What are the implications of Curriculum Learning strategy on IRL methods? Investigating Inverse Reinforcement Learning from Human Behavior

Mikhail Vlasenko[1]

Supervisor(s): Luciano Cavalcante Siebert[1]**, Angelo Caregnato Neto**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Mikhail Vlasenko
Final project course: CSE3000 Research Project
Thesis committee: Luciano Cavalcante Siebert, Angelo Caregnato Neto, Jana Weber

Abstract

Inverse Reinforcement Learning (IRL) is a subfield of Reinforcement Learning (RL) that focuses on recovering the reward function using expert demonstrations. In the field of IRL, Adversarial IRL (AIRL) is a promising algorithm that is postulated to recover non-linear rewards in environments with unknown dynamics. This study investigates the potential benefits of applying the Curriculum Learning (CL) strategy to the AIRL algorithm. For our experiments, we use a randomized partially observable Markov decision process in the form of a grid-world-like environment. Using only expert demonstrations obtained with an RL algorithm under the true reward function, we train AIRL in a variety of configurations and identify an effective curriculum. Our results show, that a well-constructed curriculum can enhance the performance of AIRL twofold in both key aspects: the speed of convergence and the efficiency of using expert demonstrations. We thus conclude that CL can be a useful addition to an AIRL-based solution. Full code is available online in the supplementary material https://github.com/mikhail-vlasenko/curriculum-learning-IRL.

1 Introduction

Inverse Reinforcement Learning (IRL) is an aspect of machine learning that enables artificial agents to infer the reward function from observed expert demonstrations. Such reward function can later be used to train a Reinforcement Learning (RL) agent in an environment. Utilizing a reward learned by IRL can be advantageous compared to employing a hand-crafted one (Coates et al., 2008). This is particularly the case in certain environments, such as driving, where defining a useful reward function may be challenging, yet expert demonstrations are comparatively easier to acquire (Abbeel & Ng, 2004).

Several IRL strategies have been proposed to recover the rewards of an environment. Notably, Ng, Russell, et al. (2000) laid the foundation for IRL methodologies. The authors proposed novel algorithms for apprenticeship learning, a setting where the agent learns from an expert's demonstrations.

Ziebart et al. (2010, 2008) proposed Maximum Entropy Inverse Reinforcement Learning - a method that takes a probabilistic, maximum entropy approach to resolve ambiguity (Ng et al., 1999) in determining the expert's intentions. This approach provides a solid mathematical foundation but is model-based and can only extract linear rewards. Thus, it finds limited applications in complex environments (Levine et al., 2011). Later, Wulfmeier et al. (2015) extended the Maximum Entropy IRL algorithm with deep convolutional neural networks (Lecun et al., 1998). Still, the performance was only demonstrated for relatively simple problems.

More recently, Fu et al. (2018) introduced an adversarial training approach to IRL in their paper about Adversarial Inverse Reinforcement Learning (AIRL). The authors show that rewards recovered by AIRL generalize better than those produced by previous methods and, crucially, are more robust to changes in the environment during training. Unlike previous methods, AIRL has been shown to perform well even in high-dimensional control tasks.

However, training models via IRL can be computationally expensive and challenging, particularly in complex environments (Wulfmeier et al., 2015). Additionally, IRL requires data from highly skilled experts for training (Arora & Doshi, 2021). This can become a significant bottleneck, as expert demonstrations can be expensive and time-consuming to acquire.

In RL, a technique known as Curriculum Learning (CL) (Bengio et al., 2009), has been shown to accelerate convergence and improve generalization (Narvekar et al., 2020; Wang et al., 2020). CL is a learning strategy, inspired by the human learning process. Its general method is to start learning from easy tasks and progressively adapt to more difficult ones. Given the success of CL in RL, it is worth investigating whether CL is able to benefit IRL, potentially improving training speed or reducing the number of necessary expert demonstrations.

Shen et al. (2022) have applied an IRL algorithm together with CL for the task of driving. In their ablation study, they show that CL brings improvements to the training results. However, they only run a single kind of curriculum - one that makes the model train on the early steps of the episode before being allowed to reach the further steps. Such a curriculum is not suitable for some applications, including for the environment discussed in this paper, as we show in Section 5.4. Also, their CL can only be applied when an RL algorithm is used alongside IRL, as the curriculum's progress depends on the trust-region optimization. Finally, Shen et al. used the IRL technique developed by Abbeel and Ng (2004), which brings such limitations as restricting the reward function to a linear combination of known features.

To address the aforementioned limitations, we propose a novel framework that applies the principles of CL to AIRL, aiming to improve the learning efficiency and performance of artificial agents. We show that a well-constructed learning curriculum allows the agent to learn the underlying reward function more effectively, reducing computational costs and decreasing the needed amount of expert data by supplementing it with demonstrations from less proficient performers.

2 Background

To understand the context and mechanisms behind the work presented in this paper, it is essential to delve into the techniques that the research is built upon. In this section, we aim to provide a brief overview of the methods that underpin our work: Proximal Policy Optimization, Adversarial Inverse Reinforcement Learning, and Curriculum Learning.

2.1 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a reinforcement learning policy optimization method (Schulman et al., 2017). It is characterized by an objective function that restricts the degree to which the policy can change during each update, ensuring the new policy does not deviate too far from the previous one. Such a design reduces the risk of harmful updates that can lead to a substantial drop in performance. Mathematically, Schulman et al. (2017) define the update function as:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

where $\hat{\mathbb{E}}_t$ denotes expectation, $\hat{A}_t$ is an estimator of the advantage function, $r_t(\theta)$ is the ratio of probabilities to take action a_t between the new and the old policy, and ϵ is a hyper-parameter.

2.2 Adversarial Inverse Reinforcement Learning

Adversarial Inverse Reinforcement Learning (Fu et al., 2018) was preferred over other approaches mentioned in Section 1, as it was shown to perform better in complex environments and respond more robustly to environment changes. In contrast, other techniques such as maximum entropy IRL were observed to struggle with acquiring rewards resilient to changes in the environment, as elaborated upon by Fu et al. (2018).

