the current policy for both evaluation sets. We furthermore compare the two GoalGAN approaches to standard Proximal Policy Optimisation (PPO) which is performed on uniformly sampled goals in order to show the advantage of using a curriculum. The results for the coverage evaluation can be seen in Figure 4.4. It is clearly visible, that the two curriculum learning approaches vastly outperform PPO. Furthermore, we see that GoalGAN achieves a better coverage of the whole reachable state space compared to Directed GoalGAN. This result is expected, as Directed GoalGAN is not intended to cover the whole state space. Furthermore, we note that Directed GoalGAN first converges to a plateau for a coverage value of about 0.4, this is because Directed

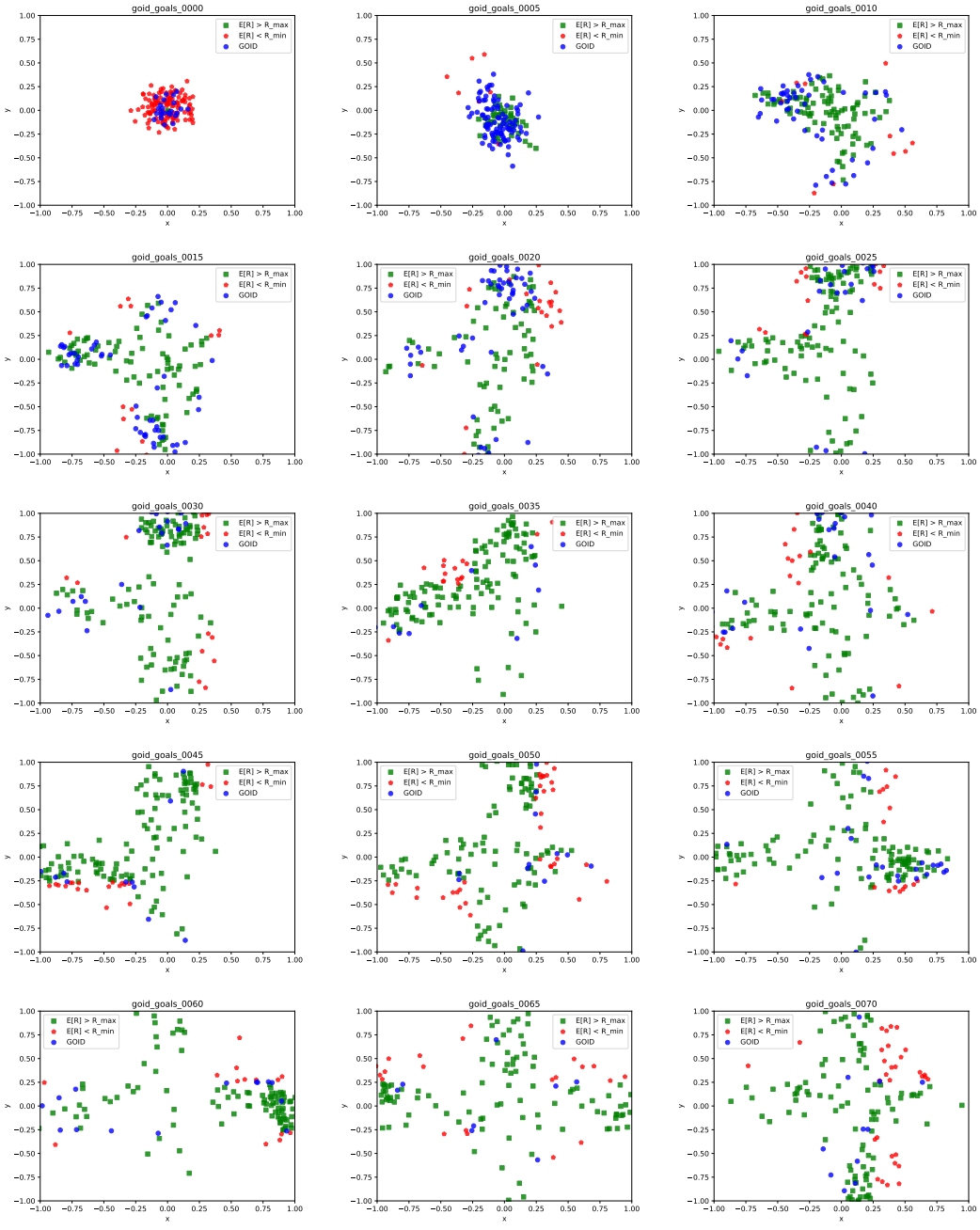


Figure 4.2: Goals generated by GoalGAN on the CrossGrid environment. Red indicates goals that are too hard, green indicates goals that have been mastered and blue indicates goal with $g \in GOID$.

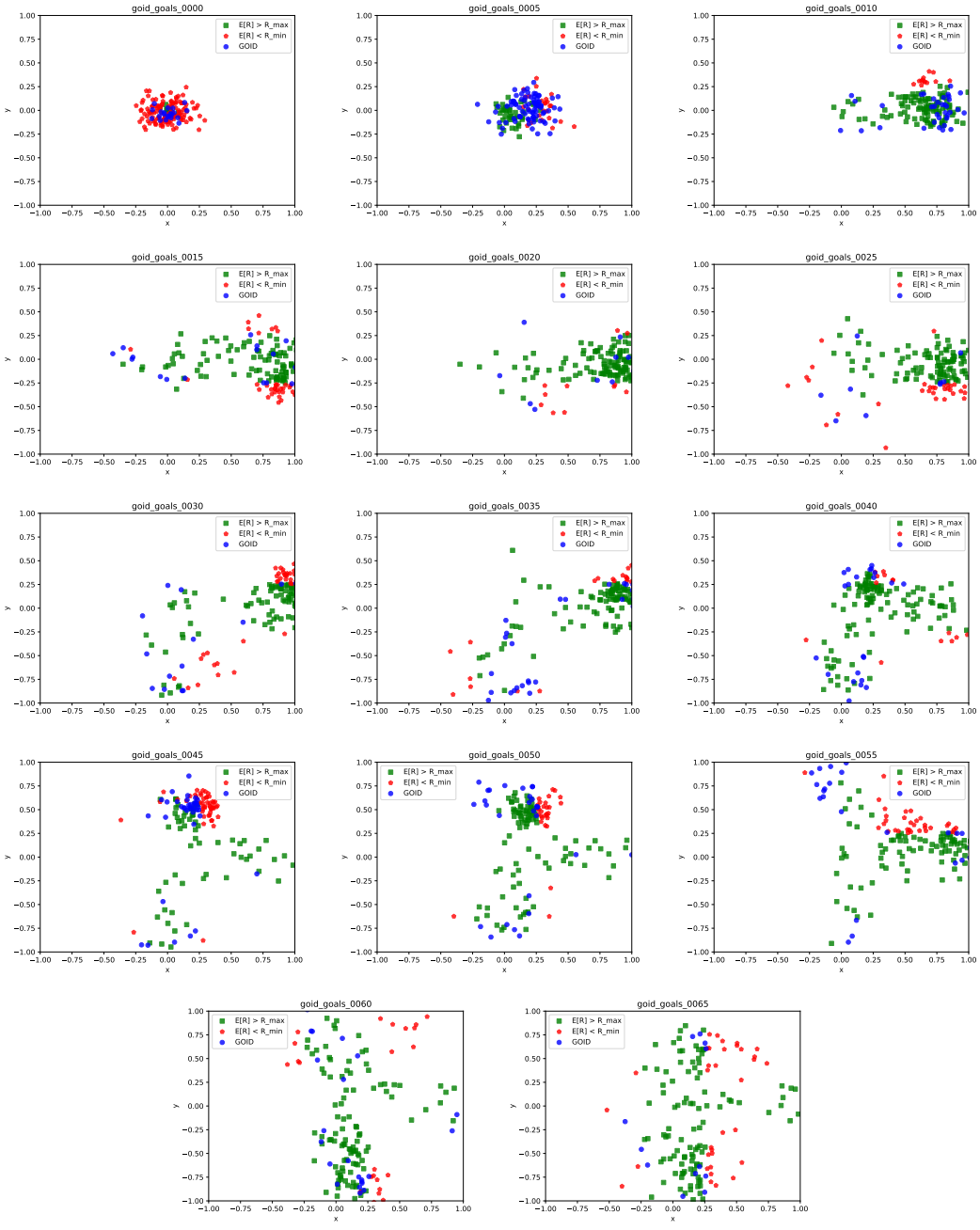


Figure 4.3: Goals generated by Directed GoalGAN on the CrossGrid environment. Red indicates goals that are too hard, green indicates goals that have been mastered and blue indicates goal with $g \in GOID$.

GoalGAN essentially first learns to reach the middle part of the cross and then the right finger, that is the target region. These two regions make up approximately 40% of the state space, which explains the plateau. Only after having learned the target region and having deemed goals within the right finger as too easy, the algorithm continues to explore new goals, as we have seen in the previous section. Figure 4.5 shows the same evaluation for the MazeGrid environment. Here we can see similar results, however in this case, standard PPO is able to perform better but is still greatly outperformed by the two curriculum generation approaches.

