Karlsruher Institut für Technologie

Autonomous Learning Robots Lab

Learning Versatile Skills using Reinforcement Learning

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik des
Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Mevlüt Onur Celik

geb. in Erlenbach am Main

Tag der mündlichen Prüfung: 24. Oktober 2025
1. Referent: Prof. Dr. Gerhard Neumann
2. Referent: Prof. Dr. Dieter Büchler

Abstract

Very early in life, humans learn how to complete a task in various ways. For instance, we are not restricted to only using our right hand to grasp a mug at its handle, but can also do so with our left hand and grasp it at the mug's bottom or top part. This intuitive ability to solve a task in versatile ways is a key feature of why humans are highly adaptive to new situations and environments. Once the mug's handle is broken, we can still grasp it at a different place and continue using it.

Based on the way humans solve a task, a natural motivation is to equip robots with such versatile skills to enable higher flexibility and adaptability in the real world, in which the environment constantly changes. Especially in the context of reinforcement learning (RL), this versatile skill discovery is interesting as there is no human expert required. Yet, most known RL methods are based on Gaussian parameterized policies that can not represent multimodal distributions, a key feature for learning versatile skills. Therefore, this thesis proposes methods for equipping robots with versatile skills using reinforcement learning for Mixture of Experts (MoE) and diffusion policies. Both policy representations can represent multimodal distributions.

First, we propose a method for learning MoE policies in the episode-based maximum entropy RL (ERL) setting, where each expert is a contextualized motion primitive. The proposed objective allows each expert to shape its own curriculum by learning a per-expert context distribution such that the experts become an expert in their respective responsible context region. We then extend these MoEs to highly non-linear experts and energy-based per-expert models to enable more complex representations thereby reducing the number of experts required to successfully solve a task. Finally, we propose a step-based maximum entropy reinforcement learning (SRL) algorithm that can train a diffusion-based policy to solve high-dimensional control tasks. Training diffusion policies are a natural alternative to MoEs as they have proven to be very effective in representing multimodal distributions while stable training. Yet, they are not straightforward to apply in the SRL setting because they do not have a tractable marginal entropy.

Zusammenfassung

Sehr früh im Leben lernen Menschen, eine Aufgabe auf verschiedene Arten zu lösen. Beispielsweise sind wir nicht darauf beschränkt, eine Tasse nur mit der rechten Hand am Griff zu greifen, sondern können dies auch mit der linken Hand tun und ihn an der Unterseite oder Oberseite greifen. Diese intuitive Fähigkeit, eine Aufgabe auf vielseitige Weise zu lösen, ist ein entscheidendes Merkmal dafür, warum Menschen sich so gut an neue Situationen und Umgebungen anpassen können. Selbst wenn der Griff des Bechers abbricht, können wir ihn an einer anderen Stelle greifen und weiter benutzen.

Ausgehend davon, wie Menschen Aufgaben lösen, liegt es nahe, Roboter mit solchen vielseitigen Fähigkeiten auszustatten, um eine höhere Flexibilität und Anpassungsfähigkeit in der realen Welt zu ermöglichen, in der sich die Umgebung ständig verändert. Gerade im Kontext des Reinforcement Learning (RL) ist die Entdeckung solcher vielseitiger Fähigkeiten besonders interessant, da kein menschlicher Experte benötigt wird. Die meisten bekannten RL-Methoden basieren jedoch auf gaußförmig parametrierten Policies, die keine multimodalen Verteilungen darstellen können – ein zentrales Merkmal zum Erlernen vielseitiger Fähigkeiten. Daher schlägt diese Arbeit Methoden vor, um Roboter mit vielseitigen Fähigkeiten auszustatten, wobei Reinforcement Learning mit Mixture of Experts (MoE) und diffusionsbasierte Policies zum Einsatz kommt. Beide Repräsentationen können multimodale Verteilungen abbilden.

Zunächst schlagen wir eine Methode zum Erlernen von MoE-Policies im episodenbasierten Maximum Entropy RL (ERL) Fall vor, bei der jeder Experte eine kontextualisierte Motion Primitive darstellt. Das vorgeschlagene Optimierungsziel erlaubt es jedem Experten, sein eigenes Curriculum zu gestalten, indem eine kontextspezifische Verteilung pro Experte gelernt wird, sodass jeder Experte auf den für ihn zuständigen Kontextbereich spezialisiert wird. Anschließend erweitern wir diese MoEs um hochgradig nichtlineare Experten und Energy Based Modelle pro Experte, um komplexere Repräsentationen zu ermöglichen und somit die Anzahl der für die erfolgreiche Lösung einer Aufgabe benötigten Experten zu reduzieren. Abschließend stellen wir einen schrittbasierten Maximum Entropy RL Algorithmus (SRL) vor, mit dem eine diffusionsbasierte Policy trainiert werden kann, um hochdimensionale Regelungsaufgaben zu lösen. Diffusionsbasierte Policies sind eine natürliche Alternative zu MoEs, da sie sich als sehr effektiv in der Darstellung multimodaler Verteilungen bei gleichzeitig stabilem Training erwiesen haben. Allerdings sind sie im SRL Fall nicht einfach anwendbar, da sie keine berechenbare marginale Entropie besitzen.

Acknowledgements

First, I would like to deeply thank my advisor Geri, who always supported me throughout my doctoral journey. Your feedback on my work was always very helpful and often got me out of a dead end. I admire your patience with which you calmly responded to my (often unfounded) concerns about my work and always gave me very helpful advice. After these conversations, I always felt more motivated and relaxed and was able to focus again on the actual topics in my research. Especially at the beginning of my doctoral journey, these discussions were very helpful in learning and improving scientific thinking. Your commitment and support at all times of day and even during your holidays helped me immensely to get this far. This also applies to rebuttal phases in the peak of summer, during which you took the time to call and help me despite being on vacation. Not to forget your relaxed and friendly manner, which has rubbed off on the group and contributed to a pleasant working atmosphere.

I would like to thank all my colleagues, and those who had already completed their doctorates for the great time during the doctoral journey. The lunches and coffee breaks together were always very entertaining and often helped me clear my mind. The discussions with you were also very insightful, and I was able to learn a lot from you. The beginning of my doctoral journey in Tübingen with Philipp B. and Max turned out to be very pleasant and instructive despite the circumstances. Thank you for that beginning and the time that followed, which came with interesting discussions. I also thank Philipp B. for the great collaboration, during which I learned many interesting things. Furthermore, I would like to thank Aleks, with whom I not only had a very great collaboration, but also shared an office, which contributed to a very pleasant working atmosphere. Not to forget your very helpful improvements to my texts; your writing skills are remarkable. I would also like to thank Fabian O. for his helpful discussions, very useful code-level tips, and the great collaboration. Bruce, I also thank you for the great collaborations, helpful tips and tricks, and the good shared dinners in local Asian restaurants. I would also like to thank Denis, who was one of the first students I had the opportunity to supervise and later as a collegue with whom I had very interesting and insightful discussions, especially in the area of diffusion models. The discussions were always very instructive, and the collaboration very pleasant. Not to forget the joint kickboxing sessions, which I will always remember as a strenuous but very fun and enjoyable time. David, I also thank you for the great collaboration and your easygoing nature, which always contributed to a pleasant working environment. I also thank Nic for the great and enjoyable time in our shared office after the first move, during which I learned a lot about 3D printing and bicycles. Your technical expertise is remarkable. I also thank Tai, Huy, Philipp D., Niklas, Hongyi, Balázs, Emiliyan, Enes, Max N., Xinkai, Vaisakh, and Michael for the very pleasant working atmosphere and the great time in the group. Finally, I would like to thank Rudi, who always gave me helpful advice and with whom collaboration was always very enjoyable.

At this point, I would also like to thank Mirkan, Jan, and Baran, with whom the time during our studies was always entertaining and fun. During our joint study sessions, the discussions were always very insightful and helped me better understand the topics. Additionally, I would like to thank my friends. The time spent together was always very entertaining and helped me take my mind off the work.

I am deeply grateful to my parents, Ayse and Ali, and my sisters, Meltem and Bahar, who played a major role in enabling me to get this far. Your emotional support and unconditional backing always helped me keep going, even when things weren't going well. Your efforts to make life as easy as possible for me so that I could focus on my work helped me enormously. Thanks to you, I was even able to make it this far in the first place.

A particularly big and heartfelt thanks goes to my wife, Zülal. You have been by my side since my bachelor's thesis and have shared all the highs and lows with me, especially during the PhD. I am deeply grateful that you always had the patience to stay by my side, supported me unconditionally, and always made the time enjoyable. Your adaptation to the circumstances, your understanding of my work, and your patience helped me focus on the PhD and finish it in the first place.

Lastly, I would like to thank Dieter Büchler who has agreed to be my second referee and I would like to thank the PhD committee who have agreed to be the examiners.

Contents

List of Figures

List of Tables

1. Introduction

When humans complete a task, they typically do so in various and different ways rather than strictly adhering to a single pattern. For instance, we do not always grasp a mug in the same manner or with the same hand. Although these variations in task execution seem trivial to humans, they provide important adaptability, enabling us to respond effectively to different situations. If our right hand is injured, we easily switch to grasping a mug with our left hand. Similarly, if the mug's handle breaks, we naturally adapt by grasping it from the bottom, or the top. Importantly, humans rarely need to relearn behaviors entirely from scratch simply due to situational changes. This versatile behavior also finds clear expression in sports. Consider table tennis, where players deliberately vary their striking styles, such as alternating between forehand and backhand strokes in similar situations, making their play less predictable and enhancing their competitive advantage. The human capacity for multimodal behavior results from versatile skills acquired throughout life, allowing tasks to be solved precisely through multiple approaches.

Inspired by this human characteristic, equipping robots with similar versatile skills is a central motivation for this thesis. Particularly compelling is the ability for robots to autonomously discover versatile skills without relying on human expert data that is usually costly because experts are not always available and access to robots is restricted. Hence, the primary question is how robots can discover versatile skills using reinforcement learning (RL).

RL emerges as a suitable learning paradigm for this purpose, as it enables policies to be trained purely through environmental interactions, removing dependence on humans. From an RL agent's perspective, acquiring *versatile* or *diverse* behaviors carries several distinct advantages. First, the ability to represent multimodal skills can lead to improved exploration and therefore to improved sample efficiency because exploring other modes that potentially lead to better performance is unlikely once the commonly used unimodal Gaussian policy converges to a specific mode. Second, versatile skills can contribute to policy robustness, enabling quick adaptation to environmental changes by discarding invalid behaviors in favor of alternatives within a diverse skill repertoire. Lastly, versatile skills offer strategic benefits in competitive environments such as robot table tennis by minimizing predictability, a principle illustrated by the varied strokes in table tennis as mentioned earlier.

However, realizing such versatile skills requires policy representations that are expressive enough to represent multimodal distributions. Mixture of Experts (MoE) policies and diffusion-based policies are two classes of models that belong to such expressive policy representations, though their optimization is complex and requires tailored RL algorithms.

Furthermore, deciding the appropriate RL paradigm for acquiring these skills is important. RL algorithms broadly fall into two categories which is distinguished based on their exploration and policy optimization strategies: step-based reinforcement learning (SRL) and episode-based reinforcement learning (ERL).

SRL is the traditional framework that optimizes expected returns in Markov Decision Processes (MDPs) (Bellman, 1957), typically under infinite horizon conditions. Because the exploration occurs directly in the raw action space at every time step, SRL methods tend to generate jittery trajectories and often suffer from local exploration in the state space, especially at the beginning of the learning process, possibly leading to suboptimal solutions (Li et al., 2024a; Raffin et al., 2022). On the upside, these methods are sample efficient because they can leverage the temporal structure in the data to update the policy, making them particularly suitable for tasks with informative, dense reward signals. Additionally, they are well-studied and most known breakthroughs in RL such as autonomous acquisition of human-level performance in board games (Mnih et al., 2015; Silver et al., 2016), real world robot manipulation tasks (Levine et al., 2016), high-precision robot learning tasks like robot table tennis (Büchler et al., 2022) or sim-to-real transfer in robotic locomotion tasks (Tan et al., 2018; Hoeller et al., 2024) are attributed to SRL methods.

Conversely, ERL methods operate on a higher abstraction level by directly exploring the parameters of a controller. A prevalent parametrization of such controllers uses motion primitives (Schaal et al., 2005; Paraschos et al., 2013; Li et al., 2023), that, deterministically generate an entire trajectory for a fixed horizon. Because exploration occurs once at the start of each episode, ERL generates time-correlated exploration with smooth trajectories that usually lead to a wider exploration of the state space (Li et al., 2024a; Otto et al., 2023). Those properties make ERL methods well suited to tasks where sparse, or time-delayed rewards are present. Additionally, ERL methods do not require Markovian reward structures because they treat the optimization as a black-box RL problem. This feature allows a more intuitive reward design based on how humans also assess the performance of a task. For instance, if the task is to jump as high as possible, the performance feedback of a jump should be the maximum height achieved during the whole jump, i.e. it is based on retrospective data rather than the sum of heights at each time step, which would be a common Markovian reward. However, ERL methods tend to be less sample efficient because they treat each evaluation of an episode and its corresponding return as a single sample.

This thesis focuses on enabling robots to learn versatile skills within both SRL and ERL frameworks, specifically leveraging the maximum entropy RL objective to incentivize exploration. More precisely, in the first part, we focus on learning MoE policies (Bishop, 2006) in the context of ERL. Because of the aforementioned properties of ERL, it is a suitable setting for learning multimodal skills using MoE policies, where we take inspiration from advances in the field of variational inference (Arenz et al., 2020) and propose a decomposition of the maximum entropy ERL objective that allows us to train each expert individually. For effective training, we extend the objective with automatic curriculum learning to ensure that each expert can shape their own pace of learning and can focus on context[1] regions it favors by gradually increasing its responsible region of the context space. Additionally, this curriculum avoids discarding experts and collapsing them into only a few experts that dominate the mixture model (Bacon et al., 2017). This automatic curriculum learning is achieved by a learnable per-expert Gaussian distribution over the context space. Optimizing both, the expert and the context distributions leads to a policy with two levels of hierarchy and allows the whole model to learn versatile skills.

Based on the same objective, we propose extending the method to the more expressive energy-based per-expert context distributions and deep networks as expert representations. The energy-based distributions can represent sharp discontinuities and multi-modality in the context space (Florence et al., 2022), which are a common case in real world applications, as the task space is usually defined in a finite space. For example, the surface of a table might be the goal position to place a mug. Here, the table's edges define the sharp boundaries of the context space and additionally, there might be some invalid context regions within the table because of objects that block the space. However, training those energy-based models is not straightforward because of the intractable normalization constant of the energy-based distribution. We propose a sample-based approximation of the normalization constant by sampling contexts from the environment's true context distribution through environment resets without execution. Subsequent importance sampling based on the per-expert energy-based context distribution allows each expert to shape its curriculum but now with the big advantage that the contexts are inherently valid as they come from the true context distribution and edges of the context space are properly represented because of the energy-based distribution's high expressiveness. In conjunction with highly non-linear experts, this expressiveness leads to a significantly reduced number of required experts to cover the whole context space and consequently achieve good performance compared to the method proposed before.

Finally, we propose using diffusion-based policies in the context of SRL as an approach capable of representing multimodal distributions. Diffusion models have recently gained a lot of attention due to their ability to represent complex multimodal distributions in high-dimensional spaces while the training procedure is very stable. However, training diffusion models in the context of maximum entropy RL is not straightforward because the objecive requires calculating the marginal entropy. This statistic is intractable, which motivates us to take inspiration from recent advances in approximate inference with diffusion models (Berner et al., 2024) to propose a tractable lower-bound on the maximum entropy objective. This lower bound allows us to backpropagate the gradients through the diffusion process. The resulting method is a policy iteration scheme that provably converges to the optimal policy.

The next section summarizes the challenges addressed in this thesis and provides an overview of the contributions alongside the structure of the document.

[1] Contexts define the task, e.g. the goal position an object needs to be put, or the goal landing position of a ball on the opponent's table side in table tennis. In theory they can also define physicall parameters such as friction values. This thesis focuses on the task defining aspect.

1.1. Thesis Contributions and Structure

From the previous discussion, we can extract the following three challenges that need to be addressed in order to enable robots to learn versatile skills using RL.

Challenge 1 (C1): Representation and Training of Multimodal Policies. Learning versatile skills requires policy representations that can capture multimodal distributions. While high-capacity representations such as Mixture of Experts (MoE) and diffusion models exist and are well-studied mostly in the supervised learning paradigm, training them in the context of reinforcement learning (RL) is not straightforward and requires novel RL methods that are adapted to the respective policy representation. Additionally, the expressiveness of the policy representation can only be exploited if the RL method explicitly incentivizes discovering different modes, which needs to be addressed in the RL algorithm alongside efficient training.
Addressed in Chapters 3, 4 and 5

Challenge 2 (C2): Retaining Multimodalities. Some modes of a multimodal policy might lead to a higher return compared to other modes earlier in the RL training process. In this case, the RL algorithm increases the likelihood of choosing these modes more often, thereby biasing the policy towards this skill. The consequence is an imbalance in the collected training data where some modes are overrepresented, leading to a more frequent update of those modes, which again increases the likelihood of choosing this mode in the next iteration even more. The result is that already discovered modes might get discarded during training, leading to a mode collapse (Bacon et al., 2017) and preventing the learning of versatile skills. This is a challenge that needs to be addressed in RL algorithms for learning versatile skills.
Addressed in Chapters 3 and 4

Challenge 3 (C3): Non-Linear Adaptation. Non-linear adaptation to inputs is a key feature why deep reinforcement learning (DRL) has been successful in recent years (Mnih et al., 2015). This non-linear adaptation has been adressed in the context of RL with unimodal Gaussian policies (Haarnoja et al., 2018b), but it is a challenge to obtain and successfully train multimodal policies with non-linear adaptation in each mode. Additionally, solving *Challenge 2* involves optimizing for input distributions over the task-defining context space. However, this context space usually has hard non-linearities that require special parameterization for successful representation.
Addressed in Chapters 4 and 5

Structure of the Thesis. In Chapter 2 we provide the necessary foundations for the thesis, including a brief introduction to RL and RL methods, the general idea of curriculum learning, and used policy representations throughout this thesis. As the used objectives draw parallels to the variational inference literature, we also provide a brief introduction

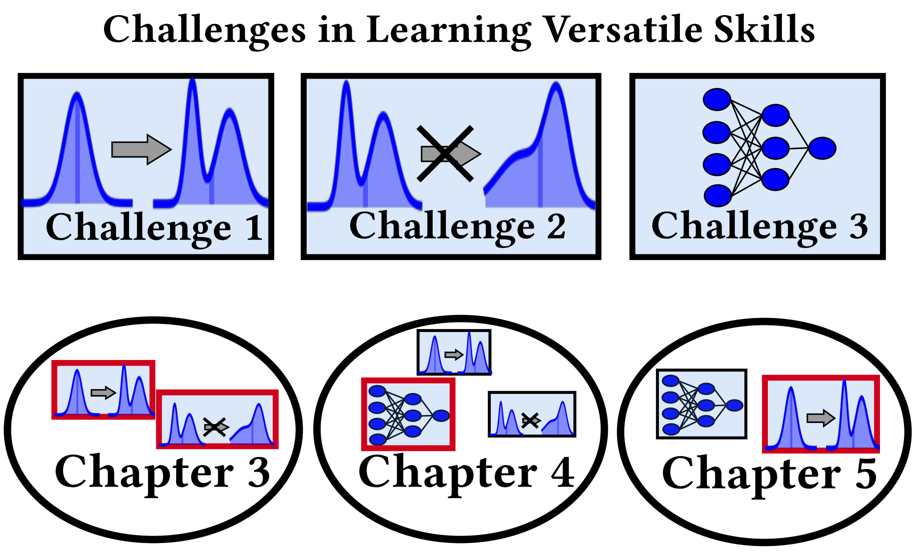


Figure 1.1.: Challenges and Contributions Overview. The figure summarizes the challenges for learning versatile skills as described in this section. Furthermore, we provide an overview of the contributions of each chapter to address the respective challenges. We have highlighted the main contributions of each chapter using a bold, red rectangle whereas the challenges that are also addressed alongside the main contributions are drawn in black rectangles. In Chapter 3 the main focus is on addressing *Challenge 1* and *Challenge 2*, whereas the main focus in Chapter 4 is on addressing *Challenge 3* alongside *Challenge 1* and *Challenge 2*. In Chapter 5 we propose a solution for *Challenge 1* using diffusion-based policies, which addresses *Challenge 3* as well.

to variational inference and how to optimize latent variable models in this framework. Chapter 3 and 4 present the methods for learning versatile skills using MoE, where Chapter 3 contributes addressing the *Challenges 1* and *2* and Chapter 4 contributes addressing the *Challenges 1,2, 3*. Chapter 5 presents an actor-critic method for learning diffusion-based policies in the context of SRL and aims to address *Challenges 1 and 3* specifically tailored for diffusion models. Finally, in Chapter 6 we summarize this thesis and provide an outlook for future research.

Following the content description of each chapter, Fig. 1.1 summarizes the challenges and provides an overview where each of the presented challenges is addressed in the thesis. The figure also emphasizes the challenges a chapter focuses on in more detail, indicating the main contributions using a bold, red rectangle.

The following three sections are reprints of the abstracts of the respective publications this thesis is based on. Those publications are discussed in Sections 3, 4 and 5 in more detail.

1.1.1. Specializing Versatile Skill Libraries using Local Mixture of Experts

Onur Celik, Dongzhuoran Zhou, Ge Li, Philipp Becker, Gerhard Neumann

A long-cherished vision in robotics is to equip robots with skills that match the versatility and precision of humans. For example, when playing table tennis, a robot should be capable of returning the ball in various ways while precisely placing it at the desired location. A common approach to model such versatile behavior is to use a Mixture of Experts (MoE) model, where each expert is a contextual motion primitive. However, learning such MoEs is challenging as most objectives force the model to cover the entire context space, which prevents specialization of the primitives resulting in rather low-quality components. Starting from maximum entropy reinforcement learning (RL), we decompose the objective into optimizing an individual lower bound per mixture component. Further, we introduce a curriculum by allowing the components to focus on a local context region, enabling the model to learn highly accurate skill representations. To this end, we use local context distributions that are adapted jointly with the expert primitives. Our lower bound advocates an iterative addition of new components, where new components will concentrate on local context regions not covered by the current MoE. This local and incremental learning results in a modular MoE model of high accuracy and versatility, where both properties can be scaled by adding more components on the fly. We demonstrate this by an extensive ablation and on two challenging simulated robot skill learning tasks. We compare our achieved performance to LaDiPS and HiREPS, a known hierarchical policy search method for learning diverse skills.

1.1.2. Acquiring Diverse Skills using Curriculum Reinforcement Learning with Mixture of Experts

Onur Celik, Aleksandar Taranovic, Gerhard Neumann

Reinforcement learning (RL) is a powerful approach for acquiring a good-performing policy. However, learning diverse skills is challenging in RL due to the commonly used Gaussian policy parameterisation. We propose **Di**verse **Skil**l **L**earning (Di-SkilL), an RL method for learning diverse skills using Mixture of Experts, where each expert formalizes a skill as a contextual motion primitive. Di-SkilL optimizes each expert and its associate context distribution to a maximum entropy objective that incentivizes learning diverse skills in similar contexts. The per-expert context distribution enables automatic curricula learning, allowing each expert to focus on its best-performing sub-region of the context space. To overcome hard discontinuities and multi-modalities without any prior knowledge of the environment's unknown context probability space, we leverage energy-based models to represent the per-expert context distributions and demonstrate how we can efficiently train them using the standard policy gradient objective. We show on challenging robot simulation tasks that Di-SkilL can learn diverse and performant skills.

1.1.3. DIME: Diffusion-Based Maximum Entropy Reinforcement Learning

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, Gerhard Neumann

Maximum entropy reinforcement learning (MaxEnt-RL) has become the standard approach to RL due to its beneficial exploration properties. Traditionally, policies are parameterized using Gaussian distributions, which significantly limits their representational capacity. Diffusion-based policies offer a more expressive alternative, yet integrating them into MaxEnt-RL poses challenges-primarily due to the intractability of computing their marginal entropy. To overcome this, we propose Diffusion-Based Maximum Entropy RL (DIME). *DIME* leverages recent advances in approximate inference with diffusion models to derive a lower bound on the maximum entropy objective. Additionally, we propose a policy iteration scheme that provably converges to the optimal diffusion policy. Our method enables the use of expressive diffusion-based policies while retaining the principled exploration benefits of MaxEnt-RL, significantly outperforming other diffusion-based methods on challenging high-dimensional control benchmarks. It is also competitive with state-of-the-art non-diffusion based RL methods while requiring fewer algorithmic design choices and smaller update-to-data ratios, reducing computational complexity.

2. Foundations

This chapter reviews the key concepts required to understand the work presented in this thesis. We begin with the fundamentals and notation of the step-based reinforcement learning (SRL) framework (Section 2.2). Next, we introduce episode-based reinforcement learning (ERL) (Section 2.3). Section 2.4 examines the maximum entropy reinforcement learning (RL) objective, followed by an overview of curriculum learning in RL (Section 2.5). In Section 2.6 we discuss the policy representations that are relevant for learning versatile skills. Finally, we conclude this chapter with an overview of optimizing latent variable models using Variational Inference together with a discussion on the parallels to maximum entropy RL (Section 2.7).

2.1. General Notation

Throughout the rest of this thesis scalars are denoted as single variables e.g., a, vectors are denoted by bold lowercase letters, e.g., $\mathbf{a}$, and matrices by bold uppercase letters, e.g., $\mathbf{A}$. We will use subscripts to denote the time step in the reinforcement learning loop, e.g., $\mathbf{a}_t$ is the action at time step t, whereas superscripts indicate the internal time steps of the diffusion model, e.g., $\mathbf{a}^k$ is the latent action at diffusion step k. In general we will use greek letters to denote parameters of a function or distribution, e.g., $\pi_{\boldsymbol{\theta}}$ is a policy with parameters $\boldsymbol{\theta}$. However, we might also need to use superscripts to denote the parameters of a function or distribution, e.g., $\pi^{\boldsymbol{\theta}}$ especially in the context of diffusion models.

Deviating notations will be stated in the text to maintain clarity.

2.2. Step-Based Reinforcement Learning (SRL)

We introduce the fundamentals of step-based reinforcement learning (SRL) in the following section, where we start by defining the Makrov Decision Process (MDP) in Section 2.2.1 and continue with the SRL's policy and objective definition together with related quantities such as the value functions in Section 2.2.2. Using these information, we provide a general overview of the SRL learning loop in Section 2.2.3. Finally, we conclude this section with a discussion on SRL methods in Section 2.2.4. We take inspiration from Sutton and Barto (2018) and Kober et al. (2013) for the definitions in this section.

2.2.1. The Markov Decision Process

Most well-known step-based reinforcement learning (SRL) algorithms are based on the Markov Decision Process (MDP) which is formally defined as the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0)$. In the following we will introduce and define the components of the MDP.

State Space. At time step t, the agent is in a state $\mathbf{s}_t \in \mathcal{S}$ that contains all information required for the agent to make an optimal decision. For example in robotics, this state can include the robot's global pose, internal joint angles, and the positions of relevant objects such as the position of a target object that needs to be manipulated. Depending on the task, $\mathcal{S}$ may be discrete or continuous, though continuous spaces are more common in robotic environments.

Action Space. The agent takes an action $a_t \in \mathcal{A}$ at time step t to interact with the environment, where the action space is defined by $\mathcal{A}$ and can be discrete or continuous depending on the task. For instance, in robotic tasks the robot can be controlled in different ways, e.g., by setting target joint positions or velocities which are then executed by a low-level controller that turns these targets into torques. In this thesis, we focus on continuous state-action spaces, as the most common practice in robotics.

Dynamics. The environment returns the next state $\mathbf{s}_{t+1}$ after executing the action $\mathbf{a}_t$ in the current state $\mathbf{s}_t$. The next state $\mathbf{s}_{t+1}$ is determined by the probability density of the transition dynamics $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, \infty)$ and is commonly unknown to the agent in RL. We define the transition dynamics as

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) = \mathcal{P}\left(\mathbf{s}' = \mathbf{s}_{t+1}|\mathbf{s} = \mathbf{s}_t, \mathbf{a} = \mathbf{a}_t\right), \tag{2.1}$$

where in some tasks the dynamics can also be deterministic.

Reward. Similarly, the agent is unaware of the **reward function** $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ that evaluates the agent's decision. Throughout this thesis, we will denote the reward function as $r(\mathbf{s}_t, \mathbf{a}_t)$, or simply as r_t as an evaluation of the action $\mathbf{a}_t$ in the state $\mathbf{s}_t$ at time step t. In general, the reward function can also just depend on the state $r(\mathbf{s}_t)$, or it can also include the next state $r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$. We will restrict ourselves to the more common case $r(\mathbf{s}_t, \mathbf{a}_t)$ in this thesis.

Initial state. Once the environment terminates, a new state is randomly sampled from the initial state distribution $\mathbf{s}_0 \sim \rho_0$. The initial state distribution is detrmined by the environment and is unknown to the agent.

Markov Property. A key feature the MDP framework satisfies is the Markov property which is formalized as

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}, ...\mathbf{s}_0) = p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t). \tag{2.2}$$

Intuitively, this means that the current state $\mathbf{s}_t$ encodes all information needed to determine the environment's next state when exectuing action $\mathbf{a}_t$ and does not require any past information. If the agent does not observe the environment's whole state information, the Markov property is not satisfied anymore and additional techniques are required to infer the environment's state based on the sensory input from the past. This setup is known as the Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998) and a common case of a POMDP is image-based RL. Here, the agent only observes images from the current scene and has to estimate the environment's state based on these images. Although conducted experiments in this thesis include tasks where the Markov property is not satisfied in the sense that the reward depends on retrospective state information, those tasks are restricted to the episode-based RL setting which inherently can handle these situations (Section 2.3).

Based on the definitions above, next we can formulate the general objective of the SRL framework.

2.2.2. Step-Based Reinforcement Learning Objective

Based on the MDP, we can define the policy, the general objective and related quantities for optimizing the policy in the step-based reinforcement learning (SRL) framework.

Policy. Commonly, deep neural networks are used to parameterize a mapping from a current state $\mathbf{s}_t$ to an action $\mathbf{a}_t \in \mathcal{A}$. This mapping is referred to as the policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$. Although deterministic policies $\mathbf{a}_t = \pi(\mathbf{s}_t)$ have been proposed in the past (Silver et al., 2014; Lillicrap et al., 2016), stochastic policies $\pi(\mathbf{a}_t|\mathbf{s}_t)$ that map to a probability density have gained more attention recently as they allow controlling the exploration behavior of the policy in the optimization (Haarnoja et al., 2018b; Abdolmaleki et al., 2018). Unless noted otherwise, throughout this thesis a subscript denotes the learnable parameters of the policy, e.g., $\pi_{\boldsymbol{\theta}}$ is a policy with parameters $\boldsymbol{\theta}$.

Trajectory. The trajectory $\boldsymbol{\tau} = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, ...)$ is defined as the sequence of state-action pairs induced by the policy π and the environment's dynamics through the relation

$$p_\pi(\boldsymbol{\tau}) = \rho(\mathbf{s}_0) \prod_{t=0}^{\infty} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi(\mathbf{a}_t|\mathbf{s}_t). \tag{2.3}$$

Hence rolling out a trajectory $\boldsymbol{\tau}$ in the environment is equivalent to sampling from $p_\pi(\boldsymbol{\tau})$.

SRL Objective. The objective of an RL agent is to find a policy π that maximizes the expected accumulated reward

$$J(\pi) = \mathbb{E}_{p_\pi}\left[G(\boldsymbol{\tau})\right] = \int_{\boldsymbol{\tau}} G(\boldsymbol{\tau}) p_\pi(\boldsymbol{\tau}) d\boldsymbol{\tau}, \quad \text{with } G(\boldsymbol{\tau}) = \sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t), \tag{2.4}$$

where $\gamma \in [0, 1)$ is the discount factor. The infinite sum over the rewards in Eq. 2.4 can lead to infinite returns, such that γ is used to reduce the importance of future rewards and consequently allows learning a useful policy.

State Value Function. The state value function $V^\pi(\mathbf{s})$ is defined as the expected return when starting in state $\mathbf{s}$ and following the policy π afterwards. Formally this is written as

$$V^\pi(\mathbf{s}) = \mathbb{E}_{p_\pi}\left[G_t | \mathbf{s}_t = \mathbf{s}\right] = \mathbb{E}_{p_\pi}\left[\sum_{k=0}^{\infty} \gamma^k r(\mathbf{s}_{t+k}, \mathbf{a}_{t+k}) \Big| \mathbf{s}_t = \mathbf{s}\right] \quad \forall \mathbf{s} \in \mathcal{S}. \tag{2.5}$$

It is a useful quantity to evaluate the quality of a state $\mathbf{s}$ under the policy π. When parameterized, the quality of the state can be assessed without the need to evaluate the policy, which is a key feature in most policy gradient methods (see Section 2.2.4).

State-Action Value Function or Q-Function. With a similar interpretation as the state value function, the state-action value function $Q^\pi(\mathbf{s}, \mathbf{a})$ is defined as the expected return when starting in state $\mathbf{s}$, executing action $\mathbf{a}$ and then following the policy π afterwards

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{p_\pi}\left[G_t | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}\right] = \mathbb{E}_{p_\pi}\left[\sum_{k=0}^{\infty} \gamma^k r(\mathbf{s}_{t+k}, \mathbf{a}_{t+k}) \Big| \mathbf{s}_t = s, \mathbf{a}_t = a\right] \quad \forall \mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}. \tag{2.6}$$

The Q-function provides a useful measure for assessing the quality of a state-action pair $(\mathbf{s}, \mathbf{a})$, where the action $\mathbf{a}$ does not need to be coming from the current policy $\pi_{\boldsymbol{\theta}}$. This difference to state value functions makes the state-action value function a quantity that is specifically useful for off-policy RL methods.

The value functions can be recursively defined using the Bellman equations (Bellman, 1957) as

$$V^\pi(\mathbf{s}_t) = \mathbb{E}_{a_t \sim \pi}\left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V^\pi(\mathbf{s}_{t+1})\right]\right], \tag{2.7}$$

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p, \mathbf{a}_{t+1} \sim \pi}\left[Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})\right], \tag{2.8}$$

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V^\pi(\mathbf{s}_{t+1})\right]. \tag{2.9}$$

These recursive equations have been used for developping many RL algorithms (see Section 2.2.4)

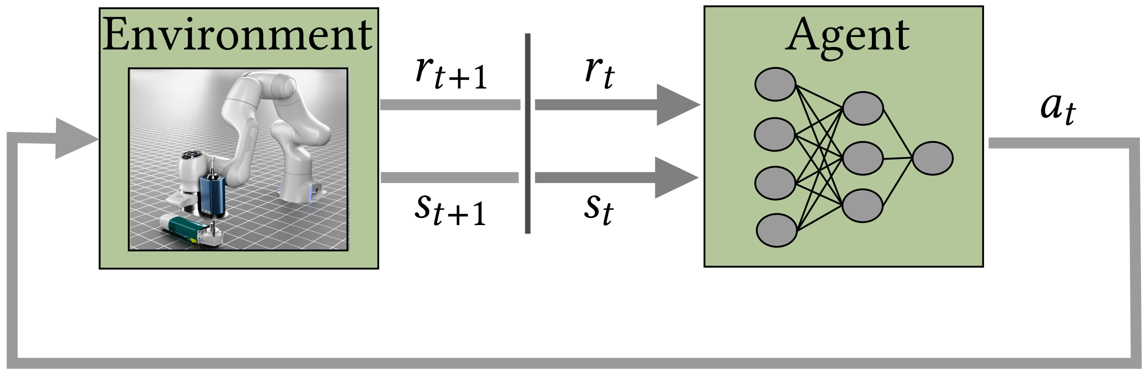


Figure 2.1.: Step-Based Reinforcement Learning (SRL) Loop. In time-step t the agent observes a state s_t and selects an action a_t, which is executed in the environment. Based on the transition dynamics $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ the environment transitions to a new state $\mathbf{s}_{t+1}$ and returns a reward r_{t+1} that is calculated by the reward function $r(\mathbf{s}, \mathbf{a})$ and that defines the task. Using the per-step data points consisting of the tuple $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$, the agent updates its policy π to maximize the expected cumulative reward. SRL is based on the Markov Decision Process (MDP) framework, in which the the state $\mathbf{s}_t$ contains all information to describe the whole system such that no retrospective information is required. All relevant quantities such as the transition dynamics and the reward function rely on this Markov property. This image is inspired and adapted from Sutton and Barto (2018).

Advantage Function. The advantage function $A^\pi(\mathbf{s}, \mathbf{a})$ is defined as the difference between the Q-function and the Value function

$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t). \tag{2.10}$$

Intuitively, the advantage function quantifies how much better an action $\mathbf{a}_t$ is than the expected value of the state $\mathbf{s}_t$ under the policy π. It is a widely used quantity especially in the policy gradient methods (see Section 2.2.4) because it allows reducing the variance of the gradient estimates without affecting the gradient.

With the formal definitions of the MDP and the SRL objective, we can now provide an general overview of the SRL learning loop, which is the core of the SRL algorithms defined in Section 2.2.4.

2.2.3. Step-Based Reinforcement Learning Loop

The core concept of step-based reinforcement learning (SRL) relies on the interaction between an agent and an environment. At each discrete time step t, an agent observes a state $\mathbf{s}_t$ and selects an action $\mathbf{a}_t$ according to its policy π_θ. The environment then returns a new state $\mathbf{s}_{t+1}$ and a reward signal r_t that are induced by the transition dynamics $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ and the reward function $r(\mathbf{s}_t, \mathbf{a}_t)$, respectively. The agent's objective is to learn a policy π_θ that maximizes the expected cumulative reward (Eq. 2.4) based solely on those per-step information (Sutton and Barto, 2018).

In most practical settings the agent is unaware of the transition dynamics p or the reward function r. However, because the task is encoded in r, maximizing the accumulated reward yields behavior that solves the task. For instance, if the reward equals the negative Euclidean distance to a goal, then maximizing this reward drives the agent to that goal. The output of an RL algorithm is a policy $\pi_{\theta^\star}$ that maps states to actions and solves the task optimally. Figure 2.1 illustrates the learning loop.

This learning loop can be related to the concept of trial and error (Klopf, 1972, 1975) which gives rise to the exploration-exploitation trade-off, a key challenge in RL. Intuitively, choosing actions that lead to high immediate rewards is the optimal behavior. However, in certain tasks, this strategy might lead to suboptimal behavior, necessitating the agent to explore actions leading to state-action regions that yield less immediate reward but higher cumulated rewards in the future (Sutton and Barto, 2018). Considering the goal-reaching example, the agent might need to first avoid an obstacle to reach the goal position, but this path might not yield the highest immediate reward if the agent first needs to move away from the goal position.

This exploration-exploitation trade-off is one of the reasons why RL is distinguished from supervised and unsupervised learning. However, also the available data structure is different in RL. Supervised learning assumes a fixed dataset of labeled input-output pairs, while unsupervised learning seeks structure in unlabelled data. In contrast, in RL the agent needs to maximize a reward signal by interacting with the environment and has to generate its data based on its current knowledge. Because the agent might not have experienced enough, the aforementioned trade-off between exploration and exploitation arises, which additionally distinguishes RL from other machine learning paradigms.

2.2.4. Step-Based Reinforcement Learning Methods

Finding an optimal strategy can be achieved by maximizing the objective in Eq. 2.4. Broadly there are two main approaches, namely the **Value-Based** and the **Policy Search** methods. Within these approaches, the algorithms are further distinguished into **on-policy** and **off-policy** methods. On-policy methods optimize a policy using data generated by that same policy, whereas off-policy methods can use data generated by a different policy to optimize the current policy. We will discuss the evolution and nowadays most known methods in the following subsections.

2.2.4.1. Value-Based Methods

Among the first methods are **Dynamic Programming (DP)** (Bellman, 1966) based approaches. Here, *Value Iteration* and *Policy Iteration* are both based on the Bellman principle of optimality (Bellman, 1966) and iteratively improve the value function and policy until convergence. DP-based methods require knowing the transition dynamics (Kober et al., 2013), such that no exploration is needed, and they usually work only for low-dimensional discrete systems because the learning iterations need to be done over all states and actions. An exception for the continuous case is the Linear Quadratic Regulator (LQR) (Kober et al., 2013).

However, the exact dynamics of a system are known in rare cases which in this case can be leveraged to learn optimal policies. For example in board games, the transition dynamics have been leveraged in the learning leading to impressive results where an agent achieved human-level performance (Silver et al., 2016, 2018).
In the more common case, the transition dynamics and the reward function are unknown thus hindering the application of classical dynamic programming. However, a Monte Carlo estimate can be used to estimate the expectation under the transition dynamics, which is done in Temporal Difference Learning based methods.

Temporal Difference Learning (TD-Learning). TD-Learning (Sutton, 1988) updates the Value function using an incremental change of the current value and a sample based estimate of the next value referred to as δ_t

$$V^{\pi}(\mathbf{s}_t) = V^{\pi}(\mathbf{s}_t) + \alpha \underbrace{(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V^{\pi}(\mathbf{s}_{t+1}) - V^{\pi}(\mathbf{s}_t))}_{\text{Temporal difference error } \delta_t}, \tag{2.11}$$

where α is a step size parameter. Hence, by using the temporal difference, TD-Learning bootstraps the new $V^{\pi}(\mathbf{s}_t)$ from the current value estimates $V^{\pi}(\mathbf{s}_{t+1})$. Usually, only one transition sample is obtained for each state. However, TD Learning can be extended to multiple samples over time bridging the pure DP approach with Monte Carlo methods. This variant is referred to as TD(λ) (Sutton and Barto, 2018).

Early works have used the temporal difference learning in the on-policy setting known as SARSA (Rummery and Niranjan, 1994; Sutton and Barto, 2018), and in the off-policy setting known as Q-learning (Watkins, 1989; Watkins and Dayan, 1992). In Q-learning, the Q-function update is independent of the data-generating policy, because the backup calculates the optimal Q-value for the next state to update the current Q-value

$$Q(\mathbf{s}_t, \mathbf{a}_t) = Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}_{t+1}} Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - Q(\mathbf{s}_t, \mathbf{a}_t) \right). \tag{2.12}$$

This key difference makes Q-learning an off-policy method. However, because a broad range of the states need to be visited for a precise update and because the optimal a_{t+1} needs to be calculated for each backup, those methods are usually restricted to discrete state-action spaces. Additionally, in their plain form, they do not scale to high-dimensional state-action spaces. Nonetheless, with the era of deep learning the latter was successfully addressed by Deep Q-Networks (DQN) (Mnih et al., 2015) that uses neural networks to approximate the Q-function. For successful training of these high-capacity function approximators, DQN uses the squared loss

$$L(\beta) = \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1} \sim \mathcal{D}} \left[\left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}_{t+1}} Q_{\bar{\beta}}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - Q_{\beta}(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \right], \tag{2.13}$$

with an additional target network $Q_{\bar{\beta}}$ to stabilize the training and a replay buffer $\mathcal{D}$, in which past experiences are saved, to improve sample efficiency and avoid catastrophic forgetting. In later works several extensions of DQN have greatly influenced the field of deep

RL methods, among which double Q-learning (Van Hasselt et al., 2016) or distributional RL (Bellemare et al., 2017) are the most known. A full summary of all key extensions is given in the work by Hessel et al. (2018).

Value-Based methods optimize the RL objective in Eq. 2.4 relying on the value functions V^π or Q^π but do not explicitly parameterize the policy π_θ. In contrast, policy search methods follow the approach of directly optimizing the policy π_θ to maximize the expected return in Eq. 2.4.

2.2.4.2. Policy Search Methods

Policy Search methods parameterize the policy π_θ and optimize their parameters $\boldsymbol{\theta}$ commonly based on the *Policy Gradient*

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha \nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) \tag{2.14}$$

$$= \boldsymbol{\theta}_i + \alpha \int_{\tau} \nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}}} G(\boldsymbol{\tau}) d\boldsymbol{\tau}, \tag{2.15}$$

to maximize the expected return in Eq. 2.4. However, evaluating the integral in Eq. 2.15 is intractable because evaluating $G(\boldsymbol{\tau})$ for every possible $\boldsymbol{\tau}$ is not possible. Therefore, the *REINFORCE* algorithm (Williams, 1992) reformulates the policy gradient as an expectation using the relation $\nabla_{\boldsymbol{\theta}} p_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{\tau}) = p_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{\tau}) \nabla_{\boldsymbol{\theta}} \log p_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{\tau})$

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = \int_{\tau} p_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{\tau}) \nabla_{\boldsymbol{\theta}} \log p_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{\tau}) G(\boldsymbol{\tau}) d\boldsymbol{\tau} \tag{2.16}$$

$$\approx \frac{1}{N} \sum_{i=0}^{N} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}^i) G(\boldsymbol{\tau}^i) \tag{2.17}$$

$$= \frac{1}{N} \sum_{i=0}^{N} \left(\nabla_{\boldsymbol{\theta}} \left(\log \rho(\mathbf{s}_0^i) + \sum_t \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) + \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t) \right) \right) \left(\sum_t \gamma^t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right) \tag{2.18}$$

$$= \frac{1}{N} \sum_{i=0}^{N} \left(\sum_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^i|\mathbf{s}_t^i) \right) \left(\sum_t \gamma^t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right), \tag{2.19}$$

such that the gradient can be estimated using Monte Carlo samples $\boldsymbol{\tau}^i \sim p_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{\tau})$. Obtaining the state-action formulation instead of the trajectory formulation in Eq. 2.19 requires applying the logarithm rules to Eq. 2.3 and noting that the initial state distribution $\rho(\mathbf{s})$ and the transition dynamics $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ do not depend on $\boldsymbol{\theta}$ such that the gradient w.r.t. $\boldsymbol{\theta}$ vanishes.
However, due to the stochasticity inherent to the policy and the transition dynamics, the gradient estimator in Eq. 2.19 has a high variance. An important extension to the

REINFORCE algorithm is therefore the Policy Gradient Theorem (Sutton et al., 1999a) which leverages the temporal structure of the return

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{N}\sum_{i=0}^{N}\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\left(\sum_k \gamma^k r(\mathbf{s}_k^i, \mathbf{a}_k^i)\right) \tag{2.20}$$

$$= \frac{1}{N}\sum_{i=0}^{N}\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\left(\sum_{k=0}^{t-1} \gamma^k r(\mathbf{s}_k^i, \mathbf{a}_k^i) + \sum_{k=t} \gamma^k r(\mathbf{s}_k^i, \mathbf{a}_k^i)\right) \tag{2.21}$$

$$= \frac{1}{N}\sum_{i=0}^{N}\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\underbrace{\left(\sum_{k=0} \gamma^k r(\mathbf{s}_{t+k}^i, \mathbf{a}_{t+k}^i)\right)}_{Q^{\pi_{\theta_{old}}}(\mathbf{s}_t^i,\mathbf{a}_t^i)}, \tag{2.22}$$

where the last line follows from the fact that rewards at time steps $< t$, i.e., past rewards, do not depend on the current action $\mathbf{a}_\mathbf{t}^\mathbf{i}$, thereby are unimportant for the gradient estimate. In addition to the Policy Gradient Theorem, subtracting a baseline $b(\mathbf{s}_t)$ from the Q-function reduces the variance of the gradient estimate further (Peters and Schaal, 2008)

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{N}\sum_{i=0}^{N}\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\left(\sum_{k=0} \gamma^k r(\mathbf{s}_{t+k}^i, \mathbf{a}_{t+k}^i) - \sum_{k=0} \gamma^k b(\mathbf{s}_{t+k}^i)\right), \tag{2.23}$$

where a common choice for the baseline is the state value function $b(s_t) = V^{\pi_{old}}(s_t)$, which leads to the advantage function $A^\pi(\mathbf{s}, \mathbf{a})$ as defined in Section 2.2.2.
The advantage function is a key concept in policy gradient methods and is widely used in most successful algorithms such as PPO (Schulman et al., 2017). Most commonly, advanced methods such as the Generalized Advantage Estimate (GAE) (Schulman et al., 2016) are used to estimate the advantage values.

