universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Imitation Learning from Multiple Perspectives - A Multi-Discriminator Framework“

verfasst von / submitted by

Susanna Maria Weinberger, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 645
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Datascience
Betreut von / Supervisor:	Ass.-Prof. Dipl.-Ing. Dr.techn. Sebastian Tschiatschek, BSc

Acknowledgements

This thesis posed a hard challenge on me, therefore I am really happy about the results I achieved in the end and grateful for the tremendous support that helped me achieving it. First and foremost I want to thank my supervisor Dr. techn. Sebastian Tschiatschek who helped me developing the idea and pushed me to grow beyond myself. Our discussions and your suggestions gave me new perspectives and supported my work immensely.

I would like to thank my family who always believed in me and supported my. Finally, thanks to my partner for providing great ideas and brainstorming with me. More importantly thanks for being on my side especially when something did not work out immediately and trying to motivate me to do my very best.

Abstract

This thesis focuses on the problem of imitation learning from observations based on expert's demonstrations from multiple perspectives provided as images. On the one hand, introducing multiple perspectives increases the amount of available information in comparison to using only one fixed perspective. On the other hand, training an agent on multiple perspectives offers the possibility to steer the focus of the learning agent to perspectives which support it best in its current learning progress by showing the agent a curated selection of perspectives. Such an approach could prove for example helpful in robotics when learning through video sequences of different perspectives from a human demonstrator.

To this end, we provide an overview of relevant related work in the field of imitation learning with special focus on learning purely from observations and formalise the imitation learning problem from multiple perspectives. We propose an extended imitation learning framework based on GAIL - Generative Adversarial Imitation Learning [HE16] - that leverages the provided information from multiple perspectives utilising different strategies for perspective selection.

We evaluate the proposed framework in extensive experiments and show empirically its ability to learn with our proposed strategies. Furthermore, we assess the impact of various strategies on imitation performance, show the capability of the framework in the face of a restricted access to expert demonstrations and demonstrate that our proposed approach outperforms relevant baselines.

Kurzfassung

Diese Arbeit widmet sich der Lösung des Problem des imitierenden Lernens mit Hilfe der Nachahmung eines Experten, wobei mehrere Perspektiven dieses Experten in Form von Videosequenzen bereitgestellt werden. Die Miteinbeziehung von mehreren Perspektiven bietet nicht nur den Vorteil eines größeren Informationsgehalts, der mit einer Perspektive nicht abgedeckt werden kann, sondern ermöglicht es auch den Fokus des lernenden Algorithmus auf die zu diesem Zeitpunkt lehrreichste Perspektive zu lenken. Dieser Ansatz kann beispielsweise in der Robotik verwendet werden, um einen Menschen im Bezug auf verschiedene Perspektiven effizient nachzuahmen.

Zu diesem Zweck geben wir einen Überblick über die derzeitige Forschung im Bereich des imitierenden Lernen und formalisieren das Problem des imitierenden Lernens aus mehreren Perspektiven. Wir stellen ein erweitertes GAIL [HE16] - Generative Adversarial Imitation Learning - Framework vor, das durch Verwendung von verschiedenen Perspektiv-Auswahl Strategien versucht den Lernprozess eines Agenten bestmöglich zu unterstützen.

Wir evaluieren unser Framework in umfangreichen Experimenten und illustrieren empirisch seine Funktionalität und Lernfähigkeit. Außerdem diskutieren wir den Effekt der verschiedenen Strategien auf die Performance, beweisen die weiterhin gute Performance bei eingeschränkter Verfügbarkeit von Expertendaten und demonstrieren eine Verbesserung der Performance durch unseren Ansatz im Vergleich zu relevanten Baselines.

Contents

List of Tables

List of Figures

List of Algorithms

1. Introduction

While humanity is struck by the abilities of artificial intelligence (AI), many ideas used in machine learning have been inspired by nature. This includes optimisation algorithms [Yan20], neural networks [WF18] and reinforcement learning [SB18]. A dog learns to bring back a stick by receiving a positive reward - a treat. A child learns not to touch the hot stove by receiving a negative reward - it hurts. The idea of learning a task by reinforcing behavior is conceptualised in reinforcement learning. Through providing a (numerical) reward but without exactly telling a machine what the goal is, the machine tries to maximize its performance. In the last years, this approach has for instance resulted in AI agents which achieve superhuman performance for board games like GO or Stratego [PDVH$^+$22, SHM$^+$16].

One challenge which can arise in reinforcement learning is the definition of the reward function such that the agent learns the wanted behaviour. While for games like Tic Tac Toe it is straightforward that the reward can be defined according to the game's outcome, it gets notably more complex when considering other problems like self-driving cars or robots [HGEJ17]. To address this challenge, the idea of imitation learning was developed. Instead of maximizing a human-defined reward, the goal of a novice - an agent, acting often randomly in the beginning - is to imitate expert behaviour given as some kind of observations, for example images, as good as possible [SB18].

Different techniques have been developed in the context of imitation learning: Behavioral Cloning deals with the problem in a supervised manner receiving state-action pairs from an expert and trying to replicate the expert's actions [TWS18a]. Inverse Reinforcement learning (IRL) on the other hand tries to solve the problem by inferring the reward that could possibly drive the expert policy. By using this inferred reward, the novice can then learn through common reinforcement learning methods [AD21].

While in the basic imitation learning setting, it is usually assumed to have full access to all expert states and all actions carried out by an expert, this assumption poses strict restrictions on the usage of already available data for expert demonstrations. This issue is addressed in Learning from Observations (LfO) [TWS19b]. In LfO, instead of relying on state-action pairs, only state information is used, often presented as raw images [TWS19b].

Current research showed already many successful approaches for different aspects of imitation learning, like the problem of differing perspectives between expert and novice [SAS17, LGAL18] also known as third person imitation learning. However, the idea of using multiple expert perspectives simultaneously and exploiting this gain of information is not present in research till now to the best of our knowledge. We can find motivation for this idea in the human learning process: Humans observe experts/other humans carrying out a task from different viewpoints to collect as much information as possible. Looking for instance at a basketball player, more knowledge about playing basketball can be gained by observing the player from the side than by watching him/her from behind where his/her hands are not visible.

We aim to close this gap by taking advantage of multiple perspectives with our proposed active imitation learning framework. Inspired by previous work we introduce an approach based on Generative Adversarial Imitation Learning (GAIL) [HE16], which itself is derived from General Adversarial Networks (GAN) [GPAM$^+$14] - adapted to a multi-perspective setting. We assume a finite number of perspectives and match always one discriminator with one perspective. This provides us with multiple discriminators, each of which is assessing the performance of the novice for one perspective respectively. The novice is trained based on rewards with a perspective selection strategy defining which discriminator output is used as reward in the current step. Ultimately, we thrive to train a novice which imitates the expert so well that it is no longer possible to differentiate expert and novice anymore in any perspective in terms of how both carry out a task.

Our contributions are as follows:

- We study different approaches to imitation learning and give an overview over the current state of the art.
- We propose an active learning GAIL-based framework for learning in the above sketched setting based on using multiple discriminators and a set of different perspective selection strategies.
- We evaluate the proposed framework empirically on different environments with a focus on strategy performance. Furthermore, to determine the models ability to deal with a realistic restricted amount of expert data, we investigate the influence of the provided amount of expert demonstrations. We also compare our multi-discriminator framework to the idea of aggregating all available information form different perspectives into one discriminator.

A paper referring to the general concept and including parts of the thesis is currently

under review for a conference and may be published in the future.

2. Fundamentals

In this chapter we provide a short overview on the fundamentals which are relevant for understanding this thesis with a focus on the basics of reinforcement learning and the idea of the GAIL approach [HE16].

2.1. Reinforcement Learning

The main idea of reinforcement learning is to find good behaviour - good in this case being defined by the maximisation of a numerical reward signal through trial & error. Although in general reinforcement learning problems can have various forms, we look at a standard scenario with discrete time steps and finite episodes [SB18]. Concretely we consider an agent that interacts in each discrete time step $t = 0, 1, 2, ...$ with an environment by performing actions $a_t \in \mathcal{A}$ where $\mathcal{A}$ is the set of possible actions, also commonly referred to as action space. After each action, the agent is provided with the actual state of the environment $s_{t+1} \in \mathcal{S}$, where $\mathcal{S}$ is the set of possible states, also commonly referred to as state space, and a numerical reward signal r_{t+1}, see Figure 2.1. Looking at this process from the outside, one can observe a trajectory [SB18] - for one complete trajectory with the final time step T we use the term episode:

$$s_0, a_0, r_1, s_1, a_1, r_2, ..., r_T, s_T$$

States s_t and actions a_t are often represented by vectors, but especially states can occur in various forms, one being images (= multi-dimensional vectors). Whilst there is no clear notation provided in literature, we will use the term and notation observation o_t in the case of explicitly referring to images and and use state s_t as a general notation for a state.

2.1.1. Markov Decision Process

While decision problems can also be considered in a broad general context, reinforcement learning problems are often limited to decision processes that fulfill the Markov property and are then referred to as Markov Decision Processes. In the context of reinforcement

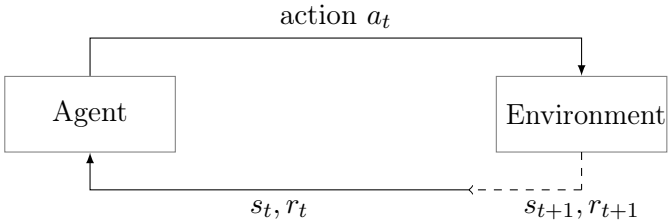


Figure 2.1.: Agent interacting with an environment.

learning, the Markov property reduces the number of states we have to take a look at in one step. Concretely speaking, the immediately preceding state must include all information of past interactions that could be relevant for the future. This means for the function

$$\mathcal{P} : \mathcal{S}^t \to [0,1], \mathcal{P}(s_t|s_{t-1}, ..., s_2, s_1) = Prob\{S_t = s_t|S_{t-1} = s_{t-1}, ..., S_2 = s_2, S_1 = s_1\}$$

describing the probability of the next state based on already observed states it must hold

$$\mathcal{P}(s_t|s_{t-1}) = \mathcal{P}(s_t|s_{t-1}, ..., s_2, s_1)^1.$$

Formally we represent the general Markov decision process as a tuple $M = (\mathcal{S}, \mathcal{A}, p, H, r)$ that fulfills the just described Markov Property where

- $\mathcal{S}$ represents the state space,
- $\mathcal{A}$ represents the action space,
- $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,1]$, $p(s'|s,a) = Prob\{S_t = s'|S_{t-1} = s, A_{t-1} = a\}$ represents the probability of ending in a state s' when being in state s and taking action a,
- H represents the horizon of the interaction - i.e. defining the length of an episode
- and r represents the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ [SB18, NB16].

We can describe now the behaviour of an agent with a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$, that maps each state-action pair on a probability. In the deterministic case, a policy associates one action to each state, meaning for each state s one pair $[s,a]$ is assigned probability 1. A probabilistic policy on the other hand maps each state to probabilities of selecting a specific action, we denote this as $\pi(a|s)$ [SB18].

[1] We overload here the notation, meaning P can refer to the probability of all preceeding states or just one immediately preceeding.

2.1.2. Return

Typically the goal of the agent is not just to optimize the immediate reward, but the total reward it gets. In particular, we introduce the concept of the return G_t, which is a (weighted) sum of rewards:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=t+1}^{H} \gamma^{k-t-1} R_k$$

The parameter $\gamma \in (0, 1)$, the discount rate[2], defines if the agent is more farsighted (γ close to 1) or puts more focus on the present & near future (γ close to 0). In our problem, we look at the case of an episodic task with a finite horizon H, however in general it is also possible to look at infinite horizon tasks with $H = \infty$. We can therefore now also describe the standard goal of an agent: finding a policy π that maximises the **expected** return, i.e.

$$\max_{\pi} \ \mathbb{E}_\pi[G_t | S_t = s] \text{ for all } s \in \mathcal{S},$$

with $\mathbb{E}_\pi$ referring to the expected return when an agent follows policy π and t being any time step [SB18].

2.1.3. UCB - Upper Confidence Bound

The k-armed bandit describes the mathematical formulation of a environment with k possible actions and one single state. Each action can be seen as pulling the arm of a one-armed bandit where each bandit returns a reward, that is based on a (e.g. normal) distribution. The goal in this scenario would be to maximize the return when playing many/infinitely many times. If we knew the means of the distributions, we could just always pull the best arm, meaning the arm with the highest mean. Without this knowledge we need to come up with strategies on how to maximise our return. A basic greedy strategy could be to pull each arm n times and afterwards just greedily select the best arm. However, with this strategy we may possibly miss the actual best arm. On the other hand, choosing the action purely random would also not lead to the maximisation of the return. We therefore would like to find a balance between **exploiting** already gained knowledge and **exploring** new actions to continuously improve the received return. This is for instance done by so-called UCB strategies, where actions are chosen according to

[2]Depending on the concrete scenario (finite/infinite horizon), 0 and 1 can be included and used as discount rate.

$$a_t = \underset{a}{\operatorname{argmax}} \left[M_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right],$$

which considers the till now observed mean $M_t(a)$ of each action, an estimate for the actual mean, and the uncertainty that underlies this mean. The uncertainty is calculated as an upper bound of the variance of the estimate and takes into account a hyper parameter c that steers the amount of exploration, the time step t we are currently looking at and $N_t(a)$ reflecting how often action a has already been chosen. This means for instance an action is less likely to be chosen if its observed mean drops or the strategy becomes more certain on the actual mean meaning the action was chosen more often [SB18].

2.1.4. Policy Gradient Methods

While for small finite state spaces there exist methods to find an optimal policy, for larger or even infinite state spaces it is intractable to find an optimal policy and we have to set our goal to finding a good approximate solution. We consider in this case parameterized policies - policies that are defined via parameters θ. A basic example would be a neural network where θ represents the used weights. Using a parameterised policy provides also the advantage being able to update θ via gradient ascent to improve our policy [SB18]. While policy gradient methods provide many advantages like being able to deal with a continuous action space [SB18], they often suffer from high variance which leads to slower convergence and instability [LZBY20].

For our purposes we use recent variants of policy gradient methods: Proximal Policy Optimization (PPO) [SWD+17] and Trust Region Policy Optimization (TRPO) [SLA+15]. These policy gradient methods include improvements in scalability, data efficiency and robustness compared to vanilla policy gradient methods.

2.2. Imitation Learning

As already discussed, reinforcement learning relies strongly on a provided reward function. However, in many scenarios it is difficult to hand-craft a good reward signal. Contrastingly, it is often clear how the best achievable final state should look like (reaching a goal in a maze, winning a game, robot fulfilling a specific task,...), even if there is no numerical signal available. In some cases even a (human) expert is available that can demonstrate a task. Therefore one possibility for learning is imitation of expert behavior.

In imitation learning an agent aims to mimic behaviour through observation of expert

demonstrations. Moreover, it is the goal to train an agent that also acts well in unseen situation and "knows" how to carry out the task [HGEJ17], meaning it learns a task and does not only copy the expert. In literature there exist multiple terms for a policy that is trained via imitation, including learning/imitator policy and novice policy. We will use those terms interchangeably.

The main approaches in imitation learning can be divided into the following categories, detailed problems and advantages of the different approaches are evaluated in Section 3:

- Behavioral Cloning (BC): BC-Approaches address the imitation learning problem by applying supervised learning methods using expert demonstrations as ground truth [RB10].
- Inverse Reinforcement Learning (IRL): Research in this area focuses on inferring a reward function from given expert behaviour to train a novice policy via reinforcement learning [Rus98].

2.2.1. Learning from Observation (LfO)

In a basic imitation learning scenario access to both expert actions and states is assumed to be possible. As expert actions may not always be available, we therefore look at the restriction to Learning from Observations - meaning deducing novice policies via imitation learning from experts by observing **only** expert states [TWS19b]. While this gives potentially access to more expert demonstrations, new issues arise especially concerning the substitution of information provided through actions. These are discussed more in detail in Section 3.

2.3. Adversarial Imitation Learning

While originally general adversarial networks (GAN's) [GPAM$^+$14] were used for image classification, these networks was also adapted to other scenarios as imitation learning. We give a general overview over the functionality of GAN's and describe in more detail the adversarial imitation learning approach our framework is based on.

2.3.1. General Adversarial Networks (GAN)

The general idea of GAN's [GPAM$^+$14] relies on the idea of a generative model competing with an adversary model. While the generative model - the generator **G** - tries to create forgeries (for example images of dogs), the adversary model - the discriminator **D** - tries

to recognize those forgeries and distinguish them from real data (for instance real dog images) it also receives [CWD+18]. While forgeries and real data come from the same data space $\mathbb{X}$, the discriminator tries to understand from which **data distribution** (real versus fake) the shown data comes from. This competition against each other leads to improvement on both sides, so for generator and discriminator, till ideally both images are not distinguishable anymore.

Usually $\mathbf{G}$ and $\mathbf{D}$ are both represented by multilayer perceptrons which leads to the notion of general adversarial **networks**. The generator $\mathbf{G}(z;\theta_g)$ defined by the parameters θ_g defines a mapping from a space of latent variables z with distribution $\mathbf{p_z}$ to the data space $\mathbb{X}$. Depending on the actual use case, the output of the generator may be of different form, we describe here the original use case of image generation, so $\mathbb{X}$ being a multi-dimensional vector space. The discriminator $\mathbf{D}(x;\theta_d)$, $\mathbf{D}\colon \mathbb{X} \to [0,1]$, defined by the parameters θ_d, on the other hand outputs a single value representing the probability that $x \in \mathbb{X}$ was generated by $\mathbf{G}$. Alternatingly, $\mathbf{D}$ is trained to correctly classify images as real and generated images and generator $\mathbf{G}$ is trained to minimize $\log(1 - \mathbf{D}(\mathbf{G}(z,\theta_g)))$. This should lead in the end to a generator $\mathbf{G}$ that is able to recover the real data distribution $\mathbf{p_{data}}$ and a discriminator which always gives the output $\frac{1}{2}$ as there is no possibility anymore to differentiate the generator's forgeries from true data. Expressed in a formula, the overall objective is

$$\min_{\mathbf{G}} \max_{\mathbf{D}} \; \mathbb{E}_{x\sim\mathbf{p_{data}(x)}}[\log(\mathbf{D}(x,\theta_d))] + \mathbb{E}_{z\sim\mathbf{p_z(z)}}[\log(1 - (\mathbf{D}(\mathbf{G}(z,\theta_g))))], \tag{2.1}$$

where $\mathbf{p_{data}}(\mathbf{x})$ describes the distribution of the true data and $\mathbf{p_z}(\mathbf{z})$ is the prior distribution of the latent variables [GPAM+14].