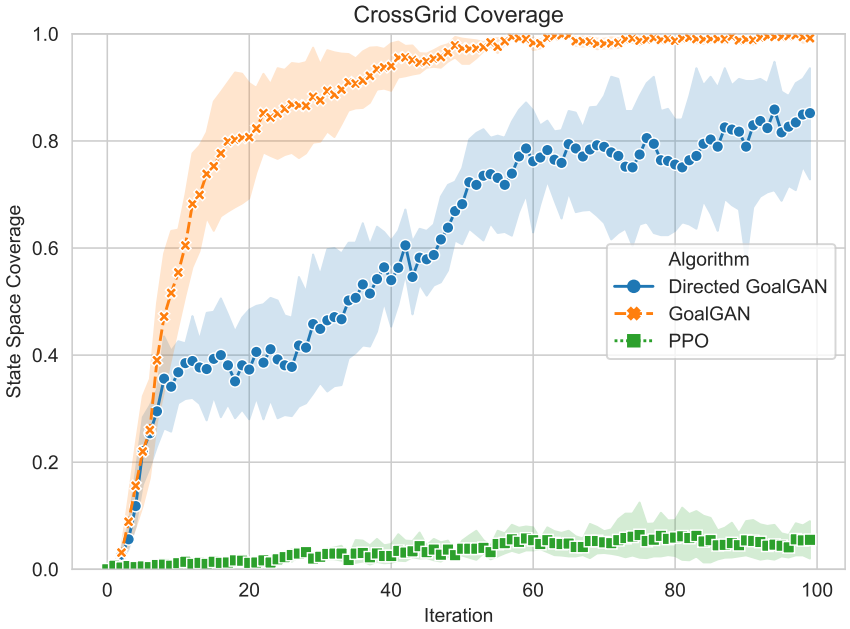


Figure 4.4: Comparison of GoalGAN, Directed GoalGAN and PPO on the state space coverage of the CrossGrid environment. Results are averaged over 5 random seeds. Shaded regions show the 95% confidence interval.

The evaluation of coverage results however is not the main objective of our experiments as we are concerned with reaching a target region. The results of our target region coverage evaluation for the two gridworld environments can be seen in Figure 4.6 for CrossGrid and Figure 4.7 for MazeGrid. In both cases we can see that Directed GoalGAN yields a better performance than GoalGAN. However, the margin between the two algorithms is significantly larger for MazeGrid. This can possibly be explained by the fact that the CrossGrid environment is easier to solve than MazeGrid as the agent is not required to turn corners.

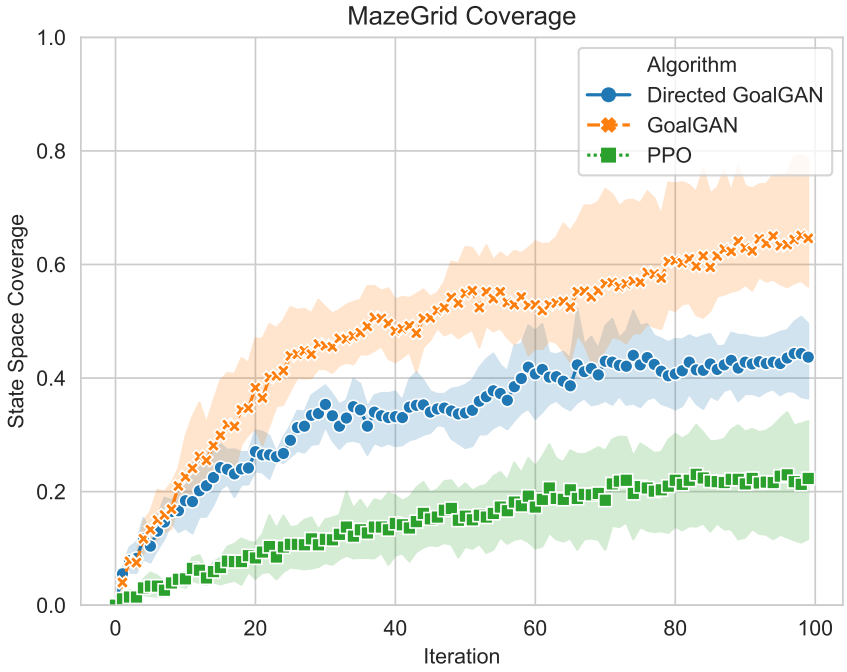


Figure 4.5: Comparison of GoalGAN, Directed GoalGAN and PPO on the state space coverage of the MazeGrid environment. Results are averaged over 5 random seeds. Shaded regions show the 95% confidence interval.

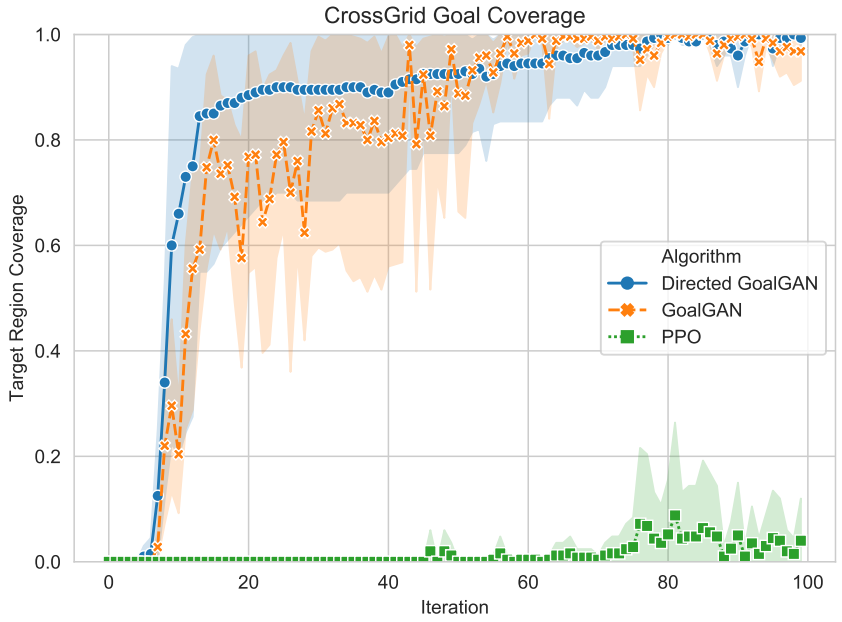


Figure 4.6: Comparison of GoalGAN, Directed GoalGAN and PPO on the target region coverage of the CrossGrid environment. Results are averaged over 5 random seeds. Shaded regions show the 95% confidence interval.

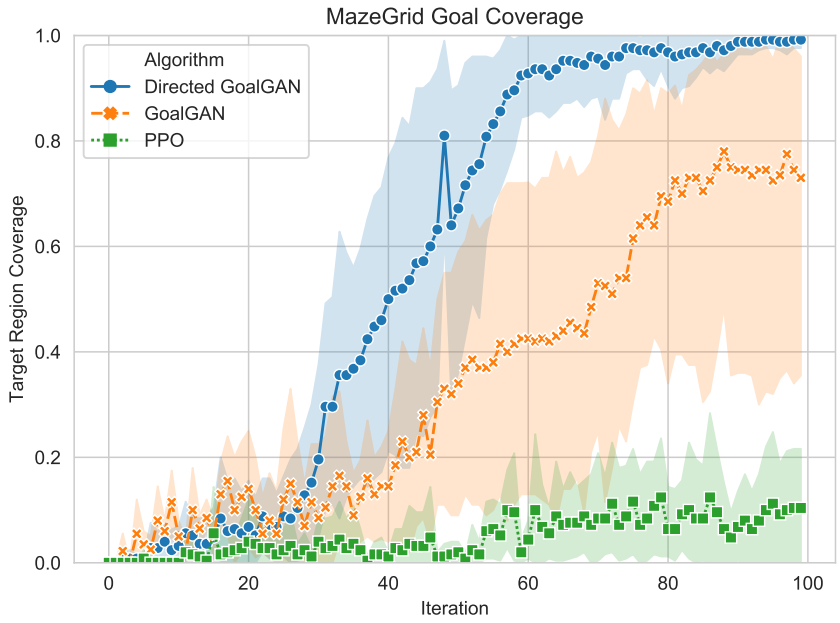


Figure 4.7: Comparison of GoalGAN, Directed GoalGAN and PPO on the target region coverage of the MazeGrid environment. Results are averaged over 5 random seeds. Shaded regions show the 95% confidence interval.

4.1.4 GoalGAN in higher-dimensional Goal Spaces

So far, we have examined GoalGAN and Directed GoalGAN on small gridworld problems. Next, we were aiming to compare them on more challenging problems. The problem we chose was hand movement of a robotic hand. For this, we used the OpenAI Gym (Brockman et al., 2016) environment for HandReach, a simulated shadow hand with 24 joints, a 63-dimensional state space, a 15-dimensional goal space and a 20-dimensional goal space. Unfortunately, we had to observe that the GoalGAN, and therefore also our Directed GoalGAN approach, is not suitable for such larger state spaces for two reasons which we will discuss in the following.