However, simply optimizing the policy parameters $\boldsymbol{\theta}$ using the gradient estimate in Eq. 2.23 poses several challenges. Commonly, the policy is parameterized as a Gaussian distribution that enables controlling the exploration behavior. However, the policy can still converge into a local optimum as the exploration of the policy is reduced too fast because the objective does not provide an incentive to keep the exploration. Furthermore, it can be challenging to find a good step size α for the policy update, as the gradient is inversely scaled with the variance of the policy which might lead to high gradients the smaller the variance becomes.

Trust Region Methods. Those challenges gave rise to **Trust Region Methods** which constrain the policy update in each iteration using a trust region constraint (Peters et al., 2010; Schulman et al., 2015, 2017; Otto et al., 2021). Most commonly the Kullback-Leibler (KL) divergence between the old policy $\pi_{\theta_{old}}$ and the new policy π_θ is used to constrain the policy update. Due to its simple implementation and great performance PPO (Schulman et al., 2017) is nowadays the most well-known on-policy policy gradient method. Its optimization problem is formalized as

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\theta_{old}}} \left[\min \left(\frac{\pi_{\theta}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\theta_{old}}(\mathbf{a}_t|\mathbf{s}_t)} A^{\pi_{\theta_{old}}}(\mathbf{s}_t, \mathbf{a}_t), \text{clip} \left(\frac{\pi_{\theta}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\theta_{old}}(\mathbf{a}_t|\mathbf{s}_t)}, 1-\epsilon, 1+\epsilon \right) A^{\pi_{\theta_{old}}}(\mathbf{s}_t, \mathbf{a}_t) \right) \right]. \tag{2.24}$$

Here, the trust region is approximated using a clipped surrogate objective where the ratio of the old and new policy is bounded by a clipping hyperparameter ϵ. However, PPO requires careful implementation details such as normalizing the observations and rewards to achieve good performance (Engstrom et al., 2020). Therefore, more recent works such as Trust Region Projection Layers (TRPL) (Otto et al., 2021) propose a more principled way by using a differentiable trust region layer to satisfy the trust region constraint per state. Formally, TRPL maximizes the constrained optimization problem

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\theta_{old}}} \left[\frac{\pi_{\theta}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\theta_{old}}(\mathbf{a}_t|\mathbf{s}_t)} A^{\pi_{\theta_{old}}}(\mathbf{s}_t, \mathbf{a}_t) \right] \quad \text{s.t.} \ \text{KL}\left(\pi_{\theta}(\cdot|\mathbf{s}) || \pi_{\theta_{old}}(\cdot|\mathbf{s})\right) \ \ \forall \mathbf{s} \in \mathcal{S}, \tag{2.25}$$

by augmenting the policy with an additional layer that projects the policy parameters $\boldsymbol{\theta}$ into the trust region.

So far, the discussed policy gradient methods are on-policy methods, as they optimize the policy using data generated by that same policy. However, in Eq. 2.23, it is also possible to approximate the state-action value function $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ using a parameterized function together in conjunction with the state value function $V^{\pi}(\mathbf{s}_t)$. In this case, the RL method is usually categorized into the field of **Actor-Critic** methods (Sutton and Barto, 2018). Actor-Critic methods have gained more attention in the context of off-policy RL methods in the past years. Those methods update the Q-function and the policy using data from a replay buffer $\mathcal{D}$ that stores past transitions from previous iterations (Haarnoja et al., 2018b; Abdolmaleki et al., 2018; Fujimoto et al., 2018). Among those methods Soft Actor Critic (SAC) (Haarnoja et al., 2018b) has proven a stable and well-performing algorithm which we will discuss in Section 2.4 in more detail.

2.3. Episode-Based Reinforcement Learning (ERL)

This section summarises the foundations of episode-based reinforcement learning (ERL) and continues with the discussion of the objective in ERL in Section 2.3.1. We conclude the discussion on ERL with an overview of ERL methods in Section 2.3.2. In the following, we take inspiration and follow Deisenroth et al. (2013) for the discussion of this section.

From Section 2.2 it is clear that step-based RL (SRL) algorithms explore by adding noise sampled from a zero mean and adaptive variance normal distribution to the action in each time step. This is a convenient exploration strategy, because it is easy to implement and works well in many tasks, especially with informative per-step reward signals.

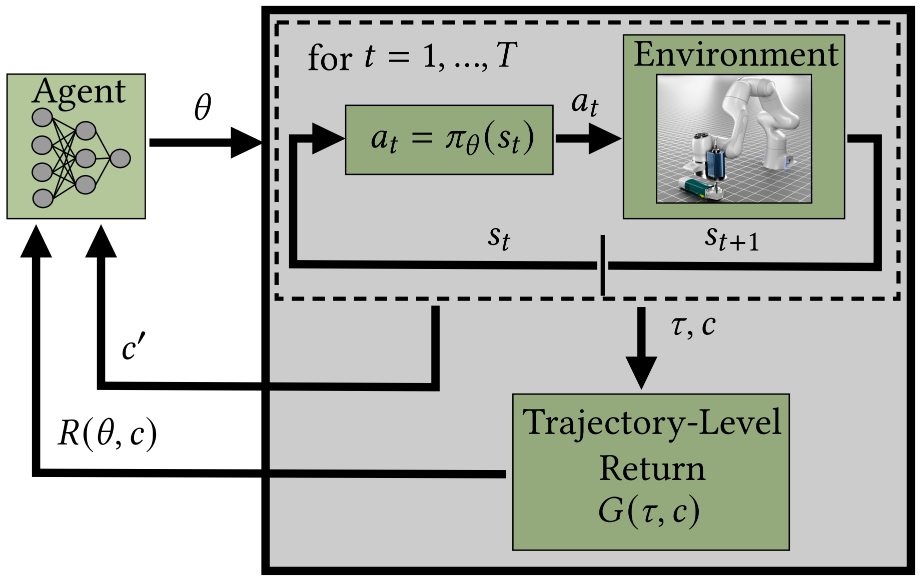


Figure 2.2.: Episode-Based Reinforcement Learning (ERL) Loop. At the beginning of each episode, the agent samples a parameter $\boldsymbol{\theta}$ from the policy $\pi_\omega(\boldsymbol{\theta}|\mathbf{c})$ which fixes the parameters of the deterministic low-level controller $\mathbf{a}_t = \pi_{\boldsymbol{\theta}}(\mathbf{s}_t)$ for the whole episode. This low-level controller is used to interact with the environment until the episode ends resulting in a trajectory $\boldsymbol{\tau}$, which is used together with the context $\mathbf{c}$ to obtain a trajectory-level return $G(\boldsymbol{\tau}, \mathbf{c})$. This trajectory-level return assesses the performance of the parameter $\boldsymbol{\theta}$ in the context $\mathbf{c}$ and is therefore referred to as $R(\boldsymbol{\theta}, \mathbf{c})$. Notably, the return $G(\boldsymbol{\tau}, \mathbf{c})$ can have any structure, e.g., it can be a Markovian return with a fixed horizon T or a non-Markovian return that requires the whole trajectory $\boldsymbol{\tau}$ to generate an assessment. Together with $R(\boldsymbol{\theta}, \mathbf{c})$, a new context $\mathbf{c}'$ that is coming from the environment after the episode ends, is returned to the agent. ERL considers the interaction with the environment together with the evaluation of the parameter $\boldsymbol{\theta}$ as black box, which is why the corresponding parts are colored in black. The samples used for updating the policy $\pi_\omega(\boldsymbol{\theta}|\mathbf{c})$ in each learning iteration are summarized as the tuple $(\mathbf{c}, \boldsymbol{\theta}, R)$, hence, the exploration in ERL is done in the parameter space $\boldsymbol{\theta}$ once at the beginning of each episode.

However, in step-based RL the exploration in the action and state-space is not time-correlated and typically not across all degrees of freedom which might lead to suboptimal and inefficient exploration mainly caused by jerky actions (Raffin et al., 2022; Li et al., 2024a). Instead of perturbing the actions in each time step, we can perturb the parameters $\boldsymbol{\theta}$ of a deterministic policy $\pi_{\boldsymbol{\theta}}(\mathbf{s}_t)$ at the beginning of an episode and keep those parameters fixed for the rest of the finite episode.
Most commonly, Motion Primitives (MPs) (Schaal, 2006; Ijspeert et al., 2013; Paraschos et al., 2013; Li et al., 2023) are used as the deterministic policy representations. MPs represent a desired trajectory $\boldsymbol{\tau}^{\text{des}}$ by a higher level abstraction in the parameter space which significantly reduces the search space (see Section 2.6.1 for a more detailed description). This approach implies a time-correlated and smooth exploration on a trajectory level, which is especially useful in robotics where jerky actions might damage the robot (Raffin et al., 2022; Li et al., 2024a). Exploring the parameters $\boldsymbol{\theta}$ of this underlying deterministic policy $\pi_\theta(\mathbf{s}_t)$ is referred to as **episode-based RL** (ERL). Although, ERL methods are not only restricted to the application of MPs as the deterministic policy $\pi_{\boldsymbol{\theta}}$ representation, this thesis focuses on MP applications in ERL only (Chapters 3 and 4).

The interaction loop of an ERL agent is illustrated in Figure 2.2, where we have colored the right part of this Figure in black to inidicate that ERL treats this part as black box and is also referred to as black box reinforcement learning approach therefore. This black-box characteristic allows a flexible structure in the return function and is not restricted to the Markov property, a feature that allows a more intuitive reward design for achieving the desired behavior. For example an environment's return to an agent that is tasked to jump as high as possible can be defined as the maximum height reached during the whole episode, which is a non-Markovian return because it requires retrospective information. This return is more intuitive than a per-step reward such as the height reached at each time step. These features frame ERL a useful tool for learning versatile skills, which we show in the Chapters 3 and 4.

In general, ERL defines a context conditioned search distribtuion $\pi_{\omega}(\boldsymbol{\theta}|\mathbf{c})$ over the parameters $\boldsymbol{\theta}$ of the deterministic policy $\pi_{\boldsymbol{\theta}}(\mathbf{s}_t)$, where $\boldsymbol{\omega}$ are the parameters of the search distribution. Those parameters of the search distribution are then optimized following the ERL objective, which we are going to discuss next.
Please note that although in ERL the policy is the deterministic low-level controller $\mathbf{a}_t = \pi_{\boldsymbol{\theta}}(\mathbf{s}_t)$, we will refer to the search distribution $\pi_{\omega}(\boldsymbol{\theta}|\mathbf{c})$ as the policy in the context of ERL in the reminder of this document, as common in literature.

2.3.1. Episode-Based Reinforcement Learning Objective

The objective for the **episode-based RL** setting can be formalized as

$$J(\pi_{\omega}) = \mathbb{E}_{p(\mathbf{c})}\left[\int_{\theta} \pi_{\omega}(\boldsymbol{\theta}|\mathbf{c}) \int_{\tau} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}|c) G(\boldsymbol{\tau}, \mathbf{c}) d\boldsymbol{\tau} d\boldsymbol{\theta}\right] \tag{2.26}$$

$$= \mathbb{E}_{p(\mathbf{c})}\left[\int_{\theta} \pi_{\omega}(\boldsymbol{\theta}|\mathbf{c}) R(\boldsymbol{\theta}, \mathbf{c}) d\boldsymbol{\theta}\right], \tag{2.27}$$

where $\mathbf{c} \in C$ is a continuous context variable that specifies the task to be solved, e.g. a position to which an object needs to be moved and is defined by the context distribution $p(\mathbf{c})$ specified by the environment and is unknown to the agent.

The aforementioned flexible return structure $G(\boldsymbol{\tau}, \mathbf{c})$ becomes more present in Eq. 2.26. Here, the optimization problem is independent of the exact definition of G as it is a black box optimization problem. For example, $G(\boldsymbol{\tau}, \mathbf{c})$ can be of *Markovian* nature with a fixed horizon T

$$G(\boldsymbol{\tau}, \mathbf{c}) = \sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c}), \tag{2.28}$$

or it can have any *non-Markovian* structure.

In Eq. 2.26 we have expressed the return $G(\boldsymbol{\tau}, \mathbf{c})$ in terms of the trajectory $\boldsymbol{\tau}$, which is a useful definition for understanding the quality assessement of a parameter $\boldsymbol{\theta}$. However, we will use the more common definition of the return as $R(\boldsymbol{\theta}, \mathbf{c})$ in Eq. 2.27 for the reminder of this thesis, as it directly illustrates the dependence on the parameters $\boldsymbol{\theta}$ and it is more compact to write.

One observation the objective in Eq. 2.27 reveals is that ERL treats each context-parameter-return tuple $(\boldsymbol{\theta}, \mathbf{c}, R(\boldsymbol{\theta}, \mathbf{c}))$ as a sample for updating the parameters $\boldsymbol{\omega}$ of $\pi_{\omega}(\boldsymbol{\theta}|\mathbf{c})$. Yet, each evaluation corresponds to executing the parameter $\boldsymbol{\theta}$ on the system for a whole episode to obtain the return $R(\boldsymbol{\theta}, \mathbf{c})$. Because ERL does not make use of the temporal structure resulting from the interaction with the environment, they are sample inefficient compared to SRL methods.

2.3.2. Episode-Based Reinforcement Learning Methods

Obtaining a policy $\pi_{\omega}(\boldsymbol{\theta}|\mathbf{c})$ that maximizes the objective in Eq. 2.27 has been tackled from different optimization perspectives in the literature. As in the step-based RL setting, the **Policy Gradient** approach has been widely researched in the episodic case (Sehnke et al., 2010). However, as the optimization in Eq. 2.27 is considered as a black box optimization problem, it also permits using gradient-free stochastic optimization methods that have been widely used in the context of episode-based RL.

2.3.2.1. Policy Gradient Methods

Similar to the *REINFORCE* algorithm in the step-based RL setting, the policy gradient version in the episode-based RL setting can be formulated as

$$\nabla_{\omega} J(\pi_{\omega}) = \int p(\mathbf{c}) \int_{\theta} \nabla_{\omega} \pi_{\omega}(\boldsymbol{\theta}|\mathbf{c}) R(\boldsymbol{\theta}, \mathbf{c}) d\boldsymbol{\theta} d\mathbf{c} \tag{2.29}$$

$$= \int p(\mathbf{c}) \int_{\theta} \pi_{\omega}(\boldsymbol{\theta}|\mathbf{c}) \nabla_{\omega} \log \pi_{\omega}(\boldsymbol{\theta}|\mathbf{c}) R(\boldsymbol{\theta}, \mathbf{c}) d\boldsymbol{\theta} d\mathbf{c} \tag{2.30}$$

$$\approx \sum_{i=0}^{N} \nabla_{\omega} \log \pi_{\omega}(\boldsymbol{\theta}^i|\mathbf{c}^i) R(\boldsymbol{\theta}^i, \mathbf{c}^i). \tag{2.31}$$

Following the same motivation as stated in Section 2.2.4.2, we can include a baseline $b(c)$ to reduce the variance of the gradient estimate in Eq. 2.31. Commonly the pendant to the value function in the step-based RL setting is used as a baseline, i.e. $b(\mathbf{c}) = V^{\pi}(\mathbf{c})$, where $V^{\pi}(\mathbf{c}) = \int_{\theta} \pi_{\omega}(\boldsymbol{\theta}|\mathbf{c}) R(\boldsymbol{\theta}, \mathbf{c})$, which leads to the advantage function

$$A^{\pi}(\boldsymbol{\theta}, \mathbf{c}) = R(\boldsymbol{\theta}, \mathbf{c}) - V^{\pi}(\mathbf{c}), \tag{2.32}$$

and finally to the episode-based policy gradient estimate

$$\nabla_{\omega} J(\pi_{\omega}) \approx \sum_{i=0}^{N} \nabla_{\omega} \log \pi_{\omega}(\boldsymbol{\theta}^i|\mathbf{c}^i) A^{\pi}(\boldsymbol{\theta}^i, \mathbf{c}^i). \tag{2.33}$$

The policy gradient in the episode-based RL setting was employed in early works such as (Peters and Schaal, 2008; Sehnke et al., 2010) and has been extended to the natural gradient using evolution strategies (ES) (Wierstra et al., 2014). More recently it has been shown to work well in more complex policy parameterizations $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c})$, such as deep neural networks (Bahl et al., 2020; Otto et al., 2023). Notably, the work by Otto et al. (2023) has extended the policy gradient approach to the trust-region constrained pendant of the step-based RL setting (Otto et al., 2021) for deep networks. It optimizes the objective

$$J(\pi_{\boldsymbol{\omega}}) = \mathbb{E}_{p(\mathbf{c}),\pi_{\boldsymbol{\omega}_{old}}(\boldsymbol{\theta}|c)}\left[\frac{\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c})}{\pi_{\boldsymbol{\omega}_{old}}(\boldsymbol{\theta}|\mathbf{c})}A^{\pi_{\boldsymbol{\omega}_{old}}}(\boldsymbol{\theta},\mathbf{c})\right] \quad \text{s.t. } \mathrm{KL}\left(\pi_{\boldsymbol{\omega}}(\cdot|\mathbf{c})||\pi_{\boldsymbol{\omega}_{old}}(\cdot|\mathbf{c})\right) \leq \epsilon \;\; \forall \mathbf{c} \in C. \tag{2.34}$$

Optimizing the policy gradient objective using a trust region constraint is specifically important as small changes in the parameter $\boldsymbol{\omega}$ usually have a big impact on the result and therefore, the policy change should be constrained during updates (Otto et al., 2023). This optimization approach is particularly relevant to this thesis, as it is used in Chapter 4 to optimize single experts for learning versatile skills.

Later works have extended the policy gradient approach to use segments of the trajectory $\boldsymbol{\tau}$ for improved sample efficiency (Li et al., 2024a) and further introduced off-policy updates (Li et al., 2025).

2.3.2.2. Gradient-Free Methods

Trust-Region Based Methods. Similar to the trust-region based policy gradient approach (Otto et al., 2023), Contextual Relative Entropy Policy Search (CREPS) (Kupcsik et al., 2013; Daniel et al., 2012) proposes optimizing a trust-region constrained optimization problem

$$\max_{\pi_{\boldsymbol{\omega}}} \int p(\mathbf{c}) \int \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c}) R(\boldsymbol{\theta},\mathbf{c}) d\boldsymbol{\theta} d\mathbf{c} \tag{2.35}$$

$$\text{s.t. } \mathbb{E}_{p(\mathbf{c})}\left[\mathrm{KL}\left(\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c})||\pi_{\boldsymbol{\omega}_{old}}(\boldsymbol{\theta}|\mathbf{c})\right)\right] \tag{2.36}$$

$$\int \pi(\boldsymbol{\theta}|\mathbf{c}) d\boldsymbol{\theta} = 1 \forall \mathbf{c}. \tag{2.37}$$

The notable difference of this approach to TRPL (Eq. 2.34) is that CREPS does not employ the trust region constraint for each context $\mathbf{c} \in C$ as TRPL (see Eq. 2.34) and it does not rely on a gradient-based approach for finding the optimal policy $\pi^*_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c})$. Instead, CREPS provides a closed-form solution for the optimal policy $\pi^*_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c})$ that follows from the Lagrangian formulation (Boyd and Vandenberghe, 2004) as

$$\pi^*_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c}) \propto \pi_{\boldsymbol{\omega}_{old}}(\boldsymbol{\theta}|\mathbf{c}) \exp\left(\frac{R(\boldsymbol{\theta},\mathbf{c})}{\lambda}\right), \tag{2.38}$$

where λ is the Lagrangian multiplier to the trust region constraint and $V(\mathbf{c})$ is the value function for the context $\mathbf{c}$. The optimal Lagrangian multiplier λ is given by optimizing the corresponding dual function. However, this closed-form solution does not permit a parametric representation of the policy such that additionally a weighted maximum likelihood estimation (MLE) needs to be done to fit a parameterized policy. CREPS' main drawback is therefore that it is not guaranteed that the new parametric policy obeys the trust region constraint.

Because of this reason, Abdolmaleki et al. (2015) and Tangkaratt et al. (2017) propose approximating the numerator in the exponential of Eq. 2.38 using a second order polynomial model such that a parametric closed form solution for $\pi_\omega(\boldsymbol{\theta}|\mathbf{c})$ can be obtained. Because of the second-order model, this method is referred to as Contextual Model Based Relative Entropy (CMORE) method and has been shown to obtain improved results over CREPS. It is of particular relevance to this thesis, as it is used in Chapter 3 for learning a linear expert that can solve multiple tasks. Yet, it is important to note that CMORE is restricted to linear policies and was therefore not used in Chapter 4 for optimizing highly non-linear policies.

Evolutionary Strategies. Another gradient-free approach for stochastic optimization problems are methods based on Evolutionary Strategies (ES) (Whitley et al., 1993; Hansen and Ostermeier, 2001; Igel, 2003; Salimans et al., 2017; Abdolmaleki et al., 2019). However, these methods are usually hard to apply in the contextual episode-based RL setting as they do not take the context $\mathbf{c}$ into account when evaluating the parameters $\boldsymbol{\theta}$. Hence, parameters that have returned a good return for one context $\mathbf{c}_1$ might not lead to high returns for another context $\mathbf{c}_2$ (Otto et al., 2023). A notable difference is contextual CMA-ES (Abdolmaleki et al., 2019) which is a contextual extension of the CMA-ES algorithm (Hansen and Ostermeier, 2001), but is restricted to linear policies.

2.4. Maximum Entropy Reinforcement Learning

In the well-known fully-observable MDP from Section 2.2.1 and the corresponding RL objective in Eq. 2.4 the optimal policy $\pi^*_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)$ is deterministic (Sutton and Barto, 2018). However, often times a stochastic policy $\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)$ that spans a distribution over the possible actions and simultaneously maximizes the expected return is preferred. The maximum entropy RL framework provides a principled way to learn such stochastic policies by augmenting the reward with the entropy of the policy distribution. This objective not only leads to a distribution over actions but also encodes a notion of controlling the exploration of the policy within the objective and hence does not require explicit treatment of exploration, e.g., by adding noise to the action of a deterministic policy (Lillicrap et al., 2016; Fujimoto et al., 2018) or as in the more common case initializing the policy as a distribution with high entropy and carefully restricting the information loss per iteration by additional techniques such as bounding the Kullback Leibler divergence in intermediate iterations (Kakade, 2001; Schulman et al., 2015, 2017; Otto et al., 2021; Peters et al., 2010). Additionally, it is shown that optimizing the maximum entropy RL yields robust policies against perturbations in the reward signal and the environment dynamics (Eysenbach and Levine, 2022), which is particularly useful in real world situations.

This thesis heavily relies on the advantages of the maximum entropy RL objective in terms of exploration for learning versatile and diverse skills, which is why we will present it in more detail in the following for the step-based and episode-based setting.

2.4.1. Step-Based Maximum Entropy Reinforcement Learning

In the step-based RL setting, the maximum entropy RL objective is defined as

$$J(\pi_\theta) = \mathbb{E}_{p_\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}\left(\pi_\theta(\cdot|\mathbf{s}_t)\right)\right)\right], \tag{2.39}$$

where $\alpha \geq 0$ is an entropy scaling parameter that controls the trade-off between maximizing the task reward $r(\mathbf{s}_t, \mathbf{a}_t)$ and the entropy $\mathcal{H}\left(\pi_\theta(\cdot|\mathbf{s}_t)\right)$ of the policy. Consequently, higher α values lead to more exploration, whereas $\alpha = 0$ recovers the standard RL objective in Eq. 2.4.

With the entropy-augmented objective in Eq. 2.39, also the state-action value function $Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$ for state $\mathbf{s}_t \in \mathcal{S}$ and action $\mathbf{a}_t \in \mathcal{A}$ in time step t changes and is defined as the soft Q-function $Q^\pi_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t)$ as (Haarnoja et al., 2017)

$$Q^\pi_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \mathbb{E}_{p_\pi}\left[\sum_{k=1} \gamma^k \left(r(\mathbf{s}_{t+k}, \mathbf{a}_{t+k}) + \alpha \mathcal{H}\left(\pi_\theta(\cdot|\mathbf{s}_{t+k})\right)\right)\right], \tag{2.40}$$

and its corresponding soft value function $V^\pi_{\text{soft}}(\mathbf{s}_t)$ as

$$V^\pi_{\text{soft}}(\mathbf{s}_t) = \alpha \log \int \exp\left(\frac{1}{\alpha} Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}')\right) d\mathbf{a}'. \tag{2.41}$$

The optimal policy that maximizes the maximum entropy RL objective is then given as the energy-based distribution (Ziebart, 2010; Haarnoja et al., 2017)

$$\pi^*(\mathbf{a}_t|\mathbf{s}_t) = \exp\left(\frac{1}{\alpha}\left(Q^\pi_{\text{soft}}(\mathbf{a}_t, \mathbf{s}_t) - V^\pi_{soft}(\mathbf{s}_t)\right)\right). \tag{2.42}$$

These relations are particularly important in Chapter 5, where a Policy Iteration scheme for the diffusion-based policy with tailored definitions are derived.

Among the known algorithms that optimize the maximum entropy RL objective in the continuous state-action domain is Soft Q-Learning (Haarnoja et al., 2017) which leverages advanced sampling techniques such as Stein variational gradient descent (Wang and Liu, 2016) to train a sampling network to sample from the optimal energy-based policy distribution in Eq. 2.42. However, training the sampling network involves importance sampling which is known to have high variance, especially in high-dimensional tasks (Messaoud et al., 2024).

Therefore, this framework was extended to a tractable family of parameterized policies in Soft Actor Critic (SAC) (Haarnoja et al., 2018b) which is among the most well-known maximum entropy RL algorithms. SAC learns a soft Q-function $Q_\phi(\mathbf{s}, \mathbf{a})$ defined as 2.40 and is therefore categorized as an off-policy actor-critic method. SAC's ease of implementation and great performance has motivated researchers to build upon this algorithm to e.g. scale the network sizes in RL (Nauman et al., 2024), or improve the performance (Bhatt et al., 2024) using batch normalization techniques.

In fact, it can be shown that framing the reinforcement learning problem as a probabilistic inference problem leads to a solution that optimizes the maximum entropy objective (Levine, 2018). Related to this, works in the field of optimal control have formulated the optimization problem as an approximate inference problem (Toussaint, 2009; Todorov, 2008; Rawlik et al., 2012), or used the maximum entropy principle to learn the reward function in the context of inverse reinforcement learning (Ziebart et al., 2008).

In Section 2.7.5 we reformulate the maximum entropy RL objective as approximate inference. This relation has provided us with inspiration from recent advancements in the field of approximate inference with diffusion models (e.g. (Berner et al., 2024; Richter and Berner, 2024)) and has led to developping Diffusion-based Maximum Entropy RL (DIME) in Chapter 5.

2.4.2. Episode-Based Maximum Entropy Reinforcement Learning

The maximum entropy RL objective can also be applied to the episode-based RL setting

$$J(\pi_\omega) = \mathbb{E}_{p(\mathbf{c})}\left[\int \pi_\omega(\boldsymbol{\theta}|\mathbf{c})(R(\boldsymbol{\theta}, \mathbf{c}) + \alpha\mathcal{H}\left(\pi_\omega(\cdot|\mathbf{c})\right) d\boldsymbol{\theta}\right]. \tag{2.43}$$

Here, instead of maximizing the entropy of the policy in the raw action space $\mathbf{a}_t$ as in the step-based RL case, the entropy of the policy π_ω over the parameters $\boldsymbol{\theta}$ is maximized. This implies that this objective tries to find all possible parameters $\boldsymbol{\theta}$ that maximize the return $R(\boldsymbol{\theta}, \mathbf{c})$. As in Section 2.4.1, the entropy scaling parameter $\alpha \geq 0$ controls the trade-off between maximizing the return and the entropy of the policy.

In Section 2.7.4 we show that the Variational Inference (VI) objective draws parallels to the maximum entropy RL objective. We build on this observation and leverage techniques from variational inference (Arenz et al., 2018, 2020; Becker et al., 2020) to optimize Mixture of Experts (MoE) policies in Sections 3 and 4 for learning diverse and versatile skills on a trajectory level.

2.5. Advancing Reinforcement Learning using Curriculum Learning

Numerical continuation methods (Allgower and Georg, 1990) are a class of numerical methods that try to find the optimal solution of a target function by first optimizing a smoothed version of the target and then gradually optimizing less smoothed versions of the target function using the found solution from the previous iteration. Motivated by these methods, Bengio et al. (2009) introduced curriculum learning as a learning paradigm that improves the learning process by presenting the learning agent with a sequence of tasks that are ordered by their difficulty. Many works have built upon this learning framework and have shown improved results in various domains where the curriculum was either defined in the increasing complexity of the model (Karras et al., 2018; Morerio et al., 2017; Sinha et al., 2020), or in the increasing task complexity (Florensa et al., 2017; Lotter et al., 2017; Sarafianos et al., 2017). Most commonly those methods expect the curriculum to be defined beforehand and do not consider the agent to adapt the curriculum during the learning process. Automatic curriculum adaptation is also known as **Self-Paced Learning** and is the more convenient way for the user as it is not required to define the pace of the difficulty of the tasks.

Self-Paced Learning. Self-Paced Learning is a crucial part of this thesis for acquiring versatile and diverse skills as discussed in Chapters 3 and 4. More concretely, the methods presented in the Chapter 3 and 4 take inspiration from the following objective to address *Challenge 2*

$$J(\pi_\omega) = \mathbb{E}_{\pi_\omega(\mathbf{c})}\left[\int \pi_\omega(\boldsymbol{\theta}|\mathbf{c})R(\boldsymbol{\theta},\mathbf{c})d\boldsymbol{\theta}\right] - \beta\mathrm{KL}\left[\pi_\omega(\mathbf{c})\middle\|p(\mathbf{c})\right], \tag{2.44}$$

where the notable difference to the standard episode-based RL objective in Eq. 2.27 is the difference in the expectation over the context distribution that goes over a learnable distribution π_ω instead of the environment's context distribution $p(\mathbf{c})$ and the augmented KL regularization that incentivizes π_ω to be near to the true distribution $p(\mathbf{c})$ of the environment. The scaling parameter β controls the trade-off between maximizing the return and minimizing the KL regularization and can be seen as the hyperparameter that controls the difficulty of the tasks in the curriculum. Higher β values encourage full coverage of the true context distribution, i.e. all possible tasks, whereas lower β values encourage the agent to choose contexts that are easier to solve and lead to higher returns. This objective was proposed in the work by Klink et al. (2020a) and has shown to work well in the episode-based RL setting.

Notably, the objective in Eq. 2.44 assumes that the agent can define the task, i.e., which context $\mathbf{c}$ to solve next during training time, which is generally not the case in the standard RL setting, where the context is defined by the environment. However, intuitively this assumption makes sense as humans also consider easier tasks first during learning and transfer their knowledge to harder tasks later on. We therefore, consider this assumption as reasonable and useful for enabling self-paced learning. During test time, the agent only observers contexts from the environment's context distribution $p(\mathbf{c})$.

We take this objective as inspiration in Chapters 3 and 4 and show how we can use it to learn a per-expert curriculum which successfully covers the whole context space minimizing the KL augmentation in Eq. 2.44. A more detailed review of curriculum learning approaches in reinforcement learning is discussed in Chapter 4.

2.6. Policy Representations

Most commonly, Gaussian parameterized policies in both the step-based and episode-based RL settings are used to represent the policy $\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)$ and $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c})$ respectively. In both setups, the mean $\boldsymbol{\mu}$ is a function of the state $\mathbf{s}_t$, or context $\mathbf{c}$, and is most commonly parameterized by a neural network. The covariance Σ is either a learnable matrix, or a function of the state $\mathbf{s}_t$, or context $\mathbf{c}$. Gaussian policies are the most common choice for representing policies as they are easy to implement and have tractable statistics, i.e. their likelihoods can be evaluated in closed form. Consequently optimizing their parameters is usually convenient.

However, in some cases, it is beneficial to use more complex policy representations such as for improved exploration, or, for capturing multi-modality that lead to completely different behaviors for the same task.

In the following, we first discuss Motion Primitives, a common representation for trajectories used in episode-based RL methods. Subsequently, we discuss two classes of latent variable models. We start with Mixture of Experts (MoE) models that we leverage as policy representations in Chapters 3 and 4 to learn versatile and diverse skills, and continue presenting diffusion-based models that can be seen as discrete latent variable models and that hold the promise to learn complex and multi-modal policies. In Chapter 5 we show how diffusion-based policies can be trained in the maximum entropy RL framework.

2.6.1. Motion Primitives

Motion Primitives (MPs). (MPs) are an elegant class of approaches that can represent the behavior of an agent using a parameterized trajectory in a compact way which is why they are broadly used in episode-based RL. Depending on the control domain of the robot, either positions can directly be set as actions, or some tracking controller can be used to follow the trajectory. MPs successfully abstract a trajectory such that they can be represented in a lower dimensional space, which significantly reduces the complexity of the search problem. Additionally, the resulting trajectory is smooth and continuous, a particularly important property in robotics to avoid jerky actions (Raffin et al., 2022; Li et al., 2023).

One of the approaches to MPs is **Dynamic Movement Primitives** (DMPs) (Schaal, 2006; Ijspeert et al., 2013). Given a parameter θ, DMPs generate a trajectory by integrating the second order differential equation

$$\gamma^2 \ddot{y}(t) = \alpha\left(\beta(g - y(t)) - \gamma \ddot{y}(t)\right) + f_\theta(x), \qquad \text{with } f_\theta(x) = x\phi(x)^T\theta, \tag{2.45}$$

where $y = y(t)$, $\dot{y} = \mathrm{d}y/dt, \ddot{y} = \mathrm{d}^2y/dt^2$ are the position, velocity, and acceleration of the system at time step t, g is the goal attractor for the state of the system, α and β are the spring-damper constants and γ is the time constant that controls the execution speed. By adapting the parameters θ, the integrated trajectory can be manipulated over the forcing function

$f_\theta(x)$. The influence of the forcing function is controlled over the exponentially decaying phase variable $x(t) = \exp(-\alpha_x/\gamma t)$. Here, $\phi(x)$ are basis functions that are fixed in the learning process. While DMPs provide an elegant framework for generating trajectories for any starting position $y(0)$ by definition, they require integrating the differential equation for each trial of θ, which makes them computationally expensive.

An alternative approach to DMPs are **Probabilistic Movement Primitives** (ProMPs) (Paraschos et al., 2013), which provide a probabilistic framework to capture the distribution over trajectories. Capturing the distribution over trajectories in terms of the parameters $\boldsymbol{\theta}$ is particularly useful in the context of imitation learning where a training set over trajectories $\boldsymbol{\tau}$ is given. However, in RL this is an uncommon situation, but in contrast to DMPs, the trajectory generation for a specific realization of the parameter $\boldsymbol{\theta}$ is particularly convenient through the linear relation

$$\boldsymbol{\tau} = \boldsymbol{\phi}^T \boldsymbol{\theta}, \tag{2.46}$$

where $\boldsymbol{\phi}$ are the basis functions over the time steps. This linear relation makes trajectory generation computationally efficient and easy to implement in the context of RL. A drawback of ProMPs is that they do not provide an intuitive way to start a trajectory from arbitrary initial positions, requiring careful treatment by e.g. always starting the trajectory from the origin and adding a difference to the initial state of the system instead of considering the absolute state trajectory. Nevertheless due to their ease how the trajectory is generated in Eq. 2.46, we have used ProMPs as a controller parameterization in Chapters 3 and 4.

Finally, **Probabilistic Dynamic Movement Primitives** (ProDMPs) (Li et al., 2023) unify the ideas and advantages of both approaches to ProDMPs by pre-calculating expensive numerical integration in DMPs and leveraging the linear relation of data generation in ProMPs, where we refer the interested reader to the detailed description in their work by Li et al. (2023). Important to notice for this thesis is that ProDMPs generate a trajectory $\boldsymbol{\tau}$ similar to the way ProMPs do, but also allow to generate the trajectory from arbitrary initial conditions and additionally allow learning a goal attractor $\mathbf{g}$ which was not possible in ProMPs and is particularly useful in goal-conditioned RL tasks. We consider ProDMPs as a controller representation in Chapter 4.

2.6.2. Mixture of Experts

Modeling complex distributions can be done using a superposition of simpler distributions such as mixture models. The next paragraph is going to discuss mixture models, where we take inspiration and follow the definitions similar to Bishop (2006).

Mixture Models. Mixture models provide a principled formalization for the idea of a superposition of simpler distributions and are categorized into the field of discrete latent variable models. Formally, a mixture model over a random variable $\mathbf{x}$ is defined as

$$p(\mathbf{x}) = \sum_{o}^{N} p(o)p(\mathbf{x}|o), \tag{2.47}$$

where $o = 1, 2, ..., N$ is a discrete latent variable that is the index specifying the component of the mixture model, and $p(o)$ is a categorical distribution $p(o) = \text{Cat}(p_1, p_2, ..., p_N)$ assigning a probability to each component o. The conditional distribution $p(\mathbf{x}|o)$ is the component distribution. Mixture models are a widely used model for representing complex distributions because given a sufficiently large number of components N, they can approximate any complex distribution (Bishop, 2006) and they can usually be trained effectively in parallel. Their easy structure gives room for interpretability, as each component o is usually responsible for a subset of the data.

Most commonly, the components $p(\mathbf{x}|o)$ are parameterized as a Gaussian distribution such that the model in Eq. 2.47 becomes a **Gaussian Mixture Model** (GMM).

In Chapter 3 we will use GMMs as a parameterized distribution over the context $\mathbf{c}$ to represent a curriculum. However, we will also consider mixture models as a policy representation in Chapter 3 and 4, where those chapters refer to the context conditioned mixture model from Eq. 2.47 as **Mixture of Experts** (MoE).

Following Eq. 2.47 we can write an MoE distribution over the parameters $\boldsymbol{\theta}$ given a context $\mathbf{c}$ as

$$\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c}) = \sum_{o=1}^{N} \pi(o|\mathbf{c})\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|o, \mathbf{c}), \tag{2.48}$$

where $\pi(o|\mathbf{c})$ is the responsibility given the context $\mathbf{c}$ and $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|o, \mathbf{c})$ is the expert distribution for the component o given context $\mathbf{c}$. We will consider Gaussian experts

$$\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|o, \mathbf{c}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{f}_{\boldsymbol{\omega}_o}(\mathbf{c}), \Sigma_o) \tag{2.49}$$

with either linear relationship in $\mathbf{f}_{\boldsymbol{\omega}}$ in Chapter 3, or a non-linear relationship using neural networks in Chapter 4. In general, we will not write $\boldsymbol{\omega}_o$ to indicate the parameters of the expert o but simply write $\boldsymbol{\omega}$ as it contains all parameters of the whole model and the discrete latent variable o assigns the respective parameters within $\boldsymbol{\omega}$ to the expert distribution $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|o, \mathbf{c})$.

The responsibility $\pi(o|\mathbf{c})$ assigns each expert o a probability for a given $\mathbf{c}$. This probability measure indicates which expert $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|o, \mathbf{c})$ is *responsible* for $\mathbf{c}$. The responsibility plays a major role in learning versatile and diverse skills as will be discussed in detail in Chapter 3.

Discrete latent variable models such as MoEs can be used to approximate complex and untractable distributions. A common approach for optimizing the parameters of the MoE is by formulating this approximate inference as an optimization problem. In Section 2.7 we present the general objective for optimizing MoEs in the context of approximate inference and discuss parallels to the episode-based maximum entropy RL objective (Section 2.4.2) which we leverage to optimize MoE policies in Chapter 3 and 4. In these chapters each expert of the MoE policy represents a skill using contextualized Motion Primitive Parameters (Section 2.6.1).

2.6.3. Diffusion Models

Diffusion models are continuous-time stochastic processes (Song et al., 2021). A general stochastic differential equation (SDE) on the time interval $[0, T]$

$$d\mathbf{x}_t = \mathbf{g}(\mathbf{x}_t)dt + \sigma_t d\mathbf{B}_t \tag{2.50}$$

describes the path of the random variable $\mathbf{x}_t$ over time t, where $\mathbf{g} : \mathcal{R}^d \times [0, T] \rightarrow \mathcal{R}^d$ is the drift term, $\sigma : [0, T] \rightarrow \mathcal{R}^+$ is the diffusion coefficient and $(\mathbf{B}_t)_{t\in[0,T]}$ is a Brownian motion. The goal is to find a suitable $\mathbf{g}$ and σ together with a suitable initial condition $\mathbf{x}_T$ such that the final distribution of $\mathbf{x}_0$ approximates a target distribution $p(\mathbf{x}_0)$ (Berner et al., 2024).

Various variants of diffusion models together with a suitable optimization scheme have been proposed in the literature (Song et al., 2021; Vargas et al., 2024; Richter and Berner, 2024; Blessing et al., 2025a,b; Vargas et al., 2023). We refer the interested reader to the survey by Blessing et al. (2024).

In this thesis, we consider Denoising Diffusion Models, where a *forward*, or *noising process* is given by the Ornstein-Uhlenbeck (OU) process (Särkkä and Solin, 2019)

$$d\mathbf{x}_t = -\beta_t \mathbf{x}_t dt + \eta\sqrt{2\beta_t} d\mathbf{B}_t, \quad \mathbf{x}_0 \sim p(\mathbf{x}_0), \tag{2.51}$$

with the diffusion coefficient $\beta : [0, T] \rightarrow \mathcal{R}^+$, Brownian motion $(\mathbf{B}_t)_{t\in[0,T]}$, and a target distribution $p(\mathbf{x}_0)$. We refer to the marginal density of this forward process at time step t as $\vec{p}_t$. This process is referred to as *noising process* because for a suitable β, the initial distribution $p_T(x_T) \approx \mathcal{N}(0, \eta^2 I)$ is approximately a zero-mean Gaussian distribution.

The corresponding *time-reversal*, or *backward generative process* to the process in Eq. 2.51 is given by the SDE

$$d\mathbf{x}_t = \left(-\beta_t \mathbf{x}_t dt - 2\eta^2 \beta_t \nabla \log \vec{p}_t(\mathbf{x}_t)\right) + \eta\sqrt{2\beta_t} d\mathbf{B}_t, \quad \mathbf{x}_T \sim \mathcal{N}(0, \eta^2 I), \tag{2.52}$$

which starts at time step T and runs backward in time (Anderson, 1982), and the marginal density of this backward process at time step t is defined as $\overleftarrow{p}_t$. This process is referred to as the *time-reversal process* because the marginal densities of the processes in Equations 2.51 and 2.52 at time t align, i.e. $\vec{p}_t = \overleftarrow{p}_t \ \ \forall t \in [0, T]$.

Therefore, first sampling $\mathbf{x}_T \sim \mathcal{N}(0, \eta^2 I)$ and simulating the process in Eq. 2.52 generates samples from the target distribution $p(\mathbf{x}_0)$. However, most commonly, $(\nabla \log \vec{p}_t(\mathbf{x}_t))_{t\in[0,T]}$

are not known, requiring an approximation of the score function $\nabla \log \vec{p}_t(\mathbf{x}_t)$, which then needs to be trained using a suitable optimization scheme. In supervised learning settings, samples of the target distribution $p(\mathbf{x}_0)$ are available, which can be used to train the score function using a score matching objective (Song et al., 2021). Yet in the context of RL, the target distribution is not known and requires generating those samples with the current policy. However, this process requires further formulations for a tractable objecive.

While learning of a parameterized function f_t^θ as an approximation for the score $\nabla \log \vec{p}_t(\mathbf{x}_t)$ in continuous time is considered in the approximate inference literature (Berner et al., 2024; Nusken et al., 2024), this thesis focuses on the discrete-time formulation in Chapter 5 which is more accessible to the RL community as it does not require knowledge in stochastic calculus.

As detailed in Chapter 5, the SDEs in Equations 2.51 and 2.52 can be discretized using the Euler-Maruyama method (Särkkä and Solin, 2019), that lead to the joint distributions of the noising and denoising process as

$$\vec{p}_{0:N}(\mathbf{x}^{0:N}) = \vec{p}_0(\mathbf{x}^0) \prod_{n=0}^{N-1} \vec{p}_{n+1|n}(\mathbf{x}^{n+1}|\mathbf{x}^n) \tag{2.53}$$

$$\overleftarrow{p}_{0:N}(\mathbf{x}^{0:N}) = \overleftarrow{p}_N(\mathbf{x}^N) \prod_{n=1}^{N} \overleftarrow{p}_{n-1|n}(\mathbf{x}^{n-1}|\mathbf{x}^n), \tag{2.54}$$

where $\vec{p}_{n+1}(\mathbf{x}^{n+1}|\mathbf{x}^n)$ and $\overleftarrow{p}_{n-1|n}(\mathbf{x}^{n-1}|\mathbf{x}^n)$ are Gaussian kernels that follow from the discretization of the SDEs in Equations 2.51 and 2.52 using the Euler-Maruyama method (see Section 5.4). We refer to the time step of the marginal distribution as a subscript and the time step of the random variable as a superscript.