AIRL uses adversarial methods similar to those used by Generative Adversarial Networks (GANs) (Goodfellow et al., 2020). The algorithm initializes two models: a policy optimization agent, and a discriminator, which also acts as a reward model. For policy optimization, this paper uses PPO, as it has been shown to outperform Trust Region Policy Optimization (TRPO) (Schulman et al., 2015; Schulman et al., 2017). The initialized PPO is then used to gather trajectories in the environment.

Subsequently, as described by Fu et al. (2018), the discriminator model utilizes expert demonstrations and the latest agent's trajectories as input. Its objective is to differentiate between the expert demonstrations and the agent's policy rollouts, much like a discriminator in a GAN distinguishes between real and fake samples. Then, the discriminator update is followed by the policy update, and the process of gathering trajectories and updating models is repeated.

Through this iterative process, AIRL strives to train a good policy for the environment. The incorporation of the reward model in the adversarial setup allows for the learning of more sophisticated and resilient reward functions than traditional IRL methods offer.

Notably, the discriminator operates on individual state transitions rather than full trajectories, since the latter formulation can result in estimates with higher variance than the former (Fu et al., 2018). The algorithm's pseudocode is described in Algorithm 1, and Figure 1 schematically presents the information flow.

More formally, the discriminator objective can be stated as

$$D_{\theta,\phi}(s,a,s') = \frac{\exp(f_{\theta,\phi}(s,a,s'))}{\exp(f_{\theta,\phi}(s,a,s')) + \pi(a|s)},$$

where $f_{\theta,\phi}$ is restricted to a reward approximator g_θ and a shaping term h_ϕ as

$$f_{\theta,\phi}(s,a,s') = g_\theta(s,a) + \gamma h_\phi(s') - h_\phi(s),$$

which is also known as the advantage.

As shown by the authors of the original paper, in case the algorithm is trained to optimality, an optimal reward function can be extracted from the discriminator as

$$f^*(\tau) = R^*(\tau) + \text{const},$$

while π recovers the optimal policy.

Additionally, the algorithm does not use actions to estimate rewards, meaning that the reward is only dependent on the state ($g_\theta(s)$). This allows us to extract rewards that are disentangled from the state transitions within the environment.

Algorithm 1 Adversarial Inverse Reinforcement Learning

```
1: Obtain expert trajectories $\tau_{E_i}$
2: Initialize policy $\pi$ and discriminator $D_{\theta,\varphi}$.
3: **for** $step \in \{1,\dots,N\}$ **do**
4:     Collect trajectories $\tau_i = (s_0, a_0, \dots, s_T, a_T)$ by executing $\pi$.
5:     Train $D_{\theta,\varphi}$ via binary logistic regression to classify expert data $\tau_{E_i}$ from samples $\tau_i$.
6:     Update reward $r_{\theta,\varphi}(s,a,s') \leftarrow \log D_{\theta,\varphi}(s,a,s') - \log(1 - D_{\theta,\varphi}(s,a,s'))$
7:     Update $\pi$ with respect to $r_{\theta,\varphi}$ using any policy optimization method.
8: **end for**
```

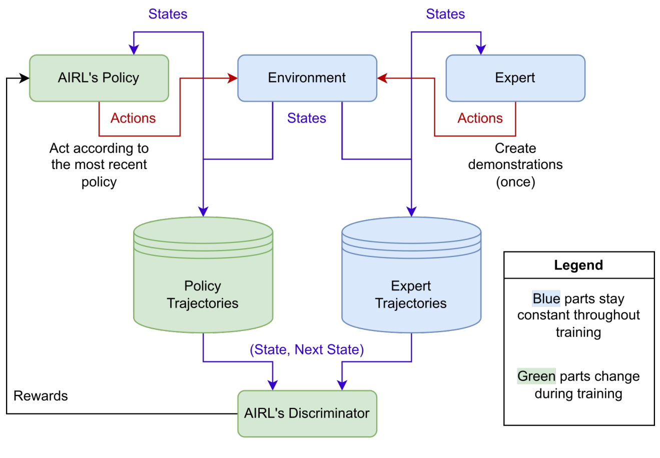


Figure 1: Diagram of AIRL algorithm. Expert trajectories are gathered once, before AIRL starts training. After AIRL's policy receives the reward, it is updated and new policy trajectories are sampled.

2.3 Curriculum learning

In this work, we incorporate the concept of Curriculum Learning (CL) to effectively enhance the learning process in our IRL framework. The core idea behind CL is to start learning from simpler tasks and progressively move to more complex ones, similar to how a curriculum operates in the education system (Bengio et al., 2009). This approach is postulated to accelerate learning and potentially result in better generalization capabilities. The model's training starts with less complex tasks demonstrated by less proficient agents, then the complexity of tasks and/or the skill level of the demonstrators is progressively increased. This approach allows the model to build upon foundational knowledge and gradually adapt to more complex situations (Narvekar et al., 2020). In doing so, we can make efficient use of available data, and possibly reduce the requirement for large volumes of expert demonstrations.

This strategy presents two potential advantages: it helps to prevent the learner from getting stuck in poor local minima (Guo et al., 2018), and it accelerates the learning process (Wang et al., 2020). In our context, a curriculum reduces the complexity of the task or the size of the state space. We discuss how exactly CL is applied in Section 3.3.

3 Methodology

This section outlines the methodology adopted in this study. The primary focus of our research is the application of CL to the AIRL algorithm. Besides this, we will delve into the importance of expert demonstrations and the technical details around the used implementation of the AIRL algorithm.

3.1 Expert Demonstrations

IRL requires demonstrations for training. Thus, before we start running any IRL algorithm, we need a way to obtain a set of trajectories from a policy that we consider to be an expert. Note that the said policy does not need to take exclusively optimal actions.

To gather the demonstrations, we could ask humans to perform actions in our environments. However, we believe that for the test environments discussed in this paper, a human would not provide any more information than an RL policy trained with a defined reward function. On the other hand, human demonstrations would be less controllable and more difficult to acquire (Zakour et al., 2021). Therefore, for the demonstrations, we train a neural network with PPO (Schulman et al., 2017) and then store its trajectories from inference runs. Other approaches for automatically gathering expert demonstrations could include: training with Trust Region Policy Optimization (TRPO) (Schulman et al., 2015), Deep Q-Network (DQN) (Mnih et al., 2013), or Q-learning (Watkins & Dayan, 1992). In our case, the performance reached by a trained PPO is well above that of a random policy, so we use this algorithm for consistency with AIRL.