2.3.2. General Adversarial Imitation Learning (GAIL)

The to our knowledge first adaption of GANs to imitation learning - Generative Adversarial Imitation Learning (GAIL) - was introduced by *Ho and Ermon* in [HE16].

Imitation learning in a GAN context can be defined via a discriminator that tries to distinguish between data, in the form of states and actions, generated by an expert policy and data generated by a novice policy. The novice's goal, similar as in the original GAN framework, is to produce data indistinguishable from the expert policy.

The goal of the discriminator $\mathbf{D}$ can be formalized as follows

$$\max_{\mathbf{D}} \; \mathbb{E}_{\pi_\theta^{\mathrm{N}}}[\log(\mathbf{D}(s,a))] + \mathbb{E}_{\pi^{\mathrm{E}}}[\log(1 - (\mathbf{D}(s,a)))], \tag{2.2}$$

where $\mathbf{D}\colon \mathcal{S} \times \mathcal{A} \to [0,1]$ outputs the probability that a state-action pair was generated by the novice, $\mathbb{E}_\pi[\cdot]$ describes the expectation with respect to state-action pairs observed when following policy π, π^{E} refers to the expert policy and π_θ^{N} is the novice's policy parameterized by some parameters θ. A perfect discriminator would assign probability 1 to state-action pairs generated by the novice and 0 to the expert's state-action pairs. Adapting the GAN formulation to imitation learning and introducing causal entropy[3] as regularization parameter we obtain the following formulation for the GAIL objective:

$$\min_{\pi_\theta^{\mathrm{N}}} \max_{\mathbf{D}} \; \mathbb{E}_{\pi_\theta^{\mathrm{N}}}[\log(\mathbf{D}(s,a))] + \mathbb{E}_{\pi^{\mathrm{E}}}[1 - \log(\mathbf{D}(s,a))] - \lambda H(\pi)$$

The optimal solution to the above problem occurs when the novice's policy generates data that can not be distinguished from the expert's data by the discriminator. In this case, for an arbitrary powerful discriminator, the novice's policy equals the expert's policy and the discriminator assigns both the probability $\frac{1}{2}$ [HE16].

In practice, the discriminator and the novice policy - often also based on a neural net - are alternatingly updated using stochastic gradient descent [HE16]. In particular, the discriminator maximizes the negative (binary) cross-entropy loss and the novice optimizes its policy using a (reformulated) output of the discriminator as reward signal (for instance using TRPO [SLA+15] or PPO [SWD+17]).

Although the approach is sometimes categorised as inverse reinforcement learning, it does not follow the idea of inferring a global reward function. An improved novice policy does not necessarily receive higher rewards as the reward depends on the also constantly changing discriminator. In an optimal learning process the novice is assigned a probability of 50% of being an expert by the discriminator, a value that can be significantly higher or lower during the actual training.

2.3.3. General Adversarial Imitation Learning in a LfO-context

When not having direct access to actions but only state information (i.e. the case of learning from observations (LfO)), the above equation has been reformulated in [TWS18b] as:

$$\min_{\pi_\theta^{\mathrm{N}}} \max_{\mathbf{D}} \; \mathbb{E}_{\pi_\theta^{\mathrm{N}}}[\log(\mathbf{D}(s,s'))] + \mathbb{E}_{\pi^{\mathrm{E}}}[1 - \log(\mathbf{D}(s,s'))] - \lambda H(\pi). \tag{2.3}$$

[3] Maximisation of the causal entropy deals with the ambiguity of the solution as many policies could describe imperfect observations. A concise explanation for the causal entropy can be found in [HE16],[BB14]. We omit a closer elaboration here as for our experiments we do not consider the regularisation i.e. set $\lambda = 0$.

The discriminator $\mathbf{D}$ is described by a function $\mathbf{D}\colon \mathcal{S} \times \mathcal{S} \to [0,1]$ which maps two states[4] (directly consecutive or interval-wise) to the probability of how likely the tuple of states was generated from the novice.

[4] While shown here for 2 states, also using longer sequences of states is possible.

3. Related Work

While there have been various techniques proposed to use the idea of imitating an expert to train an agent, to the best of our knowledge there is no approach yet to try a Multi-GAN framework with different expert viewpoints. In the following section we discuss approaches in imitation learning with a focus on learning from an observation without access to the conducted expert action. We also dedicate a part to advances in Multi-GAN frameworks (independently of imitation learning).

3.1. Imitation Learning

In contrary to standard reinforcement learning, where learning mostly relies on optimising a given reward function, imitation learning ([Sch96, BK+96]) relies on provided expert demonstrations that an agent tries to imitate. This relieves us of the need of a (manually) defined reward function, that is often not available or hard to define. Another justification for this approach lies in the similarity to the (well-working) human learning process, as we as humans also learn many tasks by simple imitation. We consider the main research areas in imitation learning: Behavioural cloning (BC) and Inverse Reinforcement Learning [TWS19b]. Another way to categorise different methods is concerning the usage of a model. While model-based techniques learn some kind of dynamics model (e.g. from state transitions to actions), model-free approaches do not explicitly learn a model [TWS19b].

3.1.1. Behavioral Cloning

Behavioral Cloning describes a supervised imitation learning process carried out through training a regressor or a classifier on given expert states and actions [RB10].

While this method has already proven its usefulness in practical complex problems such as in autonomous driving [FS18], it also comes with multiple challenges. When predicting actions using expert observations, a policy infers dependencies which might not reflect the true underlying causal structure. A concrete example may be a self driving car - although a skillful expert might stop due to seeing a pedestrian, an imitator may relate the action Break to the visible brake light [DHJL19]. Also, as behavioral cloning

is a supervised learning approach, we normally assume training and test data being i.i.d.. However, the state distribution generated by a trained policy is influenced by the data it was trained on and leads to a violation of the i.i.d. assumption. Practically, an agent might see states (as its actions changed through training) in a test situation that it has not seen in training/has only encountered to a smaller extent in training. This may lead to compounding errors that grow over time. A possibility to alleviate this issue is changing a policy only in small steps with continuous retraining on the new state distribution [RB10].

Regardless of these challenges, behavioral cloning managed also to achieve good performance when confronted with restrictions on the available data. Focusing on the issue of usually only having a small amount of expert data available per task that a robot should learn, an approach of Finn et al. [FYZ+17] dealt with this issue by building their framework in a way such that past experience with previous tasks can be reused to learn a new task. Concretely training the policy via so called meta-learning with demonstrations of other tasks enables the imitator to learn a new task with just a single video demonstration (=one-shot). This can even be extended to the scenario of only using raw video data for the new tasks, so imitation from observation. While the just described approach relies on using the same domain, this is not necessarily given, for instance when training a robot on human demonstrations. Yu et. al [YFX+18] extended the idea by dealing also with possible embodiment and perspective mismatch whilst still being able to learn new tasks with a single demonstration.

3.1.2. Inverse Reinforcement Learning

Inverse reinforcement learning [Rus98] deals with the imitation learning problem through inferring the reward from expert observations. Iteratively the learned reward function is improved and is then used to train a policy with the help of various reinforcement learning methods [AD21].

GAIL [HE16] - Generative Adversarial Imitation Learning, an adaption of the original GAN framework [GPAM+14] to imitation learning - is currently among the most popular IRL algorithms. Hereby a discriminator in the form of a neural network tries to differentiate between expert and imitator observations by assigning probabilities based on which observation it has received. This output can be then used as a reward to train a policy[1].

In the last years, many researchers worked with this framework and provided adapted ideas for different use cases. One strongly influencing part of this framework consists of the objective that a policy tries to minimize as discussed in [FLL17] (algorithm AIRL)

[1]The idea and architecture is discussed thoroughly in Section 2.3.

and [GZG20] (algorithm FAIRL). Ghasemipour et al. [GZG20] even went further and provided an intuition describing where a policy is encouraged to put its probability mass constrained on the discriminator. Concretely they elaborate on how the objective trains the policy based on the different probability mass assignments of state-action pairs (s, a) by policy and experts. While GAIL tries to discourage the imitator to put more weight on regions than the expert, AIRL nudges the imitator to follow the weighting of the expert. FAIRL focuses on regions where the expert has put more or slightly less weight than the imitator, not caring about other regions. This leads to placing low probability everywhere in the beginning and slowly approaching the true expert distribution.

Unfortunately instability is not only a challenge when using GAN's [MKKY18], but also showed to be a difficulty for the GAIL framework. A work by *Pent et al.* [PKT$^+$18] focuses on a better balance between discriminator and generator. Using a variational discriminator bottleneck, it is possible to manage the discriminators accuracy and therefore also the improve policy learning. Another problem tackled in research by *Zolna et al.* is the focus of the discriminator on the actual task [ZRN$^+$21]. The authors show that discriminators often get distracted from spurious features and provide an adapted GAIL-method (TRAIL) where the discriminator "forgets" non-task relevant features.

Within the category of adversarial methods, there do not only exist model-free methods like GAIL, but it is also possible to use GANs in a model-based approach, see [BAM16]. In the proposed approach the discriminator tries to take advantage of the state distribution and the factor of how likely action a_t is under state s_t to evaluate from which policy it received a sample. According to the authors, the advantage of their model lies especially in being able to explicitly derive the policy gradient from the gradient of the discriminator which is not possible without a model.

Many recent works have also taken approaches which focus on specific aspects of the imitation learning problem problem:

Restriction of live experimentation. While many reinforcement learning algorithms are trained on environments where algorithms can be continuously tested in an environment - an on-policy scenario - situations where live-testing is not possible or desirable, meaning training is only possible in an off-policy manner with prerecorded demonstrations, pose a completely different challenge. One typical example for restrictions of live training is imitation learning in context with health-care. Still there exist algorithms that can deal with this aggravating conditions [JBvdS20, KNT19].

Energy-Based imitation learning. Using energy-based learning - a method that reflects dependencies between variables through a scalar energy [LCH+06] - became also popular in an imitation learning context. Introducing the energy of states (or state-action pairs in an LfD context) offers new ways for imitation learning methods [LHXZ20, JBvdS20].

Non-availability of good expert demonstrations. Taking into account that not always a good expert can be found or it is expensive to first let a human master a task only to train a robot, research also follows the idea of letting robots learn not via success observations, but rather human failures. This can be done through discouraging robots to replicate failures and encourage exploration in promising state-action areas [GB12].

A general challenge, that especially inverse reinforcement learning is exposed to, is the ambiguity of reward functions. Looking only at a finite subset of all possible observations, there may exist multiple reward functions that lead to different policies which could explain the seen policies [NR+00].

Problems may also arise when processing the provided demonstrations. Expert demonstrations can be noisy [TCS20, AD21] (for instance an expert taking suboptimal actions or not observing its own state perfectly), we may have to deal with differences between expert and agent environment [SAS17] and it may also provide useful to process the expert demonstrations before deducing a reward [AD21].

3.2. Imitation Learning from Observations

While in some scenarios it may be realistic to have access to expert actions, this assumption restricts us from using existing resources like online videos [TWS19b]. Also when generating observations from scratch, the concept of learning just via observation removes the need of complicated frameworks recording actions - instead recording a video is sufficient. Using state-only observations - called Learning from Observation LfO in comparison to learning from demonstrations (state-action pairs) LfD - was discussed already early on [INS02, BUAC02] but many recent developments pushed this topic further. Imitation Learning from observation focuses on two main components [TWS19b]:

- **Perception.** As only observations are being used, it is even more important to process expert observations carefully. Especially when using images, multiple challenges may occur like different viewpoints or even an embodiment mismatch between imitator and expert.

- **Control.** An obvious focus when solving an imitation learning problem still stays how to teach an agent a task, so which algorithm/framework to use.

We will give a short general overview over LfO with a special focus on imitation learning from observation based on GAIL, as our own framework is part of this category.

Various approaches in the area of LfO are based on an inverse dynamics model This model provides a probability of an action given a state-transition (s_i, s_{i+1}) [TWS18a] that is inferred only via the agent without any interaction with the expert. The expert observations are then used to communicate to the imitator **what** the goal is - so which state is desirable in the end - not **how** exactly it should be achieved.

As for classical behavioral cloning it is necessary to have access to actions, in [TWS18a] a inverse dynamics model is trained based on a random imitator policy. The model is then used to infer expert actions, which leads to a classical behavioral cloning scenario. Using this detour, it is possible to apply behavioral cloning on deduced expert actions without exactly knowing the actual actions. Another approach based on an inverse dynamics model by *Nair et al. [NCA+17]* does not even try to infer the expert actions. Having a single predefined goal, the agent infers its actions (after exploratorily defining the model) directly through the provided expert states (s_i, s_{i+1}). While this approach tries to imitate single observations with single actions, in [PML+18] an extended idea is proposed allowing carrying out multiple actions to imitate one given expert state transition.

As mentioned for general imitation learning, also in LfO the issue of different perspectives of expert and novice may arise. In [LGAL18], this is dealt with by translating the context of an expert via convolutional encoders to the imitator's context where the reward can be inferred via the distance between imitator and expert. In another approach with the goal of learning from information from multiple viewpoints, so addressing also the challenge of perception, a time-contrastive network was trained as a basic step. A time-contrastive network defines a neural network with the goal to learn an embedding such that the embedding of images that were recorded at the same time (but maybe from different viewpoints) are close to each other. In the example of pouring coffee into a glass this means that the network needs to recognise features like the level in the glass independent of the viewpoint. In the second step an agent learns the task through a reward function constructed from the learned embedding. Due to its good performance, the algorithm was even used to train real robots [SLC+18]. This approach is especially interesting as similar to our own framework, videos from different perspectives from one demonstration are used as information, however in a quite different way.

Current research did not only deal with the just discussed idea of multiple viewpoints,

but also looked into the issue of having multiple different, partially suboptimal expert policies available [CKA20, LSE17]. In [CKA20], an idea was developed on how to get the most information out of these policies, such that the imitator exceeds the average expert. By not just using the best policy according to its value function in each state, but actually looking at the largest advantage a policy provides when carrying out a specific action, it is possible to find a policy that uniformly manages to outperform all provided oracles. *Li et al.* [LSE17] on the other hand use an approach that combines GAIL [HE16] with the idea of inferring the latent structure behind the expert observations. This can provide a disentanglement of the actual relevant information from the variability that is introduced through various (human) demonstrators with a different skill level. Further focus areas of research include:

Usage of environment-specific rewards. Environments often consist of moving towards a goal (e.g. in a maze). For those cases a natural approach for inferring the reward function is given by deducing the goal proximity of the agent through observations, for instance the euclidean distance. The distance can then be used the as reward for an imitator and provide information on how far it progressed towards the goal in each step [LSSL21].

Deduction from distributions Also in the LfO scenario when only having access to observations and not to actions anymore, distributions are a valuable source for information to train a policy. Looking at the distribution of state-only observation sequences, it is possible to train an imitator through minimizing the Kullback-Leibler Divergence [2] between the deducted probability distributions from expert and novice [BSBM22, JSA+21]. Recent related work includes also a predictive model based on a recurrent neural network that uses as reward function the distance between the actual and the predicted next state by the on expert observations trained RNN [KCTD18].

3.2.1. Imitation Learning from Observation based on GAIL

Although the original GAIL [HE16] framework relies on the provision of actions, it can also be adapted to an LfO context.

Removing the provision of an expert action is in the context of the GAIL framework influencing the discriminator and the information it achieves to assess if information originates from a expert or an imitator policy. The most basic idea in this case would be

[2] a measurement for the difference of two probability distributions [Joy11]

to just omit the action and provide one single state to a discriminator [MTT+17]. This was proven to work even in a Multi-Expert scenario where the final reward relies on a soft-weighted average of multiple discriminators where each discriminator is trained on a different expert policy [HCB+18].

Unfortunately, being able to generate similar state distributions does not necessarily give evidence of an agent imitating the actual expert behaviour well. Looking at an environment with the goal of running in a circle **clockwise**, running in a circle counterclockwise with the same speed would give the same state distribution [TWS19b] and discriminators shown only single states would not be able to differentiate between an agent running clockwise or counterclockwise. This example motivates the idea of looking instead at sequences of states as input for discriminators.

In [TWS18b] a GAIL framework *GAIfO* was proposed where discriminator and policy receive a tuple of 3 (or 4) **consecutive** states (s_t, s_{t+1}, s_{t+2}). Furthermore, the authors also look at an analogously working framework using raw visual input images (introducing CNN networks for policy and discriminator) as states. While in the case of using images as state representation it is possible to keep up with an TRPO-trained agent that also uses images as states, we see a notable performance drop in comparison to expert performance, supposedly due to limitations in learning just from visual data.

Based on this, in follow up works the authors evaluated further adaptions to their own framework. In [TGWS19], the problem of efficiency was addressed by combining GAIL and linear quadratic regulators (iLQR's) [TET12a]. Also the fact that the agent itself has often access to its own states was discussed [TWS19a]. By using this idea, only the discriminator has to deal with raw image input, while the generator can use less complex states as input. With leveraging this access, the authors managed to outperform their own previously proposed *GAIfO* algorithm.

Another GAIL approach by *B. Stadie et al.* [SAS17] addresses the problem of different viewpoints. Training a discriminator on images from different perspectives without any additions would lead to a discriminator classifying not by the actual policy, but by differences related to the perspective. The authors propose to find a domain-agnostic representation of the observations by introducing an additional domain classifier in the discriminator architecture. While still following the idea of GAIL, this second classifier is used to maximise the domain confusion in the convolutional layers of the discriminator and therefore keeping it from using information related to a specific perspective. Also the performance for different camera angles is compared in this work, showing that some environments benefit more from similar viewpoints than others. Furthermore differing from already described approaches in this case timewise slightly shifted images (s_t, s_{t+3})

are fed into the discriminator.