With the discretization of the SDEs, we can interpret the marginal distribution of the backward diffusion process as a continuous latent variable model

$$\overleftarrow{p}_0(\mathbf{x}^0) = \int \overleftarrow{p}_{0:N}(\mathbf{x}^{0:N}) d\mathbf{x}^{1:N}, \tag{2.55}$$

where the latent variables are the time-discretized random variables $\mathbf{x}^{1:N}$.

Similar to discrete latent variable models such as MoEs (Section 2.6.2), continuous latent variable models can be optimized by formulating approximate inference as an optimization problem. In Section 2.7 we present the general objective for optimizing diffusion models in the context of approximate inference and discuss parallels to the step-based maximum entropy RL objective which we leverage to optimize diffusion policies in Chapter 5.

Please note that an extension to conditional diffusion models as consiediered in RL and in this thesis is straightforward by conditioning the marginal distributions on a state $\mathbf{s}$. The corresponding notation is introduced in Chapter 5 in Section 5.3.2.

2.7. Optimizing Latent Variable Models via Variational Inference

In this thesis we propose reinforcement learning (RL) methods to optimize Mixture of Experts (MoEs) and diffusion-based policies for learning versatile skills using the maximum entropy RL objective for episode-based RL (ERL) and step-based RL (SRL) respectively. Those methods take inspiration from the Variational Inference (VI) literature, in which approximate inference is framed as an optimization problem (Murphy, 2012) and has tight connections to the maximum entropy RL objective. Therefore, this section aims to give an overview of the connection of the VI objective to the maximum entropy RL objective. For presenting the following section, we take inspiration and follow Bishop (2006) and Murphy (2012).

Variational Inference seeks to optimize the parameters of a tractable distribution $p(\mathbf{x})$ such that a target distribution $q(\mathbf{x})$ is approximated. The target distribution is intractable and unknown, but it is assumed that samples $\mathbf{x}$ can be evaluated on the unnormalized target density $\tilde{q}(\mathbf{x})$. While any distance and divergence measure $D(p(\mathbf{x})||q(\mathbf{x}))$ can be used as objective, the reverse Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) is from particular interest for this thesis. Using the KL, the objective is defined as

$$\arg\min_{p(\mathbf{x})} \mathrm{KL}\left(p(\mathbf{x})||q(\mathbf{x})\right) = \arg\min_{p(\mathbf{x})} \int p(x) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}. \tag{2.56}$$

Because the target distribution can be evaluated on samples $\mathbf{x}$ up to the normalizing constant $\mathcal{Z}$, the objective is rewritten as

$$\arg\min_{p(\mathbf{x})} \mathrm{KL}\left(p(\mathbf{x})||q(\mathbf{x})\right) = \arg\min_{p(\mathbf{x})} \int p(x) \log \frac{p(\mathbf{x})}{\tilde{q}(\mathbf{x})} d\mathbf{x} + \log \mathcal{Z}, \tag{2.57}$$

where $\log \mathcal{Z}$ is independent of x and can be ignored in the optimization therefore.

Similarly, we write the objective for the conditional case as the expected KL

$$\arg\min_{p(\mathbf{x}|\mathbf{c})} \mathbb{E}_{p(\mathbf{c})} \left[\mathrm{KL}\left(p(\mathbf{x}|\mathbf{c})||q(\mathbf{x}|\mathbf{c})\right)\right] = \arg\min_{p(\mathbf{x}|\mathbf{c})} \mathbb{E}_{p(\mathbf{c})} \left[\int p(\mathbf{x}|\mathbf{c}) \log \frac{p(\mathbf{x}|\mathbf{c})}{q(\mathbf{x}|\mathbf{c})} d\mathbf{x}\right], \tag{2.58}$$

where $\mathbf{c}$ is a context to the system. Following the same reasoning as for the marginal case, the target distribution $q(\mathbf{x}|\mathbf{c})$ can be evaluated up to a normalizing constant $\mathcal{Z}(\mathbf{c})$, leading to the objective

$$\arg\min_{p(\mathbf{x}|\mathbf{c})} \mathbb{E}_{p(\mathbf{c})} \left[\int p(\mathbf{x}|\mathbf{c}) \log \frac{p(\mathbf{x}|\mathbf{c})}{\tilde{q}(\mathbf{x}|\mathbf{c})} d\mathbf{x} + \log \mathcal{Z}(\mathbf{c})\right]. \tag{2.59}$$

By observing that $\log \mathcal{Z}(\mathbf{c})$ is independent of $\mathbf{x}$, we can ignore it during the optimization.

2.7.1. Variational Inference for Latent Variable Models

Simply inserting a latent variable model $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})dz$ into the Objective in Eq. 2.56 results in an objective that is not straightforward to optimize because the integral over the latent variable appears in the logarithm (Bishop, 2006; Arenz et al., 2018).

Instead, the chain rule for the KL divergence (Cover, 1999) can be used to rewrite the objective as

$$\mathrm{KL}\left(p_\theta(\mathbf{x})||q(\mathbf{x})\right) = \underbrace{\mathrm{KL}\left(p_\theta(\mathbf{x}, \mathbf{z})||q_\phi(\mathbf{x}, \mathbf{z})\right)}_{\mathcal{L}(\theta,\phi)} - \underbrace{\mathbb{E}_{p_\theta(\mathbf{x})}\left[\mathrm{KL}\left(p_\theta(\mathbf{z}|\mathbf{x})||q_\phi(\mathbf{z}|\mathbf{x})\right)\right]}_{\mathcal{J}(\phi,\theta)}, \tag{2.60}$$

which introduces the KL between the joint distributions $p_\theta(\mathbf{x}, \mathbf{z})$ and $q_\phi(\mathbf{x}, \mathbf{z})$ and the expected KL over the posterior distributions $p_\theta(\mathbf{z}|\mathbf{x})$ and $q_\phi(\mathbf{z}|\mathbf{x})$, where $q_\phi(\mathbf{z}|\mathbf{x})$ is also referred to as the variational distribution. Note that the joint distribution $q_\phi(\mathbf{x}, \mathbf{z})$ still relates to the target distribution $q(\mathbf{x})$ by

$$q_\phi(\mathbf{x}, \mathbf{z}) = q_\phi(\mathbf{z}|\mathbf{x})q(\mathbf{x}). \tag{2.61}$$

The optimization of the objective in Eq. 2.56 is tractable and can be done in an Expectation-Maximization (EM) like scheme (Dempster et al., 1977).

In the M-step, the parameters $\boldsymbol{\phi}$ are fixed and the parameters $\boldsymbol{\theta}$ of the latent variable model $p_\theta(\mathbf{x})$ are optimized by

$$\boldsymbol{\theta}_{i+1} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}^{(i)}), \tag{2.62}$$

and in the E-step, the parameters $\boldsymbol{\theta}$ are fixed and the parameters $\boldsymbol{\phi}$ of the variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ are optimized by

$$\boldsymbol{\phi}_{i+1} = \arg\min_{\boldsymbol{\phi}} \mathcal{J}(\boldsymbol{\phi}, \boldsymbol{\theta}^{(i)}), \tag{2.63}$$

where i denotes the iteration of the optimization.

2.7.2. Variational Inference for Mixture Models

For discrete latent variable models such as Mixture of Experts (MoEs) (Section 2.6.2), the latent variable model $p_\theta(\mathbf{x})$ from Section 2.7.1 becomes $p_\theta(\mathbf{x}) = \sum_{o=1}^{N} p_\theta(\mathbf{x}, o) = \sum_{o=1}^{N} p_\theta(\mathbf{x}|o)p_\theta(o)$ as defined in Eq. 2.47 ,where the latent variable $\mathbf{z}$ is renamed to o to better reflect the option, or component.

The optimization follows the same scheme as from Section 2.7.1, but the E-step can now be calculated in closed form. By observing that the KL divergence $\mathcal{J}$ is minimal when the arguments are equal, it is obvious that the optimal variational distribution $q_{\boldsymbol{\phi}}(o|\mathbf{x})$ is given by the posterior $p_{\boldsymbol{\theta}}(o|\mathbf{x})$

$$q_{\boldsymbol{\phi}^{(i+1)}}(o|\mathbf{x}) = p_{\boldsymbol{\theta}^{(i)}}(o|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}^{(i)}}(\mathbf{x}|o)p_{\boldsymbol{\theta}^{(i)}}(o)}{\sum_{o=1}^{N} p_{\boldsymbol{\theta}^{(i)}}(\mathbf{x}|o)p_{\boldsymbol{\theta}^{(i)}}(o)}. \tag{2.64}$$

This optimization is also proposed in the work by Arenz et al. (2018, 2020), where the optimization of a Gaussian Mixture Modell (GMM) is framed as a policy search problem. In Chapter 3 we take inspiration from this work and extend it to optimizing contextual Mixture of Experts (MoE) policies, but additionally augment the optimization with automatic curriculum learning (Section 2.5).

2.7.3. Variational Inference for Diffusion Models

For the Denoising Diffusion Model from Section 2.6.3, the latent variable model $p_{\boldsymbol{\theta}}(\mathbf{x})$ from Section 2.7.1 is given by $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int \overleftarrow{p}^{\boldsymbol{\theta}}_{0:N}(\mathbf{x}^{0:N})d\mathbf{x}_{1:N}$, where the latent variable $\mathbf{z}$ is renamed to $\mathbf{x}_{1:N}$ to reflect the time-discretized random variables and the actual variable $\mathbf{x}$ is given by $\mathbf{x}^0$. Please note that we have abused the notation and used a superscript $\boldsymbol{\theta}$ in the backward process $\overleftarrow{p}^{\boldsymbol{\theta}}_{0:N}(\mathbf{x}^{0:N})$ to indicate that the backward process $\overleftarrow{p}^{\boldsymbol{\theta}}_{0:N}(\mathbf{x}^{0:N})$ contains the parameters $\boldsymbol{\theta}$ of the approximated score model that we seek to optimize.

As the noising process $\vec{p}_{0:N}(\mathbf{x}^{0:N})$ starts at the target distribution $q(\mathbf{x}_0)$, the marginal distributions at $n > 0$ are a well suited choice for the variational distribution $q_{\boldsymbol{\phi}}(\mathbf{x}_{1:N}|\mathbf{x}^0)$, which is justified as the variational distribution is free to choose. Hence, the joint distribution $q_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{z})$ from Section 2.7.1 in Eq. 2.60 is given by $\vec{q}_{0:N}(\mathbf{x}^{0:N})$, where we have restated the joint forward distribution $\vec{p}$ to $\vec{q}$, to align with the notation in the decomposed objective in Eq. 2.60.

Additionally, as the noising process $\vec{q}_{0:N}(\mathbf{x}^{0:N})$ is an Ornstein-Uhlenbeck process, it does not have learnable parameters $\boldsymbol{\phi}$, and defines the optimal kernels at each diffusion step already. This reduces the optimization of the diffusion model to only the M-step

$$\arg\min_{\boldsymbol{\theta}} \ \mathrm{KL}\left(\overleftarrow{p}^{\boldsymbol{\theta}}_{0:N}(\mathbf{x}^{0:N})||\vec{q}_{0:N}(\mathbf{x}^{0:N})\right) \tag{2.65}$$

$$= \arg\min_{\boldsymbol{\theta}} \ \mathbb{E}_{\overleftarrow{p}^{\boldsymbol{\theta}}_{0:N}(\mathbf{x}^{0:N})}\left[\log \overleftarrow{p}_N(\mathbf{x}^N) + \sum_{n=1}^{N} \log \frac{\overleftarrow{p}^{\boldsymbol{\theta}}_{n-1|n}(\mathbf{x}^{n-1}|\mathbf{x}^n)}{\vec{q}_{n|n-1}(\mathbf{x}^n|\mathbf{x}^{n-1})} - \log q(\mathbf{x}^0)\right], \tag{2.66}$$

where we have abused the notation and used a superscript $\boldsymbol{\theta}$ in the backward process $\overleftarrow{p}^{\boldsymbol{\theta}}_{0:N}(\mathbf{x}^{0:N})$ to indicate that the parameters $\boldsymbol{\theta}$ are optimized.

Optimizing the denoising diffusion model under this objective was proposed in the time-continuous setting by Richter and Berner (2024) and written in the time-discrete setting for example by Blessing et al. (2024). We take inspiration from this objective and propose optimizing a diffusion-based policy in the maximum entropy RL setting for the step-based RL case in Chapter 5, where we adapt the Policy Iteration scheme for the entropy augmented objective tailored to the diffusion model.

2.7.4. Connection to Episode-Based Maximum Entropy RL

It is easily observable that maximizing the maximum entropy episode-based reinforcement learning Objective in Eq. 2.43 is equivalent to minimizing the KL divergence

$$L(\boldsymbol{\omega}) = \mathbb{E}_{p(\mathbf{c})}\left[\mathrm{KL}\left(\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c}) \middle\| \frac{\exp\left(R(\boldsymbol{\theta},\mathbf{c})/\alpha\right)}{Z(\mathbf{c})}\right)\right], \tag{2.67}$$

where $Z(\mathbf{c})$ is the (unknown) partition function that is independent of $\boldsymbol{\theta}$ and can be neglected in the optimization therefore. Comparing this objective to the conditional variational inference Objective in Eq. 2.58 we can see that the Eq. 2.67 has the same structure as Eq. 2.58 where the target distribution $q(x|c)$ is given by $\frac{\exp\left(R(\boldsymbol{\theta},\mathbf{c})/\alpha\right)}{Z(\mathbf{c})}$. This relation draws parallels of the episode-based maximum entropy RL setting to the variational inference problem.

2.7.5. Connection to Step-Based Maximum Entropy RL

Recall the optimal policy for the step-based maximum entropy RL objective from Section 2.4.1 is given by

$$\pi^*(\mathbf{a}_t|\mathbf{s}_t) = \exp\left(\frac{1}{\alpha}\left(Q^{\pi}_{\mathrm{soft}}(\mathbf{a}_t,\mathbf{s}_t) - V^{\pi}_{soft}(\mathbf{s}_t)\right)\right). \tag{2.68}$$

The underlying objective to this optimal policy is given by

$$L(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{s}\sim\rho(\mathbf{s})}\left[\mathrm{KL}\left(\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}) \middle\| \left(\exp\left(Q^{\pi}_{\mathrm{soft}}(\mathbf{s},\mathbf{a})/\alpha\right)/Z(\mathbf{s})\right)\right)\right], \tag{2.69}$$

with $Z(\mathbf{s}) = \int \exp\left(Q_{\mathrm{soft}}(\mathbf{s},\mathbf{a})/\alpha\right) d\mathbf{a}$ being the partition function that is independent of $\mathbf{a}$. Comparing this objective to the variational inference objective in Eq. 2.58 we observe that Eq. 2.69 has the same structure where the target distribution $q(x|c)$ becomes $\exp\left(Q^{\pi}_{\mathrm{soft}}(\mathbf{s},\mathbf{a})/\alpha\right)/Z(\mathbf{s})$. Yet, when calculating the optimal π^* to the objective in Eq. 2.69, the solution is different from the one in Eq.2.68

$$\pi^*(\mathbf{a}|\mathbf{s}) = \exp\left(\left(Q^{\pi}_{\mathrm{soft}}(\mathbf{s},\mathbf{a})/\alpha\right)\right)/Z(\mathbf{s}). \tag{2.70}$$

However, by noting that $V^{\pi}_{\mathrm{soft}}(\mathbf{s}) = \alpha \log Z(\mathbf{s})$ (see Eq. 2.41 in Section 2.4.1), we can rewrite this solution as

$$\pi^*(\mathbf{a}|\mathbf{s}) = \exp\left(\left(\frac{1}{\alpha}\left(Q^{\pi}_{\mathrm{soft}}(\mathbf{s},\mathbf{a}) - V^{\pi}_{\mathrm{soft}}(\mathbf{s})\right)\right)\right), \tag{2.71}$$

which is the same as the optimal policy in Eq. 2.68.

This relates the step-based maximum entropy RL objective to the variational inference problem, allowing us to take inspiration from the approximate inference with diffusion models literature.

2.8. Mutual Information Based Skill Discovery

In this thesis we consider discovering versatile skills that can solve a task in a variety of ways. As mentioned earlier in Section 2.2, the task is encoded in the reward function which we aim to maximize. However, there is also a field of research that considers discovering skills in the absence of a reward function. Here, usually the goal is to discriminate the states $\mathbf{s}$ into different skills that are encoded using a latent variable $\mathbf{z}$. Most commonly, the mutual information between the latent variable and the state is maximized to obtain distinct skills

$$
\begin{aligned}
I(\mathbf{z}, \mathbf{s}) &= \int p(\mathbf{z}, \mathbf{s}) \log \frac{p(\mathbf{z}, \mathbf{s})}{p(\mathbf{z})p(\mathbf{s})} d\mathbf{z} d\mathbf{s} && (2.72)\\
&= \mathcal{H}(p(\mathbf{z})) - \mathbb{E}_{p(\mathbf{s})}\left[\mathcal{H}(p(\mathbf{z}|\mathbf{s}))\right] && (2.73)\\
&= \mathcal{H}(p(\mathbf{s})) - \mathbb{E}_{p(\mathbf{z})}\left[\mathcal{H}(p(\mathbf{s}|\mathbf{z}))\right]. && (2.74)
\end{aligned}
$$

Eq. 2.73 reveals the intuition of the mutual information. A high value for the mutual information indicates that the state $\mathbf{s}$ leads to less uncertainty in $\mathbf{z}$ on average, as the expected entropy $\mathbb{E}_{p(\mathbf{s})}\left[\mathcal{H}(p(\mathbf{z}|\mathbf{s}))\right]$ is small, whereas a small value indicates that the uncertainty over $\mathbf{z}$ is high and hence, observing $\mathbf{s}$ does not lead to much information about $\mathbf{z}$. We can summarize that the mutual information encodes a measure on the uncertainty of a random variable $\mathbf{z}$ given another random variable $\mathbf{s}$. Due to the symmetry of the mutual information, this relation applies to the reverse case in Eq. 2.74 too.

Many methods in unsupervised skill discovery proposed in the literature base their objective on the mutual information (Laskin et al., 2021; Eysenbach et al., 2019; Campos et al., 2020; Lee et al., 2019; Liu and Abbeel, 2021). Most often, the mutual information between the states $\mathbf{s}$ or the trajectories $\boldsymbol{\tau}$ and the skills $\mathbf{z}$ is considered. Yet, most commonly these methods rely on the varational formulation of the mutual information that provides a tractable lower bound to the Eq. 2.73. This lower bound is required because the posterior in the expectation $\mathbb{E}_{p(\mathbf{s})}\left[\mathcal{H}(p(\mathbf{z}|\mathbf{s}))\right]$ can not be evaluated. Instead, the lower bound allows to optimize a parameterized approximation $p_{\phi}(\mathbf{z}|\mathbf{s})$ that can be evaluated (Eysenbach et al., 2019).

While the mutual information is an interesting objective for discovering distinct skills, it does not consider specific task rewards and hence these methods can learn a set of skills that are not useful for solving a specific task. Therefore, they usually choose a specific skill to solve a downstream task after discovering as many different skills as possible without a reward function. Instead, in this thesis we are interested in disocvering versatile skills that can solve a given task, instead of discovering skills in an unsupervised manner.

3. Specializing Versatile Skill Libraries using Local Mixture of Experts

In this chapter we aim to address the Challenges 1, and 2, which we state in a concise version below.

Mixture of Experts (MoE) models are particularly well-suited for acquiring versatile skills, as they can represent arbitrary complex distributions with a sufficient number of experts (Bishop, 2006). In the following, we consider MoEs with linear experts as policy representations, where each expert is a contextualized motion primitive. Additionally, we extend the model with a context distribution that is composed of per-expert context distributions that allow each expert to shape its own curriculum in the context space. The training of the resulting MoE policy is based on the curriculum augmented maximum entropy episode-based reinforcement learning (ERL) objective (Sections 2.4.2 and 2.5).

Simply updating the MoE policy with the maximum entropy objective is challenging, as the entropy term of the mixture model hinders the training of the experts independently (Bishop, 2006; Arenz et al., 2018). Inspired by previous work on variational inference (Arenz et al., 2018, 2020), this work tackles this problem by decomposing the maximum entropy objective for ERL into an individual lower bound that allows optimizing each expert independently. The resulting per-expert lower bound has a per-expert entropy bonus that encourages the experts to cover as much as possible of the motion-primitive's parameter space while maximizing the expected return. Additional reward augmentations that naturally arise through the decomposition of the objective ensure that the experts discover different modes of the parameter space incentivizing them to learn versatile behaviors. Hence, the decomposition for MoE policies addresses *Challenge 1.*

This decomposition of the objective into per-expert lower bounds naturally also provides an individual objective for updating the per-expert context distribution in a higher hierarchy. The per-expert context distribution is parameterized as a Gaussian and allows each expert to shape their own curriculum in the context space by favoring contexts from regions that are easier to solve for this expert during training, thereby allowing to become an expert in the respective local context region. Once the current contexts are solved reliably, the per-expert context distribution is incentivized to increase its entropy, thereby covering more of the context space. This incentive is provided by an entropy bonus for the per-expert context distribution that emerges from the lower bound, alongside an augmented term that punishes the current per-expert context distribution if it occupies a region that is already covered by another expert. The per-expert context distribution is a key component to address the second challenge (*Challenge 2*) because if each expert would have been forced to cover the entire context space, the few experts that are performing better would dominate the MoE model, because RL increases the likelihood of choosing decisions that yield a higher return. Consequently, the MoE model discards the other experts, which leads to a mode collapse (Bacon et al., 2017) and prevents learning versatile skills. Automatic curriculum learning prevents this mode collapse.

The aforementioned decomposition of the objective and the automatic curriculum shaping using the per-expert context distribution address the *Challenges 1 and 2* from Section 1.1, which is restated in a shorter version for the convenience of the reader in the following.

Challenge 1 (concise): Representation and Training of Multimodal Policies. Discovering versatile skills in reinforcement learning (RL) with multimodal policy representations such as Mixture of Experts (MoEs) models, requires novel RL algorithms that train these expressive policies efficiently and explicitly encourage the exploration of different modes.

Challenge 2 (concise): Retaining Multimodalities. Modes that yield higher returns earlier during the reinforcement learning (RL) training process might dominate the policy, leading to an overrepresentation of these modes and therefore to a collapse of the other modes. This mode collapse prevents learning versatile skills and needs to be addressed.

The following work was published as Specializing Versatile Skill Libraries using Local Mixture of Experts *(Onur Celik, Dongzhuoran Zhou, Ge Li, Philipp Becker, Gerhard Neumann 2021) in the 5th Conference on Robot Learning, CoRL 2021. Reprinted with permission of the authors. Wording, notation and formulations were revised in several places.*

3.1. Introduction

Human motor skills are precise and versatile, which allows us to perform motor tasks in different ways while achieving a consistent movement quality. For example, when playing table tennis, we can hit the ball in various ways while still targeting a specific landing point of the ball on the opponent's side of the table. Another example is grasping an object which lies behind an obstacle. We can grasp even small objects precisely with different grasp types while avoiding the obstacle. Such versatile skills are crucial if we want to employ robots in unstructured and dynamically changing environments. Such skills, often represented as movement primitives, were already successfully learned for challenging robot learning tasks, e.g., the ball-in-a-cup (Kober and Peters, 2014; Klink et al., 2020a) task, by a variety of policy search algorithms (Deisenroth et al., 2013). Yet, most of these algorithms cannot find multiple, versatile, and precise solutions to the multi-modal solution space, as they usually assume a Gaussian policy (Abdolmaleki et al., 2019; Klink et al., 2020a; Deisenroth et al., 2013; Kupcsik et al., 2013).

In this paper, we model versatile behavior with contextual skill libraries of motion primitives (Paraschos et al., 2013; Ijspeert et al., 2002), formalized by Mixtures of Experts (MoEs).

Here the context defines task properties, e.g., reaching different goal positions or different friction parameters of an object to manipulate (Kupcsik et al., 2013). Our goal is to learn versatile skills, i.e., different movement styles to solve a given context. Given a context, the MoE first selects an expert, i.e., a motion primitive, to execute. Subsequently, the expert adjusts the primitive's parameters and a controller executes the primitive. Such models are already commonly used (Daniel et al., 2012; End et al., 2017). Yet, the quality and versatility of the learned skill libraries remain limited using existing algorithms. Most algorithms are based on expectation-maximization (EM) (Dempster et al., 1977; Bishop, 2006) techniques (Daniel et al., 2012), and suffer from local optima and mode averaging (Bishop, 2006) which prevents the single experts from specializing in a local context region. Moreover, existing algorithms (Daniel et al., 2012; End et al., 2017) do not explicitly optimize the versatility of the library. Hence, they often yield degenerated libraries with only a single movement style. We propose a new objective for learning contextual, precise, and versatile MoE models based on a maximum entropy formulation. We also introduce a learnable context distribution, which provides a curriculum for each expert of the MoE model. We use a variational lower bound (Arenz et al., 2018) to decompose the objective into individual updates for the experts and their related local context distributions, allowing the experts to specialize in local regions of the context space and preventing mode averaging. Due to the curriculum, the MoE does not have to cover the whole context space during training, which prevents the averaging effects that negatively affect most other approaches. Yet, not covering the whole context space leads to poor performance for some contexts, which is also undesirable. Thus, we introduce a heuristic-free mechanism to add new experts during training until the whole context space is covered. Hence our algorithm provides a modular approach that learns highly precise and versatile skills that cover the whole context space.

We evaluate our approach in a simulated beer pong and a table tennis environment. Both environments allow different motion styles to solve the tasks, which are discovered by our algorithm. Moreover, we present ablation studies showing the importance of the single elements and hyperparameters of our algorithm.

3.2. Related Work

Contextual Episodic Policy Search. Episodic policy search (Deisenroth et al., 2013) aims at maximizing the expected return by optimizing the parameters $\boldsymbol{\theta}$ of a controller, e.g., a motion primitive (Schaal et al., 2005; Paraschos et al., 2013). Most approaches use a stochastic search distribution $\pi(\boldsymbol{\theta})$ over the parameter space and aim to optimize the expected return under this search distribution (Deisenroth et al., 2013), i.e.,

$$\max_{\pi(\boldsymbol{\theta})} \mathbb{E}_{\pi(\boldsymbol{\theta})}[R(\boldsymbol{\theta})],$$

where $R(\boldsymbol{\theta}) = \sum_t r_t$ is the summed reward over a whole episode obtained when executing controller parameter $\boldsymbol{\theta}$. As it is common in the literature, we will denote $\pi(\boldsymbol{\theta})$ as our policy even though it only indirectly chooses the control actions of the agent by selecting the controller parameters $\boldsymbol{\theta}$. Different optimization schemes such as policy gradients (Sehnke et al., 2010), natural gradients (Wierstra et al., 2014), stochastic search strategies (Hansen and Ostermeier, 2001; Mannor et al., 2003) or trust-region optimization techniques (Daniel et al., 2012; Abdolmaleki et al., 2015) have been used. Researchers extended these approaches to the contextual setting (Tangkaratt et al., 2017; Abdolmaleki et al., 2019), where the search distribution $\pi(\boldsymbol{\theta}|\mathbf{c})$ now depends on a context vector $\mathbf{c}$ which describes the task, e.g., a goal location to reach. The contextual objective is typically formalized as

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c})} \mathbb{E}_{p(\mathbf{c})} \left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c})} [R(\boldsymbol{\theta}, \mathbf{c})] \right],$$

where $p(\mathbf{c})$ is the given distribution over context vector and the rewards now also depends on the context. Klink et al. (2020a) introduce a curriculum into contextual policy search. By having an adaptable distribution over the contexts, they allow the agent to concentrate on easy-to-solve contexts first and then generalize to the whole context space. We refer the reader to Section 2.3 for an in-depth discussion of episode-based reinforcement learning.

Versatile Skill Learning. The Hierarchical Relative Entropy Policy Search (HiREPS) algorithm (Daniel et al., 2012) extends the classical Relative Entropy Policy Search (REPS) approach (Peters et al., 2010) to MoEs, which allows learning versatile skills in a contextual episodic policy search setting. In a similar approach, Layered Direct Policy Search (LaDiPS) (End et al., 2017) also uses MoE policies, but builds on Model-Based Relative Entropy Policy Search (MORE) (Abdolmaleki et al., 2015) instead of REPS. Both HiREPS and LaDiPS address the same problems as our approach, yet there are also considerable differences. First, HiREPS jointly optimizes the whole mixture model and introduces an additional constraint, which bounds the entropy loss of the responsibilities in each iteration. This constraint is crucial for obtaining versatile and well-performing solutions. LaDiPS uses separate updates for the different parts of the mixture but also relies on additional constraints, where the entropy of the gating is lower bounded with a constant value. In contrast, for our approach, the objective and separate updates of the individual mixture parts follow naturally from the maximum entropy formulation. Second, neither HiREPS nor LaDiPS uses a curriculum for training. Thus, in both approaches, the MoE always has to cover the whole context space and, hence, the components cannot specialize.

Variational Inference. Our work is also related to several recent advances in variational inference (Arenz et al., 2018, 2020; Becker et al., 2020). It is well known that maximum entropy RL is equivalent to inference in an appropriate probabilistic model (Levine, 2018). Similar to previous works (Neumann et al., 2011; Haarnoja et al., 2018b), we exploit this relation and draw inspiration from recent research into variational inference and density estimation for Gaussian mixture models and MoEs (Arenz et al., 2018, 2020; Becker et al., 2020). We reformulate the lower bound objectives introduced in those approaches for our maximum entropy RL setting and extend them with a curriculum for the mixture components.

Related Step-Based Approaches. In the step-based setting, the policy does not learn a function from contexts to parameters of an episodic controller but directly maps from system states to actions and the policy updates are performed with the information from each time-step. Practitioners often use deep neural networks to parameterize step-based policies, giving rise to the field of deep RL. In this context, versatile policy learning is also a very active research area (Eysenbach et al., 2019; Kumar et al., 2020; Osa et al., 2021; Campos et al., 2020). These approaches use a similar MoE model where the mixture component is only chosen at the beginning of an episode. Yet, the component is chosen randomly without conditioning on a context or state variable. Moreover, these approaches reformulate a mutual information based objective into a maximum entropy objective while we develop a more direct maximum entropy maximization.

Curriculum Learning. Researchers also worked on introducing curricula into deep RL. In a first approach Ghosh et al. (2018) proposed partitioning the initial state distribution using clustering. They then learn individual policies for each partition while keeping the partitioning fixed. Strictly, this is not a curriculum as the partitioning is not adjusted, yet it still allows specialization of the individual policies in different regions of the state space. To automatically generate and adapt a curriculum for deep RL approaches, Klink et al. (2020b) extended their approach from the episodic setting (Klink et al., 2020a) to the step-based setting. Yet, neither of these approaches addresses versatility. While we follow an episode-based approach, both methodologies have their benefits and limitations (Deisenroth et al., 2013) which are, however, not the focus of this paper. We offer further discussion and quantitative comparison to a step-based approach for a common benchmark in the experiments, therefore.

Options. A related hierarchical approach is the options framework (Sutton et al., 1999b; Bacon et al., 2017; Riemer et al., 2018). The options framework extends the standard MDP to a semi-MDP to include a temporal abstraction of low-level control policies. Given a termination condition, the executed low-level policy can be terminated and another can be turned on. Our policy structure can be seen as a simplification of the options framework where the option is only selected at the beginning of each episode. Yet, the options framework does not explicitly address learning versatile skills.

3.3. Specializing Versatile Mixture of Expert Models

To allow versatile solutions, we employ a Mixture of Experts (MoE) model as policy representation which is given as $\pi(\boldsymbol{\theta}|\mathbf{c}) = \sum_o \pi(o|\mathbf{c})\pi(\boldsymbol{\theta}|\mathbf{c}, o)$, where $\pi(o|\mathbf{c})$ is the gating distribution, assigning a probability to expert o given the context $\mathbf{c}$ and $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ is the expert distribution for expert o, which adapts the motion primitive's parameters $\boldsymbol{\theta}$ to the given context $\mathbf{c}$. In this section, we derive a lower bound to optimize each expert and its corresponding context distribution independently. In order to implement a curriculum over the context $\mathbf{c}$, we also introduce a learned context distribution $\pi(\mathbf{c}) = \sum_o \pi(\mathbf{c}|o)\pi(o)$, which is also a mixture model specified by the per-expert context distribution $\pi(\mathbf{c}|o)$ and the expert weights $\pi(o)$. By applying Bayes' Rule to replace the gating distribution $\pi(o|\mathbf{c})$, we can now rewrite the general mixture of experts model as

$$\pi_\omega(\boldsymbol{\theta}|\mathbf{c}) = \sum_o \frac{\pi_\omega(\mathbf{c}|o)\pi_\omega(o)}{\pi_\omega(\mathbf{c})}\pi_\omega(\boldsymbol{\theta}|\mathbf{c}, o), \tag{3.1}$$

where ω contains the learnable parameters of each expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$, per-expert context distribution $\pi(\mathbf{c}|o)$ and the gating distribution $\pi(o)$. This policy definition allows each expert o to adjust its curriculum by explicitly optimizing for $\pi_\omega(\mathbf{c}|o)$ (Section 3.3.3) and thus concentrating on a local region in the context space. We model the experts $\pi_\omega(\boldsymbol{\theta}|\mathbf{c}, o)$ as a linear conditional Gaussian distribution and the per-expert context distribution $\pi_\omega(\mathbf{c}|o)$ as a Gaussian. The prior weights $\pi(o)$ define a categorical distribution.

To keep notation uncluttered, we do not write the subscript ω to indicate the learnable parameters of a distribution throughout the next sections, but we simply denote every probability distribution which is adaptable through the optimization process with π as it is part of our policy $\pi(\boldsymbol{\theta}|\mathbf{c})$. Furthermore we show the full derivations for the next sections in the Appendix A.1.

3.3.1. Maximum Entropy Skill Learning with Curriculum

We consider a maximum entropy objective (Ziebart, 2010; Levine, 2018) for episodic policy search, i.e.,

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c})} \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c})}\left[\mathrm{R}(\mathbf{c}, \boldsymbol{\theta})\right] + \alpha\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c})\right]\right], \tag{3.2}$$

where $p(\mathbf{c})$ is the task specific context distribution, $\mathrm{R}(\mathbf{c}, \boldsymbol{\theta})$ is the reward function, $\mathrm{H}(\pi(\boldsymbol{\theta}|\mathbf{c})) = -\int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta}|\mathbf{c}) \log \pi(\boldsymbol{\theta}|\mathbf{c}) d\boldsymbol{\theta}$ the entropy and $\pi(\boldsymbol{\theta}|\mathbf{c})$ is our MoE model. The reward maximization enforces preciseness while the entropy bonus enforces versatility. However, the standard maximum entropy objective does not allow each mixture expert to create its own curriculum, since an optimization over the per-expert context distributions $\pi(\mathbf{c}|o)$ is not given. Inspired by the work from Klink et al. (2020a), we extend and modify the objective to

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c}), \pi(\mathbf{c})} \mathbb{E}_{\pi(\mathbf{c})}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c})}\left[\mathrm{R}(\mathbf{c}, \boldsymbol{\theta})\right] + \alpha\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c})\right]\right] - \beta\mathrm{KL}\left(\pi(\mathbf{c}) \parallel p(\mathbf{c})\right), \tag{3.3}$$

where α and β are scaling parameters, $\mathrm{H}(\pi(\boldsymbol{\theta}|\mathbf{c})) = -\int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta}|\mathbf{c}) \log \pi(\boldsymbol{\theta}|\mathbf{c}) d\boldsymbol{\theta}$ is the entropy and $\mathrm{KL}\left(\pi(\mathbf{c}) \parallel p(\mathbf{c})\right) = \int_{\mathbf{c}} \pi(\mathbf{c}) \log \frac{\pi(\mathbf{c})}{p(\mathbf{c})} d\mathbf{c}$ denotes the KL-divergence. Note the difference in the optimization variables compared to Eq. 3.2. The Kullback-Leibler (KL) term ensures that the context distribution $\pi(\mathbf{c})$ is close to the task specific context distribution $p(\mathbf{c})$ while $\pi(\mathbf{c})$ can choose to have low probability in regions of the context space where the MoE model is performing poorly. Note that this objective is similar to the negative I-projection of the joint distribution $\pi(\mathbf{c}, \boldsymbol{\theta})$ used in variational inference. Here, we also exploit the properties of the I-projection for learning the context distribution – the I-projection is mode seeking instead of mode-averaging and therefore allows for specialization on a local context area. Yet, the given objective is difficult to optimize for mixture models as the sum over the mixture experts is appearing inside the log terms of the entropy and the KL. However, similar to Arenz et al. (2018), we can replace $\pi(\boldsymbol{\theta}|\mathbf{c})$ in Objective 3.3 with our mixture model and apply Bayes theorem to arrive at

$$\max_{\pi(\mathbf{c},\boldsymbol{\theta})} \mathbb{E}_{\pi(o),\pi(\mathbf{c}|o)}\Big[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)} \overbrace{\big[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha \log \pi(o|\mathbf{c},\boldsymbol{\theta})\big]}^{\text{augmented reward for expert } o} + \overbrace{\beta \log p(\mathbf{c}) + (\beta - \alpha) \log \pi(o|\mathbf{c})}^{\text{augmented reward for context distributions}} \Big] \tag{3.4}$$
$$+ \alpha \mathbb{E}_{\pi(o),\pi(\mathbf{c}|o)}\big[\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]\big] + \beta \mathbb{E}_{\pi(o)}\big[\mathrm{H}\left[\pi(\mathbf{c}|o)\right]\big] + \beta \mathrm{H}\left[\pi(o)\right].$$

The exact derivations are given in the Appendix A.1. Note that Eq. 3.4 is equivalent to Eq. 3.3, yet, instead of the entropy for the whole mixture model, it now contains entropy terms for each hierarchy layer of the MoE model, i.e., $\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]$, $\mathrm{H}\left[\pi(\mathbf{c}|o)\right]$ and $\mathrm{H}\left[\pi(o)\right]$, which are much simpler to compute. We also introduced log responsibilities $\pi(o|\mathbf{c},\boldsymbol{\theta}) = \pi(\boldsymbol{\theta}|\mathbf{c},o)\pi(o|\mathbf{c})/\pi(\boldsymbol{\theta}|\mathbf{c})$ and $\pi(o|\mathbf{c}) = \pi(\mathbf{c}|o)\pi(o)/\pi(\mathbf{c})$, occurring in the augmented rewards. They return a high negative reward for expert o, if the context-parameter pair $(\mathbf{c},\boldsymbol{\theta})$ or the context sample $\mathbf{c}$ is already covered by another expert, pushing the expert to uncovered regions of the parameter space or context space respectively. Yet, the regions for the experts will still overlap due the entropy bonuses for $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ and $\pi(\mathbf{c}|o)$. Without this reward augmentation, each expert could be optimized completely independently in Eq. 3.4 using a maximum entropy objective. However, in this case, all experts would concentrate on learning the best solution irrespective of whether this solution has already been covered by another expert. Yet, the log responsibilities still hinder us from optimizing each expert $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ and its corresponding per-expert context distribution $\pi(\mathbf{c}|o)$ independently, since the sum over o from the mixture models $\pi(\mathbf{c})$, $\pi(\boldsymbol{\theta}|\mathbf{c})$ respectively appears in the log term. In the following sections we show, how we can overcome this limitation by introducing a lower bound inspired by variational inference (Arenz et al., 2018). As we consider each possible context as equally important, $p(\mathbf{c})$ is assumed uniformly distributed in a given interval in the following and thus, can be neglected in the objective.

3.3.2. Lower-Bound Decomposition for Expert Distributions

In order to maximize the Objective in Eq. 3.4 for each expert $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ individually, we can first extract the terms which only depend on $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ for a specific o as

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c},o)} \mathbb{E}_{\pi(\mathbf{c}|o),\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha \log \pi(o|\mathbf{c},\boldsymbol{\theta})\right] + \alpha \mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]\right]. \tag{3.5}$$

The responsibilities are still hindering us to optimize each expert $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ independently. However, similar to Arenz et al. (2018), we can obtain a tight lower-bound by introducing a variational distribution $\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})$ and replacing the responsibilities in Eq. 3.5 with $\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})$. This variational distribution is fixed during the optimization and can be computed according to the last policy model allowing us to update each expert independently. It is easy to show that after the update of the variational distribution $\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta}) = \pi_{\text{old}}(o|\mathbf{c},\boldsymbol{\theta})$, the introduced lower bound is tight. Please refer to the appendix. The resulting lower bound is a standard maximum entropy RL objective with an additional reward augmentation of $\alpha \log \tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})$ and thus can be optimized with any suitable Policy Search algorithm. Here we use a maximum-entropy-adjusted version of contextual MORE (Tangkaratt et al., 2017).

3.3.3. Lower-Bound Decomposition for Context Distributions and Prior Weights

To update $\pi(\mathbf{c}|o)$ for a specific o, we extract the relevant terms from Objective 3.4

$$\max_{\pi(\mathbf{c}|o)} \mathbb{E}_{\pi(\mathbf{c}|o)} \left[L_c(o,\mathbf{c}) + (\beta - \alpha) \log \tilde{\pi}(o|\mathbf{c})\right] + \beta \mathrm{H}\left(\pi(\mathbf{c}|o)\right), \tag{3.6}$$

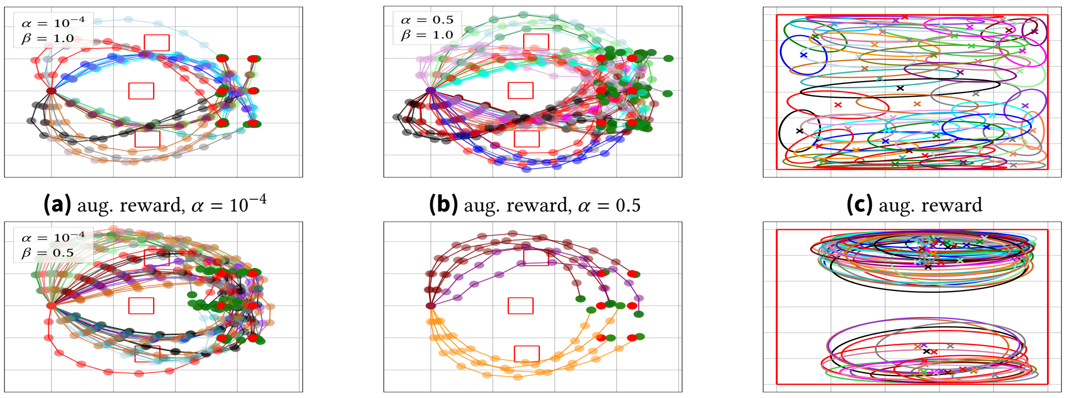
where $L_c(o,\mathbf{c}) = \mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha \log \tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right] + \alpha \mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]$ corresponds to the expected augmented maximum entropy objective of expert $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ in context $\mathbf{c}$ and $\tilde{\pi}(o|\mathbf{c}) = \pi_{\text{old}}(o|\mathbf{c})$ is a second variational distribution, which we introduced to be able to optimize for each $\pi(\mathbf{c}|o)$ individually. Similarly to the previous section, the objective given in Eq. 3.6 is a tight lower bound to the original objective where the responsibilities $\pi(o|\mathbf{c})$ are used instead of $\tilde{\pi}(o|\mathbf{c})$. We approximate the integral over $\boldsymbol{\theta}$ with a single sample, as we typically only have a single parameter sample per context available. Yet, as we still have the outer expectation $\mathbb{E}_{\pi(\mathbf{c}|o)}$ in Eq. 3.6 which we approximate by multiple context samples, the whole Monte-Carlo estimation of the expectations is still unbiased and with low variance. After the optimization step (Eq. 3.6), we obtain the optimal solution $\pi^*(\mathbf{c}|o)$ and tighten the bound by setting $\tilde{\pi}(o|\mathbf{c}) = \pi^*(o|\mathbf{c})$ and $\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta}) = \pi^*(o|\mathbf{c},\boldsymbol{\theta})$. Again, this lower bound corresponds to an augmented maximum entropy RL objective and thus, we can updated it with any suitable policy search method. Like Arenz et al. (2018), we use an adjusted version of MORE (Abdolmaleki et al., 2015).

Finally, we can formulate the objective for updating the expert weights $\pi(o)$, which resembles a lower bound of the original Objective 3.4 and corresponds to the highest hierarchy in our update scheme. The objective is given as

$$\max_{\pi(o)} \sum_{o} \pi(o) \left[\mathbb{E}_{\pi(\mathbf{c}|o)} \left[L_c(o,\mathbf{c}) + (\beta - \alpha) \log \tilde{\pi}(o|\mathbf{c})\right] + \beta \mathrm{H}\left(\pi(\mathbf{c}|o)\right)\right] + \beta \mathrm{H}\left(\pi(o)\right), \tag{3.7}$$

which is a maximum entropy RL objective for categorical distributions. Here, we use REPS (Peters et al., 2010).

To summarize, we split the initial objective in Eq. 3.4 into different hierarchies, allowing us to optimize the different terms in our mixture model individually. Starting by first updating the experts $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ using the maximization problem in Eq. 3.5, we can optimize for the per-expert context distributions $\pi(\mathbf{c}|o)$ with Objective 3.6 after tightening the bound. The experts $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ adjust the movement primitive parameters $\boldsymbol{\theta}$ given a context $\mathbf{c}$, while the per-expert context distributions $\pi(\mathbf{c}|o)$ ensure that the experts only see context samples from a local context region. Finally, we update the weight distribution $\pi(o)$ using Eq. 3.7.



(a) aug. reward, $\alpha = 10^{-4}$ **(b)** aug. reward, $\alpha = 0.5$ **(c)** aug. reward

(d) no aug. reward **(e)** HiREPS **(f)** no aug. reward

Figure 3.1.: Importance of the responsibilities in the augmented rewards and comparison to HiREPS. A snapshot of the learned policies of the planar reaching task for a 2-dim context space (illustrated in (c) and (f)) by considering $\log \tilde{\pi}(o|\mathbf{c}, \boldsymbol{\theta})$, $\log \tilde{\pi}(o|\mathbf{c})$ ((a)+(b)+(c)) and neglecting the auxiliary distributions ((d)+(f)). In ((a)+(b)+(c)) we can learn more diverse solutions (a) + (b) and cover the whole 2d-context space (c) with different entropy bonuses, whereas the solutions in ((d)+(f)) show nearly the same, partially invalid solutions by going through the red rectangles (obstacles) (d) and are not able to cover the whole context space (f) leading to poor generalization performance. The solutions by HiREPS (e) indicate less versatility compared to the solutions (a) + (b). Note that each color indicates a different expert and the red dots indicate the 6 chosen context vectors used for sampling.