3.2 Implementation of AIRL

This paper uses an implementation of AIRL that slightly deviates from the original paper. As the starting point, we use the work of Peschl et al. (2021; 2022). Additionally, we introduced a number of modifications that allow us to reach better results and run the algorithm faster.

Most importantly, we corrected reward estimation on the final step of the episode. In the original code, the environment vectorizer was combined with AIRL in an inconsistent way. The discriminator always expected a state pair (obs, obs') as though it derived from transition $(obs, action, obs')$. Where obs is given to the actor policy, which produces $action$ in response, and obs' is the observation after $action$ is performed in the environment.

However, as shown in Algorithm 2, immediately following an episode termination, the produced obs' was in fact the initial observation of a new episode, thus having no relation to obs. As a result, the reward estimation was misleading and hindered training progress. We modify the wrapper such that the returned obs' is not overwritten at the end of an episode, while an observation that should be acted on - from a new episode - can be accessed separately. The resulting improvement is discussed in Section 5.1.

Algorithm 2 Original Vector Wrapper Step Procedure

```
 1: procedure STEP(action_list)
 2:     Initialize result lists
 3:     for env in self.envs do
 4:         obs, rew, done, info ← env.step(action)
 5:         if done then
 6:             obs ← env.reset()
 7:         end if
 8:         Append obs, rew, done, info to the result lists
 9:     end for
10:     return result lists
11: end procedure
```

Observation returned from the environment step function is lost if it ends the episode.

To speed up the training process, we implemented a series of minor modifications. As part of this, we eliminated unnecessary tensor copying and batched some Neural Network (NN) evaluations. Compared to feeding states through the NN individually, batch computations allow us to leverage the hardware accelerator's capabilities and reduce the total computation time (Shawahna et al., 2019). Specific details of these minor modifications can be found in our repository[1]. Finally, we modified the NN architecture, reducing the number of layers and neurons in a layer, as the dimensions used by Peschl et al. (2022) were deemed excessively large for our task. For our experiments, we primarily use PPO with 3 hidden layers, 128 neurons each with Rectified Linear Unit (ReLU) in between, and Softmax for actor output, as the action space is discrete. The discriminator also has 3 hidden layers but with 256 neurons each and ReLU between them. The detailed visualizations of both architectures are presented in Appendix D.

3.3 Curriculum Learning for AIRL

The main contribution of this paper lies in investigating whether applying CL to the AIRL algorithm is able to accelerate training. This section elaborates on the utilization of CL, and the components within a curriculum, while Section 5 covers how it performs compared to the baseline - training entirely in the evaluation environment.

A curriculum defines which environment configurations will be used for the training and for how many steps each of those is used. For example, a curriculum that was used most

[1]GitHub repository: https://github.com/mikhail-vlasenko/curriculum-learning-IRL

in this research dictates starting the training with a scaled-down environment and subsequently transitioning to its full-sized equivalent. Mathematically, we define a curriculum as $Cu(n, [Conf_1, ..., Conf_n], [Step_1, ..., Step_n])$, where n is the number of curriculum stages, $Conf_i$ is an entity that defines an environment configuration and expert demonstrations, while $Step_i$ specifies the duration in environment steps that $Conf_i$ is used to train the model.

Once a curriculum Cu is defined, randomly initialized AIRL policy, π, and discriminator, D, begin training in $Conf_1$ environment. After completing $Step_1$ steps, π and D are retained, while the optimizers, policy trajectory dataset, and the current environment state are discarded. A new environment is then initialized using $Conf_2$, and the previously saved π and D undergo further training. This process is continued for all remaining pairs $(Conf_i, Step_i)$ in Cu. Note that, in this work, $Conf_n$ corresponds to the evaluation environment configuration for all curricula.

This procedure allows AIRL's neural networks to be initially trained on a relatively straightforward task before transitioning to a more complex setting, enabling them to dedicate fewer steps to the initial exploration in the complex setting.

One of the main challenges in applying curriculum learning is the design of an effective curriculum. This requires defining what constitutes a simple task and a complex task in the context of the problem domain and determining a suitable progression from simple to complex. In our work, we address this challenge through a combination of domain expertise and empirical evaluation. However, as shown in Section 5, some of the tested curricula in this research failed to enhance the training.

4 Environment Design

In the scope of this paper, two grid world-type environments were implemented for evaluation. First, we will describe a very simple environment that allows for an effective curriculum. Then, we present a more complex, partially observable, environment to showcase the benefits of curriculum learning in a more difficult setting. In both environments, the actor is shown as a blue circle that is able to move in one out of four directions (up, down, left, right) per step. When the actor tries to move out of the field, it stays in place, but the step is used. The target is represented as a light blue tile.

4.1 Simple Environment

The environment was designed to be easy, yet, it is randomized. It initializes the actor and the target on two random non-coinciding tiles of a grid. As supported by the findings of Zhang et al. (2018), environment randomization prevents overfitting.

In this environment, the agent observes the relative position of the target with respect to the actor, $p_{a,t} = p_t - p_a$, where $p_t \in \mathbb{R}^2$ and $p_a \in \mathbb{R}^2$ are grid positions of target and actor respectively. The episode ends when the actor reaches the goal, or when it runs out of turns. The reward is given only on the final step and is 1 if the goal is reached and -1 otherwise. The maximum number of steps, max_steps, is deliberately chosen to be large, calculated as $max\_steps = 2 \cdot size^2$, where $size$ is defined as the length of the square's side in tiles. An instance of such an environment with $size = 5$ is presented in Figure 2.

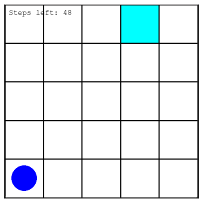


Figure 2: Example of a simple environment

4.2 Tile Reward Environment

The second presented environment, *tile reward*, builds on the previous one. Here, the number of turns is significantly limited, defined as $3 \cdot size$, and a reward between -1 and 1 is given for stepping on every unvisited tile. The episode ends when the agent steps on the goal or runs out of turns. In the latter case, the agent gets the reward of $-0.5 \cdot max\_steps$ for the final step, to encourage reaching the target. Notably, the given number of steps per episode is always enough to reach the target. Figure 3 shows two instances of the *tile reward* environment.