The first reason why GoalGAN is not suitable for high-dimensional tasks is the inaccuracy of GANs. GANs have been very successful in approximating data distributions, however, they are never able to capture them exactly. In reinforcement learning, this can have dramatic consequences as slight perturbations of a state can yield states which might be far away measured in the number of actions needed to reach them or they might even be infeasible. Generating unreachable goals, i.e. infeasible goals or goals too far away, hurts the performance of GoalGAN since many training episodes are wasted on unreachable goals.

The second reason is that the used reinforcement learning algorithm PPO is not sample-efficient enough to cope with such high-dimensional tasks within a reasonable timeframe. We show this by comparing the performance of standard Hindsight Experience Replay, which was designed for sample-efficiency in high-dimensional

tasks, with standard PPO. We collected a set of 25 states by running short rollouts executing random actions from the start state and used these states as initial goals for training PPO and HER. We then compared the performance of HER and PPO on these states. The results are shown in Figure 4.8. It can be seen, that HER is quickly able to learn how to reach these close-by states. On the other hand, PPO barely learns anything during the same time frame even though the goals were very close to the start state. Hence, GoalGAN is too inefficient to be applied to high-dimensional tasks.

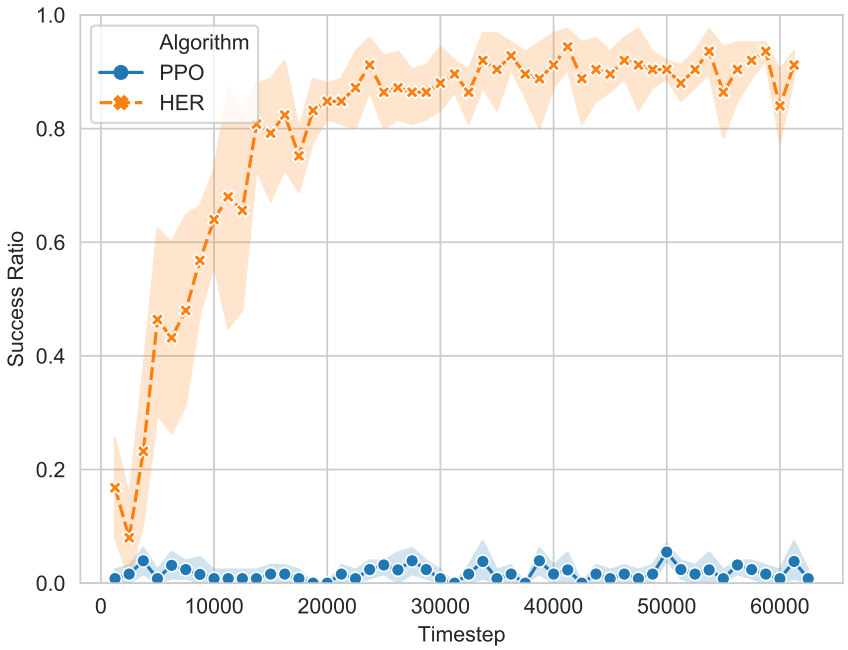


Figure 4.8: Comparison of HER and PPO on close-by states. Results are averaged over 5 random seeds. The shaded region indicates the 95% confidence interval.

4.2 Directed HER

After having presented our experimental results for our Directed GoalGAN approach, we will now continue with our second approach, Directed HER. Our experiments again started on a gridworld problem as these are great for obtaining initial results and visualising goals.

4.2.1 Gridworld

To test our Directed HER approach, we reuse the CrossGrid environment as presented in 4.1.1. We again first evaluate qualitative results by visually inspecting how the initial goals develop over time, whether this forms a curriculum and whether it is a directed curriculum.

4.2.1.1 Qualitative Results

In this section, we qualitatively analyse the behaviour of our Directed HER approach. The question we hereby want to answer is whether Directed HER is able to generate a directed curriculum. Figure 4.9 visualises the initial goals as selected by our approach. Note that the goals within the target region (on the right handside) are not sampled from achieved goals but sampled from the target region as detailled in Section 3.4. It is clearly visible that the initial goals in the beginning are concentrated around the start state as the agent is not yet capable of reaching states farther away. Over time, the initial goals clearly wander away from the start state towards the goal region and finally converge at the goal region. Hence, Directed HER generates a directed curriculum.

4.2.1.2 Quantitative Results

In our experiments we compared three different initial goal sampling strategies for Hindsight Experience Replay:

- Directed HER (our approach):
  Sample goals from achieved goals from the last iteration based on their discriminator rating.
- HER Targets:
  The initial goal is uniformly sampled from the target region.
- HER Uniform:
  The goal is uniformly sampled from the whole state space.

Figure 4.10 shows the results for the evaluation of the coverage of the whole state space. Unsurprisingly, HER Uniform achieves the best results for this metric, as it is the only algorithm intended to learn how to reach the whole state space. Our results for the target region coverage evaluation are shown in Figure 4.11. Interestingly, HER Uniform achieves a comparable performance to HER Targets. However, this finding is consistent with (Andrychowicz et al., 2017) who showed that training on multiple goals can be beneficial even if only a subset of the goals is of interest. Most

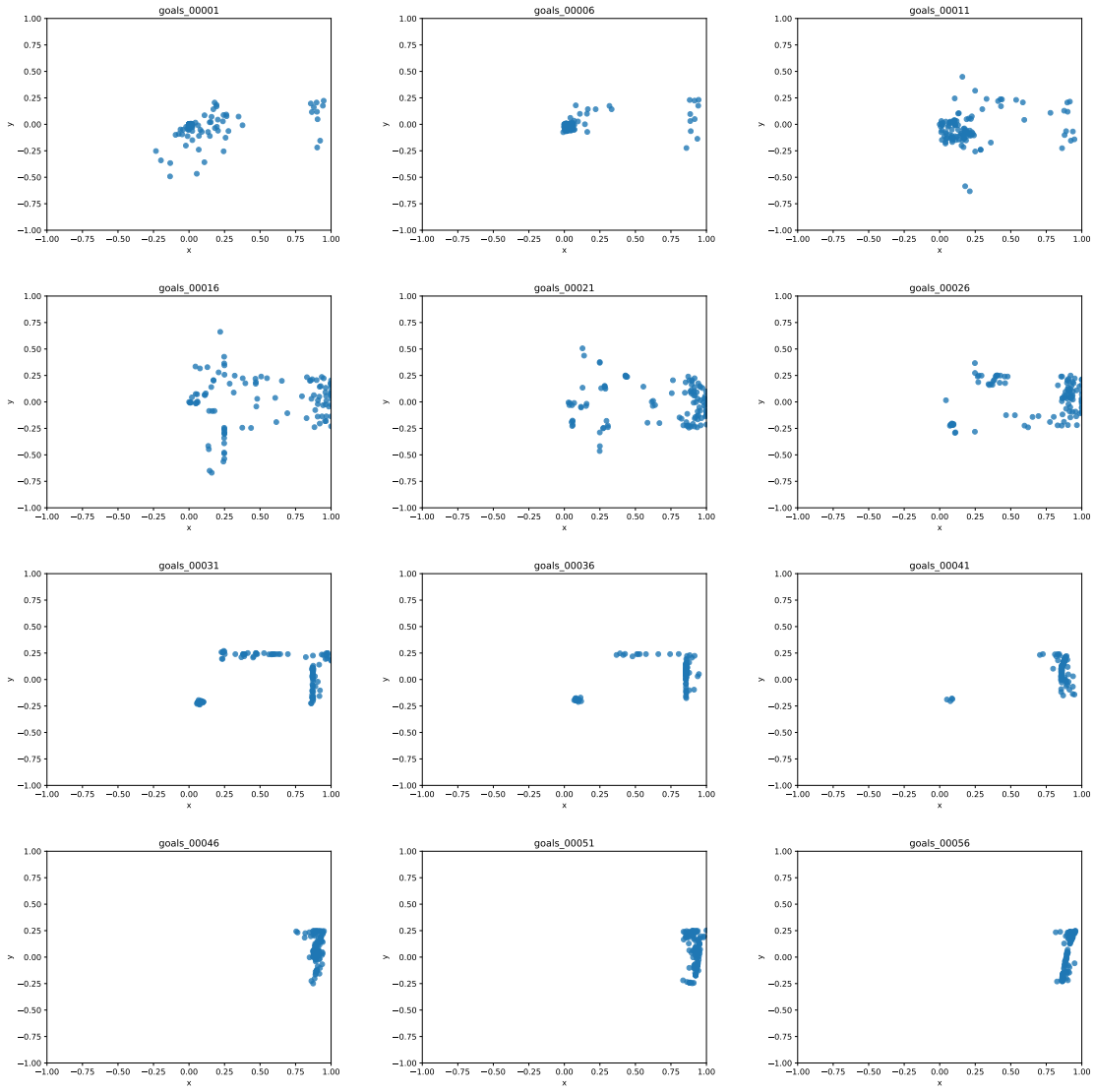


Figure 4.9: Visualisation of the initial goals selected by Directed HER over time on the CrossGrid environment.

importantly however, we can see that our approach is capable of greatly outperforming the other two methods, which confirms that an additional smart selection of initial goals can speed up the learning progress of Hindsight Experience Replay. Lastly, we observe that prioritised goal selection has no benefit compared to normal Directed HER in this case.