Algorithmic Details and Addition of Experts. We initialize our algorithm with only one expert and incrementally add experts, and their corresponding context distributions, randomly. We fix all experts except for the newly added one and optimize it for K iterations to let it discover new solutions in yet undiscovered context regions. For updating $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ (Objective 3.5) we sample from the local context distribution $\pi(\mathbf{c}|o)$. By also updating $\pi(\mathbf{c}|o)$ according to Objective 3.6, the experts can adjust their curriculum and search for their favored context regions. After K iterations we add a new expert and repeat the procedure. Due to the augmented rewards, the new expert will focus on undiscovered solutions and the local context distribution will cover uncovered areas of the context space. Note that such a simple adding procedure is only possible due to the mode-seeking properties of the I-projection, as the experts do not need to average over multiple modes but can specialize on a local context region. We also fix the weights $\pi(o)$ to be uniformly distributed among all experts during learning, since otherwise experts which are already fully trained might dominate the optimization. By allowing to fine-tune all experts every H iterations, the previously added experts can adjust to the newly added ones. After finishing adding experts, we update the weights $\pi(o)$ at the end of our optimization procedure. The variables K and the number of experts added in total are task-dependent. Please note that we need to restrict the updates of the local context distributions $\pi(\mathbf{c}|o)$ to a the valid region of the context space by introducing punishment terms in the reward function. We describe the algorithm and the respective punishments in more detail in the Appendix A.2 and A.3.

3.4. Experiments

We start by investigating the importance of the different terms of the objectives derived in Section 3.3 and subsequently evaluate the versatility and precision of the learned skills on simulated robot beer pong and table tennis experiments.

We conclude our experiments with a discussion on episode-based RL (ERL) and step-based RL (SRL) by comparing our method to PPO in different scenarios. This comparison is to highlight the advantages of each approach and does not focus on presenting versatile skills which is the main focus of the experiments before.

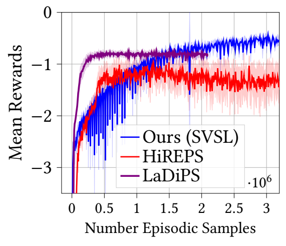
In Appendix A.3 we report all hyperparameters.

3.4.1. Ablation Studies

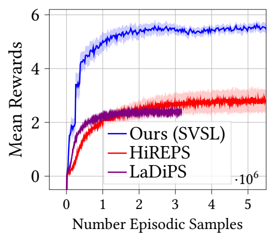
We investigate the importance of the augmented rewards on a 10-link planar reaching task with two-dimensional context space. For this purpose, we update the expert distributions $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ and the corresponding context distributions $\pi(\mathbf{c}|o)$ by optimizing the objectives in Eq. 3.5 and Eq. 3.6 with i) considering the responsibilities (as given by our algorithm) and ii) by setting $\log \tilde{\pi}(o|\mathbf{c}, \boldsymbol{\theta})$ and $\log \tilde{\pi}(o|\mathbf{c})$ to zero. Furthermore, we compare the solutions found by our method to the solutions found by the SOTA method HiREPS.

Versatility is Induced by the Augmented Rewards. In the 10-link planar reaching task (see Fig. 3.1a) the robot has to reach the red dots with its end-effector in a 2-dim context-space, while avoiding the rectangle-shaped obstacles. By adding 60 experts, we have trained both versions (i and ii) over 15 seeds/trials and have chosen the best performing parameter constellations (Fig. 3.1a for i), Fig. 3.1d for ii)). We then picked the first model and sampled for each of the shown six contexts vectors (red dots) 100 samples and plotted the corresponding mean of each sampled expert. The plots for i) (Fig. 3.1a + Fig. 3.1b) show versatile – several modes to the same context – and precise – small distance of end-effector (green dot) to goal (red dot), while avoiding obstacles– solutions while fully covering the context space (Fig. 3.1c). A trend of covering more modes with higher α can also be seen in (Fig. 3.1b). For the case without the responsibilities ii) (Fig. 3.1d), the solutions are not precise and invalid –reaching through the obstacles–. As Fig. 3.1f shows, the context distributions focus on easy-to-solve context regions (top and bottom part) and do not cover the full context space, leading to extrapolated solutions from experts that are not trained for these contexts (red dots).

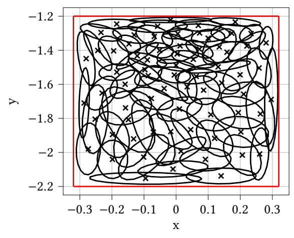
Comparison to HiREPS. We pick the best performing model among 15 seeds/trials after hyperparameter optimization. In Fig. 3.1e the mean solutions of the sampled experts – sampled in the same way as before – can be seen. HiREPS shows less diverse and qualitative solutions, where only 3 of the 60 experts were chosen in total.



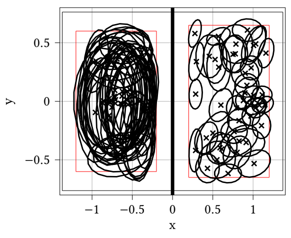
(a) Rewards- **BP**



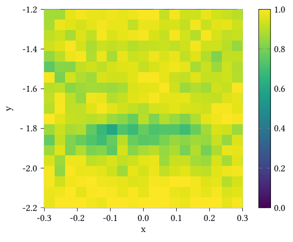
(b) Rewards- **TT**



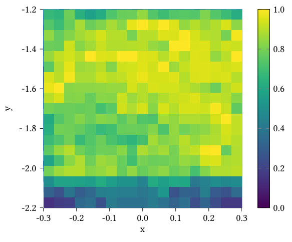
(c) Context-Space **BP**



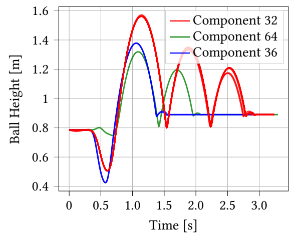
(d) Context-Space **TT**



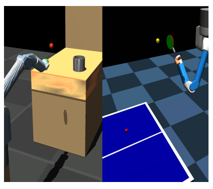
(e) Successes **BP** Ours



(f) Successes **BP** HiREPS



(g) Ball Trajectories **BP**



(h) Environments

Figure 3.2.: Beer Pong (BP) and Table Tennis (TT) Experiments. In the **BP** experiment −left of (h)− the robot has to throw the red ball into a cup on the table. In the **TT** experiment −right of (h) − the robot hast to return the ball to the opponent's table side. The learned mixture of expert (MoE) models show high quality of the skills, reflected in the high reward values (a + b) and dense coverage of the context spaces, i.e., the 2-dim. position of the cup (c) for **BP** and the 4-dim desired outgoing landing and incoming landing ball positions (d) for **TT**. For **BP** we also compare the success rates (92% in average) of our approach (e) and HiREPS (75% in average) (f) throughout the context space, where we consider a trial successful if the ball goes into the cup. For **BP** versatile skills induce versatile ball trajectories for a given context (g). Versatile strikes for the **TT** experiment are shown in Fig. 3.3.

3.4.2. Simulated Robotic Experiments

We test and compare our algorithm on simulated Beer Pong and Table Tennis environments in MuJoCo (Todorov et al., 2012), where we have chosen **HiREPS** (Daniel et al., 2012) and **LaDiPS** (End et al., 2017) as baselines. Both methods are suitable baselines since they are contextual policy search algorithms and consider optimizing a mixture model. Throughout our experiments we choose probabilistic movement primitives (ProMP) (Paraschos et al., 2013) as policy representation, where depending on the weights $\boldsymbol{\theta}$ desired trajectories are generated and subsequently tracked with a PD-controller. In our experiments, given a context $\mathbf{c}$, the experts $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ adjust the weight vector and the length of the trajectory, which are summarized in the vector $\boldsymbol{\theta}$. We consider non-Markovian rewards, in which the reward depends on the history of state and actions. This type of reward function is not applicable to common step-based RL methods which build on Markovian properties. As for analyzing the results of the experiments, we focus on the questions i) how does our algorithm perform compared to SOTA baselines, ii) are we able to cover the whole context space, and iii) are we able to learn versatile skills?

Beer Pong. The goal of the Barret WAM robot is to throw the red ball into the cup on the table. The 2-dim contexts resemble the position of the cup on the table. We incrementally add 70 experts in our method, while HiREPS and LaDiPS directly start with 70 experts.

We have run all algorithms over 20 seeds/trials and report the performance in Fig. 3.2a[1], where we plot the mean reward with two times the standard error. While HiREPS already converges after around half a million samples, our approach can quickly outperform it. Since LaDiPS uses intra-option learning, it outperforms our method in terms of sample efficiency. However, with the increasing number of experts, we achieve a higher end-reward indicating that we can learn more qualitative solutions. Our algorithm allows to cover the whole context space with the learned per expert context distributions (see Fig. 3.2c) resulting in a high end-reward. This performance is reflected in Fig. 3.2e, where we have divided the context space into a fine grid and sampled 100 times for each context an expert from which we have executed the mean. We repeated that for all 20 different models and plot the mean success rate of throwing the ball into the cup. We did the same procedure for HiREPS in Fig. 3.2f. We can observe that HiREPS has much darker rectangles, showing that the success rate in these context regions is low. Although versatility is encouraged in the joint space of the robot, different joint trajectories often yield different ball trajectories. Given one context, in Fig. 3.2g we show the z-coordinates of the ball trajectories over time resulting from sampling 20 times from the MoE model. Each expert leads to a different number of "ball-jumps". In Appendix A.3.2.1 we qualitatively show that we can learn more versatile solutions than LaDiPS and report that we can achieve a much higher expected mixture entropy (Fig. A.1), which indicates that our solutions are versatile.

Table Tennis. The task of the robot is to return different incoming balls to desired targets on the opponent's table side in different ways. We consider a four dimensional context space, including ball's initial serve position and ball's target landing position (right and left half of table Fig. 3.2d). For both parts of the context, we fix the z position and vary the x and y coordinates. We incrementally add 50 experts in our method, while HiREPS and LaDiPS directly start with 50 experts. We have run all algorithms over 20 seeds/trials and report the performance in Fig. 3.2b[1], in which the mean reward with two times the standard error is plotted. The per-expert context distributions are spread among the context space (Fig. 3.2d) and allow each expert to locally specialize on a context region. This high coverage of the context space allows for a high reward achievement, outperforming HiREPS and LaDiPS. In Fig. 3.3, we show three different striking skills sampled from our trained MoE model, for a fixed context, i.e., fixed serving and desired ball position. The first two skills use the green side of the racket to hit the ball (forehand), while the third skill uses the red side of the racket (backhand) to hit the ball. In contrast to the second strike, the first one performs a smash-like strike and ends it with the red side of the racket pointing to the camera.

[1] To reflect the model's true performance, the lastly added expert was excluded from testing, since it is not fully trained yet and would not be chosen by the model if $\pi(o)$ would not be a uniform-distribution.

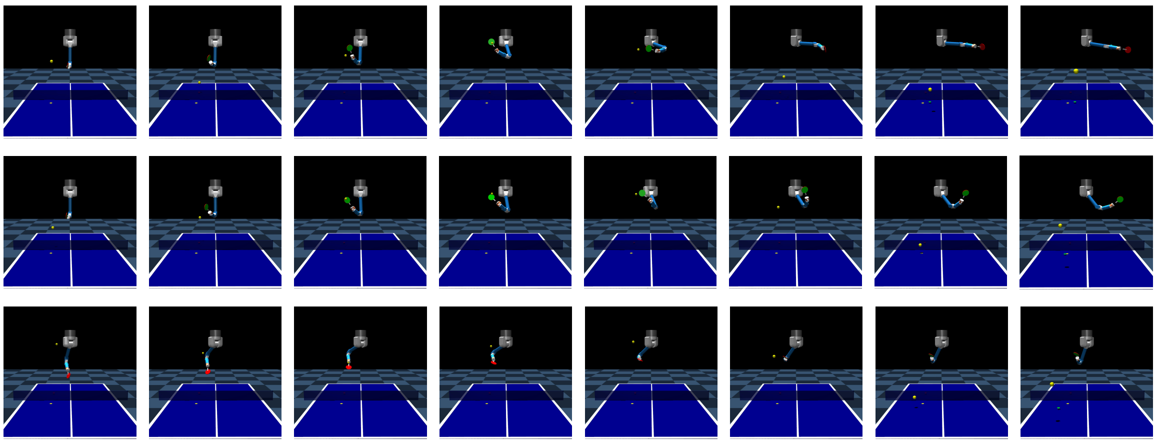


Figure 3.3.: Versatile Strikes for the Table Tennis (TT) Experiment illustrated for a fixed context. The robot can hit the (yellow) ball in various ways, also indicated through the different colors of the racket sides. Note that the red and yellow dots on the table are markers for the serving and desired landing position respectively.

3.4.3. A Comparison Between Episode-Based and Step-Based Reinforcement Learning

In Sections 2.2 and 2.3 whe have discussed the core concepts of step-based Reinforcement Learning (SRL) and episode-based Reinforcement Learning (ERL) approaches. SRL methods explore in the raw action space in each time-step, while ERL methods explore in the parameter space of the controller (Deisenroth et al., 2013) leading to correlated exploration with smooth trajectories (Li et al., 2024a). Here, we employ ProMPs as controller parameterization (Paraschos et al., 2013). While this exploration might yield benefits, usually SRL methods are more sample efficient as they can leverage the temporal structure to update the policy. Hence, it is interesting to compare these approaches. More specifically, we compare the performance of our algorithm against PPO, a well-known deep reinforcement learning method (Schulman et al., 2017). We want to analyze when episodic exploration is beneficial over step-based exploration and vice-versa for different sparsity levels of the reward.

The 5-Link Reacher. For this purpose, we use the Reacher task from OpenAI gym (Brockman et al., 2016), in which a planar robot has to reach a goal position with its tip. In order to consider a more difficult task, we extend the original reacher set-up and show the considered environment in Fig. 3.4. First, we increase the number of links from two to five and we fix the initial state position to be initialized on the horizontal to the left (see Fig. 3.4), instead of initializing on the horizontal to the right side as it is in the original version. Furthermore, we fix the initial joint velocity to be zero. For the original 2-Link

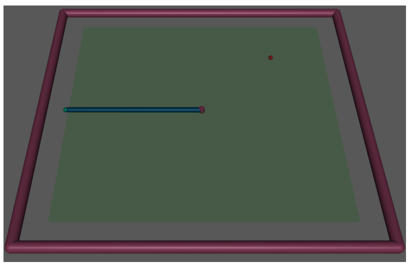


Figure 3.4.: The reacher's tip needs to reach the red goal position. Adapted from Brockman et al. (2016).

case, the goal reaching space is the circle around the agent's base, whereas in our case we consider the right half-circle around the agent's base, where the radius is the agent's total length. By initializing the agent on the left side and only considering goal positions on the right side, we make the task harder. We compare our method to PPO algorithm (Schulman et al., 2017) and use the implementation from Raffin et al. (2019).

Task Rewards. In order to examine the different policy search strategies we consider three different Markovian reward functions in the same set-up. The first reward function r_t is given by Brockman et al. (2016) and encodes task information, i.e. the tip distance to the goal, in each time-step as

$$r_t(\mathbf{s}, \mathbf{a}) = -||\mathbf{t}(\mathbf{s}) - \mathbf{g}||_2 - \sum_i a_i^2, \tag{3.8}$$

where $\mathbf{s}$ is the current state, $\mathbf{a}$ is the current action, $t(\mathbf{s})$ is the tip's current position and $\mathbf{g}$ is the goal position. One episode lasts $T = 200$ steps.

The second reward function is given as

$$r_t(\mathbf{s}, \mathbf{a}) = \begin{cases} -\sum_i a_i^2, & \text{if } t < 190 \\ -||\mathbf{t}(\mathbf{s}) - \mathbf{g}||_2 - \sum_i a_i^2, & \text{if } t \geq 190 \end{cases}. \tag{3.9}$$

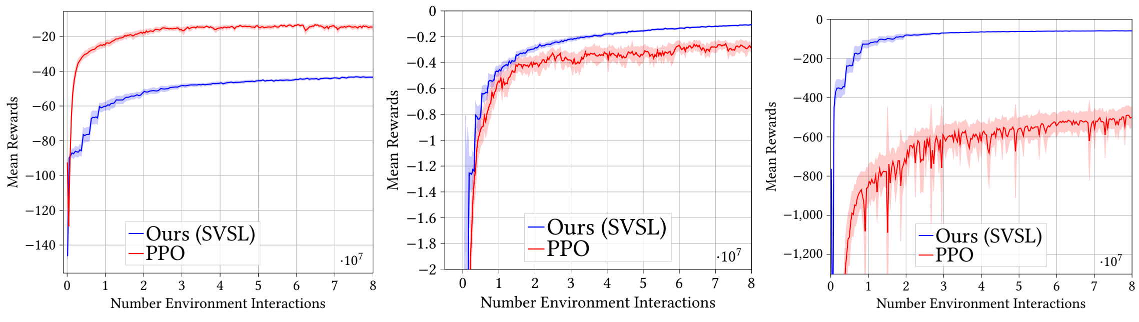
Characteristic for this reward function is that it will only return task information, i.e. tip-distance to goal, in the last 10 steps of the episode.

The third reward function is given as

$$r_t(\mathbf{s}, \mathbf{a}) = \begin{cases} -\sum_i a_i^2 - \sum_i v_i^2, & \text{if } t < 198 \\ -500 \cdot ||\mathbf{t}(\mathbf{s}) - \mathbf{g}||_2 - \sum_i a_i^2 - 1000 \cdot \sum_i v_i^2, & \text{if } t \geq 198 \end{cases}, \tag{3.10}$$

where $\mathbf{v}$ is the joint velocity at time-step t. This reward function is very similar to the reward function from Eq. 3.9. However, it only returns task information in the last two steps of the episode, which makes the task very difficult. The reward functions in Eq. 3.9 and Eq. 3.10 can be motivated from the optimal control point of view. There, usually controllers with minimum energy consumption are seeked giving rise to punishments to taken control actions in the first steps as it is done by the proposed reward functions in Eq. 3.9 and Eq. 3.10.

Observation and Contexts. For PPO we use the same observation space as described in Brockman et al. (2016), but for the Objective 3.9 and Objective 3.10 we augment this observation space with the current time-step t. The context $\mathbf{c}$ for our method is given through the two-dimensional goal position vector $\mathbf{g}$.



(a) Performances for Reward 3.8 **(b)** Performance for Reward 3.9 **(c)** Performance for Reward 3.10

Figure 3.5.: A Comparison of Episode-Based to Step-Based Policy Search. We compare our episode-based policy search method to the popular PPO. We analyze the performance on three different reward types. For (a) we consider the reward in Eq. 3.8. Here, task-information such as the distance to the goal are returned in each time-step.. For (b) we consider the reward in Eq. 3.9. Here, task-information are returned in the last 10 steps of the episode, complicating exploration. For (c) we consider the reward in Eq. 3.10, where only at the last 2 time-steps reward information are returned, which makes the task even harder. The curves show the mean and two times the standard error.

Comparison. We report the average performance and two times the standard error over 20 seeds/trials for each experiment in Fig. 3.5. The performances for the reward function in Eq. 3.8 are shown in Fig. 3.5a. Clearly, PPO outperforms our algorithm, indicating that task information in each time-step helps exploring in the raw action space to find solutions. The performances for the reward function in Eq. 3.9 is shown in Fig. 3.5b. While PPO also performs well, we can observe that our algorithm can make use of the exploration in the parameter space and outperform PPO. This effect can be seen much more clearly for the reward funcion in Eq. 3.10. Fig. 3.5c shows that PPO has clear problems to solve this task. For the settings given by the reward functions in Eq. 3.9 and Eq. 3.10 exploration in the parameter space as done by episodic policy search methods lead to better performance, whereas exploration in the raw action space leads to converging to a local optimum.

3.5. Conclusion

We proposed a new objective for learning contextual and versatile Mixtures of Experts (MoE) models. We based our objective on a maximum entropy formulation to increase the versatility of the solutions and introduced a curriculum to allow the experts to specialize. Our formulation also allows for easy online adaptation of the model complexity during training. We conducted an ablation to show the importance of the individual parts of our objective. Further, we showed that our method learns precise and versatile solutions and outperforms the baseline on sophisticated simulated robotic tasks. This work aims to present a mathematically well-founded method and demonstrates its general feasibility on various challenging tasks. Currently, the major drawback of our approach is sample efficiency, as we do not share experience between the experts. We intend to address this issue in future work, e.g., by intra-option learning. Another direction for future research is extending the approach to more complex models, such as non-linear mixtures of experts.

We expect to need fewer experts to cover the whole context space with more complex expert model representations.

4. Acquiring Diverse Skills using Curriculum Reinforcement Learning with Mixture of Experts

In this chapter we aim to address the Challenges 1, 2 and 3, which we restate in a summarized version below.

The method proposed in the following reuses the ideas of decomposing the objective and introducing an automatic curriculum learning from Chapter 3 thereby addressing *Challenge 1* and *Challenge 2* in the episode-based reinforcement learning (ERL) setting. However, a more detailed focus here is to address *Challenge 3* by extending the Mixture of Experts (MoE) policy from Chapter 3 with energy-based per-expert context distributions and deep neural networks as expert representations and proposing a tailored algorithm for training the resulting policy.

More expressive per-expert context distributions are necessary because most applications have a finite context space leading to sharp discontinuities. For example, in an environment where an agent is tasked to place an object on a table, the context space is bounded by the edges of the table and additionally, it might have regions on the table where placing the object is not possible, because other objects are in the way. Consequently, the environment's context distribution has sharp discontinuities at the edges of the table and "holes" within the table, potentially leading to highly complex distributions that need to be represented by the MoE's per-expert context distribution. Yet, so far, Gaussian parameterized context distributions in Chapter 3 were employed, which are not well suited for those edge cases. In contrast, Energy-Based models (EBM) are a rich class of distributions with high expressiveness that have proven favorable in those settings (Florence et al., 2022), but are generally not straightforward to train.

We retain the objective from Chapter 3 thereby addressing *Challenge 1*, but introduce a different sampling scheme. By repeatedly resetting the environment (without executing actions) we draw valid contexts directly from the environment's context distribution and use them to estimate the EBM's normalizer and importance sample a subset of the drawn contexts using the per-expert context distribution. Each expert's per-expert EBM still reallocates density toward high-reward contexts, preserving the automatic curriculum learning (*Challenge 2*), while avoiding the design of additional guiding rewards to push the context distribution to valid regions because the contexts are inherently valid. The resulting per-expert context distribution is an EBM that is able to represent sharp discontinuities. Additionally, this chapter extends the expert's representation to a deep neural network, enabling non-linear adaption of motion primitive parameters and optimizes these experts with trust-region black-box RL (Otto et al., 2023) for stable updates. The extensions to a per-expert energy-based distribution and deep expert representations address *Challenge 3*.

In summary, we address the *Challenges 1,2, and 3* which we have worked out in Section 1.1. Below, we provide a shorter version of these challenges for the convenience of the reader.

Challenge 1 (concise): Representation and Training of Multimodal Policies. Discovering versatile skills in reinforcement learning (RL) with multimodal policy representations such as Mixture of Experts (MoEs) models, requires novel RL algorithms that train these expressive policies efficiently and explicitly encourage the exploration of different modes.

Challenge 2 (concise): Retaining Multimodalities. Modes that yield higher returns earlier during the reinforcement learning (RL) training process might dominate the policy, leading to an overrepresentation of these modes and therefore to a collapse of the other modes. This mode collapse prevents learning versatile skills and needs to be addressed.

Challenge 3 (concise): Non-Linear Adaptation. Non-linear adaptation is a key feature for successfully learning complex skills and requires to be ensured for each mode of the multimodal policy. Additionally, highly non-linear shapes such as sharp discontinuities can naturally arise in the context space, which require special parameterization for successful representation.

The following work was published as Acquiring Diverse Skills using Curriculum Reinforcement Learning with Mixture of Experts *(Onur Celik, Aleksandar Taranovic, Gerhard Neumann 2021) in the Conference on 41st International Conference on Machine Learning, ICML 2024. Reprinted with permission of the authors. Wording, notation and formulations were revised in several places.*

4.1. Introduction

Solving tasks in diverse manners enables agents to better adapt to unknown and challenging situations. This diverse skill set is beneficial in many scenarios, such as playing table tennis, where applying different strikes (e.g. backhand, forehand, or smashing) to similar incoming balls is advantageous because the strike is less predictable for the opponent. Similarly, in scenarios with environmental changes where learned skills might be infeasible over time (e.g. grasping an object while avoiding obstacles), diverse skills provide additional adaptivity by discarding these invalid skills and relying on alternatives. This property makes them superior because complete relearning of skills is avoided.

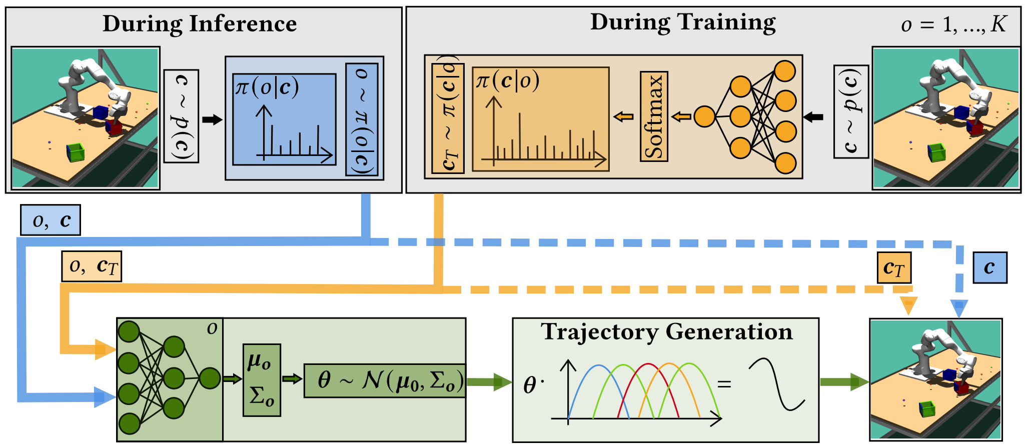


Figure 4.1.: The Sampling Procedure for Di-SkilL. During **Inference** the agent observes contexts $\mathbf{c}$ from the environment's unknown context distribution $p(\mathbf{c})$. The agent calculates the gating probabilities $\pi(o|\mathbf{c})$ for each context and samples an expert o resulting in $(o, \mathbf{c})$ samples marked in blue. During **Training** we first sample a batch of contexts $\mathbf{c}$ from $p(\mathbf{c})$, which is used to calculate the per-expert context distribution $\pi(\mathbf{c}|o)$ for each expert $o = 1, ..., K$. The $\pi(\mathbf{c}|o)$ provides a higher probability for contexts preferred by the expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$. To enable curriculum learning, we provide each expert the contexts sampled from its corresponding $\pi(\mathbf{c}|o)$, resulting in the samples $(o, \mathbf{c}_T)$ marked in orange. In both cases, the chosen $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ samples motion primitive parameters $\boldsymbol{\theta}$ for each context, resulting in a trajectory τ that is subsequently executed on the environment. Before execution, the corresponding context, e.g., the goal position of a box, needs to be set in the environment. This is illustrated by the dashed arrows, with the context in blue for inference and orange for training.

Acquiring these diverse skill sets requires learning a policy that can represent multi-modality in the behavior space. Recent advances in supervised policy learning have demonstrated the potential of training high-capacity policies capable of capturing multi-modal behaviors (Shafiullah et al., 2022; Blessing et al., 2023; Chi et al., 2023; Jia et al., 2024). These policies exhibit remarkably diverse skills and outperform state-of-the-art methods. However, Reinforcement Learning (RL) is essential to acquire skills in cases where no expert data is available, or data collection is expensive. Discovering multi-modal behaviors using RL is challenging since the policies usually rely on Gaussian parameterization and thus can only discover a single behavior.

We consider training agents that possess diverse skills, from which they can select to tackle a specific task differently. For capturing these multi-modalities in the agent's behavior space, we employ highly non-linear Mixture of Experts policies. Furthermore, we use automatic curriculum learning for efficient learning, enabling each expert to focus on a specific sub-region of the context space it favors. We introduce this curriculum shaping by optimizing for an additional per-expert context distribution that is used to sample contexts from the preferred regions to train the corresponding expert. Automatic curriculum learning has proven to increase performance by improving the exploration of agents, particularly in sparse-rewarded environments (Klink et al., 2022b).

We explore Contextual Reinforcement Learning in which a continuous-valued context describes the task (Kupcsik et al., 2013). In the example of robot table tennis (see Fig. 4.3a), a context includes the desired ball landing positions on the opponent's tableside as well as physical aspects, such as the incoming ball's velocity or friction properties. In continuous context spaces, the curriculum shaping per-expert context distributions are often parameterized as Gaussian (Klink et al., 2020a; Celik et al., 2021). However, the agent is usually unaware of the context bounds, which makes additional techniques necessary to constrain the distribution updates to stay within the context region (Celik et al., 2021). Instead, we employ energy-based per-expert context distributions, which can be evaluated for any context and effectively represent multi-modality in the context space. Importantly, our model is trained solely using context samples from the environment that are inherently valid. Our approach eliminates the need for additional regularization of the context distribution and does not require prior knowledge about the environment. Due to the overlapping probability distributions of different per-expert contexts, our resulting mixture policy offers diverse solutions for similar contexts with a high probability.

Recent research in RL has explored Mixture of Experts policies, but often these methods either train the mixture in unsupervised RL settings and then select the best-performing expert in the downstream task (Laskin et al., 2021; Eysenbach et al., 2019) or train linear experts, limiting their performance (Daniel et al., 2012; Celik et al., 2021). Our inspiration draws from recent advancements that have achieved diverse skill learning with a similar objective. However, their approach involves linear expert models with Gaussian context distributions. It requires prior knowledge of the environment to design a penalty term when the algorithm samples contexts outside the environment's bounds. These factors restrict the algorithm's performance and applicability when defining the context bounds requires knowledge, such as forward kinematics in robotics.

To summarize, we introduce a novel RL method for learning a Mixture of Experts policy that we refer to as **Di-SkilL** – **Di**verse **Skil**l **L**earning (see Fig. 4.1). Our method can generalize to the continuous range of contexts defined by the (unknown) environment's context distribution while learning diverse, and non-linear skills for solving a task defined by a specific context. Importantly, our approach operates without any assumptions about the environment. We show how we can learn multi-modal context distributions by training an energy-based model solely on context samples obtained from the environment. On multiple sophisticated simulated robot tasks, we demonstrate that we can learn diverse skills while performing on par or better than baselines.

4.2. Preliminaries

Contextual Episode-based Policy Search (CEPS). We consider learning diverse skills in the CEPS framework in which the continuous-valued context $\mathbf{c} \in C$ defines the task, e.g. a goal location to reach. The context $\mathbf{c} \sim p(\mathbf{c})$ is observed from the agent and is drawn from the environment's unknown context distribution $p(\mathbf{c})$ at the beginning of each episode. The agent's search distribution $\pi(\boldsymbol{\theta}|\mathbf{c})$ maps the context $\mathbf{c}$ to continuous-valued controller parameters $\boldsymbol{\theta} \in \Theta$, which we represent as motion primitives (MP) (Schaal, 2006;

Paraschos et al., 2013; Li et al., 2023) (see Appendix B.3). We denote $\pi(\boldsymbol{\theta}|\mathbf{c})$ as the agent's policy as common in the literature and optimize it by maximizing the objective

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c})} \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c})}[\mathrm{R}(\mathbf{c},\boldsymbol{\theta})]\right], \tag{4.1}$$

where $\mathrm{R}(\mathbf{c},\boldsymbol{\theta})$ denotes the return of a whole episode after executing the MP parameter $\boldsymbol{\theta}$ in context $\mathbf{c}$. Due to the direct return optimization, CEPS does not require the Markov assumption as in common MDPs and is therefore specifically suitable for tasks where the formulation of a Markovian reward function is difficult. We refer the reader to Section 2.3 for an in-depth discussion of episode-based reinforcement learning.

Mixture of Experts (MoE) Policy for Curriculum Learning. Due to their ability to represent multi-modality, MoE policies are a favorable choice in diverse skill learning. The common MoE policy $\pi(\boldsymbol{\theta}|\mathbf{c}) = \sum_o \pi(o|\mathbf{c})\pi(\boldsymbol{\theta}|\mathbf{c},o)$ (Bishop, 2006) contains the gating distribution $\pi(o|\mathbf{c})$ that is assigning probabilities to each expert o given context $\mathbf{c}$ during inference. However, to enable automatic curriculum learning during training, a learnable distribution $\pi(\mathbf{c}) = \sum_o \pi(\mathbf{c}|o)\pi(o)$ is required that can explicitly choose and set context samples in the environment, so each expert o can decide on which contexts it favors training (Celik et al., 2021). Using Bayes' rule $\pi(o|\mathbf{c}) = \pi(\mathbf{c}|o)\pi(o)/\pi(\mathbf{c})$ the MoE is rewritten as

$$\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c}) = \sum_o \frac{\pi_{\boldsymbol{\omega}}(\mathbf{c}|o)\pi_{\boldsymbol{\omega}}(o)}{\pi_{\boldsymbol{\omega}}(\mathbf{c})}\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c},o), \tag{4.2}$$

where we have introduced $\boldsymbol{\omega}$, that contains the learnable parameters of each expert $\pi(\boldsymbol{\theta}|\mathbf{c},o)$, the per-expert context distribution $\pi(\mathbf{c}|o)$ and the gating distribution $\pi(o)$.

The per-expert context distribution $\pi_{\boldsymbol{\omega}}(\mathbf{c}|o)$ can now be optimized and allows the expert o to choose contexts $\mathbf{c}$ it favors. We model each $\pi_{\boldsymbol{\omega}}(\mathbf{c}|o)$ as an energy-based model and each $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{c},o)$ as a neural network returning a Gaussian distribution for a context $\mathbf{c}$ (see Fig. 4.1 and Appendix B.3). However, to keep notation uncluttered, we do not write the subscript $\boldsymbol{\omega}$ to indicate the learnable parameters of a distribution throughout the next sections, but we simply denote every probability distribution which is adaptable through the optimization process with π as it is part of our policy $\pi(\boldsymbol{\theta}|\mathbf{c})$. The prior $\pi(o)$ is set to a uniform distribution throughout this work.

Self-Paced Diverse Skill Learning with MoE. Due to its ability to represent multi-modality and automatic curriculum learning, the MoE model in Eq. 4.2 is a suitable policy representation for discovering diverse skills in the same context-defined task. For explicit optimization of this policy, we are using the KL-regularized Maximum Entropy RL objective in CEPS (Celik et al., 2021)

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c}),\pi(\mathbf{c})} \mathbb{E}_{\pi(\mathbf{c})}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c})}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta})\right] + \alpha\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c})\right]\right] - \beta\mathrm{KL}\left(\pi(\mathbf{c}) \,\|\, p(\mathbf{c})\right). \tag{4.3}$$

The KL-term incentivizes the context distribution $\pi(\mathbf{c})$ to match the environment's distribution $p(\mathbf{c})$ and can be prioritized during optimization by choosing the scaling parameter β appropriately. The entropy of the mixture model incentivizes learning diverse solutions

(Celik et al., 2021) and can be prioritized with a high scaling parameter α. It is well-known that this objective is difficult to optimize for MoE policies and requires further steps to obtain a tractable lower-bound (Celik et al., 2021)

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c},o)} \mathbb{E}_{\pi(\mathbf{c}|o),\pi(\boldsymbol{\theta}|\mathbf{c},o)} \left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha \log \tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right] + \alpha \mathbb{E}_{\pi(\mathbf{c}|o)} \left[\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]\right] \tag{4.4}$$

for the expert $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ updates and a lower-bound for the per-expert context $\pi(\mathbf{c}|o)$ updates

$$\max_{\pi(\mathbf{c}|o)} \mathbb{E}_{\pi(\mathbf{c}|o)} \left[L_c(o,\mathbf{c}) + (\beta - \alpha) \log \tilde{\pi}(o|\mathbf{c})\right] + \beta \mathrm{H}\left(\pi(\mathbf{c}|o)\right), \tag{4.5}$$

where $L_c(o,\mathbf{c}) = \mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha \log \tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right] + \alpha \mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]$. The variational distributions $\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta}) = \pi_{old}(o|\mathbf{c},\boldsymbol{\theta})$ and $\tilde{\pi}(o|\mathbf{c}) = \pi_{old}(o|\mathbf{c})$ arise through the decomposition and are responsible for learning diverse solutions and concentrating on context regions with small, or no support by $\pi(\mathbf{c})$ (Celik et al., 2021). In every iteration, the variational distributions are updated in closed form to tighten the bounds. Details of the equations are in the Appendix B.1.

4.3. Related Work

Contextual Episode-based Policy Search (CEPS). CEPS is a black-box approach to reinforcement learning (RL), in which the search distribution is the agent's policy that maps the contexts to controller parameters, typically represented as motion primitives (Schaal, 2006; Paraschos et al., 2013; Li et al., 2023). One of the noteworthy advantages of CEPS lies in the independence of assumptions such as the Markovian property in common MDPs. This characteristic renders it a versatile methodology, particularly well-suited for addressing a diverse array of intricate tasks where the formulation of a Markovian reward function is difficult (Otto et al., 2023). CEPS has been explored by applying various optimization techniques, including Policy Gradients (Sehnke et al., 2010), Natural Gradients (Wierstra et al., 2014), stochastic search strategies (Hansen and Ostermeier, 2001; Mannor et al., 2003; Abdolmaleki et al., 2019), and trust-region optimization techniques (Abdolmaleki et al., 2015; Daniel et al., 2012; Tangkaratt et al., 2017), particularly in the non-contextual setting. Researchers extended the setting by incorporating linear (Tangkaratt et al., 2017; Abdolmaleki et al., 2019) and non-linear contextual adaptation (Otto et al., 2023; Li et al., 2024a), leveraging the recently introduced trust-region layers for neural networks (Otto et al., 2021). The work by (Li et al., 2024a) additionally introduces step-wise updates to improve sample-efficiency. However, all previously mentioned methods learn single-mode policies and do not address acquiring diverse skills leveraging automatic curriculum learning.

Curriculum Reinforcement Learning. Curriculum reinforcement learning can potentially increase the performance of RL agents, especially in sparse-rewarded environments (Tao et al., 2024) in which exploration is fundamentally difficult. Adapting the environment based on the agent's learning process has been proposed by several works already, e.g.

automatically generating sets of tasks or goals to increase the learning speed of the agent (Florensa et al., 2017, 2018; Sukhbaatar et al., 2018; Zhang et al., 2020; Wöhlke et al., 2020; Racaniere et al., 2020), or generating a curriculum by interpolating an auxiliary and known distribution of target tasks (Klink et al., 2022b, 2020a,b, 2024). Works propose sampling a training level from a prespecified set of environments (Jiang et al., 2021b), or unsupervised environment design (Jiang et al., 2021a; Dennis et al., 2020) based on the agent's learning process. The work by (Klink et al., 2022a) proposes improving the approximation of the state-action value function by representing it as a sum of residuals acquired in previous curriculum tasks. None of the above methods apply automatic curriculum learning on an RL problem with an MoE policy, except for the work in (Celik et al., 2021). However, they parameterize the curriculum distribution as Gaussian, suffering from low representation capacity and requiring knowledge about the environment's context distribution. Instead, we leverage energy-based models to avoid these shortcomings.

RL with Mixture of Experts (MoE). Ren et al. (2021) propose using MoE policy representation and presents a novel gradient estimator to calculate the gradients w.r.t. the MoE parameters. Huang et al. (2023) present a model-based RL approach to train latent variable models. The work presents a novel lower bound for training the multi-modal policy parameterization. Recently, Hendawy et al. (2024) proposed using MoEs for learning a shared representation in multi-task reinforcement learning, whereas Akrour et al. (2021) present how interpretable MoEs can be learned in continuous RL. These methods differ from our work in that they are not categorized in the CEPS framework, or are model-based variants and do not use automatic curriculum learning techniques. In the CEPS framework, RL with MoE policies has also been explored in the works by Daniel et al. (2012); End et al. (2017), in which an MoE model with linear experts without automatic curriculum learning is learned. Additional constraints need to be added to enforce diversity in the experts. In the work by Tosatto et al. (2021) a mixture model is used to perform RL, however, pre-recorded demonstration data is required to train the mixture model and no curriculum learning is considered. Related method to MoEs, Product of Experts was used in Hansel et al. (2023); Le et al. (2023) for motion generation.
The work by Celik et al. (2021) also uses MoE policies and relies on the maximum entropy objective as we do, however, their method only considers linear experts with Gaussian per-expert distributions which limits the performance and consequently requires many experts to solve a task. Moreover, it requires environment knowledge to hand-tune a punishment term to keep the optimization of the per-expert context distributions within the context bounds.

Quality-Diversity Optimization (QDO). Learning diverse skills has also been explored in the evolutionary strategy community, most notably with the MAP-Elites algorithm (Cully et al., 2015), where behavioral descriptors are defined to distinguish the different learned motions. Extensions (Nilsson and Cully, 2021; Faldor et al., 2023a,b) have been proposed to improve the performance of these methods. However, these methods can not easily be applied to the contextual setting where different controller parameters should be chosen in different situations such that post hoc adaptations (Keller et al., 2020; Faldor et al., 2023b) are required. In contrast to QDO methods, in our work diversity measurement naturally arises through the considered objective and does not need defining behavioral descriptors. Moreover, Di-SkilL indirectly learns a gating distribution that selects the expert after observing a context.

Unsupervised Reinforcement Learning (URL). URL also considers learning diverse policies (Yang et al., 2024; Eysenbach et al., 2021; Laskin et al., 2021; Eysenbach et al., 2019; Campos et al., 2020; Lee et al., 2019; Liu and Abbeel, 2021). The objective differs from ours and skills are trained in the absence of an extrinsic reward. We discuss parallels in the Appendix B.2.

4.4. Diverse Skill Learning

The common Contextual Episodic Policy Search (CEPS) loop (Kupcsik et al., 2013) with a Mixture of Experts (MoE) policy representation learning observes a context $\mathbf{c}$, and then selects an expert o that subsequently adjusts the controller parameters $\boldsymbol{\theta}$ given ($\mathbf{c}$, o). We consider the same process during testing time, as shown in blue color in Fig. 4.1 (see also Fig. B.1a). However, the procedure changes during training for Di-SkilL as automatic curriculum learning requires that the agent can determine which context regions it prefers to focus on. In this case, we observe a batch of context samples from the environment's context distribution $p(\mathbf{c})$. For each of these samples, every per-expert context distribution $\pi(\mathbf{c}|o)$ calculates a probability, which results in a categorical distribution over the contexts $\mathbf{c}$. We use these probabilities to sample contexts $\mathbf{c}_T$ for the corresponding expert o resulting in $(\mathbf{c}_T, o)$ sample pairs (see orange parts in Fig. 4.1 and Fig. B.1b). Each chosen expert o provides Gaussian distributions over the motion primitive parameters $\boldsymbol{\theta}$ by mapping the contexts $\mathbf{c}_T$ to mean vectors $\boldsymbol{\mu}_o$ and covariance matrices Σ_o using a parameterized neural network. We can now sample motion primitive parameters $\boldsymbol{\theta}$ from these Gaussian distributions to generate trajectories τ using a motion primitive generator. These trajectories are subsequently executed on the environment (green color in Fig. 4.1) and an episode return $R(\boldsymbol{\theta}, \mathbf{c}_T)$ is observed and used for updating the MoE (see Section 4.4.2). Yet, there exist several issues for a stable overall training of the MoE model, which requires special treatment for each $\pi(\mathbf{c}|o)$ and $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$. Algorithmic details and parameterizations of the model can be found in the Appendix B.3.

4.4.1. Energy-Based Model For Automatic Curriculum Learning

To illustrate these issues, we consider a bounded, uniformly distributed two-dimensional environment context distribution $p(\mathbf{c})$ (see example in the Appendix B.3 in Fig. B.1c). It is challenging for a Reinforcement Learning (RL) agent to automatically learn its curriculum $\pi(\mathbf{c}|o)$ within the valid context space (Celik et al., 2021). Hard discontinuities such as steps often naturally arise in $p(\mathbf{c})$ due to the environment's finite support in real-world environments. For instance, in an environment where the agent's task is to place an object in specific positions on a table, the probability of observing a goal position outside the table's surface is zero. This implies that a large subset of the context space has no probability mass. Therefore, exploration in these regions might be difficult if there is no guidance encoded in the reward. Even if it is guaranteed that $\pi(\mathbf{c}|o)$ only samples valid contexts, it still needs to be able to represent multi-modal distributions, such as illustrated in Fig. B.1d. This multi-modality can be present because of environmental circumstances or simply if experts $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ prefer contexts in spatially apart regions. For the object placing example, this could correspond to regions on the table where the object cannot be placed due to obstacles or holes. We therefore require $\pi(\mathbf{c}|o)$ being able to represent **i)** complex distributions, **ii)** multi-modality and **iii)** only explore within the valid context bounds of $p(\mathbf{c})$. We propose parameterizing each per-expert distribution $\pi(\mathbf{c}|o)$ as an energy-based model

$$\pi(\mathbf{c}|o) = \exp(\boldsymbol{\phi}_o(\mathbf{c}))/Z \tag{4.6}$$

to address the issues i) and ii), where the energy function ϕ_o is a per-expert learnable neural network. Energy-based models (EBMs) have shown to be capable of representing sharp discontinued functions and multi-modal distributions (Florence et al., 2022). Yet, they are hard to train and sample from due to the intractable normalizing constant $Z = \int_{\mathbf{c}} \exp(\boldsymbol{\phi}_o(\mathbf{c}))d\mathbf{c}$. We can circumvent and address these issues iii) by approximating the normalizing constant with contexts $\mathbf{c} \sim p(\mathbf{c})$ as $Z \approx \sum_{i=1}^{N} \exp(\phi_o(\mathbf{c}_i))$. This approximation is justified as we can sample from $p(\mathbf{c})$ by simply resetting the environment without execution. Additionally, the EBM will encounter important parts of the context space during the training by resampling a large enough batch of contexts $\mathbf{c} \sim p(\mathbf{c})$ in each iteration. Each expert can therefore sample preferred contexts from the current batch of valid contexts by calculating the probability for each of the contexts using $\pi(\mathbf{c}|o)$ as parameterized in Eq. 4.6. Updating the parameters of the EBM can readily be addressed by the standard RL objective for diverse skill learning, as described in the next section. It should be noted that explicit models such as Gaussians, or Normalizing Flows (Papamakarios et al., 2021) can also be used to parameterize $\pi(\mathbf{c}|o)$, but their support cannot be easily restricted to a bounded space with hard discontinuities defined by the environment. Therefore, sampling from an explicit $\pi(\mathbf{c}|o)$ can easily generate invalid contexts, especially if the valid distribution has hard non-linearities.