Similar to the simple version, the initial positions of the actor and the target are random. Additionally, the reward on every tile is randomized. The values are sampled from an independent uniform distribution: $reward_{i,j} \sim \mathrm{U}(-1, 1)$.

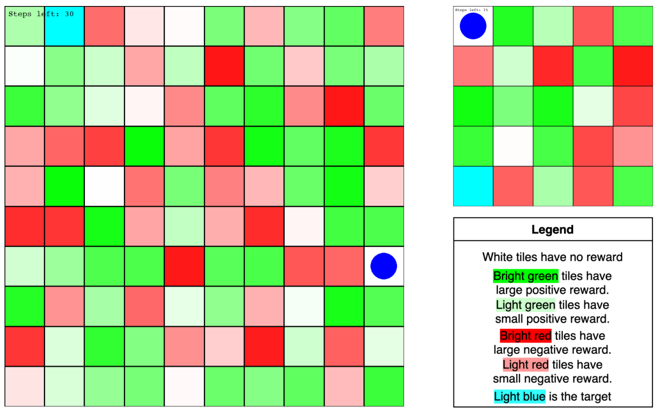

Steps left: 30
Legend
White tiles have no reward
Bright green tiles have large positive reward.
Light green tiles have small positive reward.
Bright red tiles have large negative reward.
Light red tiles have small negative reward.
Light blue is the target

Figure 3: An example starting state in the *tile reward* environment with $size = 5$ and $size = 10$.

Observation Space

For the *tile reward* environment, the agent's observation is significantly augmented. As visualized in Figure 4, it includes rewards of the 5 by 5 part of the field around the actor, as well as information on whether those tiles are walkable (tiles out of bounds, as well as the target, have reward 0). In addition to that, the agent also receives its absolute position, the absolute position of the target, and the number of turns left until the deadline. The positional observations are important since the environment is only partially observable, so the agent needs to know which direction it needs to go when the target is far away. The number of turns left is also very valuable, as it allows the agent to make a decision whether to rush directly to the target or to spend turns gathering positive rewards.

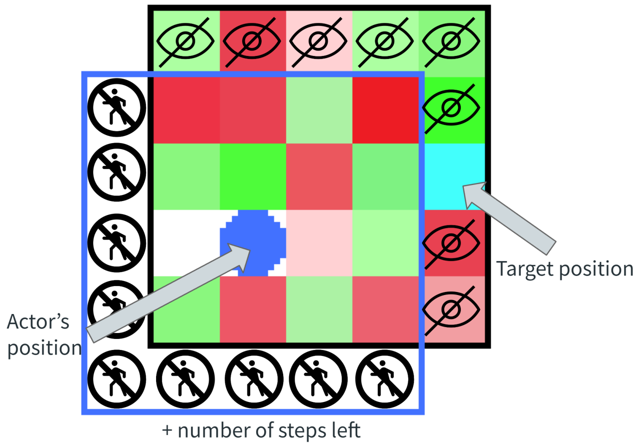


Figure 4: Observation space in the *tile reward* environment

Special Reward Patterns

Figure 5 shows two possible reward configurations that can be used in a curriculum. On the left, it shows an example of *positive stripe* reward. In such a configuration, all positive rewards are guaranteed to be on a stripe of width 2 (vertical or horizontal), while all other rewards are negative. On the right, negative rewards are arranged in a checkerboard-like pattern, such that they can always be avoided.

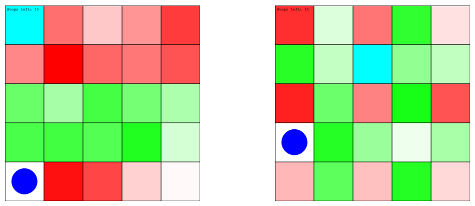


Figure 5: *Positive stripe* and *checkers* reward patterns in the *tile reward* environment.

4.3 Used Configurations

From the described environments, this study primarily uses two main configurations. Using terminology from Section 3.3, $Conf_{eval}$ is the *tile reward* environment with default reward pattern and $size = 10, max\_steps = 30$. $Conf_{small}$ is the same as $Conf_{eval}$, except with $size = 5, max\_steps = 15$. Unless otherwise stated, $Conf_i$ includes 50000 steps of expert demonstrations that are sampled with a converged PPO policy in the corresponding environment.

The results are only presented for the *tile reward* environment because we believe that the simple environment does not provide significantly interesting findings.

5 Results

The following procedure is used for evaluation: multiple AIRL policies are trained for an equal number of environment steps. Among these, one referred to as the *baseline*, is entirely trained in the evaluation environment. Other policies use different configurations, according to their curricula. Every 1024 environment steps, the performance of each policy's actor is assessed in the evaluation environment. The average reward earned per episode, as determined by the true reward function, is recorded for these resultant trajectories. Later, we refer to this as (non-discounted) true returns - $\bar{R}_{true}$ in Equation 1.

$$\bar{R}_{true} = \frac{1}{N_{episodes}} \sum_{i=1}^{N_{episodes}} \sum_{t=0}^{T_{eval}} r_{i,t} \tag{1}$$

We assess the effectiveness and the learning speed of a policy by considering the $\bar{R}_{true}$ it achieves at different stages of training. It is crucial to note that, by focusing on the true returns, we are not just assessing the proficiency of the actor policy, but also the reward model. This is because the AIRL's actor relies solely on the rewards generated by the reward model.

Training of AIRL is impacted by a multitude of random factors: random initialization of NNs, randomness in the environment, stochastic nature of the policy, etc. To reduce the influence of the random events, all described experiments are repeated at least three times, and standard deviations are presented.

5.1 Modification to the AIRL implementation

First, we present the impact of the main modification that was introduced to the implementation of AIRL that was used in this paper, in line with Section 3.2. The returns achieved by the changed and unchanged versions are shown in Figure 6. Apart from the increased returns, decreased noise is also observed. The expert that is used to sample the demonstrations for training reaches $\bar{R}_{true} = 3.9$, so an improvement from -1.5 to 3 is substantial. Therefore, for the following experiments, we use the corrected version of the implementation.