Apart from the dominating problem of perception, also other aspects are discussed in current research. A quite recent work MobILE (Model-based Imitation Learning and Exploring) [KCS21] focuses on the aspect of trade-off between imitation and exploration. With the introduction of an additional forward dynamics model the policy is learned to do more exploration in case of uncertainty (less power of the discriminator) or to focus on imitation (learning fully based on the discriminator). Also although being tested mostly on popular reinforcement learning environments like MuJoCo, [TET12b] GAIL was also proven to work in more complex settings, for instance playing Super Mario Bros. [LHI20].

Till now we only considered single-agent problems, however GAIL is also applicable on a Multi-Agent framework where multiple reinforcement learning agents interact with one environment [SRSE18]. That means even if these agents carry out different tasks, they are still indirectly influencing each other via the environment. Depending on the actual goal of the agents they may share a reward function, have separate goals and interact only via the then shown environment states or work competitively against each other where an agent receives the opposite reward of the second one by defining $r_1 = -r_2$.

3.3. Multi-GAN Frameworks

The originally proposed GAN framework [GPAM$^+$14] consists of exactly *one* discriminator and *one* generator. When adapting this famous framework, it also occurred to think about changing the number of the actors in a basic GAN setting [DGM16, HLMS19], but also for GAIL-based approaches [HCB$^+$18]. In [DGM16] a theoretical basis is given to deal with multiple discriminators. As multiple discriminators provide multiple outputs, different scenarios are analysed. While a obvious idea may be to just use the maximum of all outputs, this can lead to a too strong discriminators which hinders the policy from learning. Therefore also other ideas with softened outputs over all discriminators were tested, which pose the additional advantage of being able to adapt a softening factor over time.

Using multiple discriminators was also tested for the use-case of data parallelisation on multiple workers due to performance and/or size. Furthermore, this set-up makes it possible to deal with the scenario of crashing workers and still achieving a good performance. While the generator receives in each step input from all discriminators, a swapping procedure between discriminator parameters is used to avoid overfitting of discriminators as each learns with just one batch of training data [HLMS19]. In the context of imitation learning to our knowledge multiple discriminators were till now only

used to incorporate multiple experts [HCB+18] not different aspects of a single expert policy.

4. Problem Setting

In this thesis, we focus on the problem of imitation learning from observations when provided with multiple expert perspectives in a LfO scenario. Concretely, we consider a setting of a Markov Decision Process with a finite horizon and the goal of finding a policy to maximise the expected return, see Section 2.1. In the case of imitation learning, the reward function is not given, but the to be learned novice policy π^N tries to imitate a given expert policy π^E, see Section 2.2. The provided demonstrations can then be used with supervised learning or through inferring a reward function as in our approach [TWS19b].

In a basic imitation learning scenario we have access to all internal states $\mathcal{S}$ and actions $\mathcal{A}$ of expert and novice. However, this restricts us concerning usable expert observations, as we can only use expert-observations where both actions and states were recorded. To avoid this restriction, we can generalise the idea of imitation learning and look at a scenario where we just have access to internal states $(s_1, s_2, ...)$ and deduce a reward function depending on one or multiple states $\mathcal{R}\colon (s_t, s_{t+1}, ...) \to r$. While this relieves us of the need of knowing the concrete action, it is still required to actively track the environment when collecting expert interactions. Thus we would not be able to use historical observations that were initially not intended for reinforcement learning. Therefore we look at a setting where a novice only receives observations which are indirectly provided through a (possibly unknown) mapping from states in a d-dimensional space $\mathcal{O}\colon \mathbb{S} \to \mathbb{R}^d$, commonly RGB-images. This reflects not only more clearly the natural human approach of learning from someone else just by observing, but also enlarges the pool of learning resources that can be used as we can also rely on available expert videos.

Remembering again the human way of learning a skill through imitation via observation, we as humans have the possibility to look at an action from different perspectives that we can actively change to observe from the most informative one. These perspectives can be represented by different camera angles, but also through the actual distance to the expert. For instance, depicting a scenario of a 3D world in a 2D image may lead to missing information as important parts could be concealed. Looking at a basketball player from behind will probably not teach us how to dribble a ball as his hands are not

visible.

We therefore consider a scenario where the novice has access to expert demonstrations from various perspectives $\mathcal{V}$ in the form of image sequences $(o_1, ..., o_{|\mathcal{V}|})$ and can actively choose from the perspectives in its learning process. While it would be possible to give the novice a completely free choice (of an endless amount) of perspectives, we restrict ourselves to the scenario of a fixed number of provided perspectives. With this approach we do not need to actively query an expert (this may be expensive and/or not possible), but can use prerecorded expert sequences or do a one-time recording of the expert. Our approach builds on the idea of being able to collect more relevant information by observing a task from multiple perspectives, which only creates the overhead of multiple cameras, not more expert querying time. As it should be possible to use expert data collected beforehand, the novice does not influence the expert in any way by watching.

We have pinpointed two specific factors that support the idea of utilising multiple perspectives during training:

- When using only one perspective for training, a human decides beforehand which perspective to use to train a novice and therefore intervenes in the learning process, as the humanly preferred perspective must not necessarily be the best for a machine
- It is not always possible to provide exactly one perspective that contains all information. By using information from multiple perspectives, a novice may be able to learn more details about a single task and receive better performance.

To enable learning from multiple perspectives we introduce a perspective selection strategy ξ which decides based on available (past) information which expert perspective a novice policy is shown to learn. While there exist many ways how such a strategy could be defined, we only focus on strategies that decide based on the observed performance concerning the perspectives[1] or decide randomly. In an extension of the problem, more refined strategies could be introduced. Approaches could include introducing the strategy as an own reinforcement learning problem - so looking at a multi-agent framework - or trying to use and/or learn more about the correlations as some perspectives might show the same information (for instance multiple rotations of a bird's eye perspective). However clearly the main goal of each strategy must be to provide the novice policy always with the most informative perspective and only seldomly selecting uninformative ones.

[1] As mentioned here the novice actually has access to all expert perspectives simultaneously. The devil is here in the detail and explained more concisely in our Methodology Section 5. In short, the novice describes a group of multiple components of our algorithm that deals with the learning process and has access to all expert observations and extracts information. The **novice policy** refers to the actual learned policy and "sees" always only one perspective when learning.

5. Methodology

Our goal is to solve the imitation learning problem by using information from multiple perspectives as described in Section 4. To tackle this problem we have developed an architecture that unifies multiple components in a structured training algorithm. The complete architecture as well as its single components and the actual algorithm will be discussed in detail in this section. We base our approach on generative adversarial imitation learning [HE16] and use also ideas that have been presented in [TWS18b] and [SAS17].

5.1. Architectural Overview

To provide a general understanding of our approach we present a schematic illustration of our framework in Figure 5.1:

By interacting with a provided environment a well performing expert policy π^{E} and an in the beginning random novice[1] policy π^{N} generate observations o_t - in our case RGB-images - that represent the respective current environment states from different perspectives. Each discriminator $D_1, ..., D_{|\mathcal{V}|}$ - dealing with exactly one perspective - tries to differentiate if observations, fed in as tuples of two time-wise shifted images, are provided by an expert or a novice policy and is trained based on this information.

With the discriminators we are able to train the novice policy π_{N}: In comparison to the GAIL framework [HE16], where the novice policy is trained with a policy optimisation algorithm based on the reward that is inferred from the output of a single discriminator, we introduce an intermediate step of a perspective selection strategy ξ. This strategy decides in each training step t, based on current and past rewards, which current reward $r_t^{\mathbf{D}_1}, ..., r_t^{\mathbf{D}_{|\mathcal{V}|}}$, that is inferred from the output of the discriminators respectively, should be used in this step to train the novice policy π^{N}. By alternately training discriminators and the novice policy, their improvement leads to a novice policy that is able to imitate a well performing expert. In practice, the discriminators as well as the novice policy

[1] We overload the notion *novice* to describe all relevant components that are directly part of the training, while the novice policy π^{N} describes the actual policy that is learned.

are trained with batches of multiple observations from multiple environment episodes in each training's iteration. It should be emphasized that discriminators are trained **only** based on observations (=images) whilst the optimisation of the novice policy is using environment-internal states and actions (**not** observations) and the reward deduced from the discriminators.

While the framework is based on the idea of GAIL [HE16], the novelty of our approach lies in the introduction of multiple discriminators and a perspective selection strategy. In the following section, we provide detailed explanations about all parts of our approach.

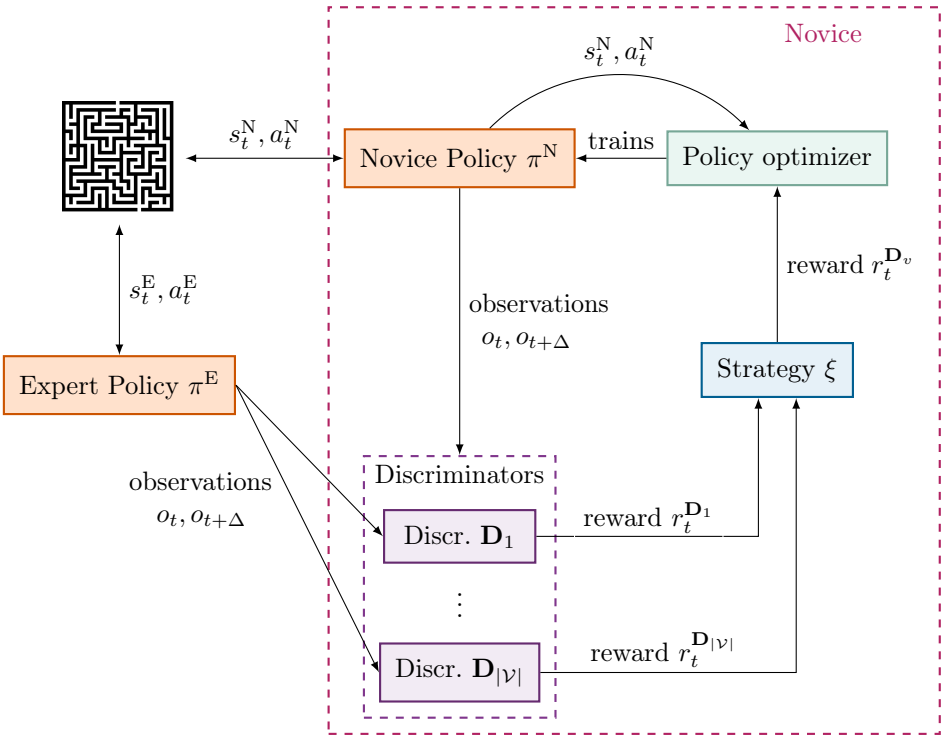


Figure 5.1.: Architectural overview of our proposed framework. Each discriminator measures the imitation success of one perspective respectively. The perspective selection strategy ξ defines which discriminator output should be provided to the novice policy π^N in policy training.

5.2. Details on components

5.2.1. Discriminator

One of the key ideas of our approach is the usage of multiple discriminators $\mathbf{D}_v$, exactly one for each perspective $v \in \mathcal{V}$. Each discriminator is trained to distinguish between

observations received from novice and expert in a particular perspective and gives a probabilistic estimate which entity provided the shown observation.

After a careful testing process of different discriminator architectures, cf. Section 6.2, we decided on a combination of a DCGAN architecture [RMC15] with ideas from [SAS17] and [TWS18b]. In particular, we substitute the missing action information - a single observation/image does not contain much information about the acting policy - by considering 2 slightly time-shifted observations/images $o_t, o_{t+\Delta}$, with Δ representing the time shift, as inputs to a discriminator. Similar to [TWS18b], we concatenate RGB-images of size $d \times d \times 3$ - in our case 2 images - and feed those into our network as $d \times d \times 6$ arrays[2]. Since some actions only lead to small differences in images, we decided to not use consecutive images, but rather pick up the idea of larger image shifts ($\Delta > 1$ as in [SAS17]). As we assume to be only given observations in the form of images and do not know the actual actions of the expert, we build on the approach of GAIL in an LfO context that was discussed in Section 2.3.3.

We show our general proposed discriminator architecture in Figure 5.2. After passing through 5 convolutional layers to extract image features and applying a sigmoid function, we receive an output $\mathbf{D}(o_t, o_{t+\Delta}) \in [0, 1]$. The discriminators are optimised with the optimizer ADAM [KB14] and a binary cross entropy loss CE given as

$$\mathrm{CE}(\mathbf{D}(o_t, o_{t+\Delta}), y_t) = -[y_t \log(\mathbf{D}(o_t, o_{t+\Delta})) + (1 - y_t) \log(1 - \mathbf{D}(o_t, o_{t+\Delta}))]$$

with y_t representing the labels of the observations (= the agent on whose actions those observations are based): 0 for expert, 1 for novice. This concretely means the output of each discriminator represents the probability that a observation tuple $(o_t, o_{t+\Delta})$ was generated by a novice. As we generally stick to the DCGAN structure proposed in [RMC15], we use *tanh* normalised images, LeakyReLU as activation function and batchnorm. When training a discriminator, we pass twice over each provided image tuple as this showed improved performance in comparison to a single pass, cf. Algorithm 1.

Although we use the output of the discriminators to provide the novice policy with a reward signal, it can not be seen as a typical reward:

- Discriminators are constantly changing during the training, meaning reward signals can differ for equivalent situations over time.
- In the desired scenario of a novice policy that perfectly imitates an expert, a discriminator would not be able to differentiate both policies anymore and assign

[2] d represents here the size of the image and can differ depending on the actual environment, see Section 6

the novice policy a probability of 50% of being an expert. Still during the actual learning process the novice policy may manage to achieve higher values although showing worse performance, as also the discriminators are improving.

While we show here the basic discriminator architecture that can be applied for $64 \times 64 \times 3$ images, experiments showed that this image size does not fit for all environments. In those cases the here shown architecture was adapted slightly, for more details see Section 6.

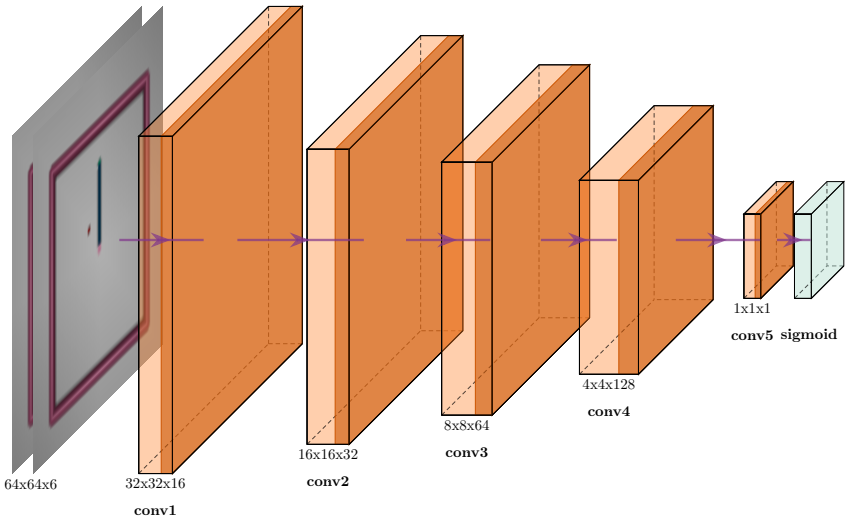


Figure 5.2.: Architecture of the DCGAN discriminators. We feed the concatenation of 2 observations namely $64 \times 64 \times 3$ RGB images into the discriminator to receive a sigmoid normalised output.

5.2.2. Environment

Our approach is designed for typical reinforcement learning environments, for instance provided by the Python library *Gym* [BCP+16]. We discuss the tested environments and perspectives more closely in Section 6, however our approach is easily adaptable to other/more environments and perspectives - we leave this generalisation open for future work. As just mentioned, the reward provided by the discriminators cannot be used as final evaluation metric. To measure performance we therefore use an environment internal reward signal which is usually used when training via reinforcement learning - we refer to this in the future as true reward. However, the true reward signal is **not seen** by any of

our components and **not used** in the training process.

5.2.3. Expert Policy

While for predefined environments there may be ready policies available, we decided on training the respective expert policies ourselves to being able to work with our adapted environments and the customized goal of reaching a high reward in a fixed amount of steps. This training is done via reinforcement learning with TRPO/PPO [SLA+15, SWD+17] by using the true reward signal of each environment.

As the goal of our novice policy π^{N} is to fool the discriminators into believing that its executed action resemble an expert action, the performance of our provided expert policy provides the upper limit our novice policy can reach.

5.2.4. Novice Policy

Although the complete novice as shown in Figure 5.1 contributes to the achieved performance, the main goal we would like to achieve is to train the actual novice policy π^{N} as efficiently as possible. While we generally measure the performance through the true reward signal, we also provide the possibility of visually inspecting the learned policy, as this helps to understand the actual still unlearned aspects.

While the discriminators receive only RGB-images as environment states, we assume a scenario, where the novice policy π^{N} has access to its own internal states - states provided by the environment, usually in vector form. This is motivated by the idea that we want to use prerecorded expert demonstrations where we do not necessarily have access to internal states, but do know all internal processes in an agent when it follows a novice policy. This facilitates the policy structure (otherwise we would need a convolutional neural network to extract image features) and the learning process. The general idea of solely using image observations was already discussed in [TWS18b], for our concrete approach we leave this open to future work.