4.4.2. Updating the Mixture of Experts Model

We update each expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ and its corresponding per-expert context distribution $\pi(\mathbf{c}|o)$ by maximizing the objectives in Eq. 4.4 and in Eq. 4.5, respectively. These decomposed objectives allow us to independently update both distributions and to retain the properties of diverse skill learning from the objective in Eq. 4.3. However, updating the distributions is not straightforward due to the bi-level optimization that leads to a dependency on both terms. This is particularly challenging for the expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ as the sampled contexts $\mathbf{c}$ can drastically change from one iteration to another if $\pi(\mathbf{c}|o)$ changes too aggressively. The same applies for updating $\pi(\mathbf{c}|o)$ as calculating the objective requires calculating an integral over $\boldsymbol{\theta}$ under the expectation of $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$. For a stable update for both distributions, we employ trust-region updates to restrict the change of both distributions from one iteration to another. These updates have been shown to improve the learning process (Peters et al., 2010; Schulman et al., 2015, 2017; Otto et al., 2021).

Expert Update. We parameterize each expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ with a single neural network and update them by a trust-region constrained optimization

$$\begin{aligned} \max_{\pi(\boldsymbol{\theta}|\mathbf{c},o)} \; & \mathbb{E}_{\pi(\mathbf{c}|o),\pi(\boldsymbol{\theta}|\mathbf{c},o)} \left[\mathrm{R}(\mathbf{c}, \boldsymbol{\theta}) + \alpha \log \tilde{\pi}(o|\mathbf{c}, \boldsymbol{\theta})\right] + \alpha \mathbb{E}_{\pi(\mathbf{c}|o)} \left[\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c}, o)\right]\right] \\ \text{s.t.} \quad & \mathrm{KL}\left(\pi(\boldsymbol{\theta}|\mathbf{c}, o) \;||\; \pi_{\text{old}}(\boldsymbol{\theta}|\mathbf{c}, o)\right) \leq \epsilon \quad \forall\, \mathbf{c} \in C, \end{aligned} \tag{4.7}$$

where the KL-bound ensures that the expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ does not differ too much from the expert $\pi_{\text{old}}(\boldsymbol{\theta}|\mathbf{c}, o)$ from the iteration before for each context $\mathbf{c}$. The entropy bonus $\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c}, o)\right]$ incentivizes $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ to fully cover the parameter space while avoiding $(\boldsymbol{\theta}, \mathbf{c})$ regions that are covered by other experts o. The latter is guaranteed by $\tilde{\pi}(o|\mathbf{c}, \boldsymbol{\theta})$ which rewards $(\boldsymbol{\theta}, \mathbf{c})$ regions that can be assigned to expert o with high probability. We efficiently update the experts using trust region layers (Otto et al., 2021, 2023).

Per-Expert Context Distribution Update. We consider the objective with the augmented rewards as shown in Eq. 4.5 for updating each context distribution $\pi(\mathbf{c}|o)$. We can not apply the trust region layers (Otto et al., 2021) in this case, as $\pi(\mathbf{c}|o)$ is a discrete distribution over the context samples $\mathbf{c}_i$ parameterized by the EBM. Yet, we can still use PPO (Schulman et al., 2017) for updating $\pi(\mathbf{c}|o)$ and simplifying our objective, as we can now calculate many terms in closed form. For this, we rewrite the objective as

$$\max_{\pi(\mathbf{c}|o)} \sum_{\mathbf{c}_i \sim p(\mathbf{c})} \pi(\mathbf{c}_i|o) L_c(o, \mathbf{c}_i) + \sum_{\mathbf{c}_i \sim p(\mathbf{c})} \pi(\mathbf{c}_i|o) \big((\beta - \alpha) \log \tilde{\pi}(o|\mathbf{c}_i) - \beta \log \pi(\mathbf{c}_i|o)\big) \tag{4.8}$$

and observe that all terms in the second sum can be calculated in closed form. Note that the first term is approximated by resampling the context samples using $\pi(\mathbf{c}|o)$ since computing $L_c(o, \mathbf{c})$ requires calculating the integral over $\boldsymbol{\theta}$ under the expectation of $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ as $L_c(o, \mathbf{c}) = \mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c}, \boldsymbol{\theta}) + \alpha \log \tilde{\pi}(o|\mathbf{c})\right] + \alpha \mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c}, o)\right]$. This expectation can only be estimated for context vectors that are actually chosen by the component. The entropy bonus in Eq. (4.8) incentivizes covering of the context space, while focusing on context regions that are not, or only partly covered by other options. The latter is guaranteed by $\tilde{\pi}(o|\mathbf{c})$ which assigns a high probability if expert o can be assigned to the context $\mathbf{c}$.

4.4.3. How does Diversity Emerge?

From the Eq. 4.7 and Eq. 4.8 it is clear that diverse behaviors, represented by the experts, emerge from the interplay of those terms in Eq. 4.7 and Eq. 4.8. We visually demonstrate the meaning of the individual terms on the 5-Link Reacher task (see Fig. 4.2d). The Reacher needs to reach a goal position in the two-dimensional space with its tip. In this task, a context represents the goal position within the context space, visualized as a red circle around the reacher's fixed first joint (Fig. 4.2a). We trained Di-SkilL with 50 experts.

In Fig. 4.2a we show the high-probability regions of the individual per-expert context distributions $\pi(\mathbf{c}|o)$, by setting the color intensity proportional to this probability. Each color

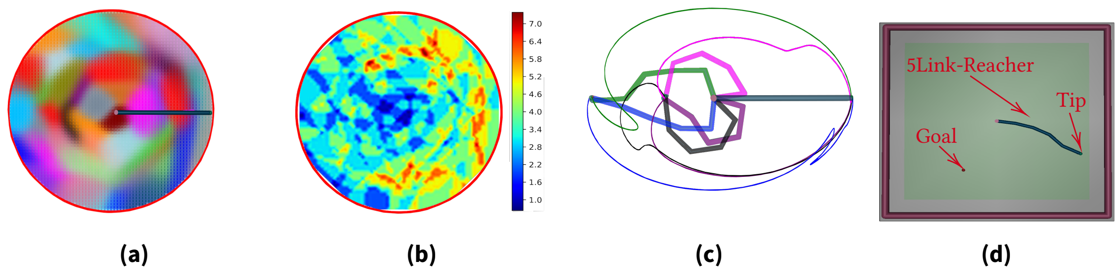


Figure 4.2.: **a)** High-probability regions of the individual per-expert context distributions $\pi(\mathbf{c}|o)$, where a color represents an expert o. The red circle marks the context space of goal-reaching positions for the 5-Link Reacher's tip. The specialization of $\pi(\mathbf{c}|o)$ is induced by $\tilde{\pi}(o|\mathbf{c})$. **b)** Different $\pi(\mathbf{c}|o)$ need to overlap for learning diverse skills. This overlapping is induced by the entropy bonus $\mathrm{H}\left[\pi(\mathbf{c}|o)\right]$. **c)** Different tip trajectories sampled in the same contexts. The trajectories and the end joint constellation are in the same color. The diversity in the parameter space is induced by $\tilde{\pi}(o|\mathbf{c}, \boldsymbol{\theta})$. **d)** Visualization of the 5-Link Reacher task (**5LR**).

represents an individual expert o. Each $\pi(\mathbf{c}|o)$ concentrates on a sub-region of the context space such that the corresponding $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ becomes an expert there. This specialization is incentivized by the term $\tilde{\pi}(o|\mathbf{c})$ in Eq. 4.8. However, for learning diverse behaviors for the same context regions, it is necessary that the per-expert context distributions $\pi(\mathbf{c}|o)$ overlap, which is motivated by the entropy term $\mathrm{H}\left[\pi(\mathbf{c}|o)\right]$ in Eq. 4.8.

These overlapping context regions are visualized in Fig. 4.2b, where we count how many experts o are active for each context. The figure shows that more experts prefer regions close to the initial position of the reacher, indicating that these contexts are easier to solve. Despite the closeness to the reacher's initial position, the agent does not have to exert much energy to reach these points. Indeed, both aspects are present in the task's reward function (see Appendix B.4 for details), explaining why the left half plane of the context space has fewer overlapping. However, the learned MoE has two or more experts active in most parts of the context region. These experts differ in their behavior (see Fig. 4.2 for examples), which is motivated by the terms $\tilde{\pi}(o|\mathbf{c}, \boldsymbol{\theta})$ and $\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c}, o)\right]$ in Eq. 4.7.

4.5. Experiments

In our evaluations, we compare Di-SkilL against the baselines **BBRL** (Otto et al., 2023) and **SVSL** (Celik et al., 2021). Whenever the environment satisfies the Markov properties, we additionally compare against **PPO** (Schulman et al., 2017). BBRL and SVSL are suitable baselines as they are state-of-the-art CEPS algorithms that can learn complex skills. BBRL can learn highly non-linear policies leveraging trust region updates. SVSL learns a linear Mixture of Experts (MoE) model and can capture multi-modality in the behavior space. We consider challenging robotic environments with continuous context and parameter spaces. The considered environments either have a non-Markovian reward function, i.e. require retrospective data for calculation, or temporally sparse reward functions, increasing the learning complexity due to more difficult exploration.

We start by providing an overview of the benchmarking environments. In an ablation study, we then show that automatic curriculum learning is an essential feature of Di-SkilL to learn high-performing skills. Lastly, on five sophisticated robot simulation tasks we report the **performance** of Di-SkilL against the baselines. The results show that Di-SkilL performs on par, or better than the baselines on all tasks. In addition to the performance analysis, we qualitatively show Di-SkilL's learned **diverse skills** on the challenging table tennis, box pushing, and reaching tasks.

4.5.1. Environments

The considered environments are visualizations in Fig. 4.3a. Throughout all environments, we used ProDMPs (Li et al., 2023) to generate trajectories (see Appendix B.3). Detail descriptions are provided in the Appendix B.4.

Table Tennis (TT). A 7-degree of freedom (DoF) robot has to learn fast and precise motions to hit the ball to a desired position on the opponent's side. The 4-dim. context consists of the incoming ball's landing position and the desired ball's landing position on the opponent's side. The TT environment requires good exploratory behavior and has a non-Markovian reward structure, making step-based approaches infeasible to learn useful skills (Otto et al., 2023).

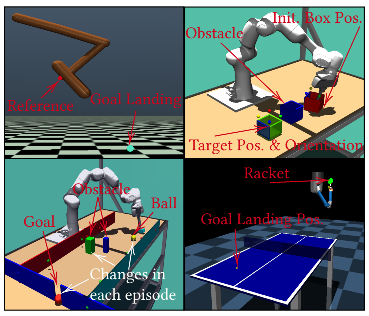
Table Tennis Hard (TT-H). We extend the TT environment to a more challenging version by varying the ball's initial velocity. This additionally increases the learning complexity, as the agent now needs to reason about the physical effects of changed velocity ranges and requires improved adaptability.

5-Link Reacher (5LR). The 5-Link reacher has to reach a goal position within all quadrants in the context space (see Fig. 4.2a) as opposed to the version in Otto et al. (2023), where the multi-modality in the behavior space (see Fig. 4.2c) was avoided by constraining the context space to the upper half of the context space. Additionally, the time-sparse reward makes this task a challenging benchmark.

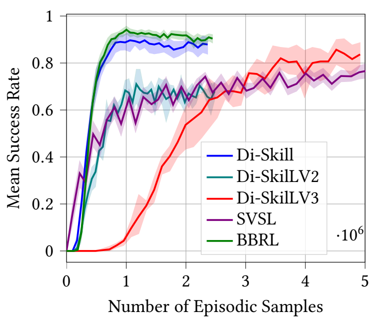
Hopper Jump (HJ). Presented in Otto et al. (2023) in which the Hopper (Brockman et al., 2016) is tasked to jump as high as possible while landing in a goal position. The HJ has a non-Markovian reward, making step-based RL methods unfeasible to learn useful policies (Otto et al., 2023).

Box Pushing with Obstacle (BPO). A 7DoF robot is tasked to push a box to a target position and rotation while avoiding an obstacle. In addition to the time-spare reward (Otto et al., 2023), our version includes the obstacle and considers a larger range of the box's target position.

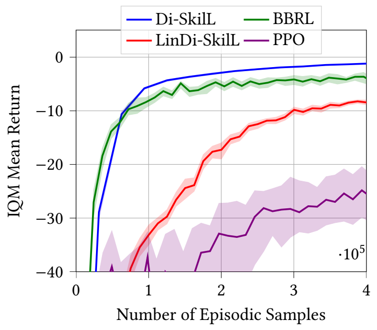
Robot Mini Golf (MG). The 7DoF robot has to hit the ball in an environment with two obstacles (static, varying), such that it passes the tight goal. The context is the obstacle's, the goal's, and the ball's position. The MG environment has a non-markovian reward, making step-based RL methods unfeasible to learn useful policies (Otto et al., 2023).



(a) Visualization of Environments



(b) Mean Success Rate (**TT**)



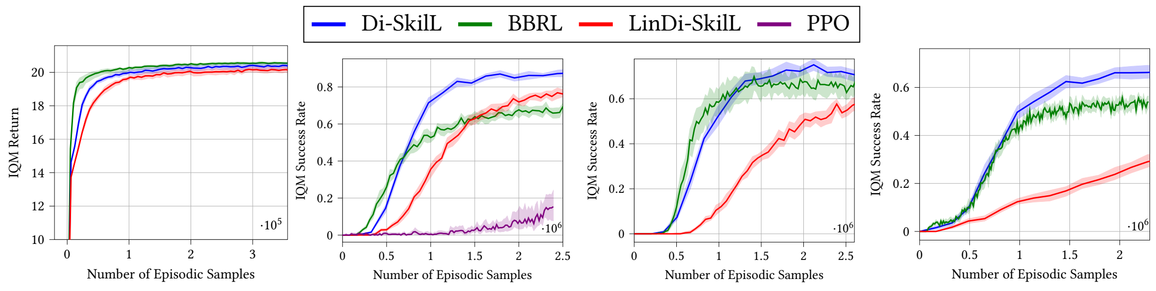
(c) IQM Mean Return (**5LR**)

Figure 4.3.: a) (left top) Hopper Jump Task (**HJ**). (Top right) Box Pushing with Obstacle (**BPO**). (Bottom Left) Robot Mini Golf (**MG**). (Bottom right) Robot table tennis (**TT**). **b)** Ablation studies, showcasing the need for automatic curriculum learning for Di-SkilL. BBRL and Di-SkilL can solve TT environment decently. Di-SkilL's variants without curriculum learning struggle to achieve a good performance. SVSL needs more samples to achieve around 80% success rate, suffering under the linear experts. **c)** Performance of Di-SkilL, BBRL, LinDi-SkilL, and PPO on 5LR with sparse in-time rewards.

4.5.2. ACL Benefits

Automatic Curriculum learning (ACL) enables Di-SkilL's experts to shape their curriculum by explicitly sampling from preferred context regions. We analyze the importance of this feature by comparing the performance of variants of Di-SkilL on the table tennis (TT) task.

For both variants *Di-SkilLV2* and *Di-SkilLV3* we disable ACL by fixing the term induced by the variational distribution to $\log \tilde{\pi}(o|\mathbf{c}) = 0$ in Eq. 4.8 and by setting the entropy scaling parameter $\beta = 2000$. Ignoring the variational distribution $\tilde{\pi}(o|\mathbf{c})$ during training eliminates the intrinsic motivation of the per-expert context distribution $\pi(\mathbf{c}|o)$ to focus on sub-regions in the context space that are not, or only partially, covered by any other $\pi(\mathbf{c}|o)$ (Section 4.4.3). Setting $\beta = 2000$ incentivizes each $\pi(\mathbf{c}|o)$ to maximize its entropy, resulting in a uniform distribution in the environment's bounded context space. For Di-SkilL we keep the ACL and set $\beta = 4$. We provide the same number of 50 context-parameter samples per expert for *Di-SkilLV2* and Di-SkilL, whereas *Di-SkilLV3* receives 260 samples per expert in each iteration. All variants possess 5 experts.
In Fig. 4.3b we report the mean success rates and the 95% confidence interval for each method on at least 4 seeds. *Di-SkilLV2* converges to a much smaller success rate, and *Di-SkilLV3* needs more samples to reach the level of Di-SkilL. BBRL and Di-SkilL achieve high success rates, while BBRL performs slightly better. SVSL shows worse performance, even though the model has 20 experts. The results indicate that ACL is an essential feature of Di-SkilL ensuring that Di-SkilL can learn high-perfroming skills with fewer samples. Moreover, SVSL's poor performance shows that Gaussian parameterized per-expert context distributions that require additionally tuned punishment terms for guided updates are together with linear experts incapable of achieving a satisfying performance.



(a) IQM Mean Return (**HJ**) **(b)** IQM Success Rate (**BPO**) **(c)** IQM Success Rate (**TT-Hard**) **(d)** IQM Success Rate (**MG**)

Figure 4.4.: Performance on the a) HJ (Hopper Jump) **b) BPO** (Box Pushing with Obstacle), **c) TT-H** (Table Tennis Hard), and **d) MG** (Robot Mini Golf) tasks. **a)** Di-SkilL performs on par with BBRL on the HJ task. **b)** The multi-modality introduced by the obstacle in the box pushing task leads to worse performance for BBRL than for Di-SkilL and LinDi-SkilL. PPO suffers under the time-sparse reward setting. **c)** While BBRL converges faster, Di-SkilL achieves a higher success rate eventually. **d)** Di-SkilL outperforms the baselines on the **MG** task. LinDi-SkilL performs poorly on the non-Markovian rewarded tasks TT-H and MG, indicating that highly non-linear policies are beneficial.

4.5.3. Analyzing the Performance and Diversity

For a detailed analysis, we have evaluated all methods on *24 seeds* for each environment and algorithm and report the *interquantile mean* (IQM) with a 95% stratified bootstrap confidence interval as suggested by Agarwal et al. (2021). Please note that SVSL requires designing a punishment function to guide the context samples in the environment's valid context region, which makes its application difficult, especially if the context influences the objects' physics. We therefore propose comparing against LinDi-SkilL instead of SVSL. LinDi-SkilL also has linear experts but benefits from Di-SkilL's energy-based per-expert context distribution $\pi(\mathbf{c}|o)$ eliminating the need for punishment functions.

The performance curve of the **HJ** task in Fig. 4.4a shows that Di-SkilL performs on par with BBRL, while BBRL converges slightly faster. Both methods can solve the task, indicating that the task doesn't require diversity. We can also see that LinDi-SkilL achieves a similar performance as BBRL and Di-SkilL, but needs more samples to converge. We provide additional analysis of this task in Appendix B.5.

Fig. 4.4b shows the performance curves on the **BPO** task. The obstacle introduces multi-modality in the behavior space which cannot be captured by a single-mode policy. This multi-modality explains why DiSkilL and LinDi-SkilL outperform BBRL, while Di-SkilL still achieves the highest success rate. PPO's poor performance indicates that time-correlated exploration as used with motion primitives is effective in sparse rewarded tasks.

A similar performance behavior can be observed in the **5LR** task. In Fig. 4.3c we report the achieved returns and observe that Di-SkilL outperforms BBRL due to the ability to capture multi-modal behaviors (e.g. reaching from different sides) while PPO suffers from the sparse rewarded setting. Moreover, LinDi-SkilL's linear experts cause slow convergence, indicating that more experts are needed to effectively cover the whole context space. For both tasks, Di-SkilL's diverse skills in the parameter space $\boldsymbol{\theta}$ induce different behaviors. Fig. 4.5 shows diverse box trajectories to several fixed goal and obstacle positions in the

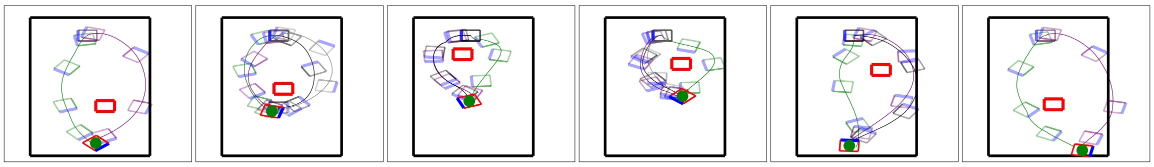
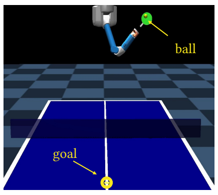
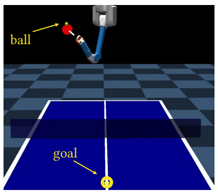


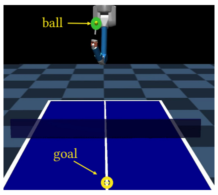
Figure 4.5.: Di-SkilL's **Diverse Skills** for the Box Pushing with Obstacle **BPO** Task. The figures visualize diverse solutions to the same contexts **c** on a table (black rectangle). The red, thick rectangle represents the obstacle. The 7DoF robot is tasked to push the box (shown in different colors for each solution found) to the goal box position (red rectangle with a green dot) and align the blue edges to match the orientation. The context consists of the 2-dim. obstacle position, the 2-dim. goal position and the 1-dim. goal orientation around the z-axis. We visualized successful box trajectories for each sampled skill from the same Di-SkilL policy with 10 experts. The diversity learned in the parameter space results in different box trajectories ranging in position and orientation.



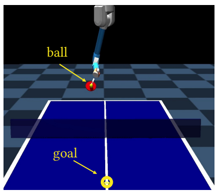
(a) Backhand Drive



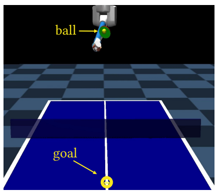
(b) Forehand Drive



(c) Backhand Block



(d) Forehand Push



(e) Backhand Smash

Figure 4.6.: Di-SkilL's **Diverse Skills** for the Table Tennis Hard **TT-H** task. We fixed the ball's desired landing position and varied the serving landing position and the ball's initial velocity. Di-SkilL can return the ball in different striking types such as backhand or forehand strikes, where hitting the ball with the green side of the racket is referred to as backhand and forehand otherwise. The shown striking styles are captured from the same Di-SkilL policy that was trained with 10 experts.

BPO task, whereas Fig. 4.2c shows different tip trajectories to several fixed goal positions in the 5LR task.

The non-Markovian rewarded tasks (**TT-H** and **MG**) show that non-linear policies as learned by BBRL and Di-SkilL are beneficial. Di-SkilL and BBRL perform similarly well on the TT-H task (see Fig. 4.4c), where Di-SkilL achieves a slightly higher end success rate compared to BBRL. However, there is a clear performance gap between Di-SkilL and BBRL on the MG task (see Fig. 4.4d) with Di-SkilL outperforming BBRL. In both tasks, LinDi-SkilL performs worse than Di-SkilL and BBRL indicating that linear experts are insufficient for solving these tasks.

Di-SkilL can discover diverse striking styles in the table tennis task (TT-H). Fig. 4.6 shows some of these learned skills. Additional strike visualizations are in Appendix B.5.

4.6. Conclusion and Future Work

In this paper, we propose Diverse Skill Learning (Di-SkilL), a novel method for learning diverse skills using a contextual Mixture of Experts. Each expert automatically learns its curriculum by optimizing for a per-expert context distribution $\pi(\mathbf{c}|o)$. We have demonstrated challenges that arise through enabling automatic curriculum learning (ACR) and proposed parameterizing $\pi(\mathbf{c}|o)$ as energy-based models (EBMs) to address these challenges. Additionally, we provided a methodology to efficiently optimize these EBMs. We also proposed using trust-region updates for the deep experts to stabilize our bi-level optimization problem. In an ablation, we have shown that ACR is necessary for efficient and performant learning. Moreover, in sophisticated robot simulation environments, we have shown that our method can learn diverse skills while performing on par or better than the baselines. Currently, the major drawback of our approach is its inability to replan, causing failures in the tasks if the robot even has small collisions with objects. We intend to address this issue in future research. To improve the sample complexity of our approach, we additionally plan to use off-policy RL techniques.

5. DIME: Diffusion-Based Maximum Entropy Reinforcement Learning

In this chapter we aim to address the Challenges 1 and 3 using diffusion-based policies, where we restate the respective challenges in a concise version below.

Chapters 3 and 4 have focused on learning versatile skills using Mixture of Experts (MoE) policies in the episode-based reinforcement learning (ERL) framework. However, training MoEs can require intensive hyper-parameter tuning, and depending on the task, they might require many experts to obtain a good performance. In contrast, diffusion models (Song et al., 2021; Ho et al., 2020; Karras et al., 2022) have shown remarkable results in generative modeling and representing highly complex multimodal distributions with a single model in high-dimensional spaces. At the same time, their training is very stable framing them as a user-friendly and promising alternative to MoE policies. However, essential statistics such as the marginal entropy of the diffusion policy are intractable, which makes key concepts such as exploration control or evaluating the maximum entropy RL objective intractable.

In contrast to the previous chapters, this chapter considers step-based RL (SRL), which is known to be more sample-efficient than ERL and more commonly used in the RL literature. We consider the maximum entropy RL objective (see Chapter 2.4) that is well studied and has become the standard approach in RL due to its incentive for exploration. Similar to the methods proposed in Chapters 3 and 4, we take inspiration from the approximate inference with diffusion models literature and propose a tractable lower bound for the maximum entropy RL objective for a policy iteration scheme. The resulting method can effectively train a diffusion-based policy and performs favorably, especially on high-dimensional control tasks, and importantly provides a framework for controlling the exploration for diffusion models.

Due to their inherent representational capacity, diffusion models can naturally represent multimodality and in conjunction with the proposed lower bound on the maximum entropy RL objective, exploration and therefore, discovering new modes is incentivized. Importantly, the proposed algorithm allows controlling the exploration behavior of the diffusion policy and can be applied without much hyperparameter tuning. Because of these features, this chapter addresses *Challenge 1.* Additionally, due to the diffusion model's principle, non-linear adaptation can be inherently represented, which addresses *Challenge 3.*

Challenges 1 and 3 from Section 1.1 are restated in a shorter version below for the convenience of the reader.

Challenge 1 (concise): Representation and Training of Multimodal Policies. Discovering versatile skills in reinforcement learning (RL) with multimodal policy representations such as diffusion models, requires novel RL algorithms that train these expressive policies efficiently and explicitly encourage the exploration of different modes.

Challenge 3 (concise): Non-Linear Adaptation. Non-linear adaptation is a key feature for successfully learning complex skills and requires to be ensured for each mode of the multimodal policy.

The following work was published as DIME: Diffusion-Based Maximum Entropy Reinforcement Learning *(Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, Gerhard Neumann) in the Conference on 42nd International Conference on Machine Learning, ICML 2025. Reprinted with permission of the authors. Wording, notation and formulations were revised in several places.*

5.1. Introduction

The maximum entropy reinforcement learning (MaxEnt-RL) objective augments the task reward in each time step with the entropy of the policy (Ziebart et al., 2008; Toussaint, 2009; Haarnoja et al., 2017, 2018b). This objective has several favorable properties among which improved exploration (Ziebart, 2010; Haarnoja et al., 2017) is crucial for RL. Recent successful model-free RL algorithms leverage these favorable properties and build upon this framework (Bhatt et al., 2024; Nauman et al., 2024) improving sample efficiency and leading to remarkable results. However, the policies are traditionally parameterized using Gaussian distributions, significantly limiting their representational capacity. On the other hand, diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021; Karras et al., 2022) are highly expressive generative models and have proven beneficial in representing complex behavior policies (Reuss et al., 2023; Chi et al., 2023). However, important metrics such as the marginal entropy are intractable to compute (Zhou et al., 2024) which restricts their usage in RL. Because of this shortcoming, recent methods propose different ways to train diffusion-based methods in off-policy RL. While these methods are discussed in more detail in the related work section, most of them require additional techniques to add artificial (in most cases Gaussian) noise to the generated actions to induce exploration in the behavior generation process. Hence, they do not leverage the diffusion model to generate potentially non-Gaussian exploration patterns but fall back to mainly Gaussian exploration. Nonetheless, there have been significant advances in training diffusion-based models for approximate inference (Berner et al., 2024; Richter and Berner, 2024). Since the policy improvement in MaxEnt-RL can also be cast as an approximate inference problem to the energy-based policy Haarnoja et al. (2017), it is a natural step to explore these parallels.

We propose Diffusion-Based Maximum Entropy Reinforcement Learning (DIME). *DIME* leverages recent advances in approximate inference with diffusion models (Richter and

Berner, 2024) to derive a lower bound on the MaxEnt objective. We propose a policy iteration framework with monotonic policy improvement that converges to the optimal diffusion policy. Additionally, building on recent off-policy RL algorithms such as Cross-Q (Bhatt et al., 2024) and distributional RL (Bellemare et al., 2017), we propose a practical version of DIME that can be used for training diffusion-based RL policies. On 13 challenging continuous high-dimensional control benchmarks, we empirically validate that DIME significantly outperforms other diffusion-based baselines on all environments and consistently outperforms other state-of-the-art RL methods based on a Gaussian policy on 10 out of 13 environments, while being computationally more efficient and requiring less algorithmic design choices as the current state of the art baseline BRO (Nauman et al., 2024).

5.2. Related Work

Maximum Entropy RL. The maximum entropy RL framework uses the entropy of the policy at each time step as an additional objective, providing a principled way of inducing exploration in the RL policy. It is different from entropy regularized RL (Neu et al., 2017), where the entropy of the policy is maximized only for the current time step. Haarnoja et al. (2017) proposed Soft-Q Learning, where amortized Stein variational gradient descent (Wang and Liu, 2016) (SVGD) is used to train a parameterized sampler that can sample from the energy-based policy. SAC (Haarnoja et al., 2018b) proposes an actor-critic RL method but frames the policy update as an approximate inference problem to the energy-based policy using a Gaussian policy parameterization. SAC has been extended to energy-based policies using SVGD in Messaoud et al. (2024), where the authors also propose a new method to estimate the entropy in closed form. While SVGD is a powerful method for learning an energy-based policy, it is harder to scale these approaches to high-dimensional control problems. For improving exploration, LSAC (Ishfaq et al., 2025) proposes leveraging Langevin Monte Carlo (Welling and Teh, 2011) in conjunction with a distributed critic objective to sample a state-action value. Haarnoja et al. (2018a) proposes learning a latent variable model as a policy representation, but relies on the change of variable formula to express the density of the policy by calculating the Jacobian of the transformations. Recent advances of SAC also define the state-of-the-art in off-policy RL in many domains, such as CrossQ (Bhatt et al., 2024) and BRO (Nauman et al., 2024). CrossQ proposed removing the target network by leveraging batch renormalization and BRO scales to large networks in RL by using several methods such as optimistic exploration (Nauman and Cygan, 2023), network resets (Nikishin et al., 2022), weight decay, and high update-to-data ratios.

Diffusion-Based Policies in RL. Early works have researched diffusion models in offline RL (Lange et al., 2012; Levine et al., 2020) as trajectory generators (Janner et al., 2022) or as expressive policy representations (Wang et al., 2023; Kang et al., 2023; Hansen-Estruch et al., 2023; Chen et al., 2023; Ding and Jin, 2024; Mao et al., 2024; Fang et al., 2025; Lu et al., 2023). More recently, diffusion models in online RL have become more popular. DIPO (Yang et al., 2023) proposes training a diffusion-based policy using a behavior cloning loss. The actions in the replay buffer serve as target actions for the policy improvement step and are updated using the gradients of the Q-function $\nabla_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$. DIPO has been extended to develop methods for learning multi-modal behaviors (Li et al., 2024b) by leveraging hierarchical clustering to isolate different behavior modes. DIPO relies on the stochasticity inherent to the diffusion model for exploration and does not explicitly control it via an objective. QSM (Psenka et al., 2024) directly matches the policy's score with the gradient of the Q-function $\nabla_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$. While their objective avoids differentiating through the whole diffusion process, the proposed objective disregards the entropy of the policy and, therefore, exploration. Consequently, QSM needs to add noise to the final action of the diffusion process. More recently, DACER (Wang et al., 2024) proposed using the data-generating process as the policy representation and backpropagating the gradients through the diffusion process. However, they do not consider a backward process as we do, and their objective for updating the diffusion model is based on the expected Q-values only. To incentivize the exploration, DACER adds diagonal Gaussian noise to the sampled actions, where the variance of this noise is controlled by a scaling term that is updated automatically using an approximation of the marginal entropy by extracting a Gaussian Mixture Model from the diffusion policy. Concurrently, QVPO (Ding et al., 2024) proposed weighting their diffusion loss with their respective Q-values after applying transformations. However, QVPO relies on sampling actions from a uniform distribution to enforce exploration.

DIME distinguishes from prior works in that we use the maximum entropy RL framework for training the diffusion policy, which was not considered before. This allows direct control of the exploration-exploitation trade-off arising naturally through this objective without the need for additional approximations. DIME is leveraging the diffusion model to generate non-Gaussian exploration actions which is in contrast to most other diffusion RL approaches that still require including Gaussian or uniform exploration noise.

Approximate Inference with Diffusion Models. Early works on approximate inference with diffusion models were formalized as a stochastic optimal control problem using Schrödinger-Föllmer diffusions (Dai Pra, 1991; Tzen and Raginsky, 2019; Huang et al., 2021) and only recently realized with deep-learning based approaches (Vargas et al., 2023; Zhang and Chen, 2021). Vargas et al. (2024); Berner et al. (2024) later extended these results to denoising diffusion models. A more general framework where both forward and backward processes of the diffusion model are learnable was concurrently proposed by Richter and Berner (2024); Nusken et al. (2024). Recently, many extensions have been proposed, see e.g. (Akhound-Sadegh et al., 2024; Noble et al., 2024; Geffner and Domke, 2023; Zhang et al., 2023; Chen et al., 2025; Blessing et al., 2025b,a). Our work can be seen as an instance of the sampler presented in Berner et al. (2024). However, our formulation allows using different diffusion samplers such as those presented in Richter and Berner (2024); Blessing et al. (2025a), while we restrict ourselves in this work to the sampler presented in Berner et al. (2024).

5.3. Preliminaries

5.3.1. Maximum Entropy Reinforcement Learning

Notation We consider the task of learning a policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$, where $\mathcal{S}$ and $\mathcal{A}$ denote a continuous state and action space, respectively using reinforcement learning (RL). We formalize the RL problem using an infinite horizon Markov decision process consisting of the tuple $(\mathcal{S}, \mathcal{A}, r, p, \rho_\pi)$, with bounded reward function $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ and transition density $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ which denotes the likelihood for transitioning into a state $\mathbf{s}' \in \mathcal{S}$ when being in $\mathbf{s} \in \mathcal{S}$ and executing an action $\mathbf{a} \in \mathcal{A}$. We follow Haarnoja et al. (2018b) and slightly overload ρ_π which denotes the state and state-action marginals induced by a policy π. Moreover, $\gamma \in [0, 1)$ denotes the discount factor. For brevity, we use $r_t \triangleq r(\mathbf{s}_t, \mathbf{a}_t)$. Lastly, we denote objective functions that we aim to maximize as J and minimize as $\mathcal{L}$.

Control as inference. The goal of maximum entropy reinforcement learning (MaxEnt-RL) is to jointly maximize the sum of expected rewards and entropies of a policy

$$J(\pi) = \sum_{t=l}^{\infty} \gamma^{t-l} \mathbb{E}_{\rho_\pi} \left[r_t + \alpha \mathcal{H}(\pi(\mathbf{a}_t|\mathbf{s}_t)) \right], \tag{5.1}$$

where $\mathcal{H}(\pi(\mathbf{a}|\mathbf{s})) = -\int \pi(\mathbf{a}|\mathbf{s}) \log \pi(\mathbf{a}|\mathbf{s}) \mathrm{d}\mathbf{a}$ is the differential entropy, and $\alpha \in \mathbb{R}^+$ controls the exploration exploitation trade-off (Haarnoja et al., 2017). To keep the notation uncluttered we absorb α into the reward function via $r \leftarrow r/\alpha$. Defining the Q-function of a policy π as

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = r_t + \sum_{l=1}^{\infty} \gamma^l \mathbb{E}_{\rho_\pi} \left[r_{t+l} + \mathcal{H}\left(\pi(\mathbf{a}_{t+l}|\mathbf{s}_{t+l})\right) \right], \tag{5.2}$$

with $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the MaxEnt objective can be cast as an approximate inference problem of the form

$$\mathcal{L}(\pi) = D_{\mathrm{KL}} \left(\pi(\mathbf{a}_t|\mathbf{s}_t) \middle| \frac{\exp Q^\pi(\mathbf{s}_t, \mathbf{a}_t)}{\mathcal{Z}^\pi(\mathbf{s}_t)} \right), \tag{5.3}$$

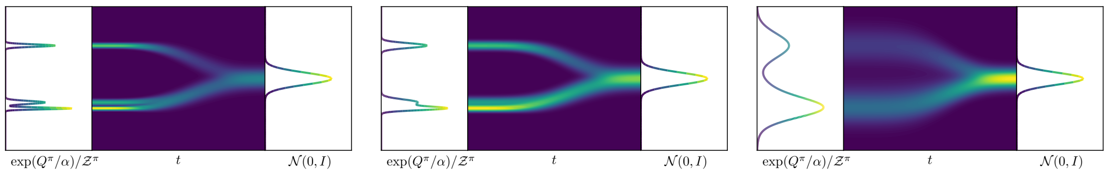
in a sense that $\max_\pi J(\pi) = \min_\pi \mathcal{L}(\pi)$. Here, D_{KL} denotes the Kullback-Leibler divergence and

$$\mathcal{Z}^\pi(\mathbf{s}) = \int \exp Q^\pi(\mathbf{s}, \mathbf{a}) \mathrm{d}\mathbf{a} \tag{5.4}$$

is the state-dependent normalization constant.

Policy iteration is a two-step iterative update scheme that is, under certain assumptions, guaranteed to converge to the optimal policy with respect to the maximum entropy objective. The two steps include policy evaluation and policy improvement. Given a policy π, policy evaluation aims to evaluate the value of π. To that end, Haarnoja et al. (2018b) showed that repeated application of the Bellman backup operator $\mathcal{T}^\pi Q^k$ with

$$\mathcal{T}^\pi Q(\mathbf{s}_t, \mathbf{a}_t) \triangleq r_t + \gamma \mathbb{E} \left[Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) + \mathcal{H}(\mathbf{a}_{t+1}|\mathbf{s}_{t+1}) \right], \tag{5.5}$$



(a) $\alpha < 1$ **(b)** $\alpha = 1$ **(c)** $\alpha > 1$

Figure 5.1.: The effect of the reward scaling parameter α. The figures in (a)-(b) show diffusion processes for different α values starting at a prior distribution $\mathcal{N}(0, I)$ and going backward in time to approximate the target distribution $\exp\left(Q^\pi/\alpha\right)/Z^\pi$. Small values for α (a) lead to concentrated target distributions with less noise in the diffusion trajectories especially at the last time steps. The higher α becomes (b) and (c), the more the target distribution is smoothed and the distribution of the samples at the last time steps becomes more noisy. Therefore, the parameter α directly controls the exploration by enforcing noisier samples the higher α becomes.

converges to Q^π as $k \longrightarrow \infty$, starting from any Q. To update the policy, that is, to perform the policy improvement step, the Q-function of the previous evaluation step, $Q^{\pi_{\text{old}}}$ is used to obtain a new policy according to

$$\pi_{\text{new}} = \arg\min_{\pi\in\Pi} D_{\text{KL}}\left(\pi(\mathbf{a}_t|\mathbf{s}_t)\middle|\frac{\exp Q^{\pi_{\text{old}}}(\mathbf{s}_t,\mathbf{a}_t)}{\mathcal{Z}^{\pi_{\text{old}}}(\mathbf{s}_t)}\right), \tag{5.6}$$

where Π is a set of policies such as a family of parameterized distributions. Note that $\mathcal{Z}^{\pi_{\text{old}}}(\mathbf{s}_t)$ is not required for optimization as it is independent of π. Haarnoja et al. (2018b) showed that for all state-action pairs $(\mathbf{s},\mathbf{a}) \in \mathcal{S}\times\mathcal{A}$ it holds that $Q^{\pi_{\text{new}}}(\mathbf{s},\mathbf{a}) \geq Q^{\pi_{\text{old}}}(\mathbf{s},\mathbf{a})$ ensuring that policy iteration converges to the optimal policy π^* in the limit of infinite repetitions of policy evaluation and improvement.

5.3.2. Denoising Diffusion Policies

For a given state $\mathbf{s} \in \mathcal{S}$, we consider a stochastic process on the time-interval $[0, T]$ given by an Ornstein-Uhlenbeck (OU) process [1] (Särkkä and Solin, 2019)

$$\mathrm{d}\mathbf{a}_t = -\beta_t\mathbf{a}_t\mathrm{d}t + \eta\sqrt{2\beta_t}\mathrm{d}\mathbf{B}_t, \quad a_0 \sim \vec{\pi}_0(\cdot|\mathbf{s}), \tag{5.7}$$

with diffusion coefficient $\beta : [0,T] \longrightarrow \mathbb{R}^+$, standard Brownian motion $(\mathbf{B}_t)_{t\in[0,T]}$, and some target policy $\vec{\pi}_0$. For $t, l \in [0,T]$, we denote the marginal density of Eq. 5.7 at t as $\vec{\pi}_t$ and the conditional density at time t given l as $\vec{\pi}_{t|l}$. Eq. 5.7 is commonly referred to as *forward* or *noising process* since, for a suitable choice of β, it holds that $\vec{\pi}_T \approx \mathcal{N}(0, \eta^2 I)$. Denoising diffusion models leverage the fact, that the time-reversed process of Eq. 5.7 is given by

$$\mathrm{d}\mathbf{a}_t = \left(-\beta_t\mathbf{a}_t\mathrm{d}t - 2\eta^2\beta_t\nabla\log\vec{\pi}_t(\mathbf{a}_t|\mathbf{s})\right) + \eta\sqrt{2\beta_t}\mathrm{d}\mathbf{B}_t, \tag{5.8}$$

[1] Please note, for clarity, we slightly abuse notation by using t to denote the time in the stochastic process. This should not be confused with the time step in RL. The distinction becomes clear when we discretize the processes.

starting from $\overleftarrow{\pi}_T = \vec{\pi}_T \approx \mathcal{N}(0, \eta^2 I)$ and running backwards in time (Nelson, 2020; Anderson, 1982; Haussmann and Pardoux, 1986). For the *backward*, *generative* or *denoising process* (Eq. 5.8), we denote the density as $\overleftarrow{\pi}$. Here, time-reversal means that the marginal densities align, i.e., $\vec{\pi}_t = \overleftarrow{\pi}_t$ for all $t \in [0, T]$. Hence, starting from $\mathbf{a}_T \sim \mathcal{N}(0, \eta^2 I)$, one can sample from the target policy $\vec{\pi}_0$ by simulating Eq. 5.8. However, for most densities $\vec{\pi}_0$, the scores $(\nabla \log \vec{\pi}_t(\mathbf{a}_t|\mathbf{s}))_{t\in[0,T]}$ are intractable, requiring numerical approximations. To address this, denoising score-matching objectives are commonly employed, that is,

$$\mathcal{L}_{\mathrm{SM}}(\theta) = \mathbb{E}\left[\beta_t \| f_t^\theta(\mathbf{a}_t, \mathbf{s}) - \nabla \log \vec{\pi}_{t|0}(\mathbf{a}_t|\mathbf{a}_0, \mathbf{s}) \|^2\right], \tag{5.9}$$

where t is sampled on $[0, T]$ and f^θ denotes a parameterized score network (Hyvärinen and Dayan, 2005; Vincent, 2011). For OU processes, the conditional densities $\nabla \log \vec{\pi}_{t|0}$ are explicitly computable, making the objective tractable for optimizing θ (Song et al., 2021). Once trained, the score network f^θ can be used to simulate the denoising process

$$\mathrm{d}\mathbf{a}_t = \left(-\beta_t \mathbf{a}_t \mathrm{d}t - 2\eta^2 \beta_t f_t^\theta(\mathbf{a}_t, \mathbf{s})\right) + \eta\sqrt{2\beta_t}\mathrm{d}\mathbf{B}_t \tag{5.10}$$

to obtain samples $\mathbf{a}_0 \sim \pi_0^\theta$ that are approximately distributed according to $\vec{\pi}_0$. Here, π_t^θ denotes the marginal distribution of Eq. 5.10 at time t. While score-matching techniques work well in practice, they cannot be applied to maximum entropy reinforcement learning. This is because the expectation in Eq. 5.9 requires samples $\mathbf{a}_0 \sim \vec{\pi}_0 \propto \exp Q^\pi$ which are not available. However, in the next section, we build on recent advances in approximate inference to optimize diffusion models without requiring samples from $\mathbf{a}_0$, relying instead on evaluations of Q^π.

5.4. Diffusion-Based Maximum Entropy RL

Here, we explain how diffusion models can be used within a maximum entropy RL framework. To that end, we express the maximum entropy objective as an approximate inference problem for diffusion models. We then use these results to introduce a policy iteration scheme that provably converges to the optimal policy. Lastly, we propose a practical algorithm for optimizing diffusion models.

5.4.1. Control as Inference for Diffusion Policies

Directly maximizing the maximum entropy objective

$$J(\overleftarrow{\pi}) = \sum_{t=l}^{\infty} \gamma^{t-l} \mathbb{E}_{\rho_\pi}\left[r_t(\mathbf{s}_t, \mathbf{a}_t^0) + \alpha \mathcal{H}(\overleftarrow{\pi}_0(\mathbf{a}_t^0|\mathbf{s}_t))\right]$$

for a diffusion model is difficult as the marginal entropy $\mathcal{H}(\overleftarrow{\pi}_0(\mathbf{a}|\mathbf{s}))$ of the denoising process in Eq. 5.8 is intractable. Please note that we use superscripts for the actions to indicate the diffusion step to avoid collisions with the time step used in RL. Moreover, we will again absorb α into the reward and use $r_t \triangleq r(\mathbf{s}_t, \mathbf{a}_t^0)$. To overcome this intractability, we propose to maximize a lower bound. We start by discretizing the stochastic processes introduced in Section 5.3.2 and use the results as a foundation to derive this lower bound. Note that while similar results can be derived from a continuous-time perspective (see e.g., Berner et al. (2024); Richter and Berner (2024); Nusken et al. (2024)), such derivation would require a background in stochastic calculus, making it less accessible to a broader audience.