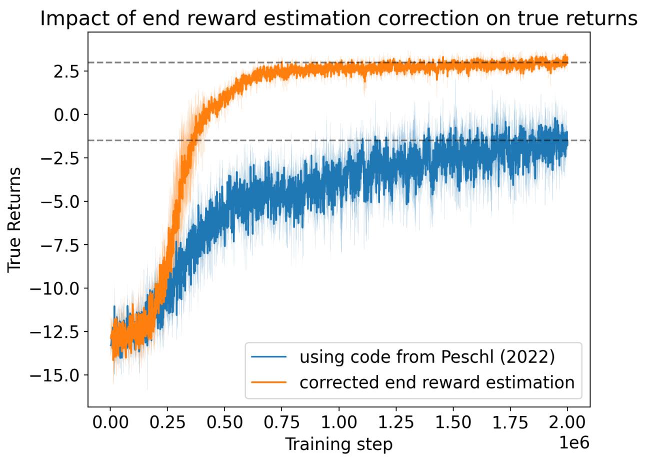


Figure 6: Graph of the true returns as the model trains. 68% confidence intervals (one σ from the mean) are shown with the half-transparent fill.

5.2 Increasing Environment Size Curriculum

The primary curriculum discussed in this paper can be stated as:

$$Cu(2, [Conf_{small}, Conf_{eval}], [x, 10^6 - x]), \quad (2)$$

using notation from Section 3.3, $Conf_{small}, Conf_{eval}$ defined in Section 4.3 and an integer x. For the total number of steps, we chose 10^6 because, with our parameters, most of the models converge by that point.

Figure 7 shows the true returns obtained by the AIRL's policy in the evaluation environment throughout training. Runs that make use of CL reach $\bar{R}_{true} = -3$ faster, but then plateau, as they are trained in an environment that is different from the evaluation one. In this experiment, $x = 2 \cdot 10^5$, so after $2 \cdot 10^5$ steps, the CL run's configuration is changed to the evaluation environment, and training is continued for the remaining $8 \cdot 10^5$ steps.

In this scenario, utilizing CL proves beneficial, as its true average return $\bar{R}_{true}$ is higher or equal to that of the baseline throughout training. Moreover, the CL method reaches near-convergence returns about 2 times faster than the baseline. We believe that the success of this curriculum stems from its effective simplification of the task. In $Conf_{small}$ the episodes are shorter, the target is on average closer, and the state space is reduced. Thus, a random policy (like the one initialized) has a better chance to perform well in the environment. Moreover, the knowledge gained in the smaller environment can be applied in the larger one, $Conf_{eval}$, as it is very similar.

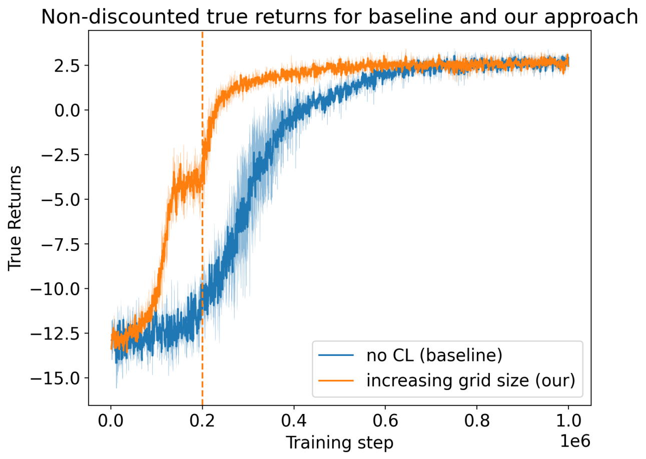


Figure 7: Graph of the true returns as models train. For curriculum, the configuration is switched from $Conf_{small}$ to $Conf_{eval}$ at $2 \cdot 10^5$ steps (the vertical line).

We also explore the influence of changing the duration of the first curriculum stage - x from Equation 2. From Figure 8, we conclude that $x = 100000$ is not enough to harvest the full benefit of CL, while $x = 300000$ is likely unnecessarily long.

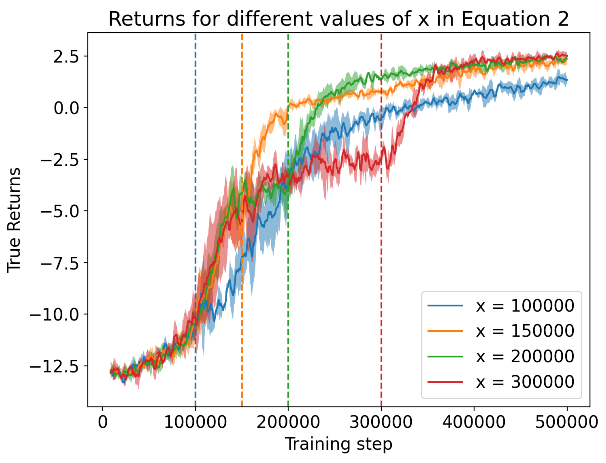


Figure 8: Graph of the true returns as models train. Dotted vertical lines of the corresponding color show when the configuration is switched. Running average smoothing is applied to improve graph readability.

5.3 Data Efficiency

Another important aspect where curriculum learning can help is reaching similar performance while using less data. As discussed before, expert data can be costly to obtain, so minimizing its amount is important in IRL applications. This section tests whether CL can help AIRL to recover better rewards when the number of demonstrations from an expert is severely limited.

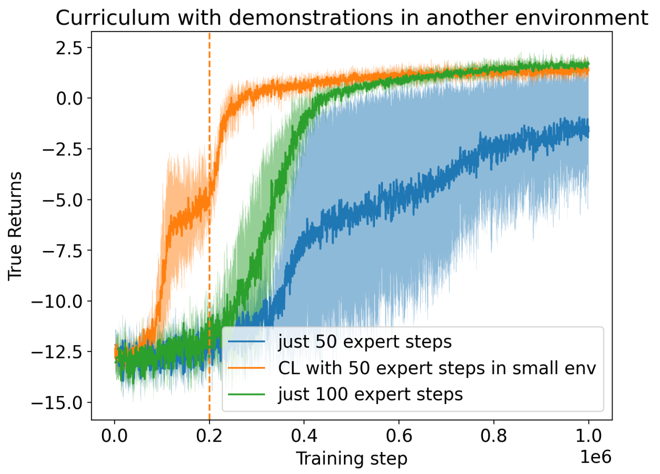
For experiments, we provide only 50 expert demonstration steps in the evaluation environment. This number was chosen empirically as an amount with which AIRL's policy rewards become significantly lower than those in the default configuration (where 50000 expert steps are provided). The small number of demonstrations introduces another source of randomness, as the dataset becomes less representative. For this reason, we create 5 sets of 50 demonstrations and run experiments 5 times - once for each set. We thus encounter a larger variance, but less bias.