5.2.5. Rewards

As reward signal provided by the discriminators $\mathbf{D}_v, v \in 1, ..., \mathcal{V}$, we define

$$r_t^{\mathbf{D}_v} = -\log(\mathbf{D}_v(o_t, o_{t+\Delta})),$$

similar as defined in the original GAIL paper [HE16]. While the theoretical background[3] for this reward function lies in Equation 2.3, this reward makes also sense intuitively: Due to the definition of our labels and the functionality of our discriminators, a well trained discriminator tries to output values close to 1 for novice observations and values close to 0 for expert observations. Our policy is optimised to achieve higher rewards - this means to encourage the novice policy to behave more expert-like we can not use the raw discriminator output as reward. To use the discriminator output as a reward signal, it is therefore necessary to assign the highest possible value to expert-like behaviour for instance through $-\mathbf{D}_v(o_t, o_{t+\Delta})$ as reward. By using $-\log(\mathbf{D}_v(o_t, o_{t+\Delta}))$ as reward, we enlarge the range of our rewards which means the policy receives clearer reward signals for really good actions.

5.2.6. Strategies

To deal with multiple perspectives, we assume that the expert is observed from multiple perspectives simultaneously and that the novice is provided with the made observations. All observations from perspective v are related to the corresponding discriminator $\mathbf{D}_v$. Each batch of novice observations used in policy training is fed through all discriminators. Based on the returned reward signals and past information, like how often the perspective was already chosen and past rewards, a strategy then decides which reward is used for the current batch. While there are in general many possibilities for how such a strategy could look like, the goal is to find a strategy that is beneficial to policy training. We assume that this is the case when a strategy provides the novice policy with the perspective it does not know enough about, meaning it performs worse on this perspective than on others. In the actual implementation we tested the following strategies:

- **Random Strategy (Rand).** As a basic strategy, we decided on a random strategy. Concretely, in each batch while training the policy, it is randomly chosen from which discriminator the policy receives the reward signal

- **Minimum Probability Strategy (MinProb).** This strategy selects the discriminator that attests the lowest performance to the novice policy **for the current batch**, meaning it assigns it the lowest probability of being an expert. Concretely the average output for each discriminator on this batch is calculated and then compared.

[3] Concrete derivations can be found in [TWS18b] and [HE16]

- **Maximum Probability Strategy (MaxProb).** This strategy picks the discriminator that currently attests the best performance to the novice policy **for the current batch**, working the same ways as MinProb, but choosing the maximum instead of the minimum.

- **UCB (UCB).** We base our strategy on UCB, see Section 2.1.3. For each batch $\tilde{\omega}^{\mathrm{N}}$ of observation-tuples $(o_t, o_{t+\Delta})$ generated by a novice policy the strategy ξ decides as follows:

$$\underset{\mathbf{D}_v, v \in \mathcal{V}}{\operatorname{argmin}} \sum_{i=1}^{|\tilde{\omega}^{\mathrm{N}}|} \frac{-\log(\mathbf{D}_v(o_t^i, o_{t+\Delta}^i))}{|\tilde{\omega}^{\mathrm{N}}|} - c\sqrt{\frac{ln(t)}{N_{v,t}}},$$

  looking at the mean of the received reward per perspective and taking into account the uncertainty by looking at $N_{v,t}$ - the number of times perspective v has been chosen so far. While we are not considering a standard best-arm selection bandit problem here, the UCB strategy still has meaningful characteristics for our problem setting. UCB leads to each perspective being selected infinitely often and focuses on the perspective in which the discriminators can best differentiate between expert and novice. Intuitively, big differences between the expert's and the novice's actions in one perspective mean that there is still much to learn for the novice concerning information shown in this perspective, while it may "know" already how to behave concerning a different perspective[4].

5.2.7. Policy Optimizer

For training the novice policy π^{N}, the selected reward - a measure of how likely the data was generated by the expert - is used as the reward signal. For optimisation we use as policy optimizer $\mathcal{T}$ Proximal Policy Optimisation (PPO) [SWD+17].

5.3. Training Algorithm

Algorithm 1 shows our proposed training algorithm which is based on the training algorithm proposed in [SAS17]. After initialising all components, we carry out training of the discriminators and the novice policy alternatingly. While we have already given

[4] Giving an example based on our implementation, see Section 6, the novice may already have understood how to move in the x-direction in the Point Environment, but still having problems to take fitting actions in the y-direction.

a general overview in Section 5.1 and show the concrete algorithm in Algorithm 1, we would like to highlight here specific details and explain used notation:

As we consider the scenario of having only restricted access to expert data, we look at a fixed number of E_d recorded environment episodes (with expert/novice policy) that can be used for training the discriminators in each iteration. For training the novice policy where no actual expert data is needed, we can more freely just train iteratively Q_p-times (for Q_p epochs) where each training improves the policy a little bit. The used batch sizes for training discriminators/novice policy b_d/b_p must not necessarily match as these batch sizes are subject to tuning in order to achieve the balance between both training processes.

For the actual samples, meaning observations created based on expert/novice policy $\pi^{\mathrm{E}}/\pi_\theta^{\mathrm{N}}$, we chose the following notation: To describe a set of observations collected we refer to Ω^E/Ω^N. We indicate the transformed set - a set of batches of observation tuples $(o_t, o_{t+\Delta})$ - with ω^E/ω^N. From ω^E /ω^N single batches $\tilde{\omega}^{\mathrm{E}}/\tilde{\omega}^{\mathrm{N}}$ can be taken, each batch has a size of b_d or b_e, depending if we look at discriminator or policy training. As mentioned in Section 5.2.4, for training the novice policy, we do also use the internal states and the actions $(\mathcal{S}, \mathcal{A})$ and indicate this with the variable I in our algorithm.

In practical scenarios it is often impossible to have endless access to expert data. In our algorithm this is reflected by the parameters η and T_e. Depending on the experiment, it is possible to generate new expert data for every training iteration ($\eta = True$) or to generate T_e environment episodes based on the expert policy (=expert episodes) in the first iteration - a global list of observations Π^{E} - and reuse this expert data for all further iterations. While in the case of the number of discriminator training's episodes for one training iteration being smaller than the available expert episodes ($E_d \leq T_e$), we use each expert episode at most once per iteration (drawn randomly without replacement), in case of $E_d > T_e$, it may happen that expert episodes are used more than once per training iteration.

We optimise the discriminators - neural networks - by using ADAM [KB14] as optimisation algorithm. To describe the calculation of the Cross Entropy Loss when imputing a batch of observation-tuples $\tilde{\omega}^{\mathrm{E}}$ generated by an expert policy into a discriminator we overload the notation and use:

$$\mathrm{CE}(\mathbf{D}(\tilde{\omega}^{\mathrm{E}}), \text{"expert"}) = \frac{1}{b_d} \sum_{t=1}^{b_d} \mathrm{CE}(\mathbf{D}(o_t, o_{t+\Delta}), 0)$$

Analogously we proceed for the novice batches $\mathrm{CE}(\mathbf{D}(\tilde{\omega}^{\mathrm{N}}), \text{"novice"})$ by substituting 0

with the novice label 1. Similarly we abbreviate the reward notation and use

$$r^{\mathbf{D}_v} = -\log(\mathbf{D}_v(\tilde{\omega}^{\mathrm{N}})) = -\frac{1}{b_p}\sum_{t=1}^{b_d}\log((\mathbf{D}_v(o_t, o_{t+\Delta}))$$

5.4. Implementation

The algorithm was implemented using Python and the reinforcement library Garage [gc19]. As stated, the general idea of the training structure of the algorithm is inspired by [SAS17], however the code is an own implementation. Differences are in the different components, the usage of Pytorch [PGM+19] instead of Tensorflow 1 [AAB+15] and an upgrade to the newer library *Garage* [gc19].

For each environment at least one separate launch file is available to run various experiments with different parameters. To be able to evaluate data of experiments, a pre-implemented, but slightly adapted experiment wrapper [gc19] is used such that all interesting data points and the used configurations are saved and can be used for evaluations. While experiments have been done only on 3 environments, the code is modular and can use any *Open AI Gym* Environment [BCP+16] with only small adaptions.

Algorithm 1: Imitation learning from multiple perspectives

Input: perspective selection strategy ξ, No. iterations K, perspectives $\mathcal{V}$, horizon H, time shift Δ, No. episodes for discriminator training E_d, No. training epochs for novice policy Q_p, batch size discriminator b_d, batch size policy b_p, boolean all expert data η, number of available expert episodes T_e

/* Initialisation */
for $v \in \mathcal{V}$ **do**
  Initialise discriminator $\mathbf{D}_v$
end
Initialise environment e policy π^{E} and novice policy π^{N}_θ; Initialise perspective selection strategy ξ and policy optimizer $\mathcal{T}$;
/* training: each iteration trains the discriminators, then the novice policy */
for $i = 1, \ldots, K$ **do**
  /* Get demonstrations from expert */
  if η=*true* **then**
    Collect observations Ω^{E} following π^{E} from E_d episodes
  else
    if *K=1* **then**
      Collect observations Π^E following π^{E} from T_e episodes
    Generate Ω^{E} by randomly choosing E_d episodes from Π_E;
  /* Get demonstrations from novice */
  Collect observations Ω^{N} from π^{N} from E_d episodes
  /* Update discriminators */
  for $\mathbf{D} \in \{\mathbf{D}_v | v \in \mathcal{V}\}$ **do**
    Prepare a list of batches ω^{E}/ ω^{N} of observation tuples $o_t, o_{t+\Delta}$ from $\Omega^{\mathrm{E}}/\Omega^{\mathrm{N}}$ with batch size b_d
    for _ *in range(2)* **do**
      for *($\tilde{\omega}^{\mathrm{E}}$, $\tilde{\omega}^{\mathrm{N}}$) in ($\omega^{\mathrm{E}}$,$\omega^{\mathrm{N}}$)* **do**
        $\mathcal{L} = \mathrm{CE}(\mathbf{D}(\tilde{\omega}^{\mathrm{E}}), \text{"expert"})$
        minimize $\mathcal{L}$ with ADAM
        $\mathcal{L} = \mathrm{CE}(\mathbf{D}(\tilde{\omega}^{\mathrm{N}}), \text{"novice"}))$
        minimize $\mathcal{L}$ with ADAM
      end
    end
  end
  /* Train novice policy */
  for _ *in range(Q_p)* **do**
    Collect observations Ω^{N} and internals I^N from π^{N} of batch size b_p
    Prepare a batch $\tilde{\omega}^{\mathrm{N}}$ by storing tuples $o_t, o_{t+\Delta}$
    Select discriminator $\mathbf{D}$ according to ξ
    rewards $= -\log(\mathbf{D}(\tilde{\omega}^{\mathrm{N}}))$
    Train the novice policy π^{N}_θ with the rewards and I^N via $\mathcal{T}$
  end
end

Result: Optimized novice policy π^{N}_θ

6. Experiments

We conducted various experiments to assess the performance of our proposed framework in different scenarios. In this chapter we give an overview of our experimental set-up, discuss discarded approaches and evaluate the the obtained results.

In our experiments we are thoroughly investigating the functionality of our proposed framework. This includes running experiments on multiple environments, evaluating the framework and especially the perspective selection strategies through various means and comparing it to alternative scenarios.

The main goals of this chapter consists therefore of the following:

- Providing an extensive description of our experiments including:
  - Environment specifications
  - Chosen hyper-parameters and reasoning behind the choice
  - Definition of used metrics and baselines in performance evaluation
- General performance evaluation for our proposed multi-discriminator framework
  - Proof of concept - our framework can imitate an expert
  - Performance evaluation of different strategies
  - Comparing our results to baselines
  - Investigation of top performance (on best run) versus average performance
- Evaluation of different ideas to improve our framework
  - Assessing the influence of the amount of expert data used for training
  - Comparison to a super-discriminator which receives all perspectives simultaneously

6.1. Environments

To assess how well our idea generalises, we ran our experiments on 3 different reinforcement learning environments. For a first proof of concept and to test various configurations, we

implemented an easy 2D *Point* environment. More elaborate evaluations were executed on 2 MuJoCo environments - *Reacher* and *Hopper* [BCP+16, TET12b]. Although relying on a pre-implementation, we adapted both environments to the needs of our experiments. While the proposed Point environment acts as test environment and as representative of a 2D environment, Reacher and Hopper evaluate a 3D scenario with movement concerning different axes.

We conducted early tested with a more extensive number of environments, however decided to use only 3 environments that cover different movement dynamics (move on a surface, move in a 3 dimensional space) for the final evaluations due to many time consuming experiments. While initially planned to nevertheless include the *Swimmer* environment [BCP+16, TET12b] where a three joint worm tries to move forward as fast as possible, this environment was discarded due to difficulties of finding a good enough expert policy for our use case.

Our framework is not applicable to all (robotic) environments: looking for instance at a card game like Black Jack the relevant information is represented by card values that do not change with perspective. We decided to run our experiments on environments that can be looked at from different viewpoints in a way that different perspectives still provide information, but not necessarily all information needed to perfectly learn a task.

Each of our reinforcement learning environment consists of the following:

Basics. Each environment includes an action space, a space space and a true reward function, that we use for evaluation. When conducting an action a_t being in state s_t, each environment returns a new state s_{t+1} and a reward r_{t+1} according to internal transition dynamics.

Rendering. As discriminators receive images as input, each environment includes a rendering function to provide a visual representation of the current environment state. We implemented the rendering in a way such that we can flexibly change the returned image sizes (by downscaling an originally large image representation) and the shown perspective in terms of camera angle and distances.

For each environment, we test concretely 4 different perspectives cf. Section 6.3.1:

- A baseline perspective supposedly showing all information: This means as a human looking at a sequence of images all relevant parts - in our case movements - can be seen directly, nothing is hidden.
- 2 perspectives providing partial information: This means looking at one perspective

not all movements are directly visible, but combining both shown perspectives the complete environment dynamics, meaning how an environment responds to taken actions, should be deducible.

- a perspective providing no relevant information at all: This means no information about the environment dynamics is shown, eg. looking in the wrong direction and just displaying the floor.

Horizon. For simplification, we only work with finite environment episodes and define their length depending on the environment. Results are easier comparable and the resulting fixed array sizes pose also advantages in implementation. Concretely this means each episode has a predefined customized length - the horizon H - independent of a policy reaching a goal or any other termination criterion.

In the next section we will give a concrete description of the used environments. The stated parameters refer to the specifications used in our experiments, in general all environments are built in a flexible way such that parameters could be easily adjusted if needed.

6.1.1. Point Environment

Environment Definition. We introduce a 2-dimensional Point environment. In a coordinate system of $[-5, 5] \times [-5, 5]$, our arena, the goal of the agent is to move a yellow point, the chaser, starting at the origin $[0, 0]$ in each episode to the goal, represented by the blue point, cf. Figure 6.1a. Although chaser and goal are originally defined as points/circles, this may differ in the shown image representation of the environment due to downscaling to a smaller image size to use the images as input for the discriminators. In each episode, the goal is randomly created at an Euclidean distance of 4.5 from the origin to be able to compare the performance of different random initialisations. This leads to the following environment specifications:

- **Action Space**: This environment works with a 2-dimensional action space that describes the movement in x and y direction. To have a continuous movement that an agent can follow, so to avoid too large changes between single observations, movements in both directions are bounded by 0.1, concretely $[x, y] \in [-0.1, 0.1] \times [-0.1, 0.1]$.
- **State Space**: As state space, we provide an agent with the current position of

the chaser, the distance between chaser and goal and the position of the goal in this epoch in the coordinate system, concretely $[x_{chaser}, y_{chaser}, dist, x_{goal}, y_{goal}] \in [-5,5] \times [-5,5] \times [0, \sqrt{2} \cdot 2 \cdot 5] \times [-5,5] \times [-5,5]$[1].

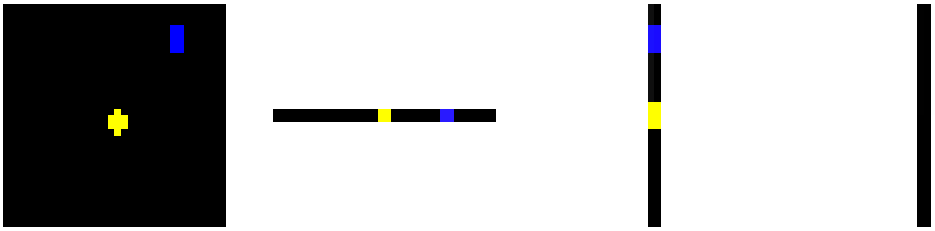
- **True Reward**: As we want to urge the agent to move towards the goal, the true reward signal is represented by the negative Euclidean distance between the current position of the agent and the goal.
- **Horizon**: Adapted to the time a perfect policy would take to reach the goal - a perfect expert can reach the goal in almost all cases in this time - we conduct our experiments with a horizon of 42 steps.
- **Image Rendering**: Assumably due to the simplicity of the environment our experiments showed better results with using smaller images. We decided in the end on an image size of 32×32 as this leads only to minimal adaptions in our proposed discriminator architecture for 64×64 images.

Our developed environment provides the possibility to easily adapt the size of the arena the chaser is moving on and the positioning of the goal. To support the discrimination between chaser, goal and background we decided on clear differentiation between those three components also concerning the RGB values. Concretely we use yellow $(255, 255, 0)$ for the chaser, blue $(0, 0, 255)$ for the goal and black $(0, 0, 0)$ for the background.

Perspectives. As there is no possibility to change camera angles in a 2D environment to just show partial information - we could change the camera angle and turn the image, but this would still include all necessary information - we represent partial information through a projection on the x (resp. y) axis, cf. Figures 6.1b & 6.1c. This means both of those perspectives include information about movement in one of the directions (first/second part of the action). The introduced non-informative perspective just shows a black image. As we are using it in combination with the 1D images, this perspective is also shown in one dimension.