The Euler-Maruyama (EM) discretization (Särkkä and Solin, 2019) of the noising (Eq. 5.7) and denoising (Eq. 5.8) process is given by

$$\mathbf{a}^{n+1} = \mathbf{a}^n - \beta_n \mathbf{a}^n \delta + \epsilon_n \quad \text{and} \tag{5.11}$$

$$\mathbf{a}^{n-1} = \mathbf{a}^n + \left(\beta_n \mathbf{a}^n + 2\eta^2 \beta_n \nabla \log \vec{\pi}_n(\mathbf{a}^n|\mathbf{s})\right) \delta + \xi_n \tag{5.12}$$

respectively, with $\epsilon_n, \xi_n \sim \mathcal{N}(0, 2\eta^2\beta_n\delta I)$. Here, δ denotes a constant discretization step size such that $N = T/\delta$ is an integer. To simplify notation, we write $\mathbf{a}^n$, instead of $\mathbf{a}^{n\delta}$. Under the EM discretization, the noising and denoising process admit the following joint distributions

$$\vec{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s}) = \vec{\pi}_0(\mathbf{a}^0|s) \prod_{n=0}^{N-1} \vec{\pi}_{n+1|n}(\mathbf{a}^{n+1}|\mathbf{a}^n, \mathbf{s}), \tag{5.13}$$

$$\overleftarrow{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s}) = \overleftarrow{\pi}_N(\mathbf{a}^N|s) \prod_{n=1}^{N} \overleftarrow{\pi}_{n-1|n}(\mathbf{a}^{n-1}|\mathbf{a}^n, \mathbf{s}), \tag{5.14}$$

in a sense that $\vec{\pi}_{0:N}$ and $\overleftarrow{\pi}_{0:N}$ converge to the law of $(\mathbf{a}_t)_{t\in[0,T]}$ in Eq. 5.7 and 5.8, as $\delta \longrightarrow 0$, respectively (Doucet et al., 2022). Here, $\vec{\pi}_{n+1|n}$ and $\overleftarrow{\pi}_{n-1|n}$ are Gaussian transition densities that directly follow from Eq. 5.11 and 5.12.

To obtain a maximum entropy objective for diffusion models, we make use of the following lower bound on the marginal entropy, that is, $\mathcal{H}(\overleftarrow{\pi}_0(\mathbf{a}_0|\mathbf{s})) \geq \ell_{\overleftarrow{\pi}}(\mathbf{a}^0, \mathbf{s})$, where

$$\ell_{\overleftarrow{\pi}}(\mathbf{a}^0, \mathbf{s}) = \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\vec{\pi}_{1:N|0}(\mathbf{a}^{1:N}|\mathbf{a}^0, \mathbf{s})}{\overleftarrow{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s})}\right]. \tag{5.15}$$

Please note that similar bounds have been used, e.g., in Agakov and Barber (2004); Tran et al. (2015); Ranganath et al. (2016); Maaløe et al. (2016); Arenz et al. (2018), or, more generally, follow from the data processing inequality (Cover, 1999). A derivation can be found in Appendix C.1. From Eq. 5.15, it directly follows that

$$J(\overleftarrow{\pi}) \geq \bar{J}(\overleftarrow{\pi}) = \sum_{t=l}^{\infty} \gamma^{t-l} \mathbb{E}_{\rho_\pi}\left[r_t + \ell_{\overleftarrow{\pi}}(\mathbf{a}_t^0, \mathbf{s}_t)\right]. \tag{5.16}$$

Next, we cast Eq. 5.16 as an approximate inference problem to make the objective more interpretable. To that end, let us define the Q-function of a denoising policy $\overleftarrow{\pi}$ with respect to the maximum entropy objective $\bar{J}$ as

$$Q^{\overleftarrow{\pi}}(\mathbf{s}_t, \mathbf{a}_t^0) = r_t + \sum_{l=1} \gamma^l \mathbb{E}_{\rho_\pi}\left[r_{t+l} + \ell_{\overleftarrow{\pi}}(\mathbf{a}_{t+l}^0, \mathbf{s}_{t+l})\right], \tag{5.17}$$

with $Q^{\overleftarrow{\pi}} : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$. With Eq. 5.17 we identify the corresponding approximate inference problem as finding $\overleftarrow{\pi}$ which minimizes (please see Appendix C.1 for derivation)

$$\bar{\mathcal{L}}(\overleftarrow{\pi}) = D_{\mathrm{KL}}\left(\overleftarrow{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s})|\overrightarrow{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s})\right), \tag{5.18}$$

where the target policy, i.e., the marginal of the noising process in Eq. 5.13 is given by the exponentiated Q-function of the diffusion policy

$$\overrightarrow{\pi}_0(\mathbf{a}^0|\mathbf{s}) = \frac{\exp Q^{\overleftarrow{\pi}}(\mathbf{s},\mathbf{a}^0)}{\mathcal{Z}^{\overleftarrow{\pi}}(\mathbf{s})}. \tag{5.19}$$

Recall from Section 5.3.2 that we aim to time-reverse the noising process, that is, to ensure for all states $\mathbf{s} \in \mathcal{S}$, it holds that $\overleftarrow{\pi}_{0:N} = \overrightarrow{\pi}_{0:N}$. Please note that this is precisely what Eq. 5.18 is trying to accomplish, i.e., we aim to learn a diffusion model $\overleftarrow{\pi}$, such that the denoising process time-reverses the noising process, and, in particular, has a marginal distribution given by $\pi_0 = \exp Q^{\overleftarrow{\pi}}/\mathcal{Z}^{\overleftarrow{\pi}}$. Lastly, from the data processing inequality, it directly follows that

$$D_{\mathrm{KL}}\left(\overleftarrow{\pi}_0(\mathbf{a}^0|\mathbf{s})\middle|\frac{\exp Q^{\overleftarrow{\pi}}(\mathbf{s},\mathbf{a}^0)}{\mathcal{Z}^{\overleftarrow{\pi}}(\mathbf{s})}\right) \leq D_{\mathrm{KL}}\left(\overleftarrow{\pi}(\mathbf{a}^{0:N}|\mathbf{s})|\overrightarrow{\pi}(\mathbf{a}^{0:N}|\mathbf{s})\right), \tag{5.20}$$

which shows the approximate inference problem in Eq. 5.18 indeed optimizes the same inference problem stated in Eq. 5.3 (also see Section 2.7.1). Next, we will use these results to develop a policy iteration scheme for diffusion models.

5.4.2. Diffusion-based Policy Iteration

We propose a policy iteration scheme for learning an optimal maximum entropy policy, similar to Haarnoja et al. (2018b). However, here we restrict the family of stochastic actors to diffusion policies $\overleftarrow{\pi} \in \overleftarrow{\Pi} \subset \Pi$. Throughout this section, we assume finite action spaces to enable theoretical analysis, but relax this assumption in Section 5.4.3. All proofs of this section are deferred to Appendix C.1.

For policy evaluation, we aim to compute the value of a policy $\overleftarrow{\pi}$. We define the Bellman backup operator as

$$\mathcal{T}^{\overleftarrow{\pi}}Q(\mathbf{s}_t,\mathbf{a}_t^0) \triangleq r_t + \gamma\mathbb{E}\left[Q(\mathbf{s}_{t+1},\mathbf{a}_{t+1}^0) + \ell_{\overleftarrow{\pi}}(\mathbf{a}_{t+1}^0,\mathbf{s}_{t+1})\right]. \tag{5.21}$$

Note that Eq. 5.21 contains the entropy-lower bound $\ell_{\overleftarrow{\pi}}$. By applying standard convergence results for policy evaluation (Sutton and Barto, 2018) we can obtain the value of a policy by repeatedly applying $\mathcal{T}^{\overleftarrow{\pi}}$ as established in Proposition 5.4.1.

Proposition 5.4.1 (Policy Evaluation). *Let $\mathcal{T}^{\overleftarrow{\pi}}$ be the Bellman backup operator for a diffusion policy $\overleftarrow{\pi}$ as defined in Eq. 5.21. Further, let $Q^0 : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ and $Q^{k+1} = \mathcal{T}^{\overleftarrow{\pi}}Q^k$. Then, it holds that* $\lim_{k\to\infty} Q^k = Q^{\overleftarrow{\pi}}$ *where $Q^{\overleftarrow{\pi}}$ is the Q value of $\overleftarrow{\pi}$.*

For the policy improvement step, we seek to improve the current policy based on its value using the Q-function. Formally, we need to solve the approximate inference problem

$$\overleftarrow{\pi}^{\text{new}} = \arg\min_{\overleftarrow{\pi}\in\overleftarrow{\Pi}} D_{\mathrm{KL}}\left(\overleftarrow{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s})|\vec{\pi}^{\,\text{old}}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s})\right) \tag{5.22}$$

for all $\mathbf{s} \in \mathcal{S}$, where $\vec{\pi}^{\,\text{old}}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s})$ is as in Eq. 5.13 with marginal density

$$\vec{\pi}^{\,\text{old}}_0(\mathbf{a}^0|\mathbf{s}) = \frac{\exp Q^{\overleftarrow{\pi}_{\text{old}}}(\mathbf{s},\mathbf{a}^0)}{\mathcal{Z}^{\overleftarrow{\pi}_{\text{old}}}(\mathbf{s})}. \tag{5.23}$$

Indeed, solving Eq. 5.22 results in a policy with higher value as established below.

Proposition 5.4.2 (Policy Improvement). *Let $\overleftarrow{\pi}_{old}, \overleftarrow{\pi}_{new} \in \overleftarrow{\Pi}$ be defined as in Eq. 5.23 and 5.22, respectively. Then for all $(\mathbf{s},\mathbf{a}) \in \mathcal{S}\times\mathcal{A}$ it holds that $Q^{\overleftarrow{\pi}_{new}}(\mathbf{s},\mathbf{a}) \geq Q^{\overleftarrow{\pi}_{old}}(\mathbf{s},\mathbf{a})$.*

Combining these results leads to the policy iteration method which alternates between policy evaluation (Proposition 5.4.1) and policy improvement (Proposition 5.4.2) and provably converges to the optimal policy in $\overleftarrow{\Pi}$ (Proposition 5.4.3).

Proposition 5.4.3 (Policy Iteration). *Let $\overleftarrow{\pi}^0, \overleftarrow{\pi}^{i+1}, \overleftarrow{\pi}^i, \overleftarrow{\pi}_* \in \overleftarrow{\Pi}$. Further, let $\overleftarrow{\pi}^{i+1}$ be the policy obtained from $\overleftarrow{\pi}^i$ after a policy evaluation and improvement step. Then, for any starting policy $\overleftarrow{\pi}^0$ it holds that $\lim_{i\to\infty}\overleftarrow{\pi}^i = \overleftarrow{\pi}^*$, with $\overleftarrow{\pi}^*$ such that for all $\overleftarrow{\pi} \in \overleftarrow{\Pi}$ and $(\mathbf{s},\mathbf{a}) \in \mathcal{S}\times\mathcal{A}$ it holds that $Q^{\overleftarrow{\pi}^*}(\mathbf{s},\mathbf{a}) \geq Q^{\overleftarrow{\pi}}(\mathbf{s},\mathbf{a})$.*

However, performing policy iteration until convergence is in practice often intractable, particularly for continuous control tasks. As such, we will introduce a practical algorithm next.

5.4.3. DIME: A Practical Diffusion RL Algorithm

To obtain a practical algorithm, we use a parameterized function approximation for the Q-function and the policy, that is, $Q_{\boldsymbol{\phi}}$ and $\pi^{\boldsymbol{\theta}}$, with parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, respectively. Here, $\pi^{\boldsymbol{\theta}}$ is represented by a parameterized score network, see Eq. 5.10. To perform approximate policy evaluation, we can minimize the Bellman residual,

$$J_Q(\boldsymbol{\phi}) = \frac{1}{2}\mathbb{E}\left[\left(Q_{\boldsymbol{\phi}}(\mathbf{s}_t,\mathbf{a}_t^0) - Q_{\text{target}}(\mathbf{s}_t,\mathbf{a}_t^0)\right)^2\right], \tag{5.24}$$

using stochastic gradients with respect to $\boldsymbol{\phi}$. We provide implementation details in Section 5.4.4. Moreover, the expectation is computed using state-action pairs collected from environment interactions and saved in a replay buffer. For policy improvement, we solve the approximate inference problem

$$\mathcal{L}(\boldsymbol{\theta}) = D_{\mathrm{KL}}\left(\pi^{\boldsymbol{\theta}}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s})|\vec{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s})\right), \tag{5.25}$$

where the target policy, i.e., the marginal of the noising process in Eq. 5.13 is given by the approximate Q-function

$$\vec{\pi}_0(\mathbf{a}^0|\mathbf{s}) = \frac{\exp Q_\phi(\mathbf{s}, \mathbf{a}^0)}{\mathcal{Z}_\phi(\mathbf{s})}, \tag{5.26}$$

where states are again sampled from a replay buffer. Further expanding $\mathcal{L}(\boldsymbol{\theta})$ yields

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\pi^\theta}\left[\log \pi_N^\theta(\mathbf{a}^N|\mathbf{s}) - Q_\phi(\mathbf{s}, \mathbf{a}^0) + \sum_{n=1}^{N} \log \frac{\pi^\theta_{n-1|n}(\mathbf{a}^{n-1}|\mathbf{a}^n, \mathbf{s})}{\vec{\pi}_{n|n-1}(\mathbf{a}^n|\mathbf{a}^{n-1}, \mathbf{s})}\right] + \log \mathcal{Z}_\phi(s), \tag{5.27}$$

showing that $\mathcal{Z}_\phi$ is not needed to minimize Eq. 5.27 as it is independent of $\boldsymbol{\theta}$. Moreover, contrary to the score-matching objective (see Eq. 5.9) that is commonly used to optimize diffusion models, stochastic optimization of $\mathcal{L}(\boldsymbol{\theta})$ does not need access to samples $\mathbf{a}_0 \sim \exp Q_\phi/\mathcal{Z}_\phi$, instead relying on stochastic gradients obtained via reparameterization trick (Kingma, 2013) using samples from the diffusion model π^θ.

5.4.4. Implementation Details

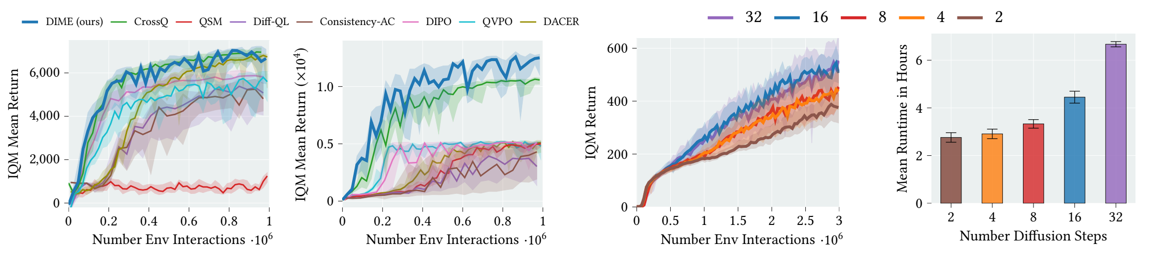
Autotuning Temperature. We follow implementations like SAC (Haarnoja et al., 2018c) where the reward scaling parameter α (also see Fig. 5.1) is not absorbed into the reward but scales the entropy term. Choosing α depends on the reward ranges and the dimensionality of the action space, which requires tuning it per environment. We instead follow prior works (Haarnoja et al., 2018c) for auto-tuning α by optimizing

$$J(\alpha) = \alpha\left(\mathcal{H}_{\text{target}} - \ell^\theta_{\mathcal{H}}\right), \tag{5.28}$$

where $\mathcal{H}_{\text{target}}$ is a target value for the mismatch between the noising and denoising processes measured by the log ratio.

Autotuning Diffusion Coefficient. Please note that the objective function in Eq. 5.27 is fully differentiable with respect to parameters of the diffusion process. As such, we additionally treat the diffusion coefficient β as a learnable parameter that is optimized end-to-end, further reducing the need for manual hyperparameter tuning. Further details on the parameterization can be found in Appendices C.4 and C.6.

***Q*-function.** Following Bhatt et al. (2024) we adopt the CrossQ algorithm, i.e., we use Batch Renormalization in the Q-function and avoid a target network for calculating Q_{target}. When updating the Q-function, the values for the current and next state-action pairs are queried in parallel. The next Q-values are used as target values where the gradients are stopped. Additionally, we employ distributional Q learning as proposed by Bellemare et al. (2017). The details are described in Appendix C.4.



(a) Ant-v3 **(b)** Humanoid-v3 **(c)** Varying the Diffusion Steps **(d)** Runtime for 1M Steps

Figure 5.2.: Learning Curves on Gym Benchmark Suite (a)-(b). We compare DIME against various diffusion baselines and CrossQ on the (a) *Ant-v3* and (b) *Humanoid-v3* from the Gym suite. While all diffusion-based methods are outperformed by DIME, DIME performs on par with CrossQ on the Ant environment. DIME performs favorably on the high-dimensional *Humanoid-v3* environment, where it also outperforms CrossQ. **Varying the Number of diffusion steps (c)-(d).** The number of diffusion steps might affect the performance and the computation time. (d) shows DIME's learning curves for varying diffusion steps. *Two* diffusion steps perform badly, whereas *four* and *eight* diffusion steps perform similar but still worse than *16* and *32* diffusion steps which perform similarly. (c) shows the computation time for 1MIO steps of the corresponding learning curves. The smaller the diffusion steps, the less computation time is required.

5.5. Experiments

In a broad range of 13 sophisticated learning environments from different benchmark suits, ranging from mujoco gym (Brockman et al., 2016), deepmind control suit (DMC) (Tunyasuvunakool et al., 2020), and myo suite (Caggiano et al., 2022), we compare DIME's performance against state-of-the-art RL baselines that employ diffusion and Gaussian policy parameterizations. The considered environments are challenging locomotion and manipulation learning tasks with up to 39-dimensional action and 223-dimensional observation spaces.

We consider **QSM** (Psenka et al., 2024), **Diffusion-QL** (Wang et al., 2023), **Consistency-AC** (Ding and Jin, 2024), **DIPO** (Yang et al., 2023), **QVPO** (Ding et al., 2024), and **DACER** (Wang et al., 2024) as baselines for diffusion-based policy representations.

Additionally, we compare against the state-of-the-art RL methods **CrossQ** (Bhatt et al., 2024) and **BRO** (Nauman et al., 2024), where we have used the provided learning curves from the latter. Both methods use a Gaussian parameterized policy and have shown remarkable results.

Finally, we analyze DIME's algorithmic features with an intensive ablation study where we clarify the role of the reward scaling parameter α, the effect of varying diffusion steps, the gained performance boost when using a diffusion policy representation over a Gaussian representation, and the effect of employing distributional Q learning.

We have run the learning curves for 10 seeds using the official code releases and report the *interquartile mean (IQM)* with a 95% stratified bootstrap confidence interval as suggested by Agarwal et al. (2021).

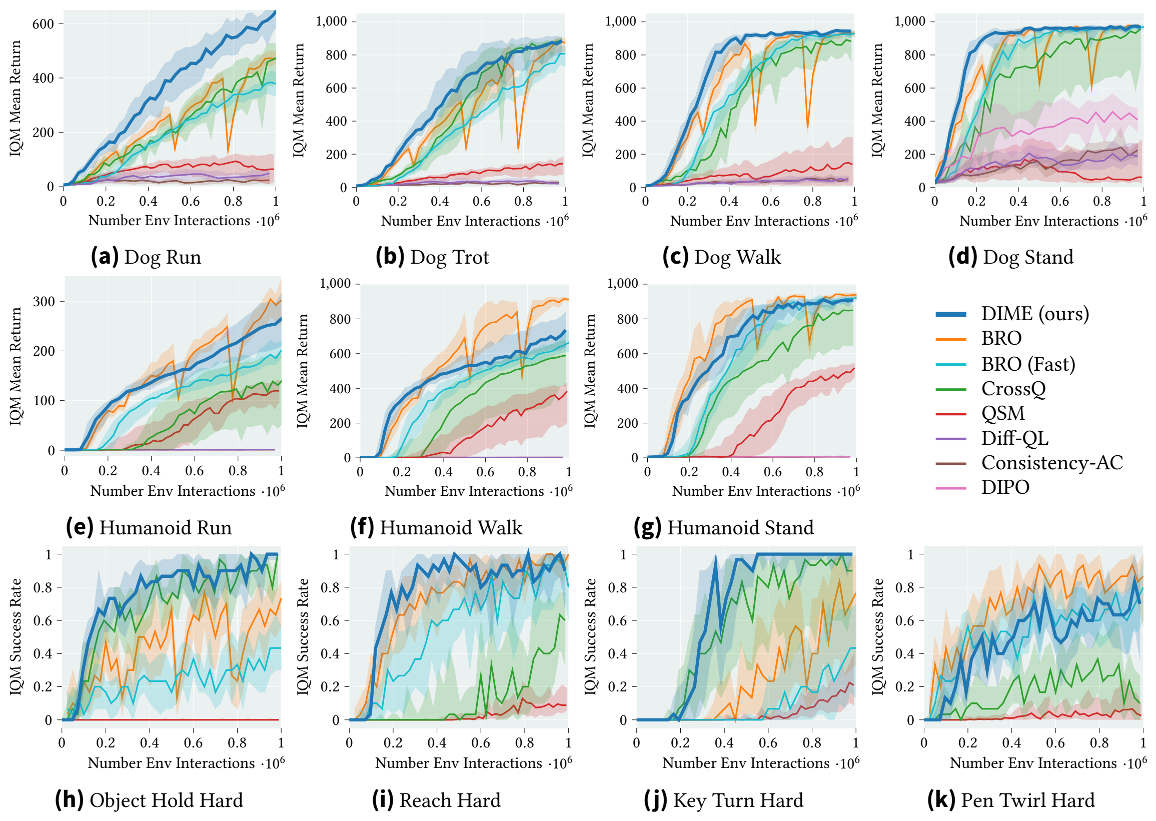


Figure 5.3.: Training curves on DMC's dog, humanoid tasks, and the hand environments from the MYO Suite. DIME performs favorably on the high-dimensional dog tasks, where it significantly outperforms all baselines (dog-run) or converges faster to the final performance. On the humanoid tasks, DIME outperforms all diffusion-based baselines, CrossQ and BRO Fast, and performs on par with BRO on the humanoid-stand task and slightly worse on the humanoid-run and humanoid-walk tasks. In the MYO SUITE environments, DIME performs consistently on all tasks, either outperforming the baselines or performing on par.

5.5.1. Performance Comparisons

We consider environments with high-dimensional observation and action spaces from three benchmark suits for a robust performance assessment (please see Appendix C.3).

Gym Environments. Fig 5.2a and Fig. 5.2b show the learning curves for the *An-tv3* and *Humanoid-v3* tasks respectively. While the diffusion-based baselines perform reasonably well on the *Ant-v3* task with DIPO outperforming the rest, they are all outperformed by DIME and CrossQ which perform comparably. On the *Humanoid-v3* DIME achieves a significantly higher return than all baselines.

DMC: Dog and Humanoid Tasks (Fig. 5.3). We benchmark on DMC suit's challenging *dog* and *humanoid* environments, where we additionally consider BRO and BRO Fast as a Gaussian-based policy baseline. BRO Fast is identical to BRO but differs only in the update-to-data (UTD) ratio of two as DIME and CrossQ. Please note that we used the online available learning curves provided by the official implementation for BRO. DIME outperforms all baselines significantly on the *dog-run* environment and converges faster to

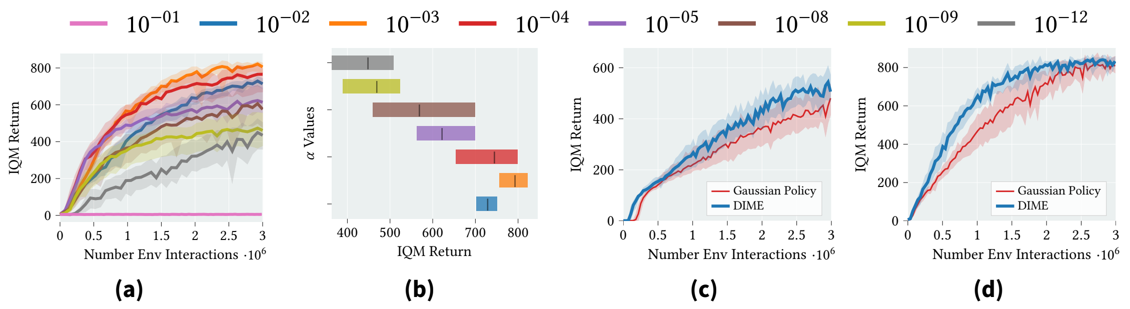


Figure 5.4.: Reward Scaling Sensitivity (a)-(b). The α parameter controls the exploration-exploitation trade-off. (a) shows the learning curves for varying values on DMC's dog-run task. Too high α values ($\alpha = 0.1$) do not incentivize learning whereas too small α values ($\alpha \leq 10^{-5}$) converge to suboptimal behavior. (b) shows the aggregated end performance for each learning curve in (a). For increasing α values, the end performance increases until it reaches an optimum at $\alpha = 10^{-3}$ after which the performance starts dropping. **Diffusion Policy Benefit (c) and (d).** We compare DIME to a Gaussian policy with the same implementation details as DIME on the (a) humanoid-run and (b) dog-run tasks. The diffusion-based policy reaches a higher return (a) and converges faster.

the same end performance on the remaining dog environments (see Fig. 5.3a - 5.3d). BRO has slightly higher average performance on the *humanoid-run* and *humanoid-walk* (see Fig. 5.3f - 5.3e)) tasks indicating that DIME performs favorably on more high-dimensional tasks like the dog environments and tasks from the myo suite. However, DIME's asymptotic behavior in the *humanoid-run* achieves slightly higher aggregated performance than BRO, where we have run both algorithms for 3M steps (Fig. 5.8c). However, BRO requires full parameter resets leading to performance drops during training and it is run with a UTD ratio of 10 which is 5 times higher than DIME. This leads to longer training times. As reported in their paper (Nauman et al., 2024), BRO needs an average training time of 8.5h, whereas DIME trains in approximately 4.5h with 16 diffusion steps on the humanoid-run with the same hardware (*Nvidia A100*).

MYO Suite (Fig. 5.3). Except for *pen twirl hard* (Fig. 5.3k), DIME consistently outperforms BRO and BRO Fast in that it converges to a higher or faster end success rate. DIME also consistently outperforms CrossQ in terms of the achieved success rates on all the tasks except for the object hold hard task 5.3h, where DIME converges faster.

5.5.2. Ablation Studies

Exploration Control. The parameter α balances the exploration-exploitation trade-off by scaling the reward signal. We analyze the effect of this parameter by comparing DIME's learning curves with different α values on the dog-run task from the DMC (see Fig. 5.4a). Additionally, we show the performance of the last return measurements for each learning curve in Fig. 5.4b. Too high α values ($\alpha = 0.1$) do not incentivize maximizing the task's return, leading to no learning at all, whereas small values ($\alpha \leq 10^{-5}$) lead to suboptimal performance because the policy does not explore sufficiently. We can also see a clear trend that starting from $\alpha = 10^{-12}$, the performance gradually increases until the best performance is reached for $\alpha = 10^{-3}$.

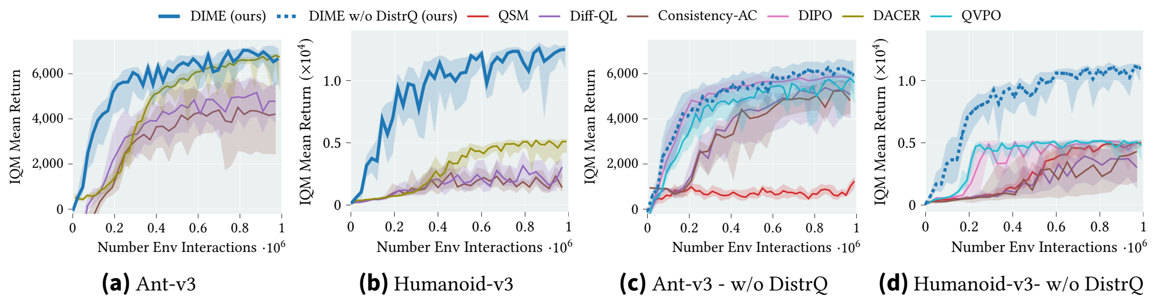


Figure 5.5.: Comparison to Diffusion Baselines with (a)-(b) and without Distributional Q (c)-(d) on the Ant-v3 and Humanoid-v3 tasks. We provide the learning curves for distributional versions for Diff-QL and Consistency-AC alongside DACER, which employs distributional Q by default on the Ant-v3 (a) and Humanoid-v3 (b) tasks. DIME converges faster on the Ant-v3 (a) task to the same performance achieved by DACER and outperforms all baselines on the more high-dimensional Humanoid-v3 (b) task. Additionally, we compare DIME without distributional Q against the diffusion baselines without distributional Q on the Ant-v3 (c) and Humanoid-v3 (d) tasks. DIME without distributional Q performs on par with the baselines DIPO and QVPO on the Ant-v3 (c) and outperforms all baselines on the Humanoid-v3 (d).

Diffusion Policy Benefit. We aim to analyze the performance benefits of the diffusion-parameterized policy compared to a Gaussian parameterization in the same setup by only exchanging the policy and the corresponding policy update. This comparison ensures that the Gaussian policy is trained with the identical implementation details from DIME as described in Sec. 5.4.4 and showcases the performance benefits of a diffusion-based policy. Fig. 5.4c and 5.4d show the learning curves of both versions on DMC's humanoid-run and dog-run environments. The diffusion policy's expressivity leads to a higher aggregated return in the humanoid-run and to significantly faster convergence in the high-dimensional dog-run task. We attribute this performance benefit to an improved exploration behavior.

Number of Diffusion Steps. The number of diffusion steps determines how accurately the stochastic differential equations are simulated and is a hyperparameter that affects the performance. Usually, the higher the number of diffusion steps the better the model performs at the burden of higher computational costs. In Fig. 5.2c we plot DIME's performance for varying diffusion steps on DMC's humanoid-run environment and report the corresponding runtimes for 1 Mio environment steps in Fig. 5.2d on an *Nvidia A100* GPU machine. With an increasing number of diffusion steps, the performance and runtime increases. However, from 16 diffusion steps on, the performance stays the same.

Analysis on Distributional Q Learning. DIME employs distributional Q Learning (Bellemare et al., 2017) to represent the Q-function as a distribution over bins. We compare DIME to baselines when using distributional Q Learning and when using the well-known Bellman residual (see Eq. 5.24) for updating the parameters of the Q-function.

We start by comparing DIME with distributional Q learning against diffusion-based baselines that employ distributional Q learning. Fig. 5.5a and Fig. 5.5b show the learning curves on the *Ant-v3* and *Humanoid-v3*, respectively, where we compare against DACER, a distributional Q variant of Diff-QL, and Consistency-AC. DIME converges faster to the same performance as DACER on the *Ant-v3* task and outperforms the baselines on the

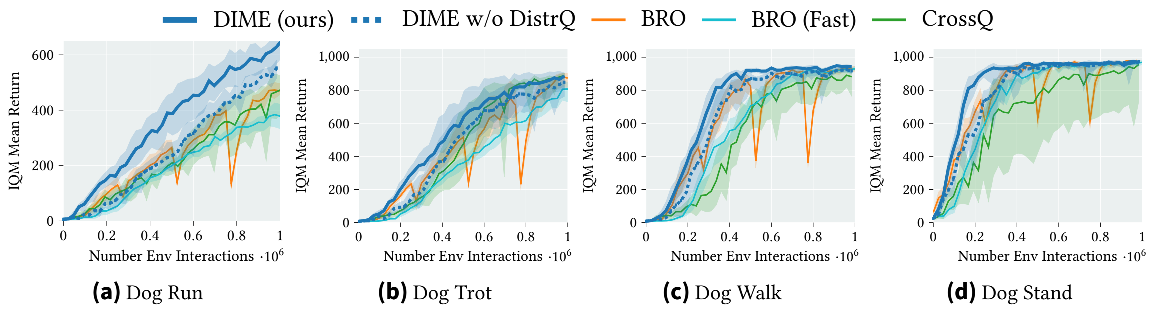


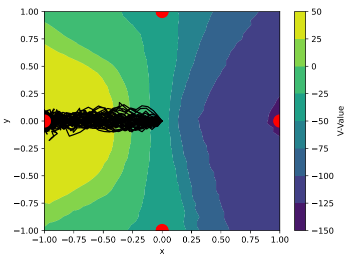
Figure 5.6.: Ablation on Distributional Q. Comparison of DIME and DIME without employing distributional Q (dashed line). While there is a small improvement when using distributional Q, DIME w/o Distributional Q still performs on par, or better than BRO, which employs quantile distributional RL. DIME w/o DistrQ outperforms CrossQ and BRO (Fast).

Humanoid-v3 task. In the setting without distributional Q Learning, i.e., when updating the parameters using the residual Bellman function, DIME performs similarly to DIPO and QVPO on the *Ant-v3* task and outperforms all baselines on the higher-dimensional *Humanoid-v3* task (Fig. 5.5c and Fig. 5.5d).

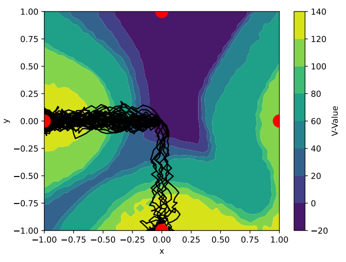
Additionally, we compare DIME with and without distributional Q Learning on the four dog environments from the DMC suite (Fig. 5.6), where we concentrate on the strong baselines BRO (Nauman et al., 2024) and CrossQ (Bhatt et al., 2024). BRO employs quantile distributional Q learning, whereas CrossQ uses the Bellman residual loss function for updating the Q-function's parameters. We have already observed that DIME with distributional Q performs favorably over the baselines. Fig. 5.6 shows a small improvement when using distributional Q. However, DIME without distributional Q (dashed line) still performs on par, or better than BRO and consistently performs better than BRO (Fast) and CrossQ. Please note that BRO and BRO (Fast) employ quantile distributional RL (Nauman et al., 2024).

5.5.3. Multimodality Analysis

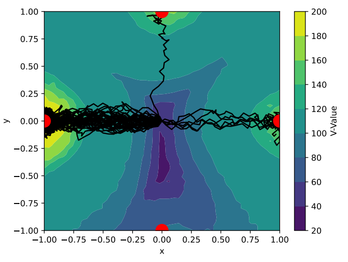
A key feature of diffusion models is their ability to model multimodal distributions. In the context of reinforcement learning, this means that the policy is able to learn multiple modes in the action space that can lead to multimodal behavior, i.e. versatile skills. So far we have analyzed DIME's performance on various benchmark tasks and its algorithmic features, but not its ability to learn multimodal behaviors. In general, analyzing wether an algorithm is able to acquire multimodal behavior is not straightforward, especially not in the off-policy setting. Here, the Q-function plays a crucial role because the policy is updated based on the Q-values. If not multiple modes are present in the Q-function, it is impossible for the policy to learn multimodal behavior. However, since the Q-function is learned based on the data collected by the policy, one can argue that the policy plays an important role for generating a good data distribution for optimizing the Q-function. More precisely, the policy is responsible for exploring state-action regions that lead to discovering those modes.



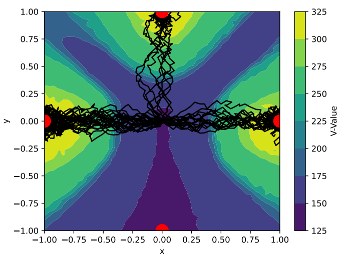
(a) $\alpha = 1.0$



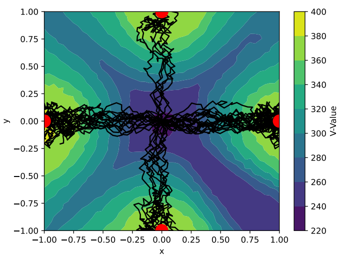
(b) $\alpha = 1.0$



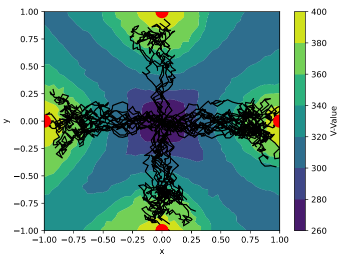
(c) $\alpha = 2.0$



(d) $\alpha = 3.0$



(e) $\alpha = 4.0$



(f) $\alpha = 5.0$

Figure 5.7.: Analyzing DIME's capabilities for learning multimodal behaviors. On a two-dimensional point mass task, we analze DIME's ability to learn multimodal behaviors. The point mass starts at the origin of the state space and is tasked to reach one of the four goals that are shown as red points. We approximate the value function using the Q-function learned by DIME for different α values. For this, we have uniformly discretized the state space into a grid wtih 2500 points and evaluated 15 actions per state, sampled from the policy to approximate the value function. Additionally, we have plotted the trajectories of the point mass for 50 rollouts. The rollouts were generated by executing the policy at the end of the training. Figures **(a)-(f)** show a clear trent that the higher the α value, the more modes are discovered in both, the value function and the resulting behavior, i.e. the trajectories of the point mass. This trend indicates that DIME is able to learn multimodal behaviors for specific α values, which alligns with the intuition from Sec. 5.5.2, where we have shown that higher α values lead to more exploration.

On a two-dimensional toy task we show that DIME is able to learn multimodal behaviors depending on the α parameter, that controls the exploration-exploitation trade-off as we have clarified in Sec. 5.5.2. In the point mass task, the agent starts always at the origin, i.e. the middle of the state space and is tasked to reach one of the four goals that are shown as red points in Fig. 5.7. We visualize the value function learned by DIME for the different α values and plot the trajectories of the point mass when executing the policy at the end of training as an upper layer on the value function visualization. Please note that the value function is a sample-based approximated of DIME's Q-function by sampling 15 actions per state. We have discretized the state space that consists of the x, y positions of the point mass into a uniform grid with a total of 2500 points. The action corresponds to the delta position on the x, y position of the point mass.

From the figures in Fig. 5.7 we can observe a clear trend. The higher the α value, the more modes are present in both, the value function and the resulting behavior, i.e. the trajectories of the point mass. This observation aligns with the intuition that the policy is responsible for the exploration behavior to discover different modes. Due to it's high-

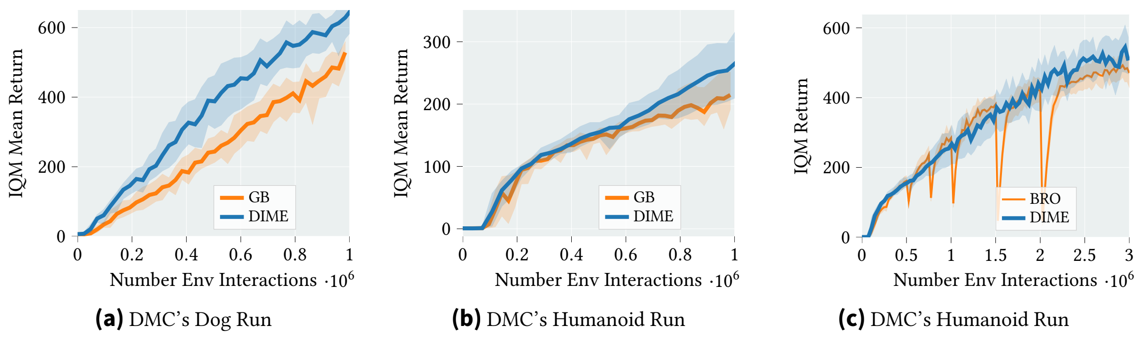


Figure 5.8.: Flexibility of the DIME framework (a)-(b) and comparison to BRO (c) on more environment interactions. As a proof of concept we have run experiments with the general bridge models (Richter and Berner, 2024) where we compare its performance (GB) against the denoising diffusion policy here labeled as DIME on the humanoid run task from the deepmind control suite. In (c) we compare DIME's perfromance agains BRO on the deempind control suite's humanoid run task with more environment interactions.

expressivity, DIME is able to represent these multiple modes that are present in the learned value function.

Fig. 5.7 not only shows that more modes are discovered the higher the α value becomes, but it also shows that the higher α becomes, the exploration within one mode increases. This can be clearly seen between the trajectories from Fig. 5.7e ($\alpha = 4.0$) and Fig. 5.7f ($\alpha = 5.0$), where a broader state space is covered while rolling out the trajectory.

For learning those multimodal behaviors consistently, we have used a target network for the Q-function to maintain the discovered modes, which otherwise were sometimes degenerated once another mode with higher return was discovered. This is an indicator that DIME needs to be extended to address Challenge 2 (Chapter 1) to consistently retain the discovered modes.

5.6. Flexibility of DIME's Framework

DIME's maximum entropy reinforcement learning framework for training diffusion policies is not specifically restricted to denoising diffusion policies but can be extended to general diffusion policies. This can be realized using the General Bridges framework as presented in Richter and Berner (2024). In this case, we can write the forward and backward process as

$$d\mathbf{a}_t = [\mathbf{f}(\mathbf{a}_t, t) + \beta\mathbf{u}(\mathbf{a}_t, \mathbf{s}, t)]\, dt + \sqrt{2\beta_t} d\mathbf{B}_t, \qquad \mathbf{a}_0 \sim \vec{\pi}_0(\cdot|\mathbf{s}), \tag{5.29}$$

$$d\mathbf{a}_t = [\mathbf{f}(\mathbf{a}_t, t) - \beta\mathbf{v}(\mathbf{a}_t, \mathbf{s}, t)]\, dt + \sqrt{2\beta_t} d\mathbf{B}_t, \qquad \mathbf{a}_T \sim \mathcal{N}(0, I), \tag{5.30}$$

with the drift and control functions $\mathbf{f}, \mathbf{u}, \mathbf{v} : \mathbb{R}^d \times [0, T] \longrightarrow \mathbb{R}^d$, the diffusion coefficient $\beta : [0, T] \longrightarrow \mathbb{R}^+$, standard Brownian motion $(\mathbf{B}_t)_{t\in[0,T]}$ and some target policy $\vec{\pi}_0$. Again we denote the marginal density of the forward process as $\vec{\pi}_t$ and the conditional density at time t given l as $\vec{\pi}_{t|l}$ for $t, l \in [0, T]$. The backward process starts from $\overleftarrow{\pi}_T = \vec{\pi}_T \sim \mathcal{N}(0, I)$ and runs backward in time where we denote its density as $\overleftarrow{\pi}$.

The respective discretizations using the Euler Maruyama (EM) (Särkkä and Solin, 2019) method are given by

$$\mathbf{a}^{n+1} = \mathbf{a}^n + [f(\mathbf{a}^n, n) + \beta \mathbf{u}(\mathbf{a}^n, \mathbf{s}, n)]\,\delta + \epsilon_n, \tag{5.31}$$

$$\mathbf{a}^{n-1} = \mathbf{a}^n - [f(\mathbf{a}^n, n) - \beta \mathbf{v}(\mathbf{a}^n, \mathbf{s}, n)]\,\delta + \xi_n, \tag{5.32}$$

where $\epsilon_n, \xi_n \sim \mathcal{N}(0, 2\beta\delta I)$, with the constant discretization step size δ such that $N = T/\delta$ is an integer. We have used the simplified notation where we write $\mathbf{a}^n$ instead of $\mathbf{a}^{n\delta}$. The discretizations admit the joint distributions

$$\vec{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s}) = \pi_0(\mathbf{a}^0|s) \prod_{n=0}^{N-1} \vec{\pi}_{n+1|n}(\mathbf{a}^{n+1}|\mathbf{a}^n, \mathbf{s}), \tag{5.33}$$

$$\overleftarrow{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s}) = \overleftarrow{\pi}_N(\mathbf{a}^N|s) \prod_{n=1}^{N} \overleftarrow{\pi}_{n-1|n}(\mathbf{a}^{n-1}|\mathbf{a}^n, \mathbf{s}), \tag{5.34}$$

with Gaussian kernels

$$\vec{\pi}_{n+1|n}(\mathbf{a}^{n+1}|\mathbf{a}^n, \mathbf{s}) = \mathcal{N}(\mathbf{a}^{n+1}|\mathbf{a}^n + [\mathbf{f}(\mathbf{a}^n, n) + \beta \mathbf{u}(\mathbf{a}^n, \mathbf{s}, n)]\,\delta, 2\beta\delta I), \tag{5.35}$$

$$\overleftarrow{\pi}_{n-1|n}(\mathbf{a}^{n-1}|\mathbf{a}^n, \mathbf{s}) = \mathcal{N}(\mathbf{a}^{n-1}|\mathbf{a}^n - [\mathbf{f}(\mathbf{a}^n, n) - \beta \mathbf{v}(\mathbf{a}^n, \mathbf{s}, n)]\,\delta, 2\beta\delta I). \tag{5.36}$$

Following the same framework presented in Section 5.4 , we can now optimize the controls **u** and **v** using the same objective

$$\bar{\mathcal{L}}(\mathbf{u}, \mathbf{v}) = D_{\mathrm{KL}}\left(\overleftarrow{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s}) | \vec{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s})\right), \tag{5.37}$$

where the target policy at time step $n = 0$ is given as

$$\pi_0(\mathbf{a}^0|\mathbf{s}) = \frac{\exp Q^{\overleftarrow{\pi}}(\mathbf{s}, \mathbf{a}^0)}{\mathcal{Z}^{\overleftarrow{\pi}}(\mathbf{s})}. \tag{5.38}$$

In practice, we optimize the control functions **u** and **v** using parameterized neural networks.

As a proof of concept, we have run experiments using the general bridge framework within the maximum entropy objective as suggested in our work. The learning curves can be seen in Fig. 5.8. While the denoising diffusion policy (DIME) performs better, the general diffusion policy (GB) is able to learn, showcasing the genral usage of the proposed framework. Additionally, the results show a potential for exploring different diffusion model types in the context of reinforcement learning.