Demonstrations from Another Environment

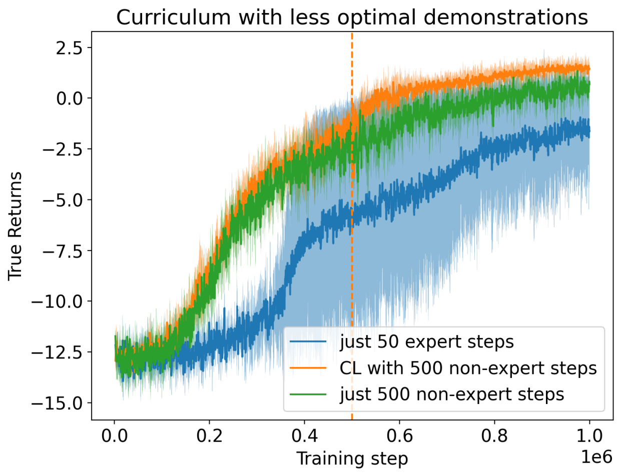
One way to improve achieved returns with limited demonstrations is to use an auxiliary set of demonstrations from another environment. For this experiment, the curriculum starts with $Conf_{small50}$, which is identical to $Conf_{small}$, except only 50 expert steps are provided. As presented in Figure 9a, the mean value achieved by CL (in orange) is significantly higher than that of the baseline (in blue). We also note that despite the large variance of the baseline, none of the baseline runs reached returns higher than CL's average. We thus conclude that our method substantially improves the outcome. Furthermore, the performance is on par with the results achieved when using 100 expert demonstrations in the evaluation environment - double the number used for CL.

Non-Expert Demonstrations

Another approach makes use of demonstrations that are sampled from a less optimal policy. Previously, all data came



(a) CL includes simplified environment



(b) CL includes worse expert

Figure 9: Non-discounted true returns as the models train. Five runs are grouped for each line. 68% confidence intervals (one σ from the mean) are shown with the half-transparent fill.

from a converged policy, reaching $\bar{R}_{true} = 3.9$ in the evaluation environment. Here, we gather 500 demonstration steps that achieve $\bar{R}_{true} = -0.35$, and use them for the first stage of the curriculum. Figure 9b shows that AIRL is on average able to get higher true returns with CL. However, the results are less conclusive in this case due to the high variance of the baseline, and relatively good performance achieved with exclusively non-expert demonstrations.

5.4 Other Curricula

Besides the discussed *increasing grid size* curriculum, other curricula were tested for the *tile reward* environment. However, they have not provided significant improvements compared to the baseline method, so we only discuss them briefly.

The sequential curriculum, as described by Shen et al. (2022), was adapted to the used evaluation environment in the following way: $Conf_1$ is the same as the evaluation configuration, $Conf_{eval}$, but the max_steps is very low. Then, subsequent $Conf_i$ gradually increase the max_steps up to 30, which is used for the evaluation environment. The performance of such a curriculum shows to be worse than the baseline, taking more environment steps to reach the same true returns, as presented in Appendix A.

We have speculated that a custom reward pattern like *positive stripe* or *checkers* could be useful for a curriculum because it is easier to reach good rewards in these configurations. However, the results show that such a curriculum does not significantly contribute to performance in the default version of the environment. We present the graphs for these curricula in Appendix B.

6 Discussion

Within this study, we focused on a comparison of our proposed approach, CL for AIRL, solely with the standard AIRL. While there are indeed multiple other IRL algorithms that could be relevant to the problems discussed, we excluded them from our experiments. As shown by Fu et al. (2018), AIRL significantly outperforms other IRL techniques, we thus believe the additional comparison is unnecessary. Moreover, some IRL algorithms are model-based, which severely complicates applying them to our environment.

Given that in CL the trained models must be applicable across all stages of the curriculum, the dimensions of the input and output need to be consistent across all environments. This imposes a limitation on the possible range of curriculum environments for a given evaluation configuration.

We understand that creating an efficient curriculum may be time-consuming, which is why such an approach may not be universally applicable. Still, the demonstrated improvements can be crucial for some applications, making CL an important component to consider.

We also acknowledge that we did not exhaustively explore certain parameters, such as NN architectures, the configuration of the evaluation environment, some hyperparameters of AIRL and PPO, and the default number of expert demonstrations. Consequently, we cannot completely rule out the potential for different outcomes with alternative parameter configurations.

7 Conclusion and Future Work

In this study, we investigate the application of curriculum learning in the context of the Adversarial Inverse Reinforcement Learning algorithm. We find that well-constructed curricula significantly reduce the number of environment steps that are necessary to learn a certain policy. Moreover, when expert demonstrations are scarce, CL is able to augment training in such a way that better true returns are reached, and thus, the reward estimation is improved. Nevertheless, if the crafted curriculum inadequately reflects the final problem, it may fail to enhance or even potentially impede the training process.

In summary, this research provides evidence that integrating CL strategies into AIRL can substantially improve performance in two main aspects: accelerating convergence speed and improving the effectiveness of the learned reward function. We hope that our work sparks further investigation into the broader application of CL in the field of IRL and helps to guide the responsible development of such methods.

In future work, we aim to investigate the feasibility of dynamically selecting the transition point to the next curriculum stage. For instance, by detecting the convergence of the model on the current configuration, we could identify opportune moments to transition to the subsequent stage. We also plan to experiment with varying learning rates for different stages or learning rate schedulers in general. Finally, more stages can be introduced to the existing curricula. It is likely that a more gradual transition between the complexity of configurations will positively impact the speed of convergence, so it may be valuable to investigate it.