4.2.2 Hand Movement: Thumbs-Up

After achieving promising results on our CrossGrid environment, we wanted to apply our Directed HER algorithm to more complex and real-world tasks. The first task is to move a robot from a start configuration into a desired target configuration, a

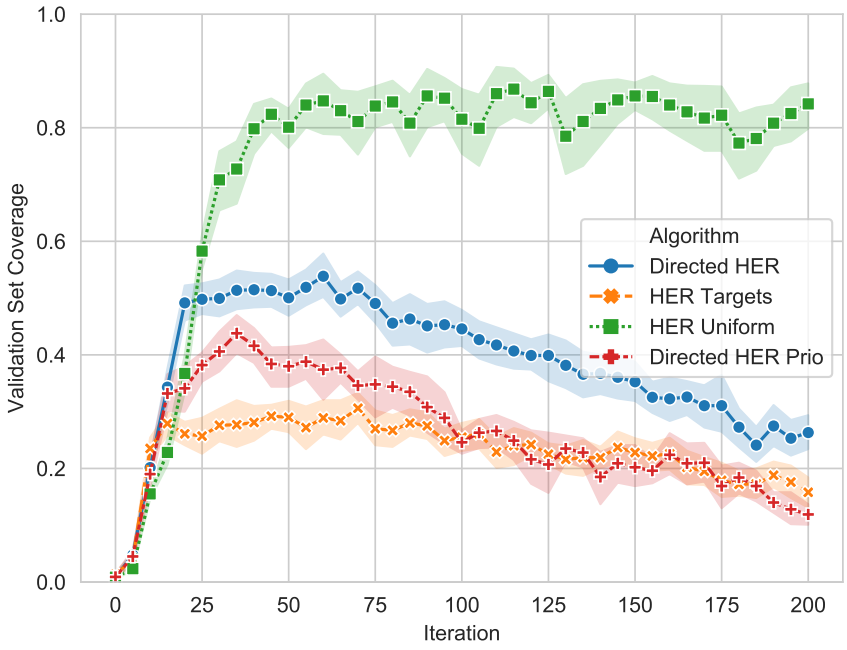


Figure 4.10: Evaluation of the coverage of the whole state space. Results are averaged over 10 random seeds and the shaded region indicates the 95% confidence interval.

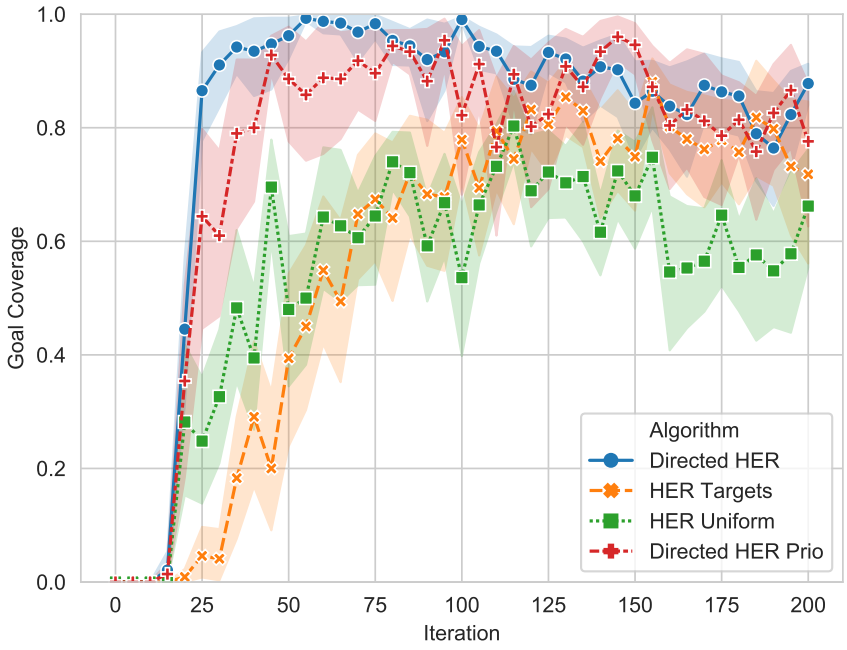


Figure 4.11: Evaluation of the coverage of the target region. Results are averaged over 10 random seeds and the shaded region indicates the 95% confidence interval.

problem often arising in robotics. More specifically, we consider the problem where a robotic hand has to transition from an initially relaxed position into a thumbs-up position.

4.2.2.1 Environment

The robotic hand environment we used is part of the OpenAI gym framework (Brockman et al., 2016) and is based on the physics-simulator Mujoco (Todorov, Erez & Tassa, 2012). Details of environment specifications are shown in Table 4.2.

State Space	$\mathbb{R}^{63}$
Goal Space	$\mathbb{R}^{15}$: xyz-positions for all fingertips
Action Space	$\mathbb{R}^{20}$
Distance Threshold ϵ	0.01
Maximum steps T	50

Table 4.2: Environment specifications for the Shadow Hand environment.

We slightly modified the original environment by switching from absolute to relative control and by decreasing the number of times an action is repeated from 20 to 5 in order to generate a more challenging problem with a longer horizon. The start position of our task is a natural, relaxed hand position and is shown in Figure 4.12. The goal of our experiment is to teach the shadow hand to transition to a thumbs-up position. In order to create a set of target goals, we hand-collected 100 example states of the shadow hand. All 100 states are slightly different but all show a clear thumbs-up. Three examples of such states are shown in Figure 4.13. In each episode the agent has 50 time steps to reach the desired goal configuration and receives a reward of -1 for every state in which the ℓ_2-distance between the achieved goal and the desired goal is larger than 0.01 and a reward of 0 for all states in which the desired configuration has been reached.

4.2.2.2 Implementation Details

We implemented our Directed HER approach using the HER implementation of OpenAI Baselines (Dhariwal et al., 2017). Each iteration consists of 25 episodes. After each episode the policy is updated using 20 batches of size 128, all other hyperparameters were identical to the ones used in (Andrychowicz et al., 2017). Before each iteration, the target discriminator is updated and used to sample new goals from the achieved goals from the previous iteration.

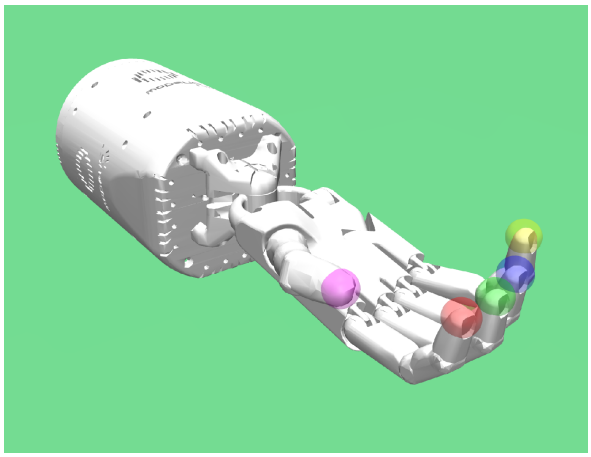


Figure 4.12: Startposition of the Shadow Hand.

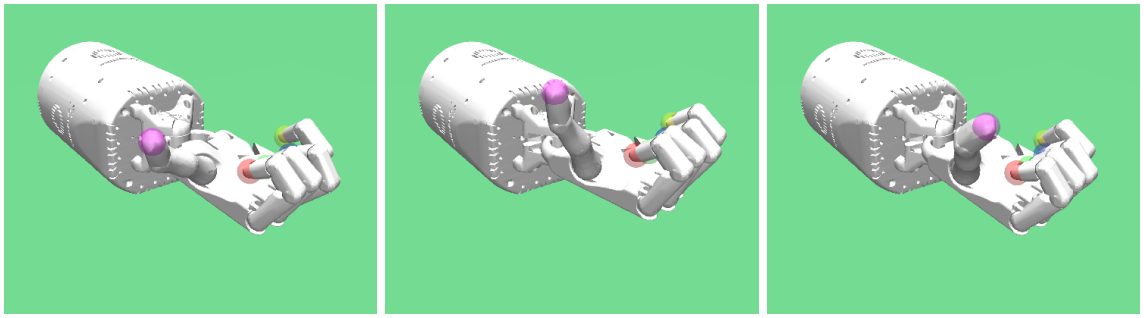


Figure 4.13: Examples of Goal Positions Thumbs-Up.