Adaptations in Discriminator Architecture. Unfortunately with our proposed framework in Section 5 we were not able to achieve good results for this environment. We tested various image sizes and received better performance when using smaller image sizes. We therefore assume problems in extracting image features when when as much irrelevant

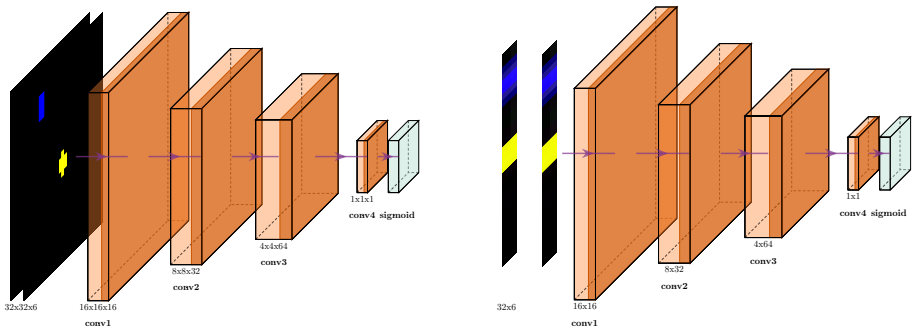
[1] As the chaser can possibly also move in the wrong direction, the maximum distance is large than the distance from the goal to the origin



(a) baseline (b) x-perspective (c) y-perspective (d) no information

Figure 6.1.: Perspectives - Point environment: (a) including all available relevant information, (b) & (c) representing perspectives with partial information, (d) showing no information - a non-informative perspective.

information is available as given in this environment. Concretely larger images lead to single neurons receiving most of the time only black background. We hence decided on using smaller images - by reducing the dimension exactly by a factor of 2, the adaptation can easily be done by just removing one layer and hence using $32 \times 32 \times 3$ images. Also for the 1D-dimensional images 32×1 there was a need of adapting the discriminator using 1D instead of 2D convolutions cf. Figure 6.2.



(a) Discriminator architecture for 2D images of size 32×32

(b) Discriminator architecture for 1D images of size 32×1

Figure 6.2.: Adaptations of the discriminator architecture for the Point environment.

Expert Policy. Before actually carrying out any experiment to imitate an expert, it is necessary to define and implement an expert policy independent on what we assume to see from the expert in the actual experiment setup. Looking at the state space of our environment, it includes the current position of the chaser (x_{chaser}, y_{chaser}), as well as the position of the goal (x_{goal}, y_{goal}). While for more complex environments the expert policy is often represented by a policy based on reinforcement learning with the true reward

signal, we can in this simple case define an (almost) perfect deterministic expert policy[2] by just taking always a maximum step - in our case ± 0.1 - in the correct direction, this means:

$$action = 0.1 \cdot \text{sgn}([x_{goal}, y_{goal}] - [x_{chaser}, y_{chaser}]) \tag{6.1}$$

We also tested, if not taking the perfect move all the time (moving randomly with probability ϵ) would improve the learning process. Theoretically a too good expert could lead to the issue of the novice not being able to learn anything due to the large differences between expert and novice in the beginning and the consequent problem of a therefore not foolable discriminator. However, this showed no significant improvement, so we conducted our experiments by moving always in the best direction.

Policy Training & Novice Policy. For the policy training we rely on pre-implemented elements from the library *Garage* [gc19]. The novice policy is trained with an adapted version of a trainer class provided in *Garage*. With the help of multiple workers that collect samples in parallel from the *Point* environment and the rewards via the perspective selection strategy from the discriminators, the trainer provides a framework including the policy optimiser $\mathcal{T}$, a Gaussian MLP policy and a Gaussian MLP value function, all also being *Garage* [gc19] implementations.

The novice policy is trained based on actions, states and rewards. Due to the fixed horizon, the policy training does not have to deal with different episode lengths/different array sizes in one experiment. Different array sizes for multiple experiments (eg. testing different horizons) do not pose a problem. For the novice policy, after some testing we decided on using 2 layers of size 32 and *tanh* as activation function, analogously we proceed with the corresponding value function.

6.1.2. Reacher Environment

Environment Description. We use the predefined MuJoCo Reacher environment [BCP+16, TET12b]. Here a two jointed robot arm tries to move its end, referred to as fingertip, by moving both joints toward a target on the plane represented by a yellow point, see Figure 6.3a. In each episode, the goal spawns randomly at an Euclidean distance of 0.2 around the origin. This leads to the following environment specifications:

[2]The expert is only almost perfect, as it does not stop moving completely when reaching the target, but still oscillates around the target with small movements

- **Action Space**: The Reacher arm moves by applying torques at both hinge joints, concretely $[t_1, t_2] \in [-1, 1] \times [-1, 1]$.
- **State Space**: The state space describes the current state of the Reacher arm as well as the absolute position of the target, the angular velocity of the arm and the relative position of the Reacher's fingertip to the target. This sums up to a 11-dimensional observation space.
  - sin/cos respectively of both parts of the arm (4)
  - position of the target (2)
  - angular velocity of both parts of the arm (2)
  - 3D - vector between fingertip and goal in the form $(x, y, 0)$ as they do never differ in the z-coordinate (3)
- **True reward**: The true reward consists of the sum of the distance between the tip of the robot arm and the goal and a penalty for taking too large actions.
- **Horizon**: By visual inspection of the trained expert policy on how long it takes on average to reach the goal, we decided on a horizon of 54. In this time frame our expert is able to reach the goal without remaining fixed in the end position for too long.
- **Image Size**: We conducted our experiments on 64×64 images. While we tested also smaller images like 32×32, this showed almost no learning effect possibly due to too small images not providing enough information.

Human visual inspection of the partial informative perspectives, see Figure 6.3b and 6.3c, proved difficulties to identify the goal as the pre-implemented goal color is similar to the depicted purplish frame. We therefore decided to change the color of the goal to yellow as shown in the images to provide a clearer differentiation between goal and frame.

Camera Positioning. For both the Reacher and the Hopper environment, we define the single perspectives via the positioning of the predefined camera in MuJoco environments. Concretely, we adapt the following parameters[3]:

- **Azimuth**: To define the viewpoint from which we look at the environment we use an angular measurement in a spherical coordinate system. Azimuth defines the horizontal rotation in degrees from a fixed start point.

[3]Details to *MuJoco* configurations can be found here: `https://mujoco.readthedocs.io/en/stable/APIreference.html`

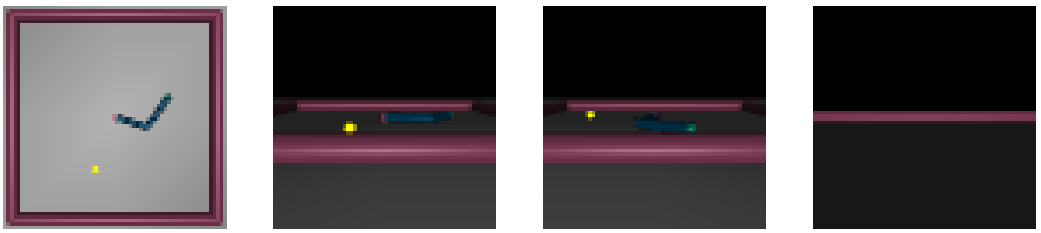
- **Elevation:** Defines the vertical rotation from rotation in degrees from a fixed start point.

- **Distance**: Defines the distance to a fixed start point in the environment.

Table 6.1.: Camera positioning in the Reacher environment.

perspective	azimuth	elevation	distance
baseline	0	-90	0.8
front perspective	0	-7	0.6
side perspective	90	-7	0.6
no information	0	0	1

Perspectives. In this 3D environment, it is possible to adapt the camera angle to control the amount of information shown. An overview over used parameters for the camera positioning is given in Table 6.1. As the Reacher arm moves only 2-dimensionally on a plane, we propose as baseline a central bird eyes view on the environment showing all information cf. Figure 6.3a. The scenario of receiving partial information is represented by looking at the environment from a side-wise perspective. To not only see the frame but also the moving Reacher arm, we chose an angle of 7 degrees between camera and the surface where the Reacher is moving. We propose two perspectives showing the environment from the front and from one side (so differing by a turn of 90° on the z-axis). Our non-informative perspective consists of just seeing the purple frame from the outside, so gaining no information due to the bad camera angle.



(a) Baseline: bird's-eye view (b) front perspective (c) side perspective (d) No information

Figure 6.3.: Perspectives - Reacher environment: (a) including all available relevant information, (b) & (c) representing perspectives with partial information, (d) showing no information - a non-informative perspective.

Expert Policy. As this environment is more complex, we propose in this case a learned expert policy. Our expert policy is based on an own-TRPO-trained MLP network, trained for 20000 epochs with batch size 1024. The policy is defined as a Gaussian MLP policy with 2 hidden layers of size 32. To approve the quality of our expert, we visually inspected the trained expert policy to evaluate how the Reacher moves and fulfills its task of moving fast to the target[4].

Policy Training & Novice Policy. We generally use the same training method as introduced for the Point environment. To being able to collect multiple episodes in parallel we had to slightly adapt the rendering from a technical perspective as the pre-implemented rendering was seemingly not build for our parallelized purpose and returned wrong results. For the Gaussian MLP policy we use 2 layers of size 32 and *tanh* as activation function, analogously we proceed with the corresponding value function.

6.1.3. Hopper Environment

Environment Description While the Reacher environment resides in 3D surroundings, it still carries out its moves rather static on the floor and does not explore a third dimension. We therefore decided on the predefined MuJoCo Hopper environment [BCP+16, TET12b] as our third environment to evaluate if our algorithm performs well when movements can also occur in the z-dimension. A two-dimensional one-legged figure - with torso, thigh, leg and a single foot - tries to make hops to the right with the goal of covering as much distance as possible, see Figure 6.4a. This leads to the following environment specifications:

- **Action Space**: Similar as for the Reacher, the actions in the Hopper environment represent the applied torques at all joints concretely $[t_0, t_1, t_2] \in [-1, 1] \times [-1, 1] \times [-1, 1]$.
- **State Space**: The current state of the Hopper is described through positions, angles and velocity of single parts of the Hopper leg. This leads to a 11-dimensional vector:
  - current height of the Hopper: corresponds to the z-coordinate of its top (1)
  - velocity of the top in x/z direction (2)
  - angles of all three joints and the top (4)

[4]Due to our specific horizon length a direct comparison to results received by others was not possible.

  - angular velocity of aforementioned parts (4)

- **True Reward**: The true reward consists of the sum of a penalty for taking to large actions, a bonus for not having fallen and the distance that it managed to move to the right.

- **Horizon**: As we work with a static camera that does not follow the movements of the Hopper, we decided on an episode length where the Hopper stays in the provided image when following the expert policy, but still covers a significant distance. This leads to a horizon of 154.

- **Image Size**: We decided based on previous experiments with the Reacher environment to use 64×64 images.

Perspectives. In this environment - the Hopper - is not moving on the floor, but in x/z direction. We fittingly adapt our perspectives and consider looking at the Hopper sideways as our baseline perspective, which includes all information, cf. Figure 6.4a. As perspectives that provide partial information remain:

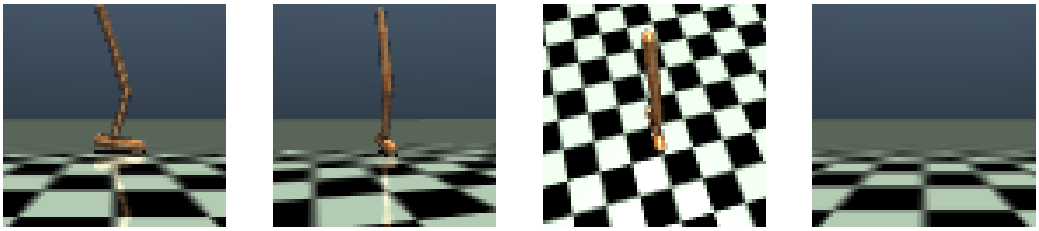
- a front perspective - Figure 6.4b: It provides information about movement and height, but cannot give clear indications about bent joints.

- a bird's-eye view - Figure 6.4c: This perspective provides information about the forward movement, however still provides not much information about bent joints and the current height of the Hopper.

We initially conducted our experiments by looking at the partial perspectives straight from the front, but decided later adapt the angle to 165° and give more information as we received bad results. For the non-informative perspective, we look at the floor from below and depict therefore an empty environment, just showing a floor with check pattern. Parameters used to define the particular perspectives can be found in Table 6.2.

Table 6.2.: Camera positioning in the Hopper environment.

perspective	azimuth	elevation	distance
baseline	90	0	0.65
front perspective	165	0	0.65
top perspective	165	-70	0.65
no information	0	0	0



(a) Baseline: side perspective (b) front perspective (c) bird's-eye view (d) No information

Figure 6.4.: Perspectives Hopper environment: (a) including all available relevant information, (b) & (c) representing perspectives with partial information, (d) showing no information - a non-informative perspective.

Expert policy. We used an MLP network (with PPO) as expert policy trained for 50000 epochs with batch size 32. Our policy is defined as a Gaussian MLP policy with 2 hidden layers of size 32. As we could not find a baseline to compare the performance of our expert with an episode length of 154, we again approved the quality of the expert through visual inspection.

Policy Training & Novice Policy. We proceed exactly analogous to the Reacher environment using the same trainer and a Gaussian MLP policy and value function with 2 layers of size 32 and *tanh* as activation function.

In Table 6.3 we summarized the most relevant parameters for all environments. Taking a closer look at the values for the horizon we would like to mention the following: Due to the training algorithm, cf. Algorithm 1, with pairs of timewise shifted images, only $(H - \Delta)$ images can be used in policy training. The last Δ images are just used as second part of the image tuples. We therefore choose the horizon always slightly larger than indicated (indications were described already for each environment).

Table 6.3.: Environment specifications concerning images and expert policy

	Point	Reacher	Hopper
horizon H	42	54	154
image size	32×32 / 32×1	64×64	64×64
expert policy	deterministic	trained MLP network	trained MLP network

6.2. Discarded Approaches

As a significant part of our work went into testing different frameworks and finding a working solution, we would like to shortly mention other tested framework ideas. While we managed quite fast to find a working solution for an easy 2D environment, cf. Section 6.1.1, we struggled with applying our first idea to more realistic and difficult 3D environments. The biggest issues turned out to be finding a good balance between policy and discriminator training and defining discriminators with a good learning capacity.

Before deciding on the final already described framework, we therefore tested the following ideas with a special focus on improving the performance of the discriminators:

Discriminator structure proposed by *Stadie et al.* [SAS17]. As we originally came to our idea via adapting the framework used in [SAS17], the most basic idea was to reuse the for us relevant parts of the discriminator framework provided in this paper. We tested the idea of a two-tier discriminator framework. The first part consists of convolutional layers that are trained to extract relevant image features. Afterwards two concatenated images $o_t, o_{t+\Delta}$ are fed through 3 dense layers to evaluate if carried out actions carried out resemble more an expert or a novice. While we managed to achieve really good results with small computational effort for our proposed toy environment with this approach, we did not find a way to translate it efficiently to a more complex scenario.

Discriminator structure proposed by *Torabi et al.* [TWS18b]. Inspired by this already working approach that shows similarities with our approach cf. Section 3, we tested the idea of using 3 consecutive grey-scale image observations. Concretely for carrying out an action at timestep t, the novice receives a reward generated by feeding a discriminator a concatenation of 3 consecutive grey-scale image observations o_{t-1}, o_t, o_{t+1}. Also after running multiple tests we were not able to reproduce the good results shown in [TWS18b]. Based on visual inspection, we assume one reason being due to more "difficult" perspectives to learn, our discriminators were not able to extract enough information from the greyscale images.

Discriminator structure proposed by *Torabi et al.* [TWS18b] with RGB images. Being not able to find a working solution with greyscale images, we decided to test the discriminator architecture from [TWS18b] with the addition of using RGB-images instead of grey-scale images, so using a concatenated image array of three consecutive RGB-images of size $d \times d \times 9$. We managed to introduce some learning with this approach,

but still received unsatisfying results without finding a concrete reason. One possibly influencing factor may be the small differences when comparing consecutive timesteps.

As testing all this frameworks did not result in a satisfying performance, we decided to combine the idea of timewise shifted images by more than one timestep with the newer architecture of a DCGAN (which is also used in [TWS18b]) as this approach showed the best results in our conducted experiments. Further performance improvements were achieved through careful parameter tuning, adapting image sizes and improving the training methods, cf. Sections 6.1 & 6.3.4.

Similarly we also conducted tests with further perspective selection strategies. The following strategies were discarded after testing with the Point environment for different reasons:

Choosing a new discriminator only every x^{th} batch. Following the idea that a policy takes more than one batch till it actually learns something, we tried choosing a new discriminator only every 5^{th} batch. This did not lead to performance improvement and was therefore discarded.

Choosing a new discriminator uniformly. Randomly choosing from multiple discriminators leads in expectation to a uniform distribution. We still investigated, if the learning process was influenced by the local irregularities that a random selection strategy introduces and tested a uniform selection strategy. Formally that means choosing discriminators strictly alternatingly for each batch, so in case of 3 discriminators a sequence of $(0, 1, 2, 0, 1, 2, 0, ...)$. Experiments showed no significant difference in performance to the random strategy, so we discarded this strategy.

Choosing the discriminator only based on past information. While our proposed strategies (MaxProb, MinProb, UCB) do use also current information provided by the discriminators, it is also possible to decide on the perspective only based on past experience (with some exploration at the beginning). While for some runs, we managed to achieve similar performance as for the already proposed strategies, quite often we experienced being stuck with the choice of perspective for the complete training's process after exploration. Looking more closely into the results, we found out that this is strongly related to the constantly changing reward function provided by discriminators. As novice and discriminator do not necessarily learn at the same speed in each iteration, it can happen that a discriminator is not able to outperform the initially achieved rewards in

the exploration phase during the whole training but still performs well. Due to this issue we discarded this strategy.

6.3. Factors for Evaluation

6.3.1. Perspectives

In our work, we try to improve performance by looking at the problem from different perspectives - represented through images recorded from different camera angles. We split them in 3 different categories: Perspectives where supposedly all relevant information for learning a task is visible, perspectives with a a restricted view of the environment dynamics, and perspectives without information.

Full information: Original/Baseline perspective. This perspective provides presumably as much information as possible to understand the environment dynamics. This means looking at it as a human no relevant aspects, in our case movements in different axes, are hidden. Named original perspective, it mostly uses the viewpoint provided by default by predefined reinforcement learning environments. As our goal is to investigate the ability of strategies to deal with multiple perspectives that show partial information, we use this perspective for baseline comparisons.