5.7. Conclusion and Future Work

In this work, we introduced DIME, a method for learning diffusion models for maximum entropy reinforcement learning by leveraging connections to approximate inference. We view this work as a starting point for exciting future research. Specifically, we explored *denoising* diffusion models, where the forward process follows an Ornstein-Uhlenbeck process. However, approximate inference with diffusion models is an active and rapidly evolving field, with numerous recent advancements that consider alternative stochastic processes. For example, Richter and Berner (2024) proposed learning both the forward and backward processes, while Nusken et al. (2024) further enhanced exploration by incorporating the gradient of the target density into the diffusion process. Additionally, Chen et al. (2025) combined learned diffusion models with Sequential Monte Carlo (Del Moral et al., 2006), resulting in a highly effective inference method. These approaches hold significant promise for further improving diffusion-based policies in RL. We have conducted already presented experiments as a proof of concept on the framework from Richter and Berner (2024) in Section 5.6, demonstrating the potential to further extensions. Finally, we note that the loss function used in this work (see Eq. 5.25) is based on the Kullback-Leibler divergence. However, in principle, any divergence could be used. For instance, the log-variance divergence (Richter and Berner, 2024) has shown promising results in optimizing diffusion models for approximate inference (Chen et al., 2025; Noble et al., 2024). Exploring alternative objectives could lead to additional performance improvements. Another interesting future research lies in investigating the effects of using more sophisticated critic structures, such as transformers, as proposed by Li et al. (2025).

6. Conclusion

In this thesis, we have proposed methods for learning versatile and diverse skills for robotics using reinforcement learning (RL). In the following, we summarize the main contributions of this thesis and conclude with a discussion and an outlook for interesting future directions.

6.1. Summary

We first started by identifying *three challenges* that occur when learning versatile skills. In the following, we summarize those challenges.

Challenge 1 (C1). Challenge 1 deals with the representation and training of multimodal policies. For learning versatile skills, we require multimodal policies that can represent different modes in the action space and that eventually lead to different behaviors. We proposed using Mixture of Experts (MoE) and diffusion policies as two classes of distributions that can represent multimodality and are particularly suitable as policy representations. However, training such policies is not straightforward and requires tailored algorithms.

Challenge 2 (C2). Challenge 2 deals with the problem of retaining multimodalities. This challenge emerges because some modes are underrepresented in the collected training data. This underrepresentation arises because those modes are not as progressed as others in training and can consequently be discarded by the RL algorithm which increases the likelihood of modes that lead to higher returns. We addressed this challenge by introducing an automatic curriculum shaping methodology that ensures that each of the experts in an MoE is guaranteed to become an expert in a specific context space.

Challenge 3 (C3). Finally, Challenge 3 deals with the problem of learning policies that can adapt using highly non-linear mappings. This non-linear adaptation is required not only for generating the actions by each expert or the diffusion policy but also in the context of curriculum learning where the context distributions need to be expressive to represent sharp discontinuities in the context space.

To address these challenges, we have proposed three algorithms that optimize MoE and diffusion policies and are capable of learning versatile skills.

In Chapter 3 we have introduced *Specializing Versatile Skill Libraries (SVSL)*. SVSL first introduces an extension with a context distribution for automatic curriculum shaping of the MoE and augments the episode-based maximum entropy objective for self-paced learning (Section 2.5). As a second step, SVSL decomposes the objective into per-expert objectives that allow the optimization of each expert of the MoE policy independently. The per-expert context distributions are also updated alongside each expert in a higher hierarchy of this decomposition and allow shaping a per-expert curriculum and therefore allowing it to become an expert in a local context region. We have shown that the emerging reward augmentations are essential for learning versatile skills and we have demonstrated that SVSL is capable of learning versatile skills in complex robotics-simulated environments, although only linear expert representations with a Gaussian-shaped context distribution are used. Because of this decomposition and automatic curriculum learning, SVSL addresses C1 and C2.

Next, we have introduced *Acquiring Diverse Skills using Curriculum Reinforcement Learning with Mixture of Experts (Di-SKilL)* (Chapter 4), which builds on SVSL's objective and extends its MoE policy with non-linear experts and energy-based representations for the per-expert context distribution. This representation enhancement requires adapting the optimization, especially for the per-expert context distribution because energy-based models (EBM) are not straightforward to train due to their normalization constant. We proposed sampling a batch of contexts from the environment's context distribution by simply resetting the environment without execution. Those samples are used to approximate the normalization constant of the EBM and for automatic curriculum learning, where the per-expert distributions put higher probabilities to contexts they favor. Additionally, we used the trust-region policy gradient approach (Otto et al., 2023) for updating the deep experts. Thereby, Di-SkilL mainly focuses on C3 but also addresses C1 and C2 by using the same decomposition as SVSL.

Finally, in Chapter 5 we have introduced *Diffusion-Based Maximum Entropy Reinforcement Learning (DIME)*, which proposes a tractable lower-bound to the step-based maximum entropy RL objective for learning a diffusion-based policy. Diffusion models have recently proven highly powerful models for generating high-dimensional data, while still representing highly multimodal distributions. Their distinct benefit over MoE policies is that they do not need several experts to represent multimodality, but can represent this using a single model. Additionally, they do not require much hyperparameter tuning and are stable in training. However, they are not straightforward to apply in the maximum entropy RL setting because they do not have a tractable marginal entropy. Therefore, DIME proposes a tractable lower bound to the step-based maximum entropy RL objective. The resulting algorithm is a hyperparameter-light algorithm that effectively trains diffusion policies that achieve state of the art performance on high-dimensional control tasks. Importantly, it provides a principled way to control the exploration of the diffusion-based policy. DIME mainly addresses C1 in the context of diffusion models and C3 due to its inherent non-linear representation capabilities.

6.2. Discussion and Outlook

The first two algorithms, SVSL and Di-SkilL (Chapters 3 and 4), have shown that training an MoE policy in the ERL setting leads to discovering versatile skills. While ERL has its benefits in exploration, it is very sample inefficient, usually requiring much more samples than SRL counterparts. This is a main drawback of SVSL and Di-SkilL that limits their application to real-world robotics. Therefore, analyzing their learning capabilities in the context of off-policy ERL settings is an interesting future direction. For instance, utilizing a transformer-based critic in combination with Motion Primitives (MPs) based policy representation has been proposed by Li et al. (2025) already and has shwon promising performance. Yet, this approach requires the Markov properties restricting it to Markovian reward functions. However, it is an interesting future direction to improve the sample efficiency of SVSL and Di-SkilL for real-world applications, especially in the context of fine-tuning policies.

On the other hand, the on-policy characteristic of SVSL and Di-SkilL makes them also suitable candidates for sim-to-real transfer, which became very active research in the era of highly parallelized simulators, where data generation is not a bottleneck anymore (Tan et al., 2018; Hoeller et al., 2024). So far this has been approached mainly in the SRL setting, but not in the ERL setting in the context of versatile skill learning setting. Therefore, this research direction holds the promise to equip robots with versatile skills in simulation and only fine-tune them in the real world for minor adjustments. Their highly flexible structure allows MoE policies to continuously add new skills over a lifetime, making them a suitable policy representation for continual learning as well. In combination with the sim-to-real transfer, this could yield a powerful framework for quickly providing robots with new skills for new situations.

So far we have used a single network for each expert in the MoE policies for both, the expert distribution and the context distribution in both algorithms SVSL and Di-SkilL. While this provides the flexible structure mentioned earlier, it could also quickly lead to a policy with too many experts such that the training becomes limited. Therefore, exploring new architectures with a common, shared backbone for both the experts and the context distributions is an important approach that needs to be analyzed. Extending this shared structure to a more flexible policy representation for example by activating or deactivating heads of the policy might be a step towards obtaining a compact, but still flexible policy structure.

Finally, it is an interesting research question on how we can extend SVSL and Di-SkilL to settings with semantic understanding. An intuitive approach is to use some of the Vision Language Models (VLMs) that have shown impressive generalization capabilities in robotics recently (Intelligence et al., 2025), but it is unclear how these models can be used in these settings.

In contrast to SVSL and Di-SkilL, DIME (Chapter 5) is an off-policy SRL algorithm that is highly sample efficient and can be considered in using real-world robotics applications. However, so far it is unclear how applicable is DIME, especially because its inference time is not analyzed yet. Although we didn't use too many diffusion steps in the experiments, the diffusion model can be a bottleneck in the control of tasks that require high-frequency control. Additionally, although DIME's training time is much less compared to some state-of-the-art methods, there is still room for improvement, for scalable real-world applications.

For instance, we can improve DIME's training time by considering a different objective for updating the diffusion model. The current objecive requires backpropagating the gradients through the whole diffusion process, which is computationally expensive and does not scale to use cases with many diffusion steps. However, this drawback can be addressed with objective functions that have been used in the field of variational inference. One such objective is the Log Variance loss (Richter et al., 2020) that allows to generate samples from any policy distribution other than the current policy, but still estimates the gradient of the Kullback-Leibler (KL) Divergence as used in DIME. This feature of the objective would allow DIME to use any samples from a different process and therefore would not require backpropagating through the whole chain.

During the analysis of DIME's capabilities to learn versatile skills, we observed that a target network for the critic is helpful because some of the modes might get lost otherwise. This is an indicator that we have to adapt DIME to also address challenge C2. While the target network is a first step towards this fix, it also restricts the algorithm as shown by Bhatt et al. (2024) and slows down the training. One solution for this is to use a mixture model for the initial noise distribution, where each mode is responsible for a mode of the target distribution, similar as done by Blessing et al. (2025b) in the context of variational inference. However, another interesting approach to this problem lies in using restricted updates to the Q-distribution. DIME leverages a distributional critic and that opens various options such as KL-regularized updates of the Q-function. In the context of this critic, we could take inspiration from the automatic curriculum learning approach of SVSL and Di-SkilL, but from the view of numerical continuation methods (Allgower and Georg, 1990) and start by optimizing a smoothed version of the critic and then gradually increase the sharpness of the critic's distribution, thereby ensuring to retain the modes.
Finally, extending DIME to fine-tuning pre-trained models is a promising future direction in RL in the era of large models and big data.

It remains an open question how RL in general will develop in the future and how we can enable the developed methods to equip robots with versatile skills. This thesis aims to provide a first step towards this goal.

Bibliography

A. Abdolmaleki, R. Lioutikov, N. Lua, L. Paulo Reis, J. Peters, and G. Neumann. Model-based relative entropy stochastic search. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 153–154, 2015.

A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

A. Abdolmaleki, D. Simões, N. Lau, L. P. Reis, and G. Neumann. Contextual direct policy search. *Journal of Intelligent & Robotic Systems*, 96(2):141–157, 2019.

F. V. Agakov and D. Barber. An auxiliary variational method. In *Neural Information Processing: 11th International Conference, ICONIP 2004, Calcutta, India, November 22-25, 2004. Proceedings 11*, pages 561–566. Springer, 2004.

R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

T. Akhound-Sadegh, J. Rector-Brooks, A. J. Bose, S. Mittal, P. Lemos, C.-H. Liu, M. Sendera, S. Ravanbakhsh, G. Gidel, Y. Bengio, et al. Iterated denoising energy matching for sampling from boltzmann densities. *arXiv preprint arXiv:2402.06121*, 2024.

R. Akrour, D. Tateo, and J. Peters. Continuous action reinforcement learning from a mixture of interpretable experts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6795–6806, 2021.

E. L. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction*, volume 13 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, Heidelberg, 1990. doi: 10.1007/978-3-642-61257-2.

B. D. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

O. Arenz, G. Neumann, and M. Zhong. Efficient gradient-free variational inference using policy search. In *International conference on machine learning*, pages 234–243. PMLR, 2018.

O. Arenz, M. Zhong, and G. Neumann. Trust-region variational inference with gaussian mixture models. *Journal of Machine Learning Research*, 21(163):1–60, 2020.

P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

S. Bahl, M. Mukadam, A. Gupta, and D. Pathak. Neural dynamic policies for end-to-end sensorimotor learning. *Advances in Neural Information Processing Systems*, 33:5058–5069, 2020.

P. Becker, O. Arenz, and G. Neumann. Expected information maximization: Using the i-projection for mixture density estimation. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=ByglLlHFDS`.

M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

R. Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

R. Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

J. Berner, L. Richter, and K. Ullrich. An optimal control perspective on diffusion-based generative modeling. *Transactions on Machine Learning Research*, 2024.

A. Bhatt, D. Palenicek, B. Belousov, M. Argus, A. Amiranashvili, T. Brox, and J. Peters. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *The Twelfth International Conference on Learning Representations*, 2024.

C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

D. Blessing, O. Celik, X. Jia, M. Reuss, M. X. Li, R. Lioutikov, and G. Neumann. Information maximizing curriculum: A curriculum-based approach for learning versatile skills. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

D. Blessing, X. Jia, J. Esslinger, F. Vargas, and G. Neumann. Beyond elbos: A large-scale evaluation of variational methods for sampling. In *Forty-first International Conference on Machine Learning*, 2024.

D. Blessing, J. Berner, L. Richter, and G. Neumann. Underdamped diffusion bridges with applications to sampling. In *The Thirteenth International Conference on Learning Representations*, 2025a.

D. Blessing, X. Jia, and G. Neumann. End-to-end learning of gaussian mixture priors for diffusion sampler. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL `https://openreview.net/forum?id=iXbUquaWbl`.

S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

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

D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters. Learning to play table tennis from scratch using muscular robots. *IEEE Transactions on Robotics*, 38(6): 3850–3860, 2022.

V. Caggiano, H. Wang, G. Durandau, M. Sartori, and V. Kumar. Myosuite – a contact-rich simulation suite for musculoskeletal motor control, 2022. arXiv preprint arXiv:2205.00588.

V. Campos, A. Trott, C. Xiong, R. Socher, X. Giró-i Nieto, and J. Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327. PMLR, 2020.

O. Celik, D. Zhou, G. Li, P. Becker, and G. Neumann. Specializing versatile skill libraries using local mixture of experts. In *Conference on Robot Learning*, pages 1423–1433. PMLR, 2021.

H. Chen, C. Lu, C. Ying, H. Su, and J. Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

J. Chen, L. Richter, J. Berner, D. Blessing, G. Neumann, and A. Anandkumar. Sequential controlled langevin diffusions. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=dImD2sgy86`.

C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

T. M. Cover. *Elements of information theory*. John Wiley & Sons, 1999.

A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

P. Dai Pra. A stochastic control approach to reciprocal diffusion processes. *Applied mathematics and Optimization*, 23(1):313–329, 1991.

C. Daniel, G. Neumann, and J. Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pages 273–281. PMLR, 2012.

M. P. Deisenroth, G. Neumann, J. Peters, et al. A survey on policy search for robotics. *Foundations and trends in Robotics*, 2(1-2):388–403, 2013.

P. Del Moral, A. Doucet, and A. Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39 (1):1–22, 1977.

M. Dennis, N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

S. Ding, K. Hu, Z. Zhang, K. Ren, W. Zhang, J. Yu, J. Wang, and Y. Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Z. Ding and C. Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

A. Doucet, W. Grathwohl, A. G. Matthews, and H. Strathmann. Score-based diffusion meets annealed importance sampling. *Advances in Neural Information Processing Systems*, 35: 21482–21494, 2022.

F. End, R. Akrour, J. Peters, and G. Neumann. Layered direct policy search for learning hierarchical skills. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6442–6448. IEEE, 2017.

L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.

B. Eysenbach and S. Levine. Maximum entropy rl (provably) solves some robust rl problems. In *International Conference on Learning Representations*, 2022.

B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

B. Eysenbach, R. Salakhutdinov, and S. Levine. The information geometry of unsupervised reinforcement learning. In *International Conference on Learning Representations*, 2021.

M. Faldor, F. Chalumeau, M. Flageat, and A. Cully. Map-elites with descriptor-conditioned gradients and archive distillation into a single policy. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 138–146, 2023a.

M. Faldor, F. Chalumeau, M. Flageat, and A. Cully. Synergizing quality-diversity with descriptor-conditioned reinforcement learning. *arXiv preprint arXiv:2401.08632*, 2023b.

L. Fang, R. Liu, J. Zhang, W. Wang, and B. Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor–critic methods. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, volume 80 of *Proceedings of Machine Learning Research*, pages 1582–1591, Stockholm, Sweden, 2018. PMLR.

T. Geffner and J. Domke. Langevin diffusion variational inference. In *International Conference on Artificial Intelligence and Statistics*, pages 576–593. PMLR, 2023.

D. Ghosh, A. Singh, A. Rajeswaran, V. Kumar, and S. Levine. Divide-and-conquer reinforcement learning. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJwelMbR-`.

T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 1851–1860. PMLR, 2018a.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018b. URL `http://proceedings.mlr.press/v80/haarnoja18b.html`.

T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018c.

K. Hansel, J. Urain, J. Peters, and G. Chalvatzaki. Hierarchical policy blending as inference for reactive robot control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10181–10188. IEEE, 2023.

N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

U. G. Haussmann and E. Pardoux. Time reversal of diffusions. *The Annals of Probability*, pages 1188–1205, 1986.

A. Hendawy, J. Peters, and C. D'Eramo. Multi-task reinforcement learning with mixture of orthogonal experts. In *The Twelfth International Conference on Learning Representations*, 2024.

M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

D. Hoeller, N. Rudin, D. Sako, and M. Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.

J. Huang, Y. Jiao, L. Kang, X. Liao, J. Liu, and Y. Liu. Schrödinger-föllmer sampler: sampling without ergodicity. *arXiv preprint arXiv:2106.10880*, 1, 2021.

Z. Huang, L. Liang, Z. Ling, X. Li, C. Gan, and H. Su. Reparameterized policy learning for multimodal trajectory optimization. *ICML*, 2023.

A. Hyvärinen and P. Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

C. Igel. Neuroevolution for reinforcement learning using evolution strategies. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 4, pages 2588–2595. IEEE, 2003.

A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. Technical report, 2002.

A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2): 328–373, 2013.

P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025.

H. Ishfaq, G. Wang, S. N. Islam, and D. Precup. Langevin soft actor-critic: Efficient exploration through uncertainty-driven critic learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=FvQsk3la17`.

M. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022.

X. Jia, D. Blessing, X. Jiang, M. Reuss, A. Donat, R. Lioutikov, and G. Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In *The Twelfth International Conference on Learning Representations*, 2024.

M. Jiang, M. Dennis, J. Parker-Holder, J. Foerster, E. Grefenstette, and T. Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021a.

M. Jiang, E. Grefenstette, and T. Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021b.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

S. M. Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

B. Kang, X. Ma, C. Du, T. Pang, and S. Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023.

T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

L. Keller, D. Tanneberg, S. Stark, and J. Peters. Model-based quality-diversity search for efficient robot learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9675–9680. IEEE, 2020.

D. P. Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

P. Klink, H. Abdulsamad, B. Belousov, and J. Peters. Self-paced contextual reinforcement learning. In *Conference on Robot Learning*, pages 513–529. PMLR, 2020a.

P. Klink, C. D' Eramo, J. R. Peters, and J. Pajarinen. Self-paced deep reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9216–9227. Curran Associates, Inc., 2020b.

P. Klink, C. D'Eramo, J. Peters, and J. Pajarinen. Boosted curriculum reinforcement learning. In *International Conference on Learning Representations*, 2022a.

P. Klink, H. Yang, C. D'Eramo, J. Peters, and J. Pajarinen. Curriculum reinforcement learning via constrained optimal transport. In *International Conference on Machine Learning*, pages 11341–11358. PMLR, 2022b.

P. Klink, C. D'Eramo, J. Peters, and J. Pajarinen. On the benefit of optimal transport for curriculum reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–15, 2024. doi: 10.1109/TPAMI.2024.3390051.

A. H. Klopf. *Brain function and adaptive systems: a heterostatic theory.* Air Force Cambridge Research Laboratories, Air Force Systems Command, United ..., 1972.

A. H. Klopf. A comparison of natural and artificial intelligence. *ACM SIGART Bulletin*, pages 11–13, 1975.

J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Learning Motor Skills*, pages 83–117. Springer, 2014.

J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

S. Kumar, A. Kumar, S. Levine, and C. Finn. One solution is not all you need: Few-shot extrapolation via structured maxent rl. *arXiv preprint arXiv:2010.14484*, 2020.

A. Kupcsik, M. Deisenroth, J. Peters, and G. Neumann. Data-efficient generalization of robot skills with contextual policy search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 27, 2013.

S. Lange, T. Gabel, and M. Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pages 45–73. Springer, 2012.

M. Laskin, D. Yarats, H. Liu, K. Lee, A. Zhan, K. Lu, C. Cang, L. Pinto, and P. Abbeel. Urlb: Unsupervised reinforcement learning benchmark. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.

A. T. Le, K. Hansel, J. Peters, and G. Chalvatzaki. Hierarchical policy blending as optimal transport. In *Learning for Dynamics and Control Conference*, pages 797–812. PMLR, 2023.

L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, and G. Neumann. Prodmp: A unified perspective on dynamic and probabilistic movement primitives. *IEEE Robotics and Automation Letters*, 8(4):2325–2332, 2023.

G. Li, H. Zhou, D. Roth, S. Thilges, F. Otto, R. Lioutikov, and G. Neumann. Open the black box: Step-based policy updates for temporally-correlated episodic reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024a.

G. Li, D. Tian, H. Zhou, X. Jiang, R. Lioutikov, and G. Neumann. TOP-ERL: Transformer-based off-policy episodic reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=N4NhVN30ph`.

Z. Li, R. Krohn, T. Chen, A. Ajay, P. Agrawal, and G. Chalvatzaki. Learning multimodal behaviors from scratch with diffusion policy gradient. *arXiv preprint arXiv:2406.00681*, 2024b.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016), San Juan, Puerto Rico, May 2–4, 2016*, 2016.

H. Liu and P. Abbeel. Aps: Active pretraining with successor features. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6736–6747. PMLR, 18–24 Jul 2021.

W. Lotter, G. Sorensen, and D. Cox. A multi-scale cnn and curriculum learning strategy for mammogram classification. In *International Workshop on Deep Learning in Medical Image Analysis*, pages 169–177. Springer, 2017.

C. Lu, H. Chen, J. Chen, H. Su, C. Li, and J. Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pages 22825–22855. PMLR, 2023.

L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *International conference on machine learning*, pages 1445–1453. PMLR, 2016.

S. Mannor, R. Y. Rubinstein, and Y. Gat. The cross entropy method for fast policy search. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 512–519, 2003.

L. Mao, H. Xu, X. Zhan, W. Zhang, and A. Zhang. Diffusion-dice: In-sample diffusion guidance for offline reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

S. Messaoud, B. Mokeddem, Z. Xue, L. Pang, B. An, H. Chen, and S. Chawla. S 2 ac: Energy-based reinforcement learning with stein soft actor critic. In *The Twelfth International Conference on Learning Representations*, 2024.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

P. Morerio, J. Cavazza, R. Volpi, R. Vidal, and V. Murino. Curriculum dropout. In *Proceedings of the IEEE international conference on computer vision*, pages 3544–3552, 2017.

K. P. Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

M. Nauman and M. Cygan. On the theory of risk-aware agents: Bridging actor-critic and economics. In *ICML 2024 Workshop: Aligning Reinforcement Learning Experimentalists and Theorists*, 2023.

M. Nauman, M. Ostaszewski, K. Jankowski, P. Miłoś, and M. Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

E. Nelson. *Dynamical theories of Brownian motion*, volume 101. Princeton university press, 2020.

G. Neu, A. Jonsson, and V. Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

G. Neumann et al. Variational inference for policy search in changing situations. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 817–824, 2011.

E. Nikishin, M. Schwarzer, P. D'Oro, P.-L. Bacon, and A. Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pages 16828–16847. PMLR, 2022.

O. Nilsson and A. Cully. Policy gradient assisted map-elites. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 866–875, 2021.

M. Noble, L. Grenioux, M. Gabrié, and A. O. Durmus. Learned reference-based diffusion sampling for multi-modal distributions. *arXiv preprint arXiv:2410.19449*, 2024.

N. Nusken, F. Vargas, S. Padhy, and D. Blessing. Transport meets variational inference: Controlled monte carlo diffusions. In *The Twelfth International Conference on Learning Representations: ICLR 2024*, 2024.

T. Osa, V. Tangkaratt, and M. Sugiyama. Discovering diverse solutions in deep reinforcement learning. *arXiv preprint arXiv:2103.07084*, 2021.

F. Otto, P. Becker, N. A. Vien, H. C. Ziesche, and G. Neumann. Differentiable trust region layers for deep reinforcement learning. *arXiv preprint arXiv:2101.09207*, 2021.

F. Otto, O. Celik, H. Zhou, H. Ziesche, V. A. Ngo, and G. Neumann. Deep black-box reinforcement learning with movement primitives. In *Conference on Robot Learning*, pages 1244–1265. PMLR, 2023.

G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann. Probabilistic movement primitives. *Advances in neural information processing systems*, 26:2616–2624, 2013.

J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21:682–697, 2008.

J. Peters, K. Mulling, and Y. Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

M. Psenka, A. Escontrela, P. Abbeel, and Y. Ma. Learning a diffusion model policy from rewards via q-score matching. In *Forty-first International Conference on Machine Learning*, 2024.

S. Racaniere, A. Lampinen, A. Santoro, D. Reichert, V. Firoiu, and T. Lillicrap. Automated curriculum generation through setter-solver interactions. In *International Conference on Learning Representations*, 2020.

A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann. Stable baselines3. `https://github.com/DLR-RM/stable-baselines3`, 2019.

A. Raffin, J. Kober, and F. Stulp. Smooth exploration for robotic reinforcement learning. In *Conference on robot learning*, pages 1634–1644. PMLR, 2022.

R. Ranganath, D. Tran, and D. Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333. PMLR, 2016.

K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.

J. Ren, Y. Li, Z. Ding, W. Pan, and H. Dong. Probabilistic mixture-of-experts for efficient deep reinforcement learning. *arXiv preprint arXiv:2104.09122*, 2021.

M. Reuss, M. Li, X. Jia, and R. Lioutikov. Goal conditioned imitation learning using score-based diffusion policies. In *Robotics: Science and Systems*, 2023.

L. Richter and J. Berner. Improved sampling via learned diffusions. In *The Twelfth International Conference on Learning Representations*, 2024.

L. Richter, A. Boustati, N. Nüsken, F. Ruiz, and O. D. Akyildiz. Vargrad: a low-variance gradient estimator for variational inference. *Advances in Neural Information Processing Systems*, 33:13481–13492, 2020.

M. Riemer, M. Liu, and G. Tesauro. Learning abstract options. *arXiv preprint arXiv:1810.11583*, 2018.

G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, Cambridge, UK, 1994. Introduces the algorithm later known as SARSA.

T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

N. Sarafianos, T. Giannakopoulos, C. Nikou, and I. A. Kakadiaris. Curriculum learning for multi-task classification of visual attributes. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2608–2615, 2017.

S. Särkkä and A. Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

S. Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *Robotics research. the eleventh international symposium*, pages 561–572. Springer, 2005.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 2016.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

F. Sehnke, C. Osendorfer, T. Rückstiess, A. Graves, J. Peters, and J. Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 21(4):551–559, May 2010. doi: 10.1016/j.neunet.2009.12.004.

N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto. Behavior transformers: Cloning *k* modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

S. Sinha, A. Garg, and H. Larochelle. Curriculum by smoothing. *Advances in Neural Information Processing Systems*, 33:21653–21664, 2020.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999a.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999b.

J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

V. Tangkaratt, H. van Hoof, S. Parisi, G. Neumann, J. Peters, and M. Sugiyama. Policy search with high-dimensional context variables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

S. Tao, A. Shukla, T.-k. Chan, and H. Su. Reverse forward curriculum learning for extreme sample and demonstration efficiency in rl. 2024.

E. Todorov. General duality between optimal control and estimation. In *2008 47th IEEE conference on decision and control*, pages 4286–4292. IEEE, 2008.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

S. Tosatto, G. Chalvatzaki, and J. Peters. Contextual latent-movements off-policy optimization for robotic manipulation skills. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10815–10821. IEEE, 2021.

M. Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pages 1049–1056, 2009.

D. Tran, R. Ranganath, and D. M. Blei. The variational gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.

S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638.

B. Tzen and M. Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Conference on Learning Theory*, pages 3084–3114. PMLR, 2019.

H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

F. Vargas, A. Ovsianas, D. Fernandes, M. Girolami, N. D. Lawrence, and N. Nüsken. Bayesian learning via neural schrödinger–föllmer flows. *Statistics and Computing*, 33(1):3, 2023.

F. Vargas, W. S. Grathwohl, and A. Doucet. Denoising diffusion samplers. In *The Eleventh International Conference on Learning Representations*, 2024.

P. Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

D. Wang and Q. Liu. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016.

Y. Wang, L. Wang, Y. Jiang, W. Zou, T. Liu, X. Song, W. Wang, L. Xiao, J. WU, J. Duan, and S. E. Li. Diffusion actor-critic with entropy regulator. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=l0c1j4QvTq`.

Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. International Conference on Learning Representations, 2023.

C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

C. J. C. H. Watkins. Learning from delayed rewards. 1989.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

D. Whitley, S. Dominic, R. Das, and C. W. Anderson. Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13:259–284, 1993.

D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696.

J. Wöhlke, F. Schmitt, and H. van Hoof. A performance-based start state curriculum framework for reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1503–1511, 2020.

L. Yang, Z. Huang, F. h. Lei, Y. Zhong, Y. Yang, C. Fang, S. Wen, B. Zhou, and Z. Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

Y. Yang, T. Zhou, Q. He, L. Han, M. Pechenizkiy, and M. Fang. Task adaptation from skills: Information geometry, disentanglement, and new objectives for unsupervised reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

D. Zhang, R. T. Chen, C.-H. Liu, A. Courville, and Y. Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *arXiv preprint arXiv:2310.02679*, 2023.

Q. Zhang and Y. Chen. Path integral sampler: a stochastic control approach for sampling. *arXiv preprint arXiv:2111.15141*, 2021.

Y. Zhang, P. Abbeel, and L. Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33:7648–7659, 2020.

H. Zhou, D. Blessing, G. Li, O. Celik, X. Jia, G. Neumann, and R. Lioutikov. Variational distillation of diffusion policies into mixture of experts. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=iiYadgKHwo`.

B. D. Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

A. Appendix for Chapter 3

A.1. Derivations

In the following we derive our objectives from Section 3.3.

A.1.1. Maximum Entropy Skill Learning with Curriculum

We start our derivation from the KL-regularized maximum entropy objective

$$\max_{\pi(\mathbf{c},\boldsymbol{\theta})} \mathbb{E}_{\pi(\mathbf{c})} \left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c})} \left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta})\right] + \alpha \mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c})\right]\right] - \beta \mathrm{KL}\left(\pi(\mathbf{c}) \parallel p(\mathbf{c})\right). \tag{A.1}$$

We use following identities

$$\begin{aligned}
\pi(\boldsymbol{\theta}|\mathbf{c}) &= \sum_o \frac{\pi(\mathbf{c}|o)\pi(o)}{\pi(\mathbf{c})}\pi(\boldsymbol{\theta}|\mathbf{c},o), \\
\log \pi(\boldsymbol{\theta}|\mathbf{c}) &= \log \frac{\pi(\boldsymbol{\theta}|\mathbf{c},o)\pi(o|\mathbf{c})}{\pi(o|\mathbf{c},\boldsymbol{\theta})}, \\
\pi(\mathbf{c}) &= \sum_o \pi(\mathbf{c}|o)\pi(o), \\
\log \pi(\mathbf{c}) &= \log \frac{\pi(\mathbf{c}|o)\pi(o)}{\pi(o|\mathbf{c})}
\end{aligned}$$

and insert them into our objective in Eq. A.1, which leads to

$$\begin{aligned}
L = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)} \left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)} \left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) - \alpha \log \pi(\boldsymbol{\theta}|\mathbf{c},o) - \alpha \log \pi(o|\mathbf{c}) + \alpha \log \pi(o|\mathbf{c},\boldsymbol{\theta})\right]\right] \\
&+ \sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)} \left[-\beta \log \pi(\mathbf{c}|o) + \beta \log \pi(o|\mathbf{c}) - \beta \log \pi(o)\right].
\end{aligned} \tag{A.2}$$

Note that we have dropped the $\log p(\mathbf{c})$ term since we assume $p(\mathbf{c})$ to be uniformly distributed in a given interval.
By further rearranging the terms we can reformulate this objective as

$$\begin{aligned}
L = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)} \left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)} \left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) - \alpha \log \pi(\boldsymbol{\theta}|\mathbf{c},o) + \alpha \log \pi(o|\mathbf{c},\boldsymbol{\theta})\right]\right] \\
&+ \sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)} \left[-\beta \log \pi(\mathbf{c}|o) + (\beta - \alpha) \log \pi(o|\mathbf{c}) - \beta \log \pi(o)\right].
\end{aligned} \tag{A.3}$$

As stated in the paper, we can not optimize this objective. Therefore we introduce the auxiliary distributions $\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})$ and $\tilde{\pi}(o|\mathbf{c})$ as

$$\begin{aligned} L = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) - \alpha\log\pi(\boldsymbol{\theta}|\mathbf{c},o) + \alpha\log\pi(o|\mathbf{c},\boldsymbol{\theta}) + \alpha\log\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right.\right. \\ &\left.\left.-\alpha\log\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right]\right] + \sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[-\beta\log\pi(\mathbf{c}|o) - \beta\log\pi(o) + (\beta-\alpha)\log\pi(o|\mathbf{c})\right. \\ &\left.+(\beta-\alpha)\log\tilde{\pi}(o|\mathbf{c}) - (\beta-\alpha)\tilde{\pi}(o|\mathbf{c})\right]. \end{aligned}$$

We can rearrange the terms further to

$$\begin{aligned} L = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha\log\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right] + (\beta-\alpha)\log\tilde{\pi}(o|\mathbf{c})\right] \\ &+ \alpha\mathbb{E}_{\pi(o),\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathcal{H}\left(\pi(\boldsymbol{\theta}|\mathbf{c},o)\right)\right] + \beta\mathbb{E}_{\pi(o)}\left[\mathcal{H}\left(\pi(\mathbf{c}|o)\right)\right] + \beta\mathcal{H}\left(\pi(o)\right) \\ &+ \alpha\mathbb{E}_{\pi(\mathbf{c}),\pi(\boldsymbol{\theta}|\mathbf{c})}\left[\mathrm{KL}\left(\pi(o|\mathbf{c},\boldsymbol{\theta})||\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right)\right] + (\beta-\alpha)\mathbb{E}_{\pi(\mathbf{c})}\left[\mathrm{KL}\left(\pi(o|\mathbf{c})||\tilde{\pi}(o|\mathbf{c})\right)\right]. \quad \text{(A.4)} \end{aligned}$$

A.1.2. Lower-Bound Decomposition for Expert Distributions

By observing that not all terms depend on $\pi(\boldsymbol{\theta}|\mathbf{c},o)$, we extract only the important ones from the objective in Eq. A.4 as

$$\begin{aligned} \tilde{L}_c = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha\log\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right]\right] + \alpha\mathbb{E}_{\pi(o),\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathcal{H}\left(\pi(\boldsymbol{\theta}|\mathbf{c},o)\right)\right] \\ &+ \alpha\mathbb{E}_{\pi(\mathbf{c}),\pi(\boldsymbol{\theta}|\mathbf{c})}\left[\mathrm{KL}\left(\pi(o|\mathbf{c},\boldsymbol{\theta})||\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right)\right]. \end{aligned}$$

Since the KL is always positive, we can write the lower bound to this objective as

$$L_c = \sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha\log\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right]\right] + \alpha\mathbb{E}_{\pi(o),\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathcal{H}\left(\pi(\boldsymbol{\theta}|\mathbf{c},o)\right)\right],$$

such that $\tilde{L}_c \geq L_c$ always holds. After maximizing L_c w.r.t. $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ we tighten the lower bound by updating the responsibilities as described in the paper.

A.1.3. Lower Bound Decomposition for Per-Expert Context Distributions

By neglecting all terms which do not depend on $\pi(\mathbf{c}|o)$ in objective (A.4), we can write the objective for optimizing w.r.t. $\pi(\mathbf{c}|o)$ as

$$\begin{aligned} \tilde{L}_k = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[L_c(o,\mathbf{c}) + (\beta-\alpha)\log\tilde{\pi}(o|\mathbf{c})\right] + \beta\mathbb{E}_{\pi(o)}\left[\mathcal{H}\left(\pi(\mathbf{c}|o)\right)\right] \\ &+ (\beta-\alpha)\mathbb{E}_{\pi(\mathbf{c})}\left[\mathrm{KL}\left(\pi(o|\mathbf{c})||\tilde{\pi}(o|\mathbf{c})\right)\right], \end{aligned}$$

where

$$L_c(o,\mathbf{c}) = \mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha\log\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right] + \alpha\mathcal{H}(\pi(\boldsymbol{\theta}|\mathbf{c},o)).$$

Again we can observe that the KL term is always positive and we can write the lower bound to $\tilde{L}_k$ as

$$L_{c,k} = \sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[L_c(o,\mathbf{s}) + (\beta-\alpha)\log\tilde{\pi}(o|\mathbf{c})\right] + \beta\mathbb{E}_{\pi(o)}\left[\mathcal{H}\left(\pi(\mathbf{c}|o)\right)\right],$$

such that $\tilde{L}_k \geq L_{c,k}$ holds. By setting the auxillary distributions to the responsibilities of the updated model, we tighten the bound.

A.1.4. Lower Bound Decomposition for Prior Weights

Finally we can write down the objective for updating the prior weights $\pi(o)$ as

$$L_p = \sum_o L_{c,k}(o) + \beta\mathcal{H}\left(\pi(o)\right), \tag{A.5}$$

where

$$L_{c,k}(o) = \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[L_c(o,\mathbf{s}) + (\beta-\alpha)\log\tilde{\pi}(o|\mathbf{c})\right] + \beta\mathrm{H}\left(\pi(\mathbf{c}|o)\right).$$

Some context regions naturally lead to low reward due to for example high action regularizations. Still, these context regions should be discovered, even if the auxiliary rewards $\tilde{\pi}(o|\mathbf{c})$ are not sufficient. For this purpose, we calculate a value function, which reveals the mean reward in a context. If solely one expert covers this context region, it will get a high value for updating its weight $\pi(o)$, although the task reward might be bad compared to other experts in other regions. We use importance sampling, to calculate the Value function

$$V(\mathbf{c}) = \sum_o \frac{\pi(o|\mathbf{c})}{\hat{\pi}(o|\mathbf{c})}\int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta}|\mathbf{c},o)\mathrm{R}(\mathbf{c},\boldsymbol{\theta})d\boldsymbol{\theta}, \tag{A.6}$$

where $\hat{\pi}(o|\mathbf{c})$ are the gating probabilities before starting to update the weights $\pi(o)$ at the end of our algorithm (see Algorithm 1). We completely resample the small replay buffer for all experts before starting to update $\pi(o)$. We use the Nadaraya Watson predictor to get $V(\mathbf{c})$. Thus, the update rule for $\pi(o)$ is given by

$$L_p = \sum_o L_{c,k}(o) - V(\mathbf{c}) + \beta\mathcal{H}\left(\pi(o)\right). \tag{A.7}$$

A.2. Algorithmic Details

We describe algorithmic details in the algorithm box 1. We start with one expert, which is randomly added and incrementally add experts after training the lastly added expert for K iterations. We update the expert and context distributions individually. Note that there is no strict order on updating the experts or context distributions first.

Every H iterations we fine-tune all experts, if more than one expert is available. This allows the already trained experts to adjust for the newly added one. After finishing training the lastly added expert k, we check if it converged to a local optimum. There are different ways on checking that. For example by comparing the entropy, or achieved task reward to already trained experts. This step is not absolutely necessary since for the upcoming fine tune steps, the expert might improve. For the Beer Pong and Table Tennis task, we have disabled the deletion step in line 13.

As last step we update the prior weights $\pi(o)$ and delete all experts, which are below a threshold value (e.g. 10^{-5}) for $\pi(o)$.

Note that in some experiments, it might happen that highly versatile skills result in worse rewards, e.g. through action regularization, although the task is solved successfully. In order to avoid deleting these diverse skills from the skill library at the end of the optimization, a higher β value, β_w for the entropy of $\pi(o)$ in Eq. A.7 can be used. This higher β_w value will encourage to put weights on skills which are not achieving highest reward. Although those skills still will have lower weight compared to other experts, they are kept in the skill set and might be useful. For example when the model should be adapted to an environmental change. We made use of a different β_w value only once in the Beer Pong task (see Experimental Details A.3).

Note that we have used a variant of MORE (Abdolmaleki et al., 2015) for optimizing Obj. 3.6, as described in (Arenz et al., 2020) and a variant of Contextual MORE Tangkaratt et al. (2017) for optimizing Obj. 3.5 as described in (Becker et al., 2020).

Also note, that for updating each expert we use a small replay buffer. The newly taken samples replace the oldest samples in the buffer. For the updates we simply use all samples in the buffer for each expert.

A.3. Experimental Details

We describe the different hyper parameters and environment specifications used in the experiments part here. Please note: We included a context punishment in all reward functions for all experiments for contexts which were sampled out of the context range defined by the environment. This encourages the context distributions $\pi(\mathbf{c}|o)$ to stay in the valid context region as defined from the environment. We report the reward functions and the environment specifications in the sub section for each experiment.

Algorithm 1 Versatile Skill Learning

```
Input: $\alpha$, $\beta$, $\beta_w$, N, K, H
Output: $\pi(\boldsymbol{\theta}|\mathbf{c})$
1: for $k = 1$ to N do
2:     $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$, $\pi(\mathbf{c}|o) \leftarrow$ randomly_add_expert()
3:     $\pi(o) = 1/k, \ \forall o$
4:     for $i = 1$ to $K$ do
5:         if i% H is 0 then
6:             update_all_experts($\alpha$) to obj. (3.5) each
7:             update_all_context_distributions($\alpha$, $\beta$) to obj. (3.6) each
8:         else
9:             $\pi(\boldsymbol{\theta}|\mathbf{c}, o) \leftarrow$ update_expert_k($\alpha$) to obj. (3.5)
10:            $\pi(\mathbf{c}|o) \leftarrow$ update_context_distribution_k($\alpha$, $\beta$) to obj (3.6)
11:        end if
12:    end for
13:    if expert_k in local optimum then
14:        delete_expert_k()
15:    end if
16: end for
17: $\pi(o) \leftarrow$ update_prior_weights($\beta_w$) to obj. (A.7)
18: delete_redundant_experts()
```

As for the hyper parameters for all experiments for our Algorithm, we provide a summary in Table A.1. Following the notation from Algorithm 1, β_w is the entropy coefficient for $\pi(o)$ in Objective A.7, N is the number of incrementally added experts, K is the number of iterations one expert is trained and H describes the number that every H. iteration all experts are updated.

For the hyper parameters for all experiments for HiREPS and LaDiPS, we provide a summary in Table A.2 and A.3 respectively.

Hyper Parameter (Ours)	α	β	β_w	N	K	H
Planar Reacher (Ablation)	varies (see section 3.4.1)	1.0	1.0	60	350	50
Beer Pong Task	0.001	0.5	2.5	70	750	50
Table Tennis Task	0.00001	0.2	0.2	50	800	50

Table A.1.: Hyperparameters of our algorithm for all environments.

Hyper Parameter (HiREPS)	ϵ	κ	N
Planar Reacher (Ablation)	0.5	0.99	60
Beer Pong Task	0.3	0.9	70
Table Tennis Task	0.3	0.9	50

Table A.2.: Hyperparameters of HiREPS for all environments.

Hyper Parameter (LaDiPS)	ϵ	ϵ_{gating}	α	N
Beer Pong Task	0.1	0.05	$\log 30$	70
Table Tennis Task	0.01	0.02	$\log 30$	50

Table A.3.: Hyperparameters of LaDiPS.

A.3.1. Ablation Studies

For the Planar Reaching task we consider a two dimensional context space. Note that we have not used any Movement Primitive representation for the policy. Instead, the sampled θ from our search distributions directly represent the angles of each joint of the reacher task.

We consider a two dimensional context space x, y, where $x_{min} = 4.5$, $x_{max} = 7$ and $y_{min} = -6$, $y_{max} = 6$.

The reward function is given by:

$$R(\boldsymbol{\theta}, \mathbf{c}) = -||\boldsymbol{\theta}||_2^2 - 2 \cdot ||\mathbf{f}(\boldsymbol{\theta}) - \mathbf{c}||_2^2 - l_c(\mathbf{c}) - l_o(\boldsymbol{\theta}),$$

where

$$l_c(\mathbf{c}) = \begin{cases} 10, & c \notin C \\ 0, & else \end{cases}, \quad l_o(\mathbf{c}) = \begin{cases} 3, & b_{1:10} \in O \\ 0, & else \end{cases},$$

where l_c are the costs if a context $\mathbf{c}$ is not within the valid context range C and l_o are the costs, if one of the ten links b_i has a collision with the obstacles of the environment. $\mathbf{f}(\boldsymbol{\theta})$ are the forward kinematics of the planar reacher. Note that the angles $\boldsymbol{\theta}$ are always normalized into a range $[-\pi, \pi]$.

A.3.2. Beer Pong

For the Beer Pong task we consider a two dimensional context space x, y positions of the cup on the table, where $x_{min} = -0.32$, $x_{max} = 0.32$ and $y_{min} = -2.2$, $y_{max} = -1.2$. We use ProMPs as policy parameterizations. We also learn the length of the trajectory, which might lead to invalid, i. e. negative trajectory length. We use a punishment term, if the trajectories are out of a reasonable range and do not execute this sample on the environment. We do the same for the context samples. Furthermore, we do not execute trajectories, which violate the joint constraints of the robot. Since these samples are not executed on the environment, they are not counted as "taken samples". For the case that those restrictions are not violated, the reward function is given as

$$R(\boldsymbol{\theta}, \mathbf{c}) = \begin{cases} -4 - min(||p_{c,top} - p_{b,1:T}||_2^2) - 0.5||p_{c,bottom} - p_{b,T}||_2^2 - 10^{-4}\tau, & \text{if cond. 1} \\ -2 - min(||p_{c,top} - p_{b,1:T}||_2^2) - 0.5||p_{c,bottom} - p_{b,T}||_2^2 - 10^{-4}\tau, & \text{if cond. 2} \\ ||p_{c,bottom} - p_{b,T}||_2^2 + 1.5 \cdot 10^{-4}\tau, & \text{if cond. 3} \end{cases},$$

where $p_{c,top}$ is the position of the top edge of the cup, $p_{c,bottom}$ is the ground position of the cup, $p_{b,t}$ is the position of the ball at time point t and τ is the squared mean torque over all joints during one rollout. The different conditions are:

- cond. 1: The ball is not in the cup and had no table contact
- cond. 2: The ball is not in the cup and had table contact
- cond. 3: The ball is in the cup.