4.2.2.3 Results

Similarly to previous experiments, we again start by qualitatively analysing the performance of our algorithm. Herein, we visually inspect the initial goals that our algorithm selects in each iteration and evaluate whether they look increasingly similar to the target states. Afterwards, we compare our Directed HER approach to normal Hindsight Experience Replay with either uniformly selected goals or goals sampled from the target set.

4.2.2.3.1 Qualitative Results The first questions we again need to answer are whether our approach generates a curriculum, i.e. if the proposed initial goals are of increasing difficulty, and whether this curriculum is directed towards the target region. For this purpose, we run our algorithm and render the set of goals that our algorithm proposes as initial goals and inspect them visually. The results of this evaluation are depicted in Table 4.3. The shown images are the goals which we deemed to be visually most similar to the target region. Therefore, the shown images are essentially the best of all sampled goals in each iteration. It is clearly visible that the best goals of each iteration are more and more similar to the desired target pose. As these best goals correspond to the best states visited by the agent, this means that

the agent reaches states closer and closer to the target region with every iteration. Hence, our algorithm indeed does generate a curriculum of increasing difficulty. This curriculum is furthermore also directed as we intended, since other movement directions, e.g. two fingers bent and two fingers stretched are not selected as goals. In the next section we will now evaluate whether this curriculum actually helps to accelerate the training process.

4.2.2.3.2 Quantitative Results As for the gridworld experiment, we once again compare our Directed HER approach to HER with uniform goal selection and HER with initial goals sampled from the set of target goals. Additionally, we also test another initial goal sampling strategy called HER L2. This algorithm samples the initial goals in from the same set as our Directed HER approach with the key difference that instead of a discriminator rating, the points with the smallest ℓ_2 distance to a randomly selected target state are selected. We include this algorithm in our comparison to show that the similarity measure indeed has to be learned and cannot simply be assumed to be a distance metric. In contrast to our performed gridworld experiments, we cannot evaluate the coverage of the whole state space as the state space of the shadow hand is too large to cover it with a grid and evaluating the policy for every grid point. As we primarily care about reaching the target region, this state space coverage metric is neglectable. Instead, we only measure the coverage of the target region. For this, we divide our target set of 100 thumbs-up poses into a training set consisting of 75 states and a validation set consisting of 25 states. The training set is used in our Directed HER approach in order to train the discriminator and to augment the set of sampled goals and in the HER Targets approach these 75 are used in order to sample the initial goal. The validation goals however are solely used for evaluating the performance of the algorithms. Figure 4.14 visualises our findings. While our Directed HER approach indeed does generate a directed curriculum as shown in the previous section and furthermore greatly outperforms HER L2 and normal HER with uniform goal selection, it unfortunately yields worse performance than HER Targets.

4.2.3 In-Hand Cube Manipulation

While hand movement is an interesting and challenging problem, HER can still easily learn this task fairly quickly without an additional directed curriculum. We therefore tested our algorithm on another more challenging task, namely the in-hand manipulation of a cube where a robotic hand needs to rotate a cube by 90 degrees around the x-axis.

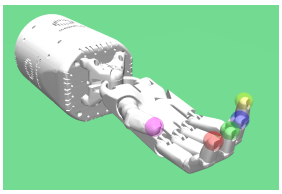
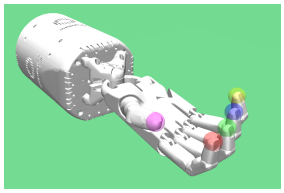
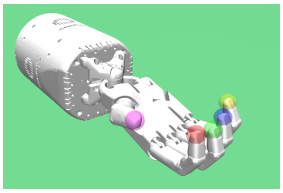
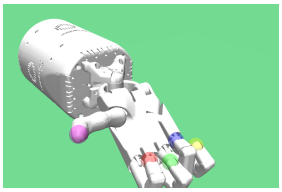
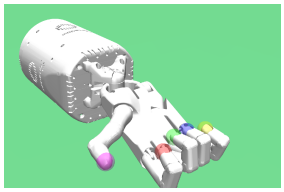
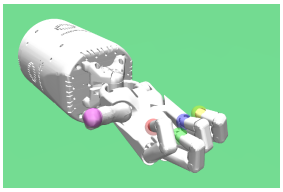
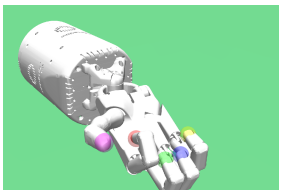
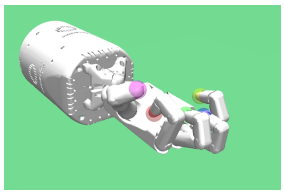
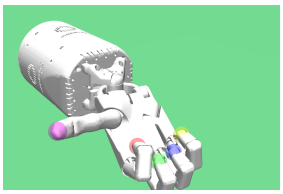
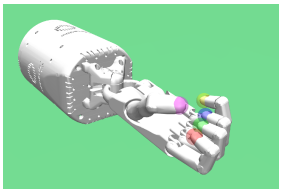
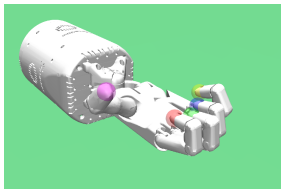
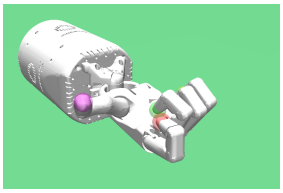
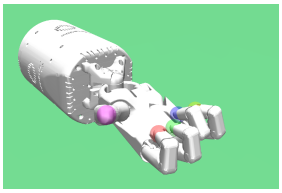
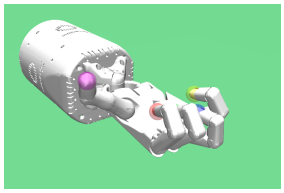
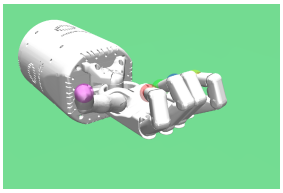
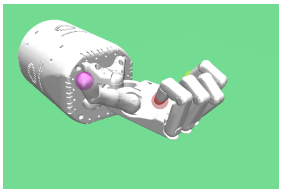
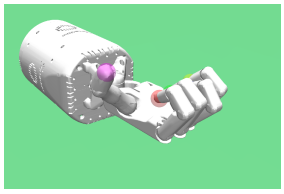
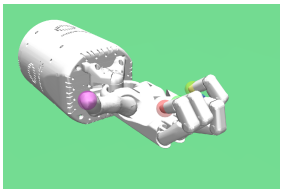
Epoch	Examples of sampled initial goals.		
1			
3			
5			
7			
9			
11			

Table 4.3: Best initial goals selected by HER of each epoch. All goals were visually assessed and the most similar ones to the targets were chosen.

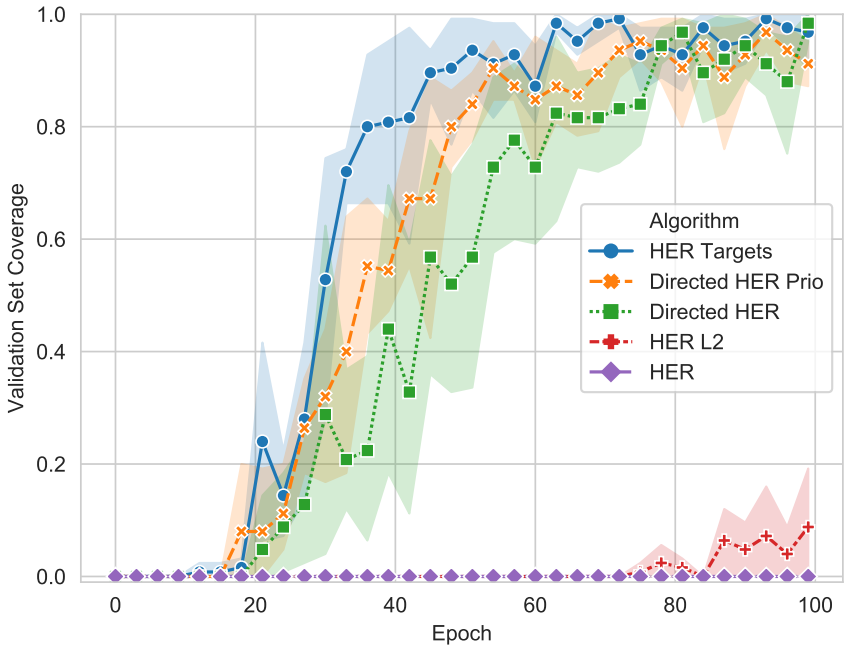


Figure 4.14: Thumbs-Up Experiment: Coverage of the Validation Set. Results are averaged over 5 random seeds. Shaded regions indicate the 95% confidence interval.