Partial information: Side/Top/Front perspective. When imitating an expert, we are not always provided with one viewpoint that makes it possible to follow all environment dynamics. We therefore look at perspectives that partly show changes between the image observations, but do not display those completely. This may be to look only at one dimension in the case of a 2D environment, see Figure 6.1, or to change the camera angle in a way that relevant changes in the image are not clearly visible anymore.

No information: Non-informative perspective. The non-informative perspective does not provide any information relevant for learning (as this can also happen in real life). To represent a more realistic scenario, we do not just show a black screen in general, but look in a direction were we do not see anything relevant, like for the Hopper only the surroundings, but not the hopping leg.

6.3.2. Baselines

We compare our results to different baselines:

Optimal: Expert policy. As our algorithm tries to create similar image observations as the expert (and hence also performing similar actions), by construction the expert policy is the upper bound that a novice can achieve, except from outperforming it by chance. In our experiments we use the average of the true reward over 1000 epochs received when following the expert policy.

Full information: Original perspective. In all our environments, it is possible to look at task from a perspective that provides the agent with all/most of what is happening when performing actions. We use this perspective as baseline. This baseline best describes the goal that is realistically to be reached or outperformed by our strategies. Relying on the same learning architecture, comparing the performance of strategies to the performance of using the original perspective best shows how well information can be extracted from multiple perspectives/if information extraction from multiple perspectives can outperform one single perspective. Worse performance than the expert policy of this baseline reflects therefore rather a non-optimal general framework and not necessarily problems with our idea of multiple perspectives.

Partial information. We try to show that it is possible to extract relevant information from different perspectives, especially when some relevant factors are not included in all provided perspectives. This means using multiple perspectives that include partial information - with a good strategy - should provide us with better results than using just one of those perspectives. Hence we propose as lower baseline, that should outperform an unlearned policy, training on only one partial information perspective perspective. We choose the perspectives shown in (b) in the respective environment Figures 6.1, 6.3 and 6.4.

Unlearned policy. One of the most basic goal when performing a reinforcement learning problem is to outperform a random policy. Especially during parameter tuning we saw that a bad choice of parameters may even lead to a worse than random performance. As lower baseline we therefore use 10 randomly initialised unlearned policies and show the average when following each policy for 1000 epochs.

6.3.3. Evaluation Metrics

True reward. To evaluate the performance we use the non-discounted sum of the true reward that is given by a reinforcement learning reward signal. As we are only using

finite episodes, we decided on not using any discount rate for evaluation - a factor of 0.99 is used for policy training.

GAN reward. Even though the GAN reward that we use for training the policy does not explicitly reflect the final performance of the novice policy, it gives us valuable insights on the balance between training of discriminators and policy and differences between discriminators dealing with different perspectives. This metric provides us with the reward $r^{\mathbf{D}_v}$ of the respectively chosen perspective in the novice policy training.

Normalised score. To better compare performance of different strategies, but also all 3 environments, we introduce a normalised scoring of the true reward. Concretely, this results in a minmax-scaling of the true reward with the expert performance representing the maximum and therefore receiving a score of 1 and the random policy performance representing the minimum represented by a score of 0. The achieved performance by our framework can then be seen as percentage of achieved expert performance, for example 0.7 referring to 70 % expert performance.

Percentage of chosen perspective. In each training batch of the novice policy a new perspective/discriminator which returns the reward is chosen - looking at the distribution of chosen discriminators in the whole process especially in combination with the GAN reward can give us valuable insights in the functionality of each strategy.

Although not explicitly stated in the reported experiments, during building our algorithm and hyper-parameter tuning, we also took into account the recorded losses during training of discriminators and novice policy.

6.3.4. Fine Tuning

Algorithm Parameters When testing our framework we came to the realisation that unfortunately it is not possible to generalise the parameter choice for multiple environments completely. Before running our final experiments, we therefore performed hyperparameter-tuning with the strategy MinProb and the 2 partial information and the non-informative perspective. Our selection of tuned parameters and the final choice is shown in Table 6.4.

- H **- Horizon:** The choice of this parameter was already discussed in section 6.1 and is influenced by the time an expert needs to reach a goal in the environment and the movement out of the picture in the case of the Hopper environment.

Table 6.4.: Algorithmic parameters per environment.

	Point	Reacher	Hopper
K **- number of iterations**	40	40	40
Q_p **- policy training epochs**	50	15	8
b_p **- batch size policy**	200	300	150
E_d **- episodes discriminator**	50	10	20
b_d **- batch size discriminator**	64	64	150
Δ **- image shift**	2	5	6
H **- horizon**	42	54	154
T_e **- available expert episodes**	50	50	50

- Δ **- Image Shift:** To being able to deduce the influence of an action on the environment taken by a policy, each discriminator receives a tuple of 2 images. Although a single action clearly influences the observation vector, this difference is not necessarily visible on an image. Depending on the environment, each action leads to bigger or smaller change in a rendering of an environment, this is reflected via the choice of Δ.

- **Sample Parameters** Q_p**,** b_p**,** E_d**,** b_d**:** A common difficulty when training GAN's is to find the balance between a generator (in our case a policy) and a discriminator [GPAM+14]. We tried to deal with this issue by finding a good balance between images shown to the discriminator E_d, the epochs of policy training Q_p and which batch size is used respectively b_d/b_p. Due to high runtime of each experiments and generally huge fluctuations in results, we decided on an exploratory approach to find good parameters - for example we stopped experiments that clearly showed unsatisfactory results already early on instead of performing a complete grid search.

- **Iterations** K**:** While we also tried different parameters for the number of iterations in the algorithm, in the end we agreed on an on average good value for all environments to have comparable graphics between all environments.

- **Available Expert Episodes** T_e**:** In the realistic scenario we want to cover, there is only a restricted amount of expert episodes available. We choose 50 as a basic value, but ran different experiments to judge the influence of this parameter, see Section 6.4.5.

Discriminator. In our first experiments we encountered the problem of bad performing discriminators independent of other chosen parameters or the discriminator architecture.

Hence we adapted our training algorithms with some tricks that have already proved successful when training GANs ([SGZ+16, RMC15]):

- **Initialising discriminators.** As we use a DCGAN, we decided to also initialise our discriminators as proposed in the initial paper [RMC15]. This includes using batchnorms, using LeakyReLu [MHN+13] and initialising all weights with a standard normal distribution.

- **Normalising input.** We first tried using the raw RGB input with values in the interval $[0, 255]$ and received very poor results. Referring again to the original DCGAN structure, the discriminator receives *tanh*-normalised input by the generator, normalising our input with the *tanh* function significantly improved our results [RMC15].

- **Using soft labels/label smoothing.** Neural networks trained with smoothed labels have been found to be less vulnerable to adversarial examples and showed better performance [SGZ+16, SVI+16]. Instead of just training the discriminators with labels 0 for expert images and 1 for novice images, we assign uniform distributed labels in the interval $[0, 0.2)$ and $[0.8, 1)$ to our images.

UCB - Parameter Selection. As can be seen in Section 5.2.6, the UCB strategy includes the parameter c that strongly influences the strategy regarding how much exploration is done. Concretely, with growing c, the strategy is more likely to explore rather than just choosing the discriminator returning the highest reward. We ran 5 runs on the Point environment for multiple values of c and decided to use c=10 as this seems to show the best performance taking into account final performance, stability and learning speed after 30 iterations cf. Figure 6.5.

6.4. Performance Evaluation

We conducted multiple experiments to investigate different factors of our proposed framework. Although, as stated before, internal parameters may vary due to differences in complexity of our environments, we show comparable results for our environments running 40 algorithm iterations on each environment. To investigate the general performance we carried out each experiment 10 times. However, due to instability issues of the algorithm (very common for GAN based algorithms see [GPAM+14]), it may happen that a policy learns nothing or even completely breaks down and shows worse results than following

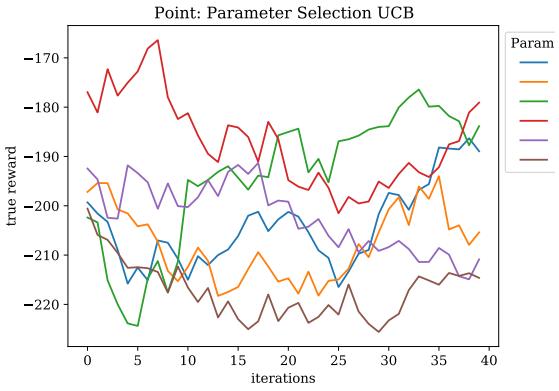


Figure 6.5.: Performance of the UCB strategy using different choices for the exploration parameter c averaged over 5 runs.

random actions. Therefore - if not stated otherwise - we show only the average of the best 5 runs (out of the conducted 10 runs) for each experiment in our visualisations. To show the variance between multiple runs some visualisations also include the standard error that occurs when averaging over multiple runs.

6.4.1. Proof of Concept

Although the general GAIL framework has already successfully worked in previous experiments (also in an LfO context [HE16, TWS18b]), this does not necessarily prove that learning is also possible in case of receiving rewards from multiple discriminators.

We tested a simple scenario to show that our algorithm is actually fully functioning, able to improve policy performance and to extract information even if not all relevant information can be found in an image: For each environment we take both perspectives with partial information and run the experiment with the strategy RAND. To get already a first understanding on how this strategy performs in comparison to baselines, we show also the expert performance and the performance of a random policy.

As can be seen in Figure 6.6, our algorithm has the ability to improve the performance on all given environments, but the random strategy seems not being able to lead to a good learning process for more complex 3D environments. Not only is there only a small improvement visible, but for the Reacher environment we do only get slightly better than taking random actions and the results are not continuously improving in the case of the Hopper environment.

Although these results may look dissatisfying, they are not voting against our approach

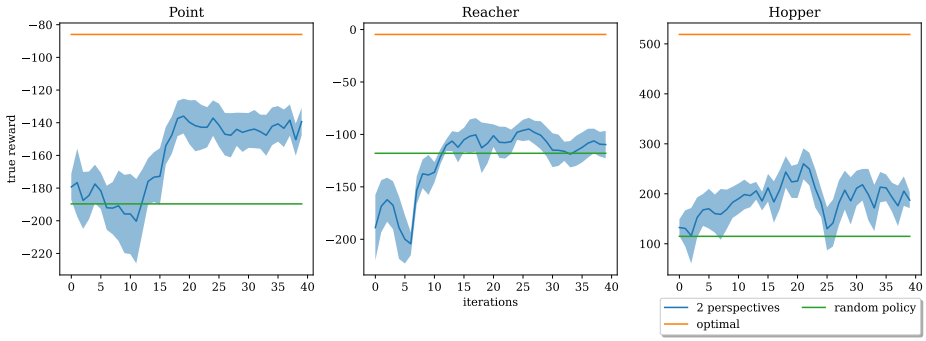


Figure 6.6.: Proof of Concept. We run our experiment extracting information from 2 perspectives including partial information with using the strategy RAND. A learning process is visible, however the achieved performance is not satisfying.

in general, as our goal is to find strategies that work better than choosing randomly.

6.4.2. Evaluation of Perspective Selection Strategies

Our core interest lies in the comparison of different strategies cf. Figure 6.7. Therefore, we compare the effectiveness of all proposed strategies concerning performance. For all experiments we used two perspectives including partial information and our introduced perspective including no information. Concretely this relates to perspectives (b)-(d) from Figures 6.1, 6.3 and 6.4.

Intuitively, our expectations before conducting the experiments were the following: As there is no distinction possible in the non-informative perspective, the discriminator receiving images from this perspective should not able to differentiate between expert and novice and will therefore always assign both policies a 50% probability, which translates to a reward signal of 0.69[5]. On contrary, the discriminators receiving perspectives with partial information should be able to differentiate between expert and novice and therefore return a lower reward than 0.69 to observations generated following a novice policy. Taking this into account, this would mean that the strategies MINPROB and UCB (except for some exploration) should always choose the perspective where the novice policy shows the most differing actions - visible through the provided image tuples. This theoretical analysis would assume MINPROB and UCB being the best performing strategies. MAXPROB on the other hand would mostly choose the non-informative strategy as it returns the highest reward signal and should assumably not learn well in comparison.

[5]The returned reward to the policy is $-\log(\mathbf{D}(o_t, o_{t+\Delta}))$.

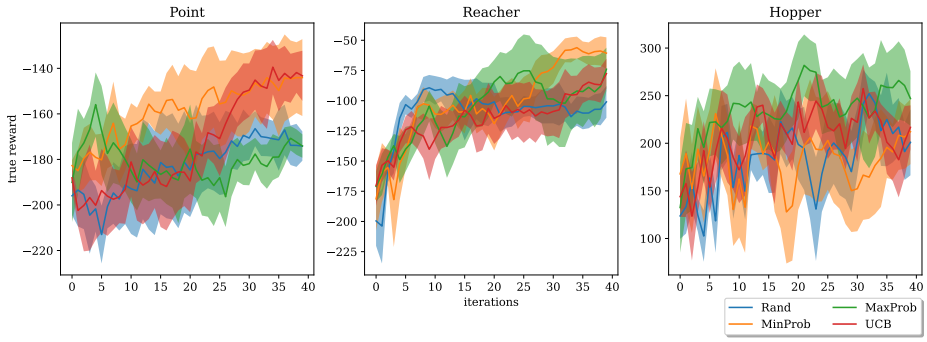


Figure 6.7.: Experiment on comparing different perspective selection strategies for policy training. We observe that all strategies show a learning process, however there is no strategy clearly outperforming the others looking at all 3 environments

Looking at the empirical results in Figure 6.7, we can see that this proves only partially true. Although for the Point environment our intuition is evidenced by the shown results (lower performance of RAND and MaxProb), looking at the Reacher and Hopper environment all strategies seem to show quite similar performance. Unfortunately we observe again unstable learning on the Hopper environment.

As our intuition was not fully supported by our experiments, we decided to more closely examine reasons for this results. Therefore, we examine more closely the distribution of chosen discriminators over all batches used for policy training (see line 28 in Algorithm 1). Looking at Table 6.5, we see for the Point environment the anticipated behavior: MaxProb strongly tends to choose the non-informative perspective, while MinProb and UCB focus on the other perspectives with UCB showing more exploration. However, the results for Reacher and Hopper do not support our hypothesis completely. Especially when learning to move the Reacher arm, the non-informative perspective was chosen surprisingly often for MinProb and UCB.

Table 6.5.: Percentage of chosen strategy for each discriminator: perspective 1 relates to (b), perspective 2 relates to (c), no - information relates to (d) in the respective environment Figures 6.1,6.3 and 6.4

	Point				Reacher				Hopper			
perspective	Rand	MinProb	MaxProb	UCB	Rand	MinProb	MaxProb	UCB	Rand	MinProb	MaxProb	UCB
perspective 1	32.4 %	40.7 %	15.0 %	42.3%	34.2 %	24.5 %	47.4 %	28.9 %	32.8 %	20.3 %	46.6 %	34.9 %
perspective 2	32.9 %	55.9 %	11.5 %	42.9 %	33.9 %	37.6 %	29.1 %	35.5%	33.0 %	54.9 %	17.8 %	42.0 %
no information	34.7 %	3.4 %	73.5 %	14.8 %	31.9 %	37.9 %	23.5 %	35.6 %	34.3 %	24.8 %	35.6 %	23.2 %

We investigated this further and found the following explanation. Exemplary, we show the chosen discriminator proportion in each (novice policy) training iteration for the best

run of the Reacher environment when using the UCB strategy and the achieved GAN reward during the training process in Figure 6.8. We can observe two different kinds of iterations by looking at the chosen discriminators per iteration. In some iterations, the algorithm uses mostly perspectives that introduce additional information, in other iterations only the non-informative perspective, depicted in green, is chosen[6].

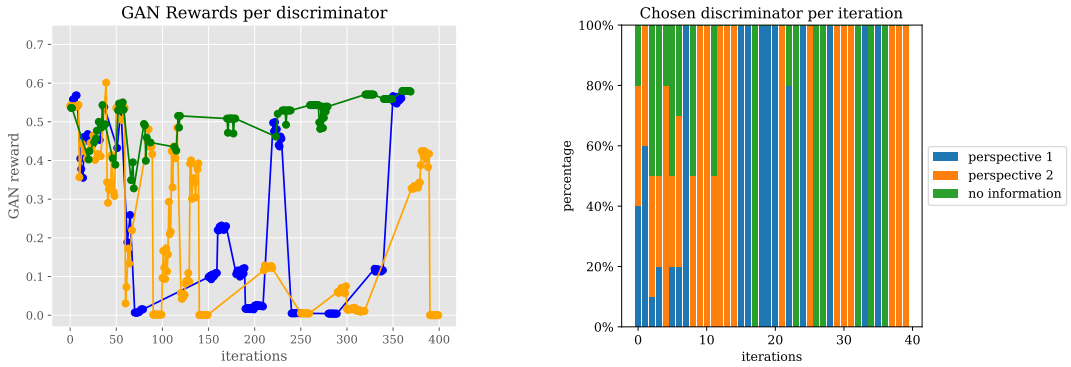


Figure 6.8.: Analysis of chosen discriminators in the best iteration on the Reacher environment using the strategy UCB. The left figure represents all used rewards (average over one batch) in novice policy training which means in this case 10 batches for each of the 40 iterations of the algorithm. The right sides provides the consolidated view on how often each discriminator was chosen proportionally per iteration.

This means that in some iterations the policy learns faster than the discriminator, more concretely, even after another training iteration of the discriminators, the novice policy still manages to trick them. After a learning-wise useless iteration for the policy, the discriminators learn enough such that the novice is again shown the informative perspectives. Having this in mind, we tested different hyper parameter scenarios to give the discriminators more time to learn. Unfortunately, we did not manage to find a better balance and ended up with scenarios of over-performing discriminators, such that the novice policy was not able to learn anything at all.