8 Responsible Research

Despite the subtitle of the paper mentioning human data, no data from humans was used in the research. The expert demonstrations were gathered exclusively using RL models, and the training environments did not ask for any human feedback.

8.1 Ethical Considerations

In conducting this exploration, we have committed to maintaining the essential principles of ethical research within the machine learning landscape. The implementation of curriculum learning strategies within the realm of IRL, while promising, does carry the potential for misuse or unintended consequences.

For instance, biased or ethically compromised curricula could result in harmful outcomes when applied in real-world scenarios through IRL algorithms. We hope that our research's potential applications are oriented toward ethically acceptable and beneficial goals. One such application could be more human-like navigation capabilities for artificial agents.

8.2 Reproducibility

In the spirit of promoting open science, we made a concerted effort to ensure the reproducibility of our research. A detailed and transparent account of the methodologies applied, including the development of curriculum learning strategies, the setup of IRL methods, and our experimentation process, are all provided.

In addition, we make a commitment to the FAIR (Findable, Accessible, Interoperable, and Reusable) principles, by making all data and code used within the study publicly available. This not only allows our contribution to be used by fellow researchers for further studies but also promotes a collaborative scientific community.

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. *Association for Computing Machinery*. https://doi.org/10.1145/1015330.1015430

Arora, S., & Doshi, P. (2021). A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, *297*. https://www.sciencedirect.com/science/article/pii/S0004370221000515

Bengio, Y., et al. (2009). Curriculum learning. *Association for Computing Machinery*. https://doi.org/10.1145/1553374.1553380

Coates, A., Abbeel, P., & Ng, A. Y. (2008). Learning for control from multiple demonstrations. *ICML*. http://heli.stanford.edu

Fu, J., Luo, K., & Levine, S. (2018). Learning robust rewards with adversarial inverse reinforcement learning. *International Conference on Learning Representations*. https://openreview.net/forum?id=rkHywl-A-

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*, 139–144. https://doi.org/10.1145/3422622

Guo, S., Huang, W., Zhang, H., Zhuang, C., Dong, D., Scott, M. R., & Huang, D. (2018). Curriculumnet: Weakly supervised learning from large-scale web images. *Proceedings of the European conference on computer vision (ECCV)*. https://github.com/guoshengcv/CurriculumNet.

Lecun, Y., Bottou, E., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

Levine, S., Popovic, Z., & Koltun, V. (2011). Nonlinear inverse reinforcement learning with gaussian processes. *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper_files/paper/2011/file/c51ce410c124a10e0db5e4b97fc2af39-Paper.pdf

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. http://arxiv.org/abs/1312.5602

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., & Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, *21*, 1–50.

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *ICML*.

Ng, A. Y., Russell, S., et al. (2000). Algorithms for inverse reinforcement learning. *International Conference on Machine Learning*, *1*, 663–670.

Peschl, M. (2021). *Aligning ai with human norms multi-objective deep reinforce-ment learning with active* *prefer-ence elicitation thesis report msc applied mathematics*. Delft University of Technology. http://repository.tudelft.nl/.

Peschl, M., Zgonnikov, A., Oliehoek, F. A., & tudelftnl C Luciano Siebert. (2022). *Moral: Aligning ai with human norms through multi-objective reinforced active learning; moral: Aligning ai with human norms through multi-objective reinforced active learning*. https://github.com/mlpeschl/moral_rl.

Schulman, J., Levine, S., Moritz, P., Jordan, M., & Abbeel, P. (2015). Trust region policy optimization. *Proceedings of the 32nd International Conference on Machine Learning*.

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

Shawahna, A., Sait, S. M., & El-Maleh, A. (2019). *Fpga-based accelerators of deep learning networks for learning and classification: A review*. https://doi.org/10.1109/ACCESS.2018.2890150

Shen, Y., Li, W., & Lin, M. C. (2022). Inverse reinforcement learning with hybrid-weight trust-region optimization and curriculum learning for autonomous maneuvering, 7421–7428. https://doi.org/10.1109/iros47612.2022.9981103

Wang, X., Chen, Y., & Zhu, W. (2020). A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. http://arxiv.org/abs/2010.13166

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

Wulfmeier, M., Ondruska, P., & Posner, I. (2015). Maximum entropy deep inverse reinforcement learning. *CoRR*, *abs/1507.04888*. http://arxiv.org/abs/1507.04888

Zakour, M., Mellouli, A., & Chaudhari, R. (2021). Hoisim: Synthesizing realistic 3d human-object interaction data for human activity recognition. *2021 30th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 1124–1131. https://doi.org/10.1109/RO-MAN50785.2021.9515349

Zhang, C., Vinyals, O., Munos, R., & Bengio, S. (2018). A study on overfitting in deep reinforcement learning. *arXiv*. http://arxiv.org/abs/1804.06893

Ziebart, B. D. (2010). *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University. http://reports-archive.adm.cs.cmu.edu/anon/anon/home/ftp/usr/ftp/ml2010/CMU-ML-10-110.pdf

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. (2008). Maximum entropy inverse reinforcement learning. *Aaai*. http://repository.cmu.edu/robotics

A Sequential Curriculum

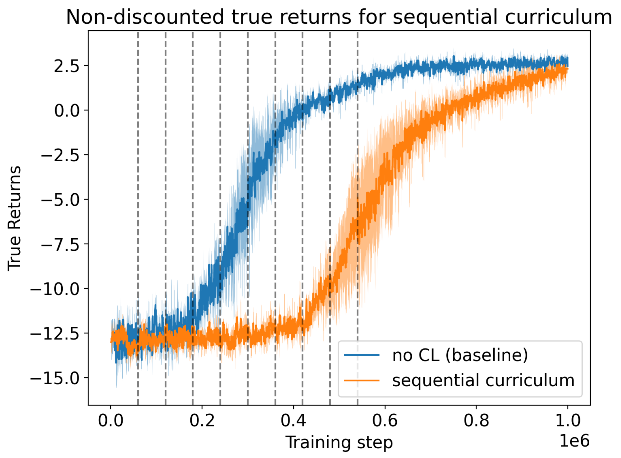


Figure 10: Graph of the true returns as models train. $max\_steps = [2, 4, 6, 8, 10, 12, 15, 20, 25, 30]$.