4.2.3.1 Environment

Once again, this environment was taken from the OpenAI Gym framework (Brockman et al., 2016) and is implemented using Mujoco (Todorov et al., 2012). The start state of the environment is visualised in Figure 4.15. We again hand-collected a set of 100 target states and split this set into a training set containing 75 states and a validation set consisting of 25 states. Three examples of these states are shown in Figure 4.16. State, action and goal spaces are again equivalent to those presented in (Plappert et al., 2018). In each episode the agent has 75 timesteps to reach the desired goal configuration and receives a reward of -1 for each state in which the desired goal is not reached and a reward of 0 if the goal is reached.

State Space	$\mathbb{R}^{61}$
Goal Space	$\mathbb{R}^7$: xyz-positions of the cube and rotation represented as quaternions.
Action Space	$\mathbb{R}^{20}$
Distance Threshold ϵ	0.05
Rotation Threshold ϵ	0.1
Maximum steps T	75

Table 4.4: Environment specifications for the cube manipulation environment.

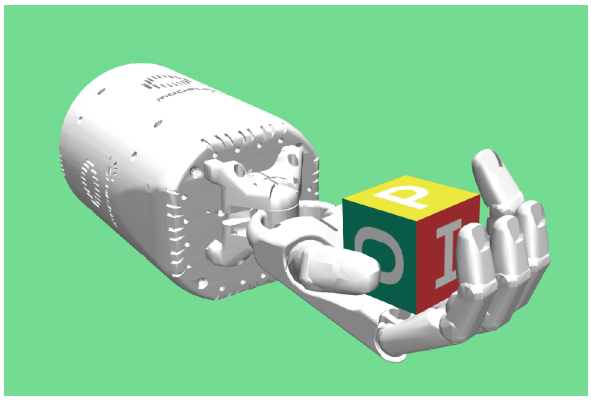


Figure 4.15: Startposition of the in-hand cube manipulation environment.

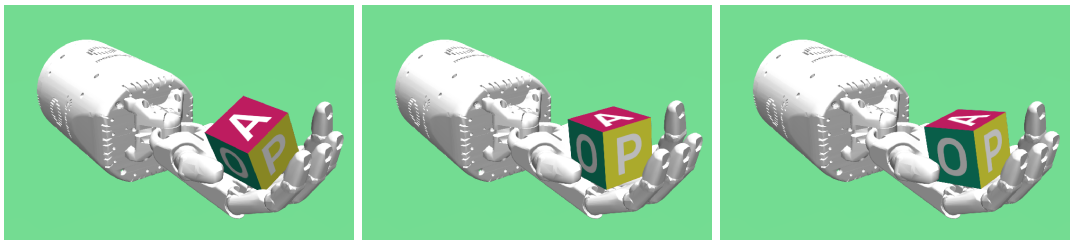


Figure 4.16: Examples of Target Positions for the Cube Rotation Task.

4.2.3.2 Implementation Details

We again implemented our Directed HER approach using the HER implementation of OpenAI Baselines (Dhariwal et al., 2017). Each iteration consists of 50 episodes. After each episode the policy is updated using 20 batches of size 128, all other hyperparameters were identical to the ones used in (Andrychowicz et al., 2017). Before each iteration, the target discriminator is updated and used to sample new goals from the achieved goals from the previous iteration.

4.2.3.3 Results

As for all previous experiments, we again split the report of our results into a qualitative analysis part and a quantitative analysis part. We will again start the evaluation with the qualitative results where we inspect whether the sampled initial goals create a directed curriculum converging to states similar to the given target states. In the quantitative part we will then measure whether our approach is able to speed-up learning of the task on hand.

4.2.3.3.1 Qualitative Results We once again inspect the generated curriculum by visualising the initial goals sampled by our Directed HER approach. A curriculum for cube rotation ideally achieves to rotate to cube by a little bit more each epoch.

Furthermore, we hope that the generated curriculum is generated towards the goal region, and therefore the rotation should progress mainly around the x-axis while other rotation directions should ideally not be explored at all. We again manually pick the best three goals form all goals sampled in one iteration. These selected goals are visualised in Table 4.5. It can be seen that the results are as expected, the agent achieves further and further rotated states over time and the rotation progresses only around the x-axis. Therefore, Directed HER indeed manages to generate a directed curriculum. However, the performance still needs to be numerically compared with the original HER method in order to determine whether this curriculum actually helps to learn faster.

4.2.3.3.2 Quantitative Results The quantitative evaluation of our algorithm was conducted in equivalent fashion as the thumbs-up evaluation (see Section 4.2.2.3.2). We again compare the performance of Directed HER with HER Targets, HER L2 and HER Uniform by measuring the fraction of validation states which can be reached over time. The results of this experiment are shown in Figure 4.17. It can be seen that our Directed HER approach fails completely on this task, however the Directed HER approach with prioritised goal sampling manages to outperform even HER Targets, which has not been the case for our Thumbs-Up experiment. This successful experiment brings us to the conclusion that the thumbs-up experiment was too easy for HER and did not need an additional directed curriculum. We furthermore conclude that a directed curriculum is able to improve the performance of HER on complex tasks such as in-hand manipulation.

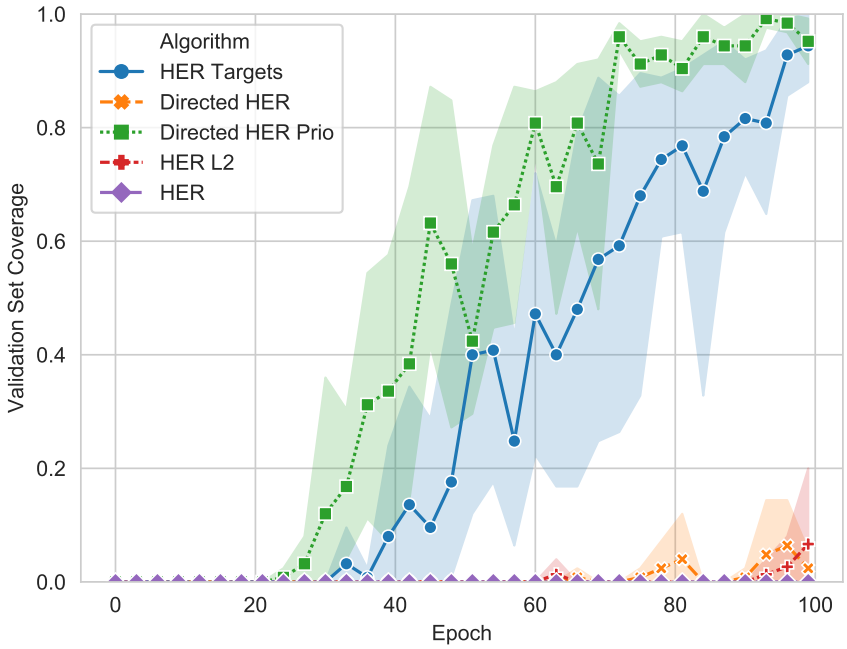


Figure 4.17: Cube Rotation: Coverage of the Validation Set. Results are averaged over 5 random seeds. Shaded regions indicate the 95% confidence interval.