Another factor we identified which could influence our performance, is the label order in discriminator learning. Remembering our Algorithm 1, we train each discriminator alternatingly with expert and novice batches, starting always with an expert batch and ending with a novice batch. Thinking about the completely analogous non-informative

[6]If we would average here again over 5 runs, this effect would not be visible as the "non-informative" iterations do not necessarily occur at the same time in various runs. Therefore we show here only the best run according to the final performance. However, tests with other runs/strategies showed similar results.

perspective for expert and novice, this would mean that the discriminator associated with this perspective always leans more towards assigning a novice label, as this is the last training impulse it is trained with. Looking at the GAN rewards per discriminator in Figure 6.8 this proves true. Assigning an image tuple 50 % probability would result in a reward of about 0.69 as $-log(0.5) \approx 0.69$. Our figure shows that every time the nonsense perspective, represented in green, is chosen, its value is clearly below 0.69. As the problem seems to be connected with the non-informative perspective, we decided on investigating two ideas further:

- Under the assumption that we would like to exploit the capabilities of the strategies UCB and MinProb, we developed the idea of switching the discriminator learning process in a way, that each discriminator is shown first a novice and then an expert batch. This assumably leads to higher rewards for less informative perspectives and leads to choosing those less often in case of UCB and MinProb.

- As shown, it seems problematic for our algorithm to deal with a complete non-informative perspective. While we would like to avoid to actually choose a perspective beforehand, humans still could most likely recognize and exclude completely nonsense perspectives beforehand. Therefore, we also had a look at the same experiment using only partial information perspectives for both discriminator learning methods (being shown first expert/novice batches).

For further visualisations of these proposed experiments similar to Figure 6.7 and Table 6.5, we refer to the appendix and show here again the exemplary experiment for the Reacher environment when training first on novice labels and a comparison of the final performance of all four proposed experiments after 40 iterations.

Looking at Figure 6.9, we can see that the GAN rewards of the discriminator learning only based on the non-informative perspective are now on a higher level as we intended. On the other hand, the percentage of the chosen discriminator still shows many iterations, where the algorithm never chooses a useful perspectives. A possible solution to this issue could be using a different set of parameters (different balance between discriminator and policy learning, shorter/longer iterations,...), but unfortunately also here even after conducting multiple experiments with different parameters we did not find better hyper parameters.

In Figure 6.10 we give an overview of the previously proposed scenarios with 2/3 perspectives and training the discriminator first with expert/novice labels. To being able to show all experiments simultaneously, we decided here on the already described normalised score to show the performance in comparison to an expert and an unlearned

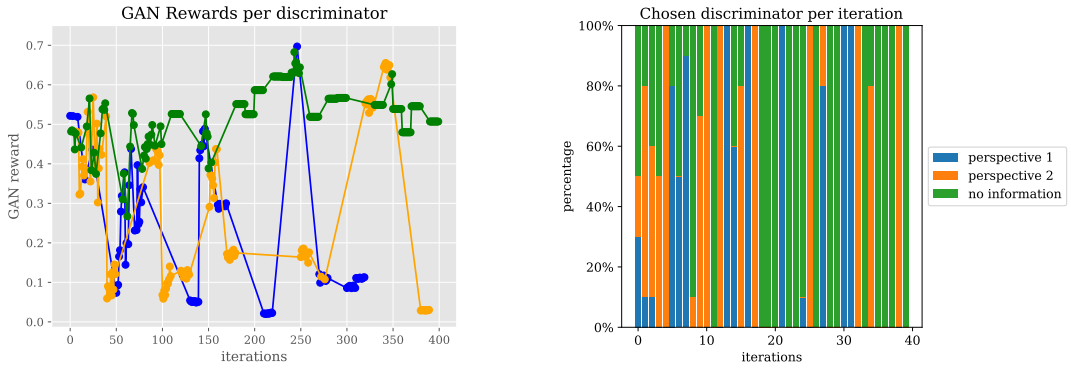


Figure 6.9.: Analysis of chosen discriminators of the best iteration on the Reacher environment using strategy UCB by first training on novice labels in discriminator training. Again the left figure represents all used rewards in novice policy training, while the bar chart shows the proportion of the choice of each discriminator per iteration.

policy as baseline. Furthermore we also include the average performance after 40 iterations of a policy trained on the original perspective as baseline.

Although our algorithm reaches at its best only 60 % of the expert performance, the figure also shows, especially when looking at the baselines where all information is given for learning, that the restrictions of the framework do not lie in the idea of using multiple perspectives, but rather in the concrete implementation of the algorithm. Particularly the UCB strategy manages to outperform both baseline methods for all proposed scenarios and environments. Nevertheless there remain some unclear phenomena, one being the huge differences in similar situations such as the MinProb strategy for the Reacher environment.

Moreover, one may assume to achieve better performance when training on only two informative perspectives as the density of information is higher. This seems not always to be the case. Let's investigate this further and look at the distributions of the GAN reward under the UCB strategy when applied on the Reacher environment, cf. Figure 6.11. We can clearly observe a wider distribution of the GAN reward values when learning on 2 perspectives. This leads to the conclusion that the non-informative perspective acts as some kind of boundary on the GAN reward, which may regularise the learning process in the case of a strong policy in comparison to the discriminators. Another possible factor may be that we chose the parameters of the algorithm based on 3 perspectives and applied those in the 2-perspective scenario.

In an optimal plot all 3 perspectives should show a mean of 0.69 with a very small

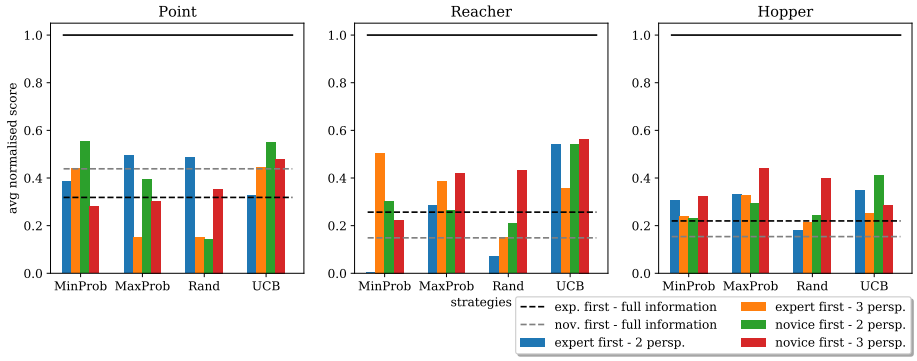


Figure 6.10.: Comparing the final performance after 40 runs of our strategies in 4 different training's settings. We compare achieved results with baselines including a random policy (score 0), the expert policy (score 1) and 2 baselines that train on our full information perspective.

range and interquartile range for the non-informative perspective as the discriminator should not be influenced by the training. The partial information perspectives, while also symmetrically distributed around the mean, would have a larger (interquartile) range representing the learning of the novice policy in each iteration. Closest to optimal performance (as indicated also in earlier experiments) comes our experiment with 3 perspectives when choosing the novice first.

As the policy is now presented with an easier goal of not having to cope with a non-informative perspective, adapting the sample parameters accordingly (and/or c in the UCB strategy) would probably lead to a better performance.

6.4.3. Comparison to Baselines

To understand the significance of the achieved results for applying various strategies, we compare the respective best strategy (highest reward after 40 iterations) for 3 perspectives to our baselines, cf. Section 6.3.2. As can be seen in Figure 6.10, this occurred for all environments when training first on novice labels. Concretely this means we chose UCB for Point and Reacher environment and MaxProb for the Hopper environment. Although the expert describes the absolute maximum that can be achieved concerning performance, we consider our results to be good when we get close to or surpass our full information baseline as this means information extraction is as efficient done as when provided a single source with complete information.

Looking at Figure 6.12, we see that our strategy outperforms the partial information as

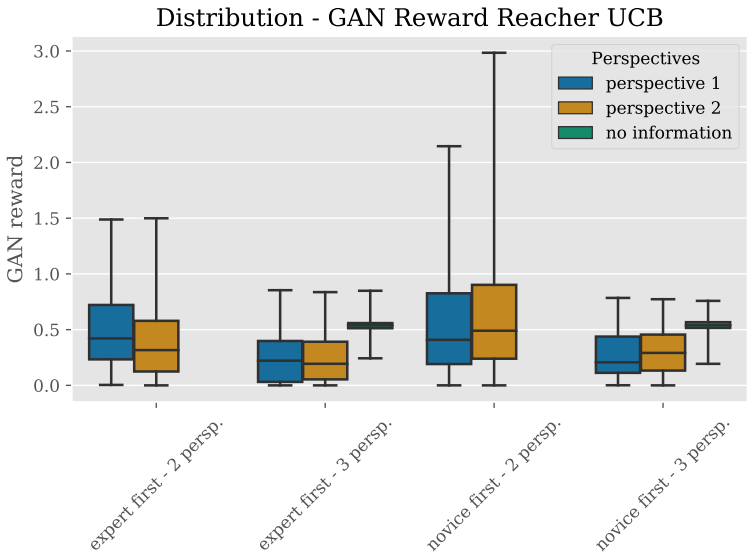


Figure 6.11.: Distribution of the GAN reward in the novice training process for the Reacher environment when applying the strategy UCB. We clearly see a wider distribution when training only with 2 perspectives, which leads to the assumption of the non-informative perspective acting as some kind of boundary.

well as the full information baseline in all cases, most clearly for the Reacher environment[7]. This means we managed to reach one of our main goal - extracting efficiently information from multiple perspectives. Each strategy shown in this last figure even had to deal with the additional obstacle of differentiating the useful perspectives from the useless one provided.

6.4.4. Evaluation of Performance versus Stability

Although the already shown results prove that our novice algorithm is able to learn, the results are generally still significantly worse than the expert policy. To understand if there is a general limit of achievable performance or the shown graphs reflect the missing stability - till now we mostly showed averaged results - we also would like to present the already shown graphs referring only to the best out of all 10 runs respectively. Looking at Figure 6.13, we see not only the expected improvements but also that for Point and Reacher environment we receive good results of over 80 % using random and expert policy as lower/upper limits. Unfortunately, again the novice policy trained to act in

[7]For better comparison we also train baselines with novice labels first if applicable.

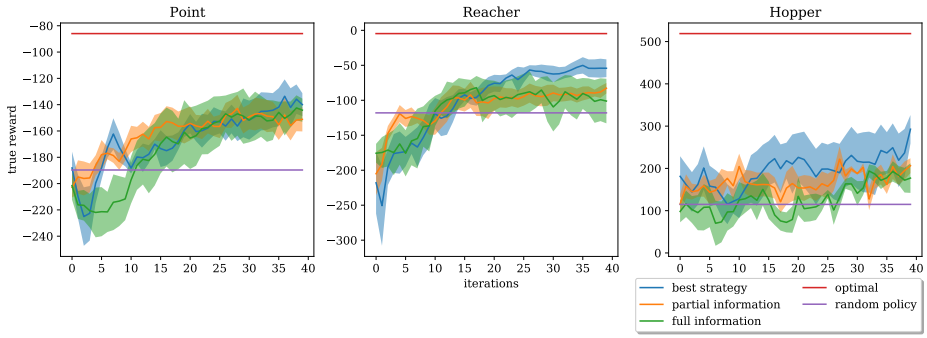


Figure 6.12.: Comparison of best performing strategy to baselines. We chose the best performing strategies per environment Point - UCB, Reacher - MaxProb, Hopper - UCB and compare them to our baselines. We can observe that for all environments we manage to outperform the full information baseline as well as the random policy and the partial information baseline.

the Hopper environment stays behind as is also shown by the best strategy being the random strategy. Looking at all results we achieved for the Hopper environment till now - showing some, but not a satisfying learning process - we assume that we did not find a good hyper parameter configuration for this environment. Another reason could be that this more complex environment would even need more information in terms of larger images to improve its learning process.

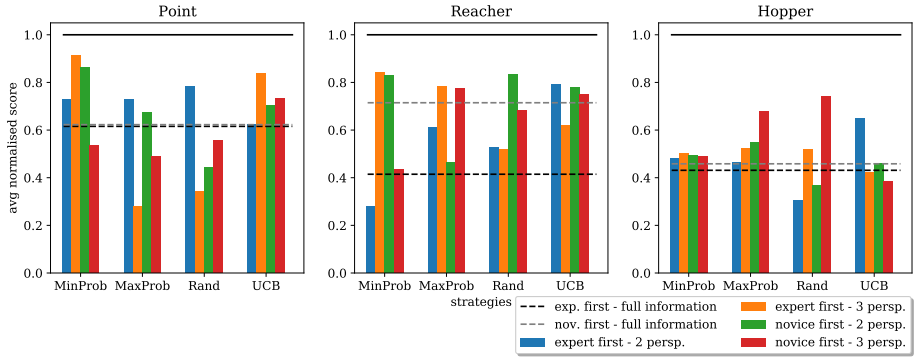


Figure 6.13.: Comparing the final performance for the best out of 10 runs respectively. We compare achieved results with baselines including a random policy (score 0), the expert policy (score 1) and 2 baselines that train on our full information perspective. For the Point and the Reacher environment the results show that we manage good performance on single runs, but taking into account other results reliable stability is not given.

6.4.5. Experiment on the Influence of the Amount of Expert Data

Up till now, to understand the functionality of our proposed framework, we provided our algorithm with 50 episodes of expert data for discriminator training. In a real world scenario, it might not be possible to have endless access to access data, even 50 complete episodes may not be available. Therefore we investigated the influence of the amount of provided expert observations by looking at the cases of unrestricted access to expert data which means expert observations are created always newly if needed and a stronger restriction of only 10 available expert episodes again using the best strategy per environment. Comparison of the learning graphs in Figure 6.14 shows that the environments can also deal well with a small amount of expert data.

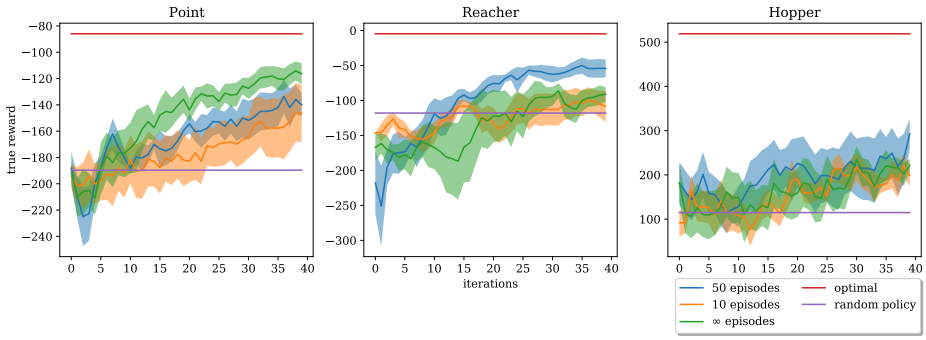


Figure 6.14.: Comparison of the influence of the amount of expert data using the respective best strategy. In the conducted experiments the available amount of expert observation episodes to train the discriminators with differs between 10, 50 and an endless amount of episodes (meaning observations are always created newly if needed).

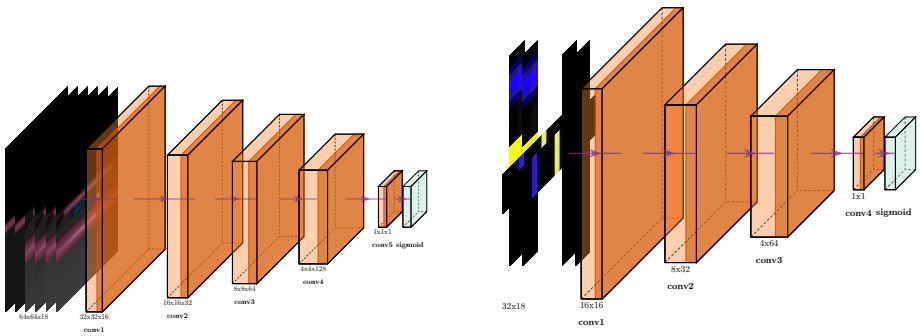
Although only the Point environment depicts a clear positive dependency of the final performance on the number of observations, looking at the standard error in Table 6.6, also the Reacher environment benefits by showing a smaller variability between runs when providing more expert observations. Reflecting on the functionality of the environments, we can find a possible reasoning behind the small influence that the number of expert observations has on the Hopper environment. While for Point and Reacher, the goal can be positioned differently in each epoch, the Hopper leg has always the non-changing same goal of moving to the right.

Table 6.6.: Influence of the provided amount of expert data - standard error depending on number of provided expert episodes

	Point	**Reacher**	**Hopper**
10 episodes	21.65	22.38	23.98
50 episodes	8.71	12.75	33.64
∞ episodes	8.08	10.24	14.37

6.4.6. Comparison to a Super-Discriminator

The core part of our idea underlies the assumption that a good strategy - defined by us - can help the novice policy to learn efficiently. However, the flexibility of neural networks also provides us with the possibility of training one discriminator on multiple perspectives simultaneously by adapting its architecture. Therefore we compared our best strategy to a "Super"-Discriminator, that uses a stacked array of all partial perspectives as input and hence receives all provided images simultaneously. The basic structure of this discriminator (2 variants, one for 2D images for Hopper/Reacher, one for the 1D images of the Point environment) - again an adaption of our originally proposed DCGAN framework - can be found in Figure 6.15.



(a) Super-Discriminator for 2D images (Reacher, Hopper)

(b) Super-Discriminator for 1D images (Point)

Figure 6.15.: Adapted discriminator architectures of the Super-Discriminator. As the Super-Discriminator has as input 3 image pairs it is necessary to slightly adapt the discriminator architecture.

Looking at the results in Figure 6.16, the Super-Discriminator manages to outperform our strategy for both Point and Hopper environment and shows comparable performance for the Reacher environment. Taking into consideration the run time for training, also here the Super-Discriminator manages to outperform our best strategy for one complete

training run. For our concrete scenario of environments and using 3 perspectives, this leads to the conclusion that the overhead of using multiple discriminators and strategies does not prove value in terms of performance and run time. While we did not find a configuration of our approach that is able to surpass a Super-Discriminator, our flexible approach can likely be improvable in future works to surpass or match the Super-Discriminators as discussed in the conclusion.