B Other Curricula

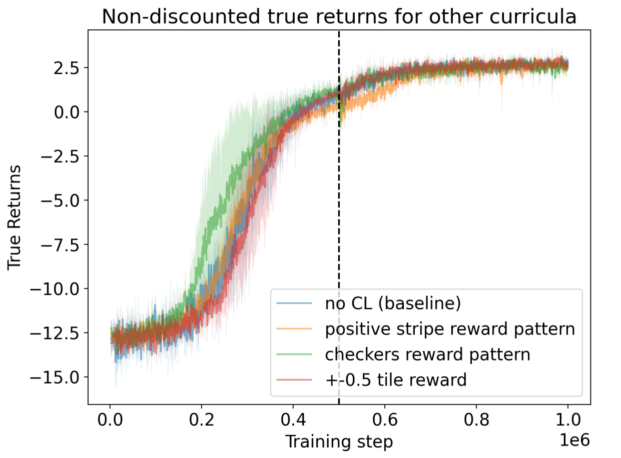


Figure 11: Graph of the true returns as models train.

C Used Compute Power

For all experiments, we use a machine with an NVIDIA GeForce RTX 4090, AMD Ryzen 7 7700X, and 32GB of RAM. We run 64 environments in parallel to speed up training. One train run (usually 10^6 environment steps) takes approximately 20 minutes. We believe that the bottleneck comes primarily from the update of PPO, as its batches are single trajectories from episodes, meaning it processes at most 30 steps at once.

D Neural Network Architecture Diagrams

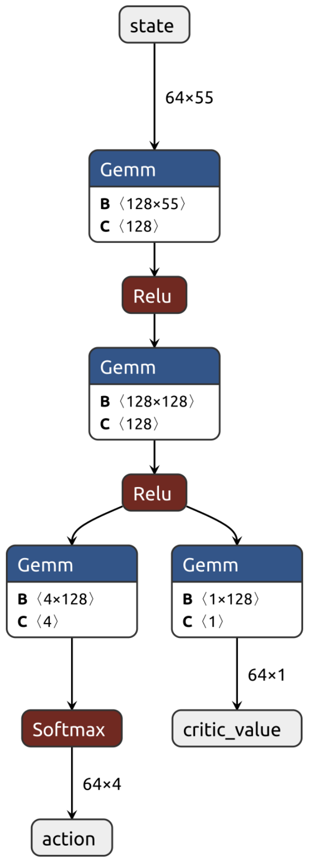


Figure 12: Network architecture of the PPO policy. On the diagram, 64 is the batch size (and thus can be substituted for another integer). 55 is the observation dimension of the *tile reward* environment.

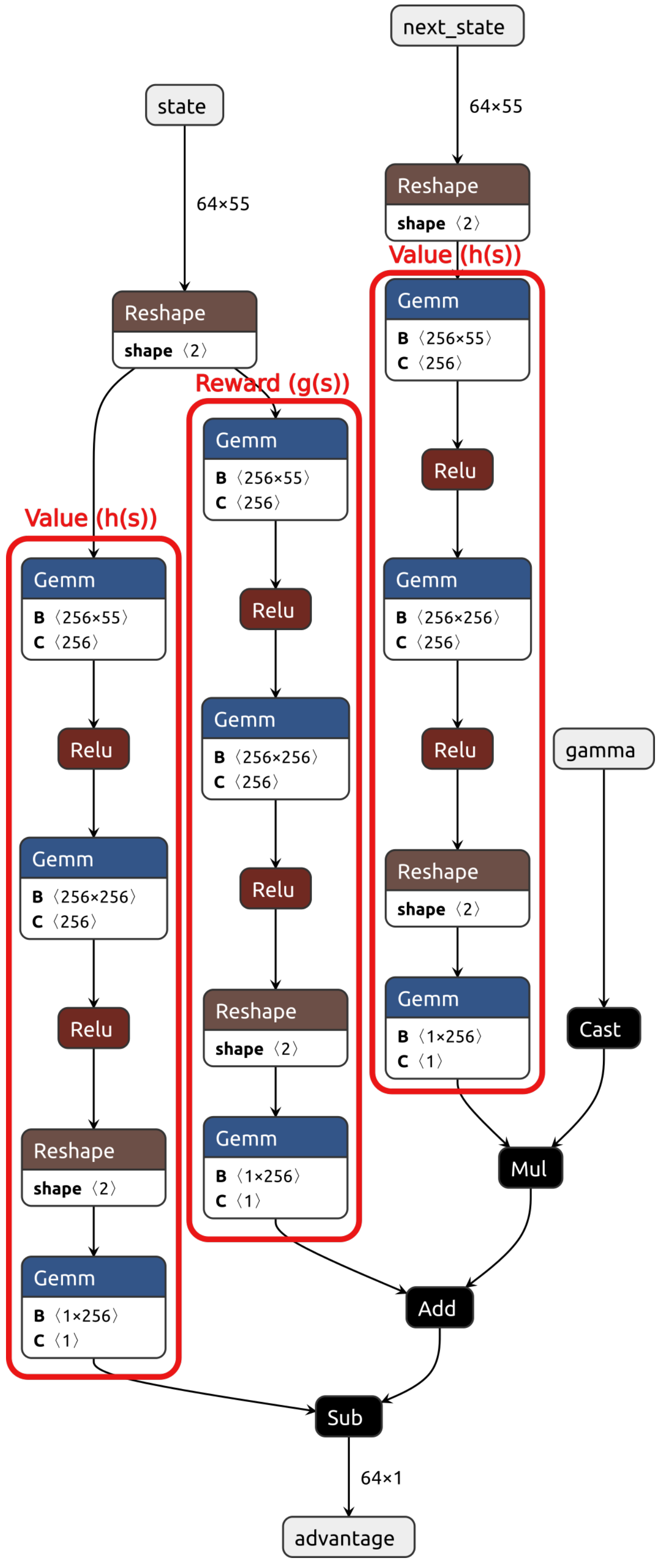


Figure 13: Network architecture of the AIRL discriminator. On the diagram, 64 is the batch size (and thus can be substituted for another integer). 55 is the observation dimension of the *tile reward* environment.