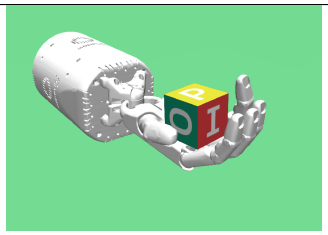
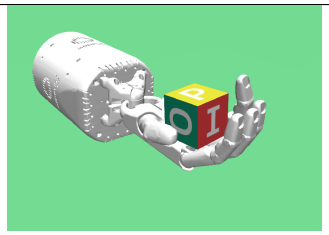
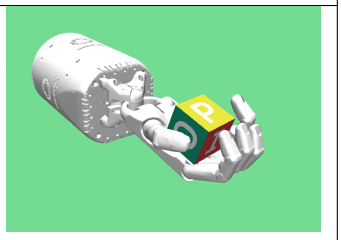
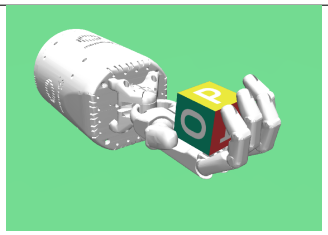
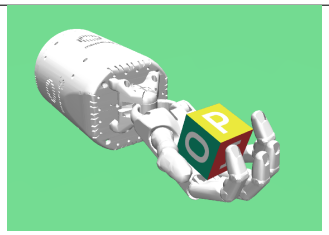
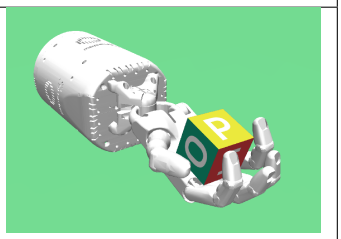
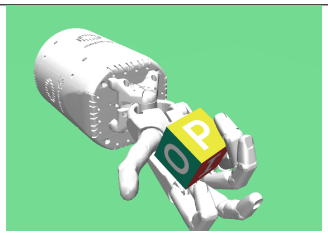
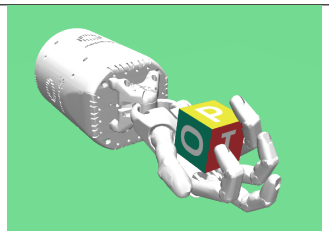
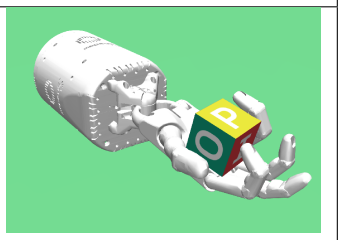
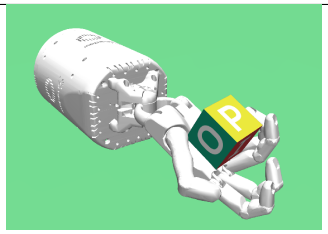
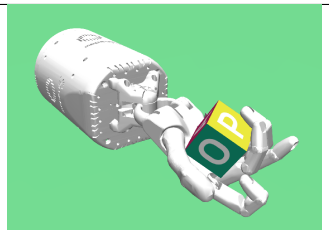
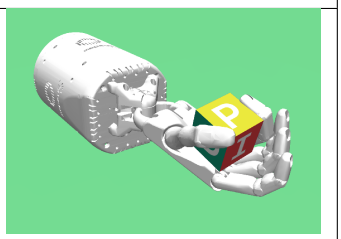
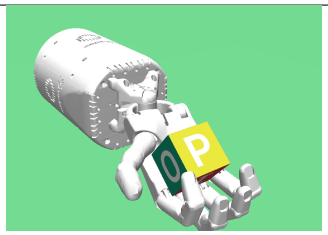
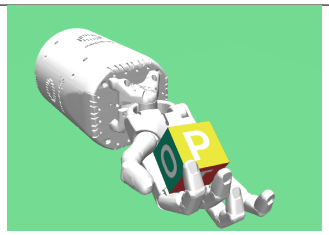
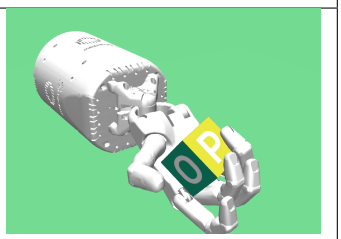
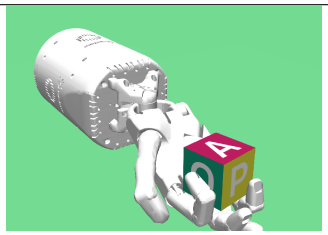
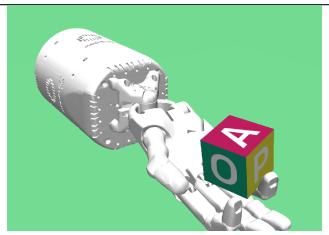
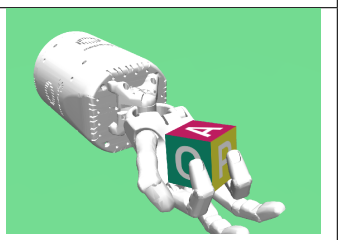
Epoch	Examples of sampled initial goals.		
1			
11			
21			
31			
41			
51			

Table 4.5: Best initial goals selected by HER of each epoch. All goals were visually assessed and the most similar ones to the targets were chosen.

4.3 Summary

In this chapter, we presented experimental evaluations of our contributions. We have shown that Directed GoalGAN is able to generate a directed curriculum which almost exclusively explores goals towards the target region. Furthermore, Directed GoalGAN is able to reach the target region faster than GoalGAN. Unfortunately, we had to observe that GoalGAN is not viable for high-dimensional goal spaces. We furthermore examined our Directed HER approach on a gridworld and two robotics tasks. The results show that Directed HER is in some cases able to outperform standard HER.

Chapter 5

Conclusion and Future Work

In this project we examined how existing automatic curriculum learning approaches can be modified in order to bias the curriculum progression towards a pre-defined target region. We introduce the idea of training a discriminator network, inspired by generative adversarial networks, to compute a measure of similarity or usefulness. This measure is then used to evaluate the usefulness of a set of goal states and we then primarily train on goals of high interest in order to direct the curriculum towards the targets.

We applied this general principle to the two curriculum methods GoalGAN and HER. Our experiments show that this goal bias can sometimes help to speed-up learning, but however sometimes only yields comparable or even slightly worse performance than undirected approaches.

Curriculum learning is a very convenient framework for training reinforcement learning agents as it alleviates learning of a difficult task. We furthermore believe, that even though some of our experiments showed no improvement compared to undirected methods, the idea of generating a curriculum which mainly expands towards a target region has high potential.

Future work could adapt our idea to learn a measure of usefulness through the training of a discriminator network, but employ the idea in different ways. (Plappert et al., 2018) proposed a research request in which the replay goals for Hindsight Experience Replay are not chosen randomly, which favours states with high visitation frequency and therefore states reachable within shorter policy rollouts, but instead to select the replay goals in a way that favours states which are most interesting to learn from, similar to prioritized experience replay (Schaul, Quan, Antonoglou & Silver, 2015). It would be interesting, if this prioritisation measure could be learned in a similar fashion to our discriminator training.

An extension of this project could furthermore explore the benefits of our approach on tasks with longer planning horizon such as object assembly. As our robotics tasks

have a fairly short horizon, we suspect that a curriculum is not necessarily needed to learn the task, which could be a reason why our Directed HER approach only yielded minor performance improvements if any.

Appendices

Appendix A

Ethics Checklist

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		x
Does your project involve the use of human embryos?		x
Does your project involve the use of human foetal tissues / cells?		x
Section 2: HUMANS		
Does your project involve human participants?		x
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from Human Embryos/Foetuses i.e. Section 1)?		x
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		x
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		x
Does it involve processing of genetic information?		x
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		x
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		x
Section 5: ANIMALS		
Does your project involve animals?		x

	Yes	No
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		x
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		x
Could the situation in the country put the individuals taking part in the project at risk?		x
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		x
Does your project deal with endangered fauna and/or flora /protected areas?		x
Does your project involve the use of elements that may cause harm to humans, including project staff?		x
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		x
Section 8: DUAL USE		
Does your project have the potential for military applications?	x	
Does your project have an exclusive civilian application focus?	x	
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		x
Does your project affect current standards in military ethics - e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		x
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		x
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		x
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		x
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		x
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		x
Will your project use or produce goods or information for which there are data protection, or other legal implications?		x
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		x

Appendix B

Ethics Considerations Summary

The only questions from the ethics checklist we marked with yes are:

- ”Does your project have the potential for military applications?”
  As this project is in the area of machine learning / reinforcement learning and deals with robotics, our approaches could potentially be used in military robots. We are dealing with cases in which an agent is supposed to learn how to reach a target region or configuration, while this can be used in a military robot as well, it would not be a dangerous advance in military applications.
- ”Does your project have an exclusive civilian focus?”
  For this question, we researched the definition on https://ec.europa.eu/research/participants/portal/doc/call/h2020/ds-04-2015/1645170-explanatory_note_on_exclusive_focus_on_civil_applications_en.pdf. Our project does have potential military applications, as any AI and robotics related research projects do, however as we only intend to use our approaches for general research in robotics and explicitly do not intend to propose a method which advances military robotics, our project indeed only focuses on civilian use cases.

All other questions could safely be marked with no, as we do not process any personal or sensitive data and do not deal with any physical materials or living creatures, neither animals or humans.

Appendix C

Experiment: Goal Development MazeGrid

Here we show the goal development of GoalGAN (Figure C.1 versus Directed GoalGAN (Figure C.2 for the MazeGrid environment. The results confirm the presented results for CrossGrid as presented in Section 4.1.3.1. We can again see that the curriculum generated by GoalGAN expands into all possible directions while our Directed GoalGAN approach is able to generate a directed curriculum which almost exclusively explores the necessary pathways.

Figure C.1: Goals generated by GoalGAN on the MazeGrid environment. Red indicates goals that are too hard, green indicates goals that have been mastered and blue indicates goal with $g \in GOID$.