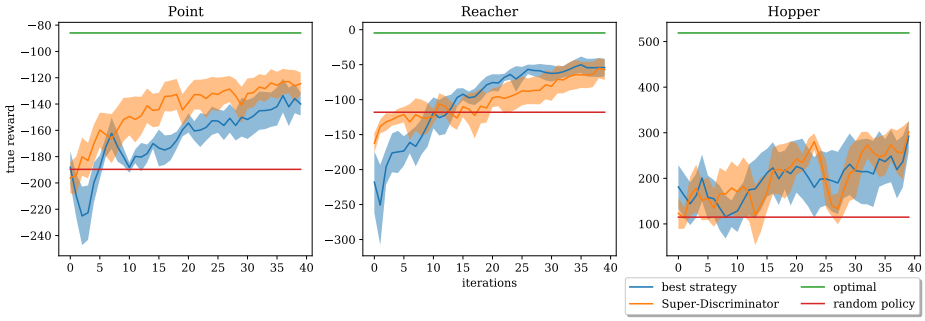


Figure 6.16.: Comparison of our best strategy to a Super-Discriminator, which receives always all perspectives simultaneously.

7. Conclusions

In this thesis we developed and studied the approach of a strategic expert perspective choice to improve the performance of imitation learning, especially in a situation with restricted access to information. We provided an extended summary on background focusing on the GAIL framework [HE16], particularly in the context of learning from observation with no access to expert actions. Furthermore, we examined various imitation learning methods with focus on similar frameworks as ours (GAIL based), but also investigated closely other methods that rely on images as available expert demonstrations. While we found many approaches that address different problems in imitation learning, like embodiment mismatch [SAS17] or compounding error [RB10], up to our knowledge there is no approach trying to take advantage of multiple expert perspectives simultaneously.

We motivated the idea of multi-perspective imitation learning and introduced the idea of using strategies to choose the most informative perspective when training a policy. Practically, we proposed a GAIL-based framework that relies on the idea of multiple discriminators which try to differentiate between observations from provided expert's demonstration and a novice policy. Hereby each discriminator deals with observations from exactly one perspective. We train the novice policy by defining multiple strategies that define which reward signal - from which discriminator - the policy receives.

The framework and adapted reinforcement learning environments that fit to our purpose were implemented in Python based on the libraries Pytorch [PGM+19] and Garage [gc19] and various experiments were conducted to assess our framework. We hereby also focused on implementing the complete framework as flexible and comprehensible as possible to give others the possibility to easily extend it and test different configurations.

Although we struggled with a general problem of GAIL - instability [GPAM+14] - our experiments showed that it is generally possible to extract information from different perspectives even if each perspective only provides parts of relevant information. We assessed the performance of our proposed strategies with the expectation that using the reward signals from discriminators that currently are assigning a low reward to observations generated by the novice policy would yield the best learning process. Although the behaviour is not exactly as expected (for some environments strategies not following this

idea show better results), using a UCB based strategy that chooses the discriminator assigning the policy the lowest performance next to some exploration, shows mostly good results. By adapting our implementation to our findings we managed to even surpass the final performance of a baseline providing all relevant environment information in some environments. We also focused on the influence of the amount of available expert data and showed that our framework is applicable in a realistic scenario of restricted expert data. Furthermore, restricting the amount of provided expert's demonstrations did not show a significant reduction of performance in comparison to endless available data. We concluded our experiments by a comparison to using all perspectives simultaneously to see the relevance of our strategies concerning the factor of extracting **relevant** information.

While we carried out a vast amount of experiments, there still remain many open questions and research directions based on our general idea. Although we base our theoretical foundation on already proven concepts of GAIL in a learning from observations context [TWS18b] and the usage of multiple discriminators shown in [DGM16], we focus on the experimental results of our strategy-focused framework and omitted a thorough theoretical analysis.

Furthermore, it is possible to extend our framework in various ways. Although we focus on predefined strategies, an extension would be to include the strategies directly in the framework meaning they would be defined by the machine without human intervention. This could be done through introducing an additional value for the action space of each environment and therefore letting the policy decide directly. An even more advanced approach could be a multi-agent framework with a second reinforcement learning agent taking on the role of the strategy.

Our novice currently has access to all perspectives, trains each discriminator in every training step and also the strategies use all perspectives to decide which to show the novice policy. As rendering the image representations of the environment is computationally expensive, reducing the number of needed images would improve the efficiency. Possible adaptations of our framework to reduce the amount of needed images include strategic training of discriminators meaning also basing the discriminator training on a strategy and development of well-performing strategies that rely only on already received rewards - meaning observations would just be fed through one discriminator and not all when deciding which discriminator to use in policy training.

Bibliography

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[AD21] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[BAM16] Nir Baram, Oron Anschel, and Shie Mannor. Model-based adversarial imitation learning. *arXiv preprint arXiv:1612.02179*, 2016.

[BB14] Michael Bloem and Nicholas Bambos. Infinite time horizon maximum causal entropy inverse reinforcement learning. In *53rd IEEE Conference on Decision and Control*, pages 4911–4916, 2014.

[BCP+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[BK+96] Paul Bakker, Yasuo Kuniyoshi, et al. Robot see, robot do: An overview of robot imitation. In *AISB96 Workshop on Learning in Robots and Animals*, volume 5. Citeseer, 1996.

[BSBM22] Damian Boborzi, Christoph-Nikolas Straehle, Jens S Buchner, and Lars Mikelsons. Imitation learning by state-only distribution matching. *arXiv preprint arXiv:2202.04332*, 2022.

[BUAC02] Darrin C Bentivegna, Ales Ude, Christopher G Atkeson, and Gordon Cheng. Humanoid robot learning and game playing using pc-based vision. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 3, pages 2449–2454. IEEE, 2002.

[CKA20] Ching-An Cheng, Andrey Kolobov, and Alekh Agarwal. Policy improvement via imitation of multiple oracles. *Advances in Neural Information Processing Systems*, 33:5587–5598, 2020.

[CWD+18] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[DGM16] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016.

[DHJL19] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[FLL17] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[FS18] Wael Farag and Zakaria Saleh. Behavior cloning for autonomous driving using convolutional neural networks. In *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 1–7, 2018.

[FYZ+17] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on robot learning*, pages 357–368. PMLR, 2017.

[GB12] Daniel H Grollman and Aude G Billard. Robot learning from failed demonstrations. *International Journal of Social Robotics*, 4(4):331–342, 2012.

[gc19] The garage contributors. Garage: A toolkit for reproducible reinforcement learning research. `https://github.com/rlworkgroup/garage`, 2019.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[GZG20] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pages 1259–1277. PMLR, 2020.

[HCB+18] Peter Henderson, Wei-Di Chang, Pierre-Luc Bacon, David Meger, Joelle Pineau, and Doina Precup. Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[HE16] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[HGEJ17] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[HLMS19] Corentin Hardy, Erwan Le Merrer, and Bruno Sericola. Md-gan: Multi-discriminator generative adversarial networks for distributed datasets. In *2019 IEEE international parallel and distributed processing symposium (IPDPS)*, pages 866–877. IEEE, 2019.

[INS02] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1398–1403. IEEE, 2002.

[JBvdS20] Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Strictly batch imitation learning by energy-based distribution matching. *Advances in Neural Information Processing Systems*, 33:7354–7365, 2020.

[Joy11] James M Joyce. Kullback-leibler divergence. In *International encyclopedia of statistical science*, pages 720–722. Springer, 2011.

[JSA+21] Andrew Jaegle, Yury Sulsky, Arun Ahuja, Jake Bruce, Rob Fergus, and Greg Wayne. Imitation by predicting observations. In *International Conference on Machine Learning*, pages 4665–4676. PMLR, 2021.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KCS21] Rahul Kidambi, Jonathan Chang, and Wen Sun. Mobile: Model-based imitation learning from observation alone. *Advances in Neural Information Processing Systems*, 34:28598–28611, 2021.

[KCTD18] Daiki Kimura, Subhajit Chaudhury, Ryuki Tachibana, and Sakyasingha Dasgupta. Internal model from observations for reward shaping. *arXiv preprint arXiv:1806.01267*, 2018.

[KNT19] Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*, 2019.

[LCH+06] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[LGAL18] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125. IEEE, 2018.

[LHI20] Wanxiang Li, Chu-Hsuan Hsueh, and Kokolo Ikeda. Imitating agents in a complex environment by generative adversarial imitation learning. In *2020 IEEE Conference on Games (CoG)*, pages 702–705. IEEE, 2020.

[LHXZ20] Minghuan Liu, Tairan He, Minkai Xu, and Weinan Zhang. Energy-based imitation learning. *arXiv preprint arXiv:2004.09395*, 2020.

[LSE17] Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. *Advances in Neural Information Processing Systems*, 30, 2017.

[LSSL21] Youngwoon Lee, Andrew Szot, Shao-Hua Sun, and Joseph J Lim. Generalizable imitation learning from observation via inferring goal proximity. *Advances in Neural Information Processing Systems*, 34:16118–16130, 2021.

[LZBY20] Yanli Liu, Kaiqing Zhang, Tamer Basar, and Wotao Yin. An improved analysis of (variance-reduced) policy gradient and natural policy gradient methods. *Advances in Neural Information Processing Systems*, 33:7624–7636, 2020.

[MHN+13] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA, 2013.

[MKKY18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[MTT+17] Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

[NB16] Ann Nowé and Tim Brys. A gentle introduction to reinforcement learning. In *Scalable Uncertainty Management: 10th International Conference, SUM 2016, Nice, France, September 21-23, 2016, Proceedings 10*, pages 18–32. Springer, 2016.

[NCA+17] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2146–2153. IEEE, 2017.

[NR+00] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[PDVH+22] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sas«ank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[PKT+18] Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. *arXiv preprint arXiv:1810.00821*, 2018.

[PML+18] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 2050–2053, 2018.

[RB10] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[Rus98] Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998.

[SAS17] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. *arXiv preprint arXiv:1703.01703*, 2017.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Sch96] Stefan Schaal. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

[SGZ+16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[SHM+16] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016.

[SLA+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[SLC+18] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[SRSE18] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. *Advances in neural information processing systems*, 31, 2018.

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[TCS20] Voot Tangkaratt, Nontawat Charoenphakdee, and Masashi Sugiyama. Robust imitation learning from noisy demonstrations. *arXiv preprint arXiv:2010.10181*, 2020.

[TET12a] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[TET12b] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[TGWS19] Faraz Torabi, Sean Geiger, Garrett Warnell, and Peter Stone. Sample-efficient adversarial imitation learning from observation. *arXiv preprint arXiv:1906.07374*, 2019.

[TWS18a] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[TWS18b] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[TWS19a] Faraz Torabi, Garrett Warnell, and Peter Stone. Adversarial imitation learning from state-only demonstrations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2229–2231, 2019.

[TWS19b] Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

[WF18] Yu-chen Wu and Jun-wen Feng. Development and application of artificial neural network. *Wireless Personal Communications*, 102(2):1645–1656, 2018.

[Yan20] Xin-She Yang. *Nature-inspired optimization algorithms.* Academic Press, 2020.

[YFX+18] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

[ZRN+21] Konrad Zolna, Scott Reed, Alexander Novikov, Sergio Gomez Colmenarejo, David Budden, Serkan Cabi, Misha Denil, Nando de Freitas, and Ziyu Wang. Task-relevant adversarial imitation learning. In *Conference on Robot Learning*, pages 247–263. PMLR, 2021.

A. Appendix

A.1. Additional Experimental Results

While the figures for thoroughly discussed results are shown in our evaluation in Section 6.4, we would like, for completeness, to also show the additional figures of only briefly mentioned experiments. Concretely, the here shown figures depict in the main section not shown experiments testing the influence of which label the discriminator receives first for 2 and 3 perspectives.

A.1.1. Experiment: Discriminator training starting with novice labels, 3 perspectives

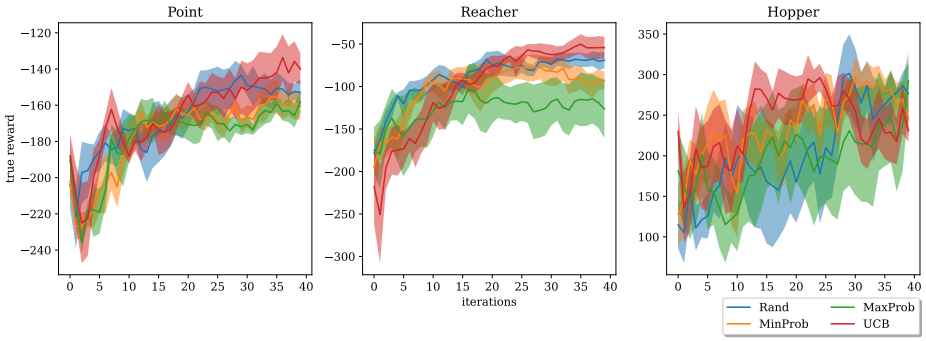


Figure A.1.: Experiment on comparing different strategies for policy training when training discriminators first on novice images on 3 perspectives

Table A.1.: Percentage of chosen strategy for each discriminator - starting with novice - 3 perspectives: perspective 1 relates to (b), perspective 2 relates to (c), no - information relates to (d) in the respective environment Figures 6.1, 6.3 and 6.4

	Point				Reacher				Hopper			
perspective	RAND	MINPROB	MAXPROB	UCB	RAND	MINPROB	MAXPROB	UCB	RAND	MINPROB	MAXPROB	UCB
perspective 1	32.8 %	50.3 %	24.3 %	40.7%	33.3 %	30.8 %	43.8 %	23.0 %	34.0 %	27.5 %	31.3 %	32.0 %
perspective 2	33.7 %	45.0 %	16.2 %	40.0 %	34.0 %	33.6 %	32.8 %	29.8%	33.6 %	46.8 %	13.2 %	36.0 %
no information	33.5 %	4.7 %	59.5 %	19.3 %	32.7 %	35.6 %	29.4 %	47.2 %	32.4 %	25.7 %	55.5 %	32.0 %

A.1.2. Experiment: Discriminator training starting with novice labels, 2 perspectives

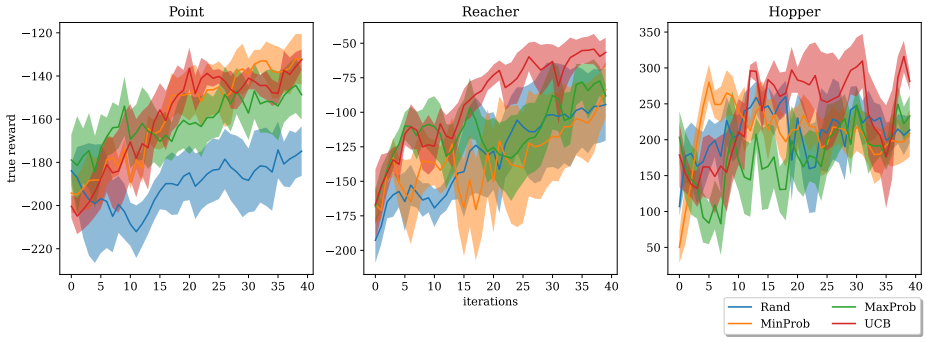


Figure A.2.: Experiment on comparing different strategies for policy training when training discriminators first on novice images on 2 perspectives

Table A.2.: Percentage of chosen strategy for each discriminator - starting with novice - 2 perspectives: perspective 1 relates to (b), perspective 2 relates to (c) in the respective environment Figures 6.1, 6.3 and 6.4

	Point				Reacher				Hopper			
perspective	RAND	MINPROB	MAXPROB	UCB	RAND	MINPROB	MAXPROB	UCB	RAND	MINPROB	MAXPROB	UCB
perspective 1	50.4 %	52.0 %	51.9 %	47.8%	49.3 %	39.7 %	62.0 %	47.9%	49.4 %	27.8 %	76.9 %	37.8 %
perspective 2	49.6 %	48.0 %	48.1 %	52.2 %	50.7 %	60.3 %	38.0 %	52.1%	50.6 %	72.2 %	23.1 %	62.2 %

A.1.3. Experiment: Discriminator training starting with expert labels, 2 perspectives

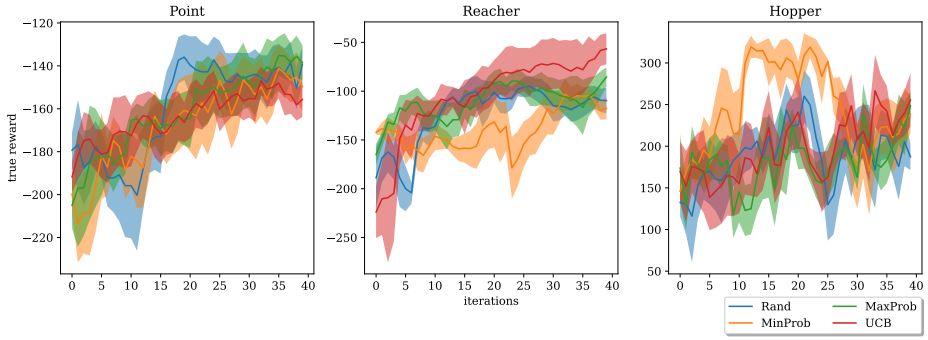


Figure A.3.: Experiment on comparing different strategies for policy training when training discriminators first on expert images using 2 perspectives

Table A.3.: Percentage of chosen strategy for each discriminator - starting with expert labels - 2 perspectives: perspective 1 relates to (b), perspective 2 relates to (c) in the respective environment Figures 6.1, 6.3 and 6.4

	Point				Reacher				Hopper			
perspective	RAND	MINPROB	MAXPROB	UCB	RAND	MINPROB	MAXPROB	UCB	RAND	MINPROB	MAXPROB	UCB
perspective 1	50.6 %	48.5 %	46.3 %	53.7%	47.7 %	32.3 %	53.5 %	44.0 %	48.2 %	19.6 %	68.8 %	38.0 %
perspective 2	49.4 %	51.5 %	53.7 %	46.3 %	52.3 %	67.7 %	46.5 %	56.0%	51.8 %	80.4 %	31.2 %	62.0 %