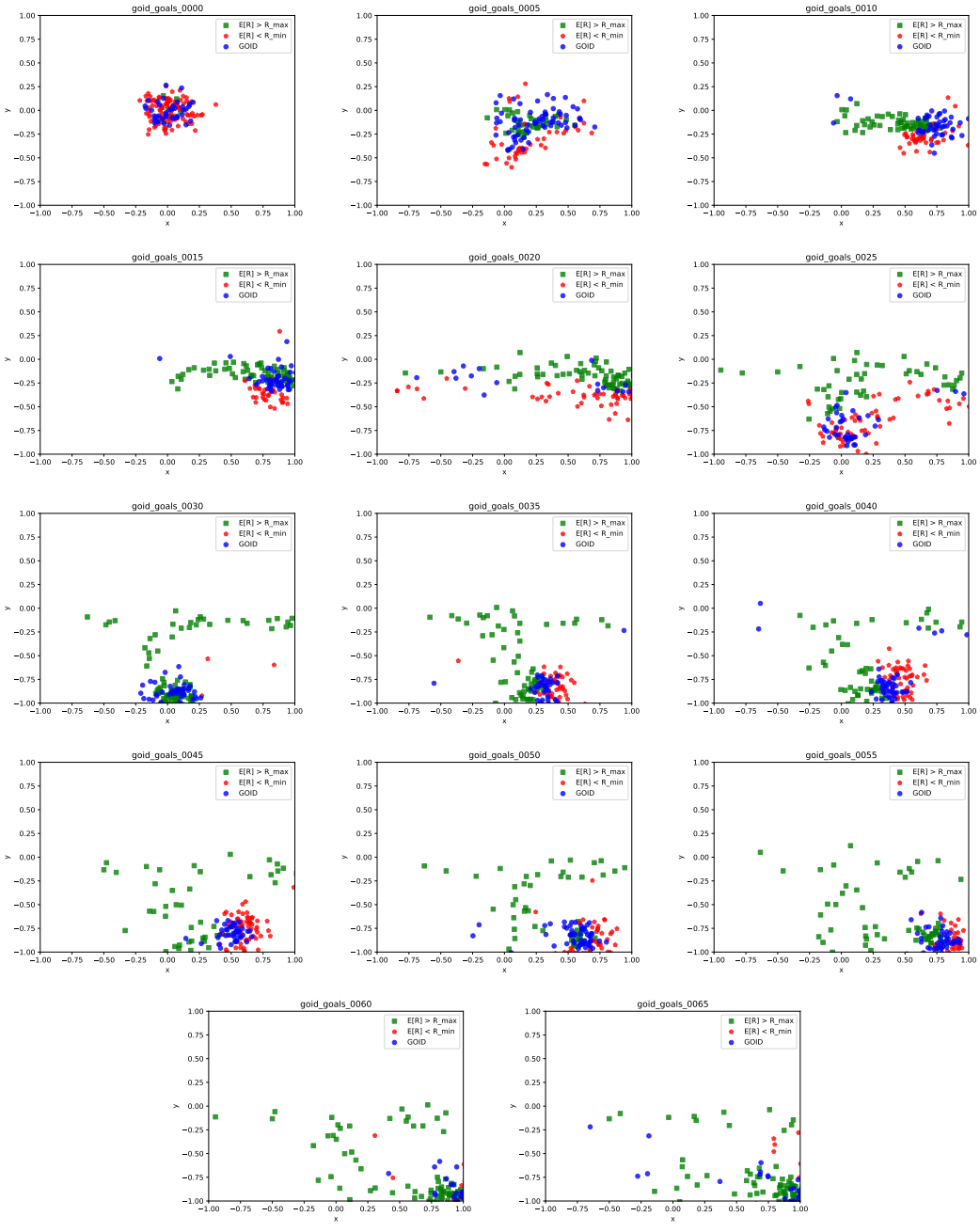


Figure C.2: Goals generated by Directed GoalGAN on the MazeGrid environment. Red indicates goals that are too hard, green indicates goals that have been mastered and blue indicates goal with $g \in GOID$.

Bibliography

Abbeel, P. & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on machine learning*. Banff, Alberta, Canada: ACM. doi:10.1145/1015330.1015430

Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive science*, *9*(1), 147–169.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., ... Zaremba, W. (2017). Hindsight experience replay. In *Advances in neural information processing systems* (pp. 5048–5058).

Arjovsky, M., Chintala, S. & Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv: 1701.07875*.

Barto, A. G., Sutton, R. S. & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5), 834–846.

Bengio, Y., Louradour, J., Collobert, R. & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 41–48). ACM.

Bengio, Y., Yao, L., Alain, G. & Vincent, P. (2013). Generalized denoising auto-encoders as generative models. In *Advances in neural information processing systems* (pp. 899–907).

Bogdanovic, M., Markovikj, D., Denil, M. & De Freitas, N. (2015). Deep apprenticeship learning for playing video games.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Cabi, S., Colmenarejo, S. G., Hoffman, M. W., Denil, M., Wang, Z. & de Freitas, N. (2017). The intentional unintentional agent: Learning to solve many continuous control tasks simultaneously. *CoRR*, *abs/1707.03300*. arXiv: 1707.03300. Retrieved from http://arxiv.org/abs/1707.03300

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., ... Wu, Y. (2017). Openai baselines. *GitHub, GitHub repository*.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M. & Abbeel, P. (2017). Reverse curriculum generation for reinforcement learning. In *Conference on robot learning* (pp. 482–495).

Ganin, Y., Kulkarni, T., Babuschkin, I., Eslami, S. & Vinyals, O. (2018). Synthesizing programs for images using reinforced adversarial learning. *arXiv preprint arXiv:1804.01118*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Gu, S., Holly, E., Lillicrap, T. & Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Robotics and automation (icra), 2017 ieee international conference on* (pp. 3389–3396). IEEE.

Held, D., Geng, X., Florensa, C. & Abbeel, P. (2017). Automatic goal generation for reinforcement learning agents. *arXiv preprint arXiv:1705.06366*.

Held, D., Geng, X., Florensa, C. & Abbeel, P. (2018). Automatic goal generation for reinforcement learning agents. *International Conference on Machine Learning*.

Ho, J. & Ermon, S. (2016). Generative adversarial imitation learning. (pp. 4565–4573).

Karpathy, A. & Van De Panne, M. (2012). Curriculum learning for motor skills. In *Canadian conference on artificial intelligence* (pp. 325–330). Springer.

Kuefler, A., Morton, J., Wheeler, T. & Kochenderfer, M. (2017). Imitating driver behavior with generative adversarial networks. In *Intelligent vehicles symposium (iv), 2017 ieee* (pp. 204–211). IEEE.

Kumar, A., Paul, N. & Omkar, S. (2018). Bipedal walking robot using deep deterministic policy gradient. *arXiv preprint arXiv:1807.05924*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z. & Smolley, S. P. (2017). Least squares generative adversarial networks. In *Computer vision (iccv), 2017 ieee international conference on* (pp. 2813–2821). IEEE.

Mirza, M. & Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Ostrovski, G. et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

Molchanov, A., Hausman, K., Birchfield, S. & Sukhatme, G. (2018). Region growing curriculum generation for reinforcement learning. *arXiv preprint arXiv: 1807.01425*.

Ng, A. Y. [A. Y.], Russell, S. J. et al. (2000). Algorithms for inverse reinforcement learning. In *Icml* (pp. 663–670).

Ng, A. Y. [Andrew Y], Harada, D. & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml* (Vol. 99, pp. 278–287).

Nie, S., Wang, Z. & Ji, Q. (2017). A Generative Restricted Boltzmann Machine Based Method for High-Dimensional Motion Data Modeling. *ArXiv e-prints*. arXiv: 1710.07831 [cs.CV]

Peters, J. & Schaal, S. (2006). Policy gradient methods for robotics. In *Intelligent robots and systems, 2006 ieee/rsj international conference on* (pp. 2219–2225). IEEE.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., ... Zaremba, W. (2018). Multi-goal reinforcement learning: Challenging robotics environments and request for research. *CoRR*, *abs/1802.09464*. arXiv: 1802.09464. Retrieved from http://arxiv.org/abs/1802.09464

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B. & Lee, H. (2016). Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*.

Sammut, C., Hurst, S., Kedzier, D. & Michie, D. (1992). Learning to fly. In *Machine learning proceedings 1992* (pp. 385–393). Elsevier.

Sanger, T. D. (1994). Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE Transactions on Robotics and Automation*, *10*(3), 323–333. doi:10.1109/70.294207

Schaul, T., Quan, J., Antonoglou, I. & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. & Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning* (pp. 1889–1897).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D. & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *Icml*.

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A. & Fergus, R. (2017). Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*.

Tang, Y. & Salakhutdinov, R. R. (2013). Learning stochastic feedforward neural networks. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 26* (pp. 530–538). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/5026-learning-stochastic-feedforward-neural-networks.pdf

Todorov, E., Erez, T. & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *Intelligent robots and systems (iros), 2012 ieee/rsj international conference on* (pp. 5026–5033). IEEE.

Vroman, M. C. (2014). *Maximum likelihood inverse reinforcement learning*. Rutgers The State University of New Jersey-New Brunswick.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K. & de Freitas, N. (2016). Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*.

Watkins, C. J. & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3-4), 279–292.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3-4), 229–256.

Yu, L., Zhang, W., Wang, J. & Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. (pp. 2852–2858).

Zaremba, W. & Sutskever, I. (2014). Learning to execute. *CoRR*, *abs/1410.4615*. arXiv: 1410.4615. Retrieved from http://arxiv.org/abs/1410.4615

Zhang, H., Xu, T., Li, H., Zhang, S., Huang, X., Wang, X. & Metaxas, D. (2017). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint*.

Ziebart, B. D., Maas, A. L., Bagnell, J. A. & Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Aaai* (Vol. 8, pp. 1433–1438). Chicago, IL, USA.