For the case that context samples were sampled out of the range we provide a high negative reward

$$R_c(\boldsymbol{\theta}, \mathbf{c}) = -30 - d_c^2,$$

where d_c^2 is the distance of the current context $\mathbf{c}$ to the valid context region.

For the case that the duration of the trajectory was sampled out of the range we provide a negative reward as

$$R_\theta(\mathbf{c}, \boldsymbol{\theta}) = \begin{cases} -30 - 10 \cdot (l - 0.1)^2, & \text{if } l < 0.1 \\ -30 - 10 \cdot (l - 1.3)^2, & \text{if } l > 1.3 \end{cases},$$

where l is the duration of the Trajectory in seconds. If the joint constraint limits are violated by the ProMP's trajectory, we give a punishment of -30.

A.3.2.1. On Versatility of our Solutions on the Beer Pong Task

In this section we want to compare the learned solutions of LaDiPS (End et al., 2017) and our method. As LaDiPS uses experience-sharing between experts, it outperforms our method in terms of sample efficiency, but does not find as qualitative solutions as our method, which is reflected in the end-rewards (see Fig. 3.2a), where we can clearly achieve a higher value. In addition to the more qualitative solutions, we are able to learn a mixture model with higher expected entropy, which can be seen in Fig. A.1. Our method is able to achieve more qualitative skills with a much higher entropy compared to LaDiPS and HiREPS, indicating that the proposed solutions by our method are more versatile than the others. Please note that we have defined a fine grid of contexts and sampled for each context $\mathbf{c}$ 1000 parameter vectors $\boldsymbol{\theta}$ from the mixture models to calculate the expected entropy. We have repeated this procedure for all 20 different trials/seeds and report the mean expected entropy together with two times the standard error. In addition to the expected mixture entropy we have conducted a qualitative comparison, where we have picked the first model for both, our method and LaDiPS and have let it run for twelve different contexts. For each context we have sampled 100 times from the mixture model and executed the mean of the sampled experts on the environment. Note that we only have taken those ball trajectories into account that lead to a successful solution. For LaDiPS all observed solutions for all contexts showed only one mode, in which the ball bounced once on the table and then landed in the cup. For our method the same solution could be

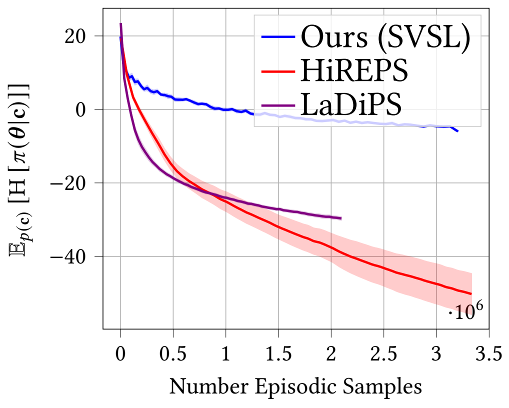


Figure A.1.: Expected Mixture Entropies.

seen for six contexts. For three contexts, our method threw the ball directly into the cup, whereas for another three contexts three different ball trajectories could be observed: i) direct throw, ii) bouncing ones on the table ii) bouncing once on the table and once on the wall.

A.3.3. Table Tennis

For the Table Tennis task we consider a four dimensional context space $[\mathbf{b}^{\mathbf{des}}, \ \mathbf{b}^{\mathbf{inc}}] \in \mathbb{R}^4$. We have the desired landing position of outgoing ball $\mathbf{b}^{\mathbf{des}} = [x^{des}, \ y^{des}] \in \mathbb{R}^2$ and the initial positions of the incoming ball $\mathbf{b}^{\mathbf{inc}} = [x^{inc}, \ y^{inc}] \in \mathbb{R}^2$ where $x^{des} \in [-1.2, \ -0.2]$, $y^{des} \in [-0.6, \ 0.6]$, $x^{inc} \in [-1.2, \ -0.2]$, $y^{inc} \in [-0.65, \ 0.65]$. We fix the initial ball velocity and z^{inc} position so we can transform the initial ball position to the incoming ball landing position on the table. We use ProMPs as policy parameterizations and learn the length of the trajectory, which might lead to invalid, i. e. negative trajectory length. We use a punishment term, if the length of trajectories are out of a reasonable range and do not execute this sample on the environment. We do the same for the context samples. Furthermore, we do not execute trajectories, which violate the joint constraints of the robot and return a punishment. Since these samples are not executed on the environment, they are not counted as "taken samples". For the case that those restrictions are not violated, the reward function is given as

$$R(\boldsymbol{\theta}, \mathbf{c}) = \begin{cases} 0.2 \cdot (1 - tanh(\min_t(||\mathbf{b_t} - \mathbf{r_t}||_2^2)), & \text{if cond. 1} \\ 2 \cdot (1 - tanh(\min_t(||\mathbf{b_t} - \mathbf{r_t}||_2^2)) & \\ + (1 - tanh(\min_t(||\mathbf{b^{des}} - \mathbf{b_{t,xy}}||_2^2)), & \text{if cond. 2} \\ 2 \cdot (1 - tanh(\min_t(||\mathbf{b_t} - \mathbf{r_t}||_2^2)) & \\ + 4 \cdot (1 - tanh((||\mathbf{b^{des}} - \mathbf{b^{land}}||_2^2)) + 1_{\{b_x^{land} < 0\}}, & \text{if cond. 3} \end{cases},$$

where $\mathbf{b_t} \in \mathbb{R}^3$ is the position of the ball at time point t, $\mathbf{b_{t,xy}} \in \mathbb{R}^2$ is the x, y position of the ball at time point t, $\mathbf{r_t} \in \mathbb{R}^3$ is the position of the racket at time point t, $\mathbf{b^{land}} \in \mathbb{R}^2$ is the real x, y landing position point. The different conditions are:

- cond. 1: The ball had no racket contact.
- cond. 2: The ball had racket contact but then no table contact.
- cond. 3: The ball had racket contact and then table contact.

For the case that context samples were sampled out of the range we provide a high negative reward

$$R_c(\boldsymbol{\theta}, \mathbf{c}) = -20 \cdot d_c^2,$$

where d_c^2 is the distance of the current context $\mathbf{c}$ to the valid context region.

For the case that the duration of the trajectory was sampled out of the range we provide a negative reward as

$$R_\theta(\mathbf{c}, \boldsymbol{\theta}) = \begin{cases} -3||l - 0.1||, & \text{if } l < 0.1 \\ -3||l - 5||, & \text{if } l > 5 \end{cases},$$

where l is the duration of the Trajectory in seconds.

For the case that the trajectory was sampled out of the joint constraints we provide a negative reward as

$$R_c(\boldsymbol{\theta}, \mathbf{c}) = -1 \cdot \frac{1}{T} \sum_i^T d_{qpos,t},$$

where $d_{qpos,t}$ is the distance of the planned joint positions to the valid joint position region.

A.3.3.1. On Versatility of our Solutions on the Table Tennis Task

We compare the versatility of learned solutions of LaDiPS (End et al., 2017), HiREPS (Daniel et al., 2012) and ours. We plot the expected mixture entropies of each algorithm in Fig. A.2a. For each of the 1600 uniformly context samples we have sampled 1000 parameter vectors θ from the mixture model to calculate the mixture entropy. We repeat this for all models resulting from 20 seeds/trials and plot the average of the entropies in Fig. A.2a. Please note that the fluctuations of the entropy curve in Fig. A.2a of our algorithm are due to adding experts during training.

We are able to achieve the highest entropy while clearly outperforming both methods in terms of the achieved task reward (see Fig. A.2b), which indicates that we are able to learn more qualitative skills.

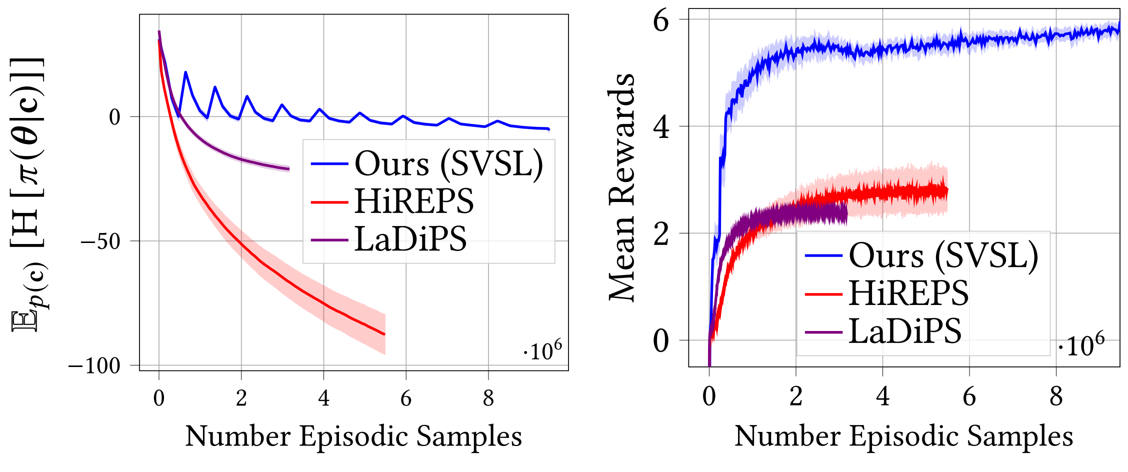


Figure A.2.: Expected Mixture Entropies (left) and **performance on the table tennis task** (right). Note that the performance graph is the same as in Fig. 3.2b, but with the full graph of our method to train 70 experts. The graph in Fig. 3.2b was clipped in the x-axis to maintain overview.

A.3.3.2. Hyperparameters

The hyperparameters for our algorithm are given in the following Table. Please note that we used punishments,

Hyper Parameter (Ours)	α	β	β_w	N	K	H
Reacher Reward (3.8)	$1e^{-6}$	15	15	40	250	50
Reacher Reward (3.9)	0.01	1	2.5	40	250	50
Reacher Reward (3.10)	1	25	25	40	800	50

Table A.4.: Hyperparameters of our algorithm for the reacher environment.

if contexts were sampled out of the context range (which is the right half circle around the base with radius 0.5), we provided a quadratic punishment to the distance of the sample to the valid context bound. We multiplied this distance with a factor of 1000 for the case of reward function (3.8, 3.9) and with a factor of 50000 for the case of reward function (3.10).

We normalized the observations and rewards for PPO and used the same hyperparameters for all three experiments, which we summarized in the following table

Hyp. Params (PPO)	α	N	GAE λ	γ	K	ϵ	network structure
Reacher Reward (3.8)	$1e^{-4}$	16384	0.95	0.99	64	0.2	[64, 64]
Reacher Reward (3.9)	$1e^{-4}$	16384	0.95	0.99	64	0.2	[64, 64]
Reacher Reward (3.10)	$1e^{-4}$	16384	0.95	0.99	64	0.2	[64, 64]

Table A.5.: Hyperparameters of PPO for the reacher environment. Note that α is the learning rate, N is the number of rollouts, γ is the discount factor, K is the batch size, ϵ is the clip value for the importance ratio. The layer structure is for both: the policy network as well as the value function network.

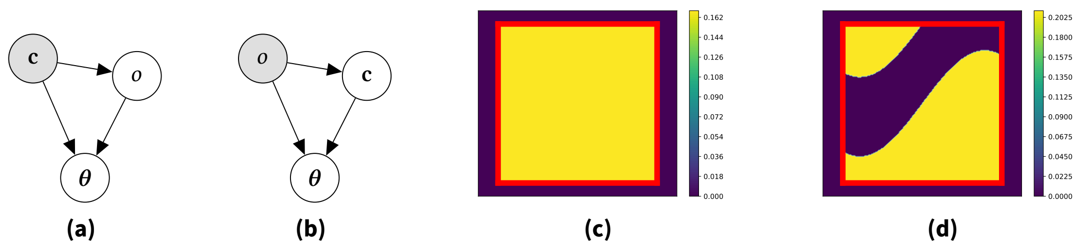


Figure B.1.: Probabilistic Graphical Models (PGMs) during **inference a)** and **training b)**. During **a))** the model observes the contexts $\mathbf{c}$ from the environment. An expert o is sampled from $\pi(o|\mathbf{c})$, which leads to an adjustment of the motion primitive parameters $\boldsymbol{\theta}$ by $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$. We iterate over each expert during **(b)**, sample the contexts $\mathbf{c}$ and $\boldsymbol{\theta}$ from the per-expert distribution $\pi(\mathbf{c}|o)$ and $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ respectively. Sampling from $\pi(\mathbf{c}|o)$ allows shaping the expert's curriculum. **c)** illustrates the environment's context distribution $p\left(\mathbf{c}\right)$ and a possibly optimal $\pi(\mathbf{c}|o)$ **(d))** in two-dim. space. Yellow areas indicate high and purple zero probability. The illustrations show that optimizing $\pi(\mathbf{c}|o)$ requires dealing with i) step-like non-linearities, ii) multi-modality, iii) bounded within the red rectangle support of $p\left(\mathbf{c}\right)$, complicating exploration.

B. Appendix for Chapter 4

B.1. Additional Information to Self-Paced Diverse Skill Learning with MoE

The general self-paced diverse skill learning objective

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c}),\pi(\mathbf{c})} \mathbb{E}_{\pi(\mathbf{c})}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c})}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta})\right] + \alpha \mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c})\right]\right] - \beta \mathrm{KL}\left(\pi\left(\mathbf{c}\right) \parallel p\left(\mathbf{c}\right)\right)$$

can be reformulated to

$$\begin{aligned}\max_{\pi(\mathbf{c},\boldsymbol{\theta})} \mathbb{E}_{\pi(o),\pi(\mathbf{c}|o)}&\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha \log \pi(o|\mathbf{c},\boldsymbol{\theta})\right] + \beta \log p\left(\mathbf{c}\right) + (\beta - \alpha) \log \pi(o|\mathbf{c})\right] \\ &+ \alpha \mathbb{E}_{\pi(o),\pi(\mathbf{c}|o)}\left[\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]\right] + \beta \mathbb{E}_{\pi(o)}\left[\mathrm{H}\left[\pi(\mathbf{c}|o)\right]\right] + \beta \mathrm{H}\left[\pi(o)\right], \qquad \text{(B.1)}\end{aligned}$$

by inserting $\pi(\boldsymbol{\theta}|\mathbf{c})$, $\pi\left(\mathbf{c}\right)$ from Eq. 4.2 into Eq. B.1 and applying Bayes theorem. This objective is not straightforward to optimize for Mixture of Experts MoE models and requires further steps to introduce a lower bound (see Section 4.2) that can be efficiently optimized. Please note that the variational distributions in Eq. 4.4 and Eq. 4.5 can be calculated in closed form by the identities

$$\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta}) = \pi_{old}(o|\mathbf{c},\boldsymbol{\theta}) = \frac{\pi_{old}(\boldsymbol{\theta}|\mathbf{c},o)\pi_{old}(o|\mathbf{c})}{\pi_{old}(\boldsymbol{\theta}|\mathbf{c})},$$
$$\tilde{\pi}(o|\mathbf{c}) = \pi_{old}(o|\mathbf{c}) = \frac{\pi_{old}(\mathbf{c}|o)\pi(o)}{\pi_{old}(\mathbf{c})}.$$

We refer the interested reader to Celik et al. (2021) for a detailed derivation.

B.2. Additional Related Work

Unsupervised Reinforcement Learning. Another field of research that considers learning diverse policies is unsupervised reinforcement learning (URL). In URL the agent is first trained solely with an intrinsic reward to acquire a diverse set of skills from which the most appropriate is picked to solve a downstream task. More related to our work is a group of algorithms that obtain their intrinsic reward based on information-theoretic formulations (Laskin et al., 2021; Eysenbach et al., 2019; Campos et al., 2020; Lee et al., 2019; Liu and Abbeel, 2021). However, their resulting objective is based on the mutual-information and differs from the objective we maximize. The learned skills in the pre-training aim to cover distinct parts of the state-space during pre-training in the absence of an extrinsic task reward which implies that skills are not explicitly trained to solve the same task in different ways. Those methods operate within the step-based RL setting which differs from CEPS.

B.3. Additional Information to Diverse Skill Learning

B.3.1. The Parameterization of the Mixture of Experts (MoE) Model

In the following, we provide details on the parameterization of the MoE model.

Parametrization of the expert $\pi(\boldsymbol{\theta}|\mathbf{c},o)$. We parameterize each expert $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ as a Gaussian policy $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\gamma}}(\mathbf{c}),\Sigma_{\boldsymbol{\gamma}}(\mathbf{c}))$, where the mean $\boldsymbol{\mu}_{\boldsymbol{\gamma}}(\mathbf{c})$ and the covariance $\Sigma_{\boldsymbol{\gamma}}(\mathbf{c})$ are functions of the context $\mathbf{c}$ and parameterized by a neural network with parameters $\boldsymbol{\gamma}$. Although the covariance $\Sigma_{\boldsymbol{\gamma}}(\mathbf{c})$ is formalized as a function of the context $\mathbf{c}$, we have not observed any advantages in doing so. In our experiments, we therefore parameterize the covariance as a lower-triangular matrix $\mathbf{L}$ and form the covariance matrix $\Sigma = \mathbf{L}\mathbf{L}^T$.

Parameterization of the per-expert context distribution $\pi(\mathbf{c}|o)$. The reader is referred to Section 4.4 for details on the parameterization of $\pi(\mathbf{c}|o)$

Parameterization of the prior $\pi(o)$. We fix the prior $\pi(o)$ to a uniform distribution over the number K of available experts and do not further optimize this distribution. This is a useful definition to increase the entropy of the mixture model.

Parameterization of the context distribution $\pi(\mathbf{c})$. Due to the relation $\pi(\mathbf{c}) = \sum_o \pi(\mathbf{c}|o)\pi(o)$, $\pi(\mathbf{c})$ is defined by $\pi(\mathbf{c}|o)$ and does not need explicit modelling.

Parameterization of the gating distribution $\pi(o|\mathbf{c})$. Due to the relation $\pi(o|\mathbf{c}) = \frac{\pi(\mathbf{c}|o)\pi(o)}{\pi(\mathbf{c})}$ we do not need an explicit parameterization of $\pi(o|\mathbf{c})$ and can easily calculate the probabilities for choosing the expert o given a context $\mathbf{c}$.

B.3.2. Using Motion Primitives in the Context of Reinforcement Learning

Motion Primitives (MPs) are a low-dimensional representation of a trajectory. For instance, instead of parameterizing a desired joint-level trajectory as the single state in each time step, MPs introduce a low-dimensional parameter vector $\boldsymbol{\theta}$ which concisely defines the trajectory to follow. The generation of the trajectory depends on the method that is used. Probabilistic Movement Primitives (ProMPs) (Paraschos et al., 2013) for example define the desired trajectory as a simple linear function $\boldsymbol{\tau} = \boldsymbol{\Phi}^T\boldsymbol{\theta}$, where $\boldsymbol{\Phi}$ are time-dependent basis functions (e.g. normalized radial basis functions). Dynamic Movement Primitives (DMPs) (Schaal, 2006) rely on a second-order dynamic system that provides smooth trajectories in the position and velocity space. Recently Probabilistic Dynamic Movement Primitives (ProDMPs) were introduced by Li et al. (2023) and combines the advantages of both methods, that is the easy generation of trajectories and smooth trajectories. We therefore rely on ProDMPs throughout this work.

In the context of reinforcement learning, the policy $\pi(\boldsymbol{\theta}|\mathbf{c})$, or in our case an expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ defines a distribution over the parameters $\boldsymbol{\theta}$ of the MP depending on the observed context $\mathbf{c}$. This allows the policy to quickly adapt to new tasks defined by $\mathbf{c}$.

B.3.3. Algorithm Details

Detailed descriptions of the algorithm during training and during inference are provided in the algorithm boxes Alg. 2 and Alg. 2, respectively. In each iteration during training, we sample a batch of contexts $\mathbf{c}$ from the environment by resetting it. We then iterate over each expert and evaluate the probabilities of these contexts $\mathbf{c}$ on each per-expert context distribution $\pi(\mathbf{c}|o)$ and sample then training contexts $\mathbf{c}_T$ from them. From the corresponding expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ we sample motion primitive parameters $\boldsymbol{\theta}$ and evaluate the samples $(\mathbf{c}_T, \boldsymbol{\theta})$ on the environment and observe a return $\mathrm{R}(\mathbf{c}, \boldsymbol{\theta})$ which we use to update the experts $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ and the per-expert context distributions $\pi(\mathbf{c}|o)$ by maximizing Obj. 4.7 and Obj. 4.8 respectively. During inference, we observe contexts $\mathbf{c}$ from the environment, calculate the gating distributions $\pi(o|\mathbf{c}) = \frac{\pi(\mathbf{c}|o)\pi(o)}{\pi(\mathbf{c})}$ from which we sample the expert o. We then either take the mean or sample an $\boldsymbol{\theta}$ from this expert and execute it on the environment.

Algorithm 2 Di-SkilL Training

```
Input: $\alpha$, $\beta$, N(max. iterations), K(num. experts),T(num. samples per expert)
Output: $\pi(\boldsymbol{\theta}|\mathbf{c})$
 1: for $k = 1$ to N do
 2:     $\mathbf{c} \sim p(\mathbf{c})$ (context batch by environment resetting)
 3:     for $o = 1$ to $K$ do
 4:         $\mathbf{c}_T \sim \pi(\mathbf{c}|o)$ (context batch from EBM)
 5:         $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}|\mathbf{c}_T, o)$
 6:         $\mathrm{R}(\mathbf{c}, \boldsymbol{\theta}) \leftarrow \mathrm{eval}(\boldsymbol{\theta}, \mathbf{c}_T)$
 7:         $\pi(\boldsymbol{\theta}|\mathbf{c}, o) \leftarrow$ Obj. 4.7
 8:         $\pi(\mathbf{c}|o) \leftarrow$ Obj. 4.8
 9:     end for
10: end for
```

Algorithm 3 Di-SkilL Inference

```
Input: $\pi(\boldsymbol{\theta}|\mathbf{c})$
 1: $\mathbf{c} \sim p(\mathbf{c})$ (observe contexts from environment)
 2: $o \sim \pi(o|\mathbf{c})$, where $\pi(o|\mathbf{c}) = \frac{\pi(\mathbf{c}|o)\pi(o)}{\pi(\mathbf{c})}$
 3: $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}|\mathbf{c}, o)$
 4: $\mathrm{R}(\mathbf{c}, \boldsymbol{\theta}) \leftarrow \mathrm{eval}(\boldsymbol{\theta}, \mathbf{c})$
```

B.4. Experimental Details

B.4.1. Environment Details

B.4.1.1. Table Tennis Easy

Environment. We use the same table tennis environment as presented in Otto et al. (2023), in which a 7 Degree of Freedom (DoF) robot has to return a ball to a desired ball landing position. The **context** is the four-dimensional space of the ball's initial landing position ($x \in [-1, -0.2]$, $y \in [-0.65, 0.65]$) on the robot's table side and the desired ball landing position ($x \in [-1.0, -0.2]$, $y \in [-0.6, 0.6]$) on the opponent's table side. The robot is controlled with torques on the joint level in each time step. The torques are generated by the tracking controller (PD-controller) that tracks the desired trajectory generated by the motion primitive. We consider three basis functions per joint resulting in a 21-dimensional parameter ($\boldsymbol{\theta}$) space. We additionally allow the agent to learn the trajectory length and the starting time step of the trajectory. Note that the starting point allows the agent to define when after the episode's start the generated desired trajectory should be tracked. Induced by the varying contexts, this is helpful to react to the varying time the served ball needs to reach a positional space that is convenient to hit the ball with the robot's racket. Overall the **parameter space** is 23 dimensional. The task is considered successful if the returned ball lands on the opponent's side of the table and within ≤ 0.2m to the goal location.

The **reward function** is unchanged from Otto et al. (2023) and is defined as

$$R_{task} = \begin{cases} 0, & \text{if cond. 1,} \\ f_2(\mathbf{p}_r, \mathbf{p}_b) & \text{if cond. 2,} \\ f_3(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) & \text{if cond. 3,} \\ f_4(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) & \text{if cond. 4,} \\ f_5(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) & \text{if cond. 5,} \end{cases}$$

where $\mathbf{p}_r$ is the executed trajectory position of the racket center, $\mathbf{p}_b$ is the executed position trajectory of the ball, $\mathbf{p}_l$ is the ball landing position, $\mathbf{p}_{goal}$ is the target position. The individual functions are defined as

$$\begin{aligned} f_2(\mathbf{p}_r, \mathbf{p}_b) &= 0.2 - 0.2g(\mathbf{p}_r, \mathbf{p}_b) \\ f_3(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) &= 3 - 2g(\mathbf{p}_r, \mathbf{p}_b) - h(\mathbf{p}_l, \mathbf{p}_{goal}) \\ f_4(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) &= 6 - 2g(\mathbf{p}_r, \mathbf{p}_b) - 4h(\mathbf{p}_l, \mathbf{p}_{goal}) \\ f_5(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) &= 7 - 2g(\mathbf{p}_r, \mathbf{p}_b) - 4h(\mathbf{p}_l, \mathbf{p}_{goal}), \end{aligned}$$

where $g(\mathbf{x}, \mathbf{y}) = \tanh(\min ||\mathbf{x} - \mathbf{y}||^2)$ and $h(\mathbf{x}, \mathbf{y}) = \tanh(||\mathbf{x} - \mathbf{y}||^2)$. The different conditions are

- cond. 1: the end of the episode is not reached,
- cond. 2: the end of the episode is reached,
- cond. 3: cond.2 is satisfied and the robot did hit the ball,
- cond. 4: cond.3 is satisfied and the returned ball lands on the table,
- cond. 5: cond.4 is satisfied and the landing position is at the opponent's side.

The episode ends when any of the following conditions are met

- the maximum horizon length is reached
- ball did land on the floor without hitting
- ball did land on the floor or table after hitting

The whole desired trajectory is obtained ahead of environment interaction, making use of this property we can collect some samples without physical simulation. The reward function based on this desired trajectory is defined as

$$r_{traj} = -\sum_{(i,j)} |\tau_{ij}^d| - |q_j^b|, \quad (i,j) \in \{(i,j) \mid |\tau_{ij}^d| > |q_j^b|\},$$

where τ^d is the desired trajectory, i is the time index, j is the joint index, q^b is the joint position upper bound. The desired trajectory is considered as invalid if $r_{traj} < 0$, an invalid trajectory will not be executed on the robot. The overall reward is defined as:

$$r = \begin{cases} r_{traj}, & r_{traj} < 0 \\ r_{task}, & \text{otherwise.} \end{cases}$$

SVSL. SVSL requires designing a guiding punishment term for context samples that are not in a valid region. For the four-dimensional context space in table tennis, this can be done using quadratic functions (as proposed in the original work Celik et al. (2021)):

$$R_c(\mathbf{c}) = -20 \cdot d_c^2,$$

where d_c^2 is the distance of the current context $\mathbf{c}$ to the valid context region.

SVSL Hyperparameters All hyperparameters are summarized in the Table B.1.

Hyperparameters are listed in the Table B.2.

B.4.1.2. Table Tennis Task Hard

Environment. We extend the table tennis environment described in Appendix B.4.1.1 by additionally including the ball's initial velocity in the context space making the task harder as the agent has to react to ranging velocities now. We define the initial velocity $v_x \in [1.5\frac{m}{s}, 4\frac{m}{s}]$. Note that every single constellation within the resulting context space is a valid context. However, there exist ball landing positions that can not be set along with a subset of the initial velocity range. This makes designing a guiding punishment term for SVSL especially difficult. We adopt the **parameter space** and the **reward function** as defined in the standard table tennis environment as described in Appendix B.4.1.1.

Hyperparameters are listed in the Table B.4.

B.4.2. Hopper Jump

Environment. We use the same hopper jump environment as presented in (Otto et al., 2023), in which the hopper Brockman et al. (2016) has to jump as high as possible and land at a specified position. The **context** is the four-dimensional space of the last three joints of the hopper and the goal landing position $[j_3, j_4, j_5, g]$, where the ranges are from $[-0.5, -0.2, 0, 0.3]$ to $[0.0, 0.0, 0.785, 1.35]$. The hopper is controlled the same as in (Brockman et al., 2016). Here, we consider three basis functions per joint and a goal basis resulting in a **parameter space** ($\boldsymbol{\theta}$) of 12 dimensions. The reward is non-markovian and is unchanged from (Otto et al., 2023).

In each time-step t the action cost

$$\tau_t = 10^{-3} \sum_i^K (a_t^i)^2,$$

is provided. The variable $K = 3$ corresponds to the number of degrees of freedom. At the end of the episode, a reward containing retrospective information about the maximum height in the z-direction of the center of mass achieved h_{max}, the goal landing position of the heel p_{goal}, the foot's heel position when having contact with the ground after jumping the first time $p_{\text{foot, contact}}$ is given. Additionally, per-time information such as $p_{\text{foot, t}}$ describing the position of the foot's heel in world-coordinates is given. The resulting reward function is

$$R_{tot} = -\sum_{t=0}^{T} \tau_t + R_{height} + R_{gdist} + R_{cdist} + R_{healthy},$$

where

$$\begin{aligned} R_{height} &= 10 h_{max}, \\ R_{gdist} &= ||p_{foot,T} - p_{goal}||_2, \\ R_{cdist} &= ||p_{foot,contact} - p_{goal}||_2, \\ R_{healthy} &= \begin{cases} 2 & \text{if } z_T \in [0.5, \infty] \text{and } \theta, \gamma, \phi \in [-\infty, \infty] \\ 0 & \text{else.} \end{cases} \end{aligned}$$

The healthy reward is the same as provided by (Brockman et al., 2016).

Hyperparameters are listed in the Table B.5.

B.4.2.1. Box Pushing with Obstacle Task

Environment. We increase the difficulty of the box pushing environment as presented in (Otto et al., 2023), by changing major parts of the context space. The goal of the box pushing task is to move a box to a specified goal location and orientation using the seven DoF Franka Emika Panda. The newly **context space** (compared to the original version in (Otto et al., 2023)) are described in the following. We increase the box' goal position range to $x_g \in [0.3, 0.6], y_g \in [-0.7, 0.45]$, and keep the goal orientation angle $\phi \in [0rad, 2\pi rad]$. Additionally, we include an obstacle between the initial box and the box's goal. The range of the obstacle position is $x_o \in [0.3, 0.6], y_o \in [-0.3, 0.15]$. Note that we guarantee a distance of at least 0.15m between the obstacle's position and the initial position as well as at least 0.15m between the obstacle's position and the box's goal position.

The robot is controlled via torques on the joint level. We use four basis functions per DoF, resulting in a **parameter space** of 28 dimensions. We consider an episode successful if the box's orientation around the z-axis error is smaller than 0.5 rad and the position error is smaller than 0.05m.

The **sparse-in-time reward function** is up to a scaling parameter the same as presented in (Otto et al., 2023). We describe the whole reward function in the following.

The box's distance to the goal position is

$$R_{\text{goal}} = ||\mathbf{p} - \mathbf{p}_{goal}||,$$

where $\mathbf{p}$ is the box position and $\mathbf{p}_{goal}$ is the goal position. The rotation distance is defined as

$$R_{\text{rotation}} = \frac{1}{\pi} \arccos |\mathbf{r} \cdot \mathbf{r}_{goal}|,$$

where $\mathbf{r}$ and $\mathbf{r}_{goal}$ are the box orientation and goal orientation quaternion respectively. The incentive to keep the rod within the box is defined as

$$R_{\text{rod}} = \text{clip}(||\mathbf{p} - \mathbf{h}_{pos}||, 0.05, 10),$$

where $\mathbf{h}_{pos}$ is the position of the rod tip. Similarly, to incentivize to maintain the rod in a desired rotation, the reward

$$R_{\text{rod\_rotation}} = \text{clip}(\frac{2}{\pi} \arccos |\mathbf{h}_{rot} \cdot \mathbf{h}_0|, 0.25, 2)$$

is defined, where $\mathbf{h}_{rot}$ and $\mathbf{h}_0 = (0, 1, 0, 0)$ are the current and desired rod orientation in quaternion respectively. To incentivize the robot to stay within the joint and velocity bounds, the error

$$\text{err}(\mathbf{q}, \mathbf{q}) = \sum_{i \in \{i || q_i| > |q_i^b|\}} (|q_i| - |q_i^b|) \qquad + \sum_{j \in \{j || \dot{q}_j| > |\dot{q}_j^b|\}} (|\dot{q}_j| - |\dot{q}_j^b|)$$

is used, where $\mathbf{q}$, $\mathbf{q}$, $\mathbf{q}^b$, and $\mathbf{q}^b$ are the robot's joint positions and velocities as well as their respective bounds. To learn low-energy motions, the per-time action (torque) cost

$$\tau_t = \sum_i^K (a_t^i)^2,$$

is used. The resulting temporal sparse reward is given as

$$R_{\text{tot}} = \begin{cases} -R_{\text{rod}} - R_{\text{rod\_rotation}} - 0.02\tau_t - \text{err}(\mathbf{q}, \mathbf{q}) & t < T, \\ -R_{\text{rod}} - R_{\text{rod\_rotation}} - 0.02\tau_t - \text{err}(\mathbf{q}, \mathbf{q}) & \\ -350R_{\text{goal}} - 200R_{\text{rotation}} & t = T, \end{cases}$$

where $T = 100$ is the horizon of the episode. The reward gives relevant information to solve the ask only in the last time step of the episode, which makes exploration hard.

Further Visualizations of learned skills. We show additional plots of the box's trajectories in the box pushing task in Fig. B.2.

Hyperparameters are listed in the Table B.6.

B.4.3. Extended 5-Link Reacher Task

Environment. In the 5-Link Reacher task, a 5-link planar robot has to reach a goal position with its tip. The reacher's initial position is straight to the right. This task is difficult to solve, as it introduces multi-modality in the behavior space. (Otto et al., 2023) avoided this multi-modality by constraining the y coordinate of the goal position to $y \geq 0$, i.e. the first two quadrants. We adopt the 5Link-Reacher task by increasing the context space to the full space, i.e. all four quadrants. We consider 5 basis functions per joint leading to a 25-dimensional **parameter space**. We consider the **sparse reward function** presented in (Otto et al., 2023) as

$$R_{\text{tot}} = \begin{cases} -\tau_t & t < T, \\ -\tau_t - 200R_{\text{goal}} - 10R_{\text{vel}} & t = T, \end{cases}$$

where

$$R_{\text{goal}} = ||\mathbf{p} - \mathbf{p}_{goal}||_2$$

and

$$\tau_t = \sum_i^K (a_t^i)^2.$$

The sparse reward only returns the task reward in the last time step T and additionally adds a velocity penalty $R_{\text{vel}} = \sum_i^K (\dot{q}_T^i)^2$. The joint velocities are denoted as$\mathbf{q}$. This velocity penalty avoids overshooting in the last time step.

Hyperparameters are listed in the Table B.3.

B.4.4. Robot Mini Golf Task

Environment. In the robot mini golf task the agent needs to hit a ball while avoiding the two obstacles, such that it passes the tight goal to achieve a bonus. The **context space** consists of the ball's initial x-position $x_{ball} \in [0.25m, 0.6m]$, the XY positions of the green obstacle $x_{obs} \in [0.3, 0.6]$ and $y_{obs} \in [-0.5, -0.1]$ and the x positions of the goal $x_{ball} \in [0.25, 0.6]$. The **parameter space** is 29 dimensional resulting from the 4 basis functions per joint and an additional duration parameter which allows the robot to learn the duration of the trajectory. The robot starts always at the same position. The **reward** function consists of three stages:

$$R_{task} = \begin{cases} -0.0005 \cdot \tau_t & \text{if cond. 1,} \\ 0.2 - 0.2\tanh\left(\min ||\mathbf{p}_r - \mathbf{p}_b||\right) & \text{if cond. 2,} \\ 2 - 2\tanh\left(\min ||\mathbf{p}_b - \mathbf{p}_g||\right) & \\ -\tanh\left(||\mathbf{p}_{b,y} - \mathbf{p}_{thresh,y}||\right) & \text{if cond. 3,} \\ 6 & \text{if cond. 3,} \end{cases}$$

where the individual conditions are

- cond. 1: the end of the episode is not reached,
- cond. 2: the end of the episode is reached and the robot did not hit the ball,
- cond. 3: the end of the episode is reached and the robot has hit the ball, but the ball didn't pass the goal
- cond. 4: the end of the episode is reached, robot has hit the ball and the ball has passed the goal for at least 0.75m

The episode ends when the maximum horizon length $T = 100$ is achieved. We again make use of the advantage that we obtain the whole desired trajectory ahead of the environment interaction, such that we can collect some samples without physical simulation. The reward function based on this desired trajectory is defined as

$$r_{traj} = \sum_{(i,j)} |\tau_{ij}^d| - |q_j^b|, \quad (i,j) \in \{(i,j) \mid |\tau_{ij}^d| > |q_j^b|\},$$

where τ^d is the desired trajectory, i is the time index, j is the joint index, q^b is the joint position upper bound. The desired trajectory is considered as invalid if $r_{traj} < 0$, an invalid trajectory will not be executed on the robot. Additionally, we provide a punishment, if the agent samples invalid duration times

$$r_{dur} = -3\left(\max(0, t_d - td, max) + \max(0, t_{d,mint} - t_d)\right),$$

where $t_{d,max} = 1.7s$, $t_{d,min} = 0.45s$ and t_d is the duration in seconds chosen by the agent. The overall reward is defined as:

$$r = \begin{cases} r_{traj}, -20(r_{traj} + r_{dur}) - 5 & \text{if invalid duration,} \\ & \text{or trajectory} \\ r_{task}, & \text{otherwise.} \end{cases}$$

Hyperparameters are listed in the Table B.7.

B.5. Additional Evaluations

We provide additional diverse skills to the Box Pushing Obstacle task in Fig. B.2. In Fig. B.4 we provide additional diverse strikes to fixed ball's desired landing positions on the **TT-H** task.

Furthermore, we analyze Di-SkilL's performance on the hopper jump task in more detail. In Fig. B.3a we observe that the mean return is on par with BBRL, similar to the achieved goal distance in Fig. B.3c. However, there is a small gap in the max height, where BBRL jumps slightly higher (see Fig. B.3b. Given that the mean return is on par, one would expect that the maximum jump height is on par as well. However, Di-SkilL optimizes the remaining terms in the objective of the hopper jump task such as the healthy reward (see Appendix B.4), which explains this gap.

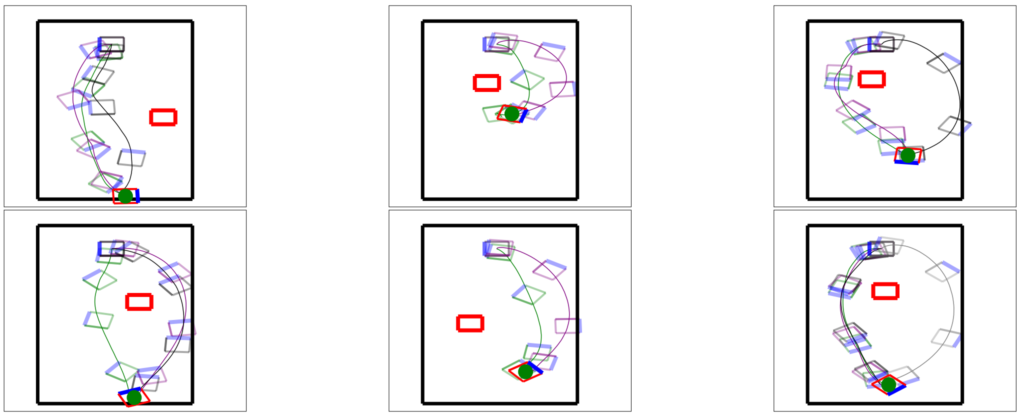
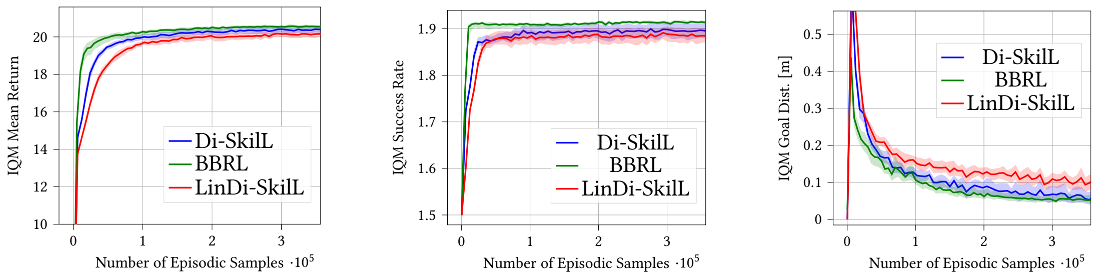


Figure B.2.: Additional Diverse Skills for the Box Push Obstacle Task learned by Di-SkilL. We fix the contexts and sample experts which we subsequently execute. This leads to diverse behaviors in the motion primitive parameter spaceθ which leads to different trajectories of the pushed box on the table.



(a) IQM Mean Return **HJ** **(b)** IQM Max. Height Jump **HJ** **(c)** IQM Goal Distance **HJ**

Figure B.3.: Additional Analysis of the Hopper Jump (HJ) task.

B.6. Hyperparameters

We list the hyperparameters for all algorithms on all environments in the following tables.

add expert every iteration	1000
fine tune all experts every iteration	50
number expert adds	1
number initial experts	1
number total experts	20
number traj. samples per expert per iteration	200
α	0.0001
β	0.5
expert KL-bound	0.01
context KL-bound	0.01

Table B.1.: Hyperparameters for SVSL on TT

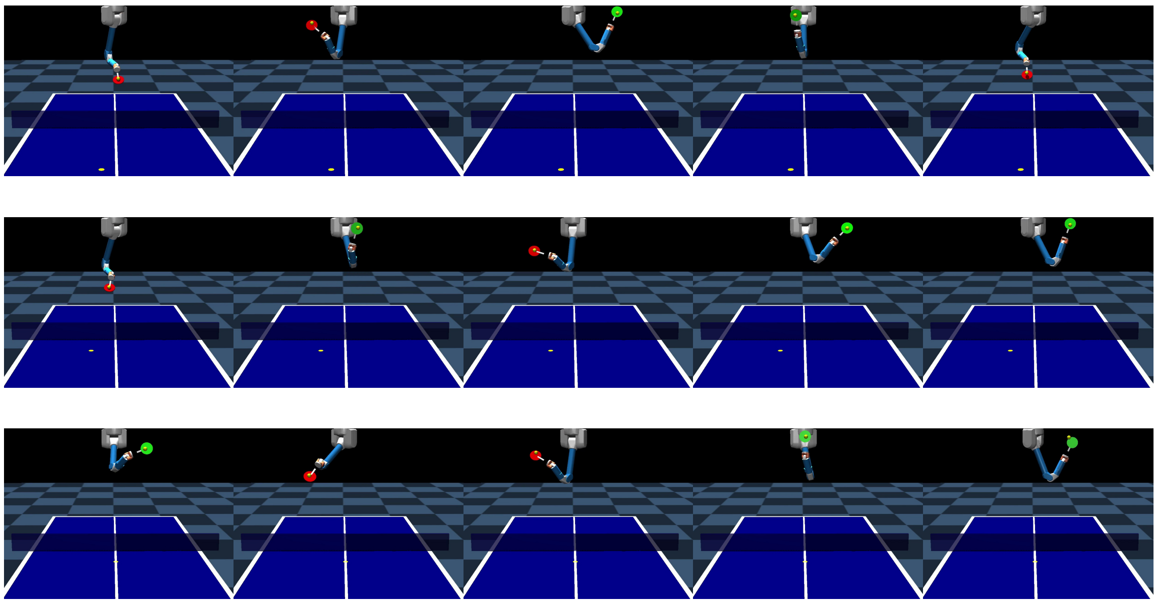


Figure B.4.: Di-SkilL's **Diverse Skills** for the **TT-H** task. We fixed the ball's desired landing position and varied the serving landing position and the ball's initial velocity. Di-SkilL can return the ball in different striking types. Note that each row represents a different desired ball landing position.

	Di-SkilL	**BBRL**
critic activation	tanh	tanh
hidden sizes critic	[8,8]	[32, 32]
initialization	orthogonal	orthogonal
lr critic	0.0003	0.0003
optimizer critic	adam	adam
ciritc epochs	100	100
activation context distribution	tanh	–
epochs context distribution	100	–
hidden sizes context distr	[16,16]	–
initialization	orthogonal	–
lr context distribution	0.0001	–
optimizer context distr	adam	–
batch size per expert	50	209
number samples from environment distribution	5000	–
number samples per expert	50	209
normalize advantages	True	True
expert activateion	tanh	tanh
epochs	100	100
hidden sizes expert	[64]	[32]
lr policy	0.0003	0.0003
covariance type	full	full
alpha	0.001	–
beta	4	–
number experts	5	–
covariance bound	0.005	0.001
mean bound	0.05	0.05
projection type	KL	KL
trust region coefficient	100	25

Table B.2.: Hyperparameters for Di-SkilL and BBRL on TT.

	Di-SkilL	**BBRL**	**LinDi-SkilL**	**PPO**
critic activation	tanh	tanh	tanh	tanh
hidden sizes critic	[32,32]	[32, 32]	[32, 32]	[32, 32]
initialization	orthogonal	orthogonal	orthogonal	orthogonal
lr critic	0.0003	0.0003	0.0003	0.0003
optimizer critic	adam	adam	adam	adam
ciritc epochs	100	100	100	10
activation context distr.	tanh	–	tanh	–
epochs context distr.	100	–	100	–
hidden sizes context distr.	[16,16]	–	[16, 16]	–
initialization	orthogonal	–	orthogonal	–
lr context distr.	0.0001	–	0.0001	–
optimizer context distr.	adam	–	adam	–
batch size per expert	25	240	25	512
nr. mini batches	–	–	–	32
nr. samples from env. distr.	5000	–	5000	–
number samples per expert	25	240	25	16384
normalize advantages	True	True	True	True
expert activateion	tanh	tanh	–	tanh
epochs	100	100	100	10
hidden sizes expert	[32,32]	[64,64]	–	[32, 32]
lr policy	0.0003	0.0003	0.0003	0.0003
covariance type	full	full	full	diagonal
alpha	0.01	–	0.01	–
beta	8	–	8	–
number experts	10	–	10	–
covariance bound	0.001	0.005	0.0005	–
mean bound	0.05	0.05	0.05	–
projection type	KL	KL	KL	–
trust region coefficient	100	25	100	–
discount factor	1	1	1	1

Table B.3.: Hyperparameters for Di-SkilL, BBRL, LinDi-SkilL, and PPO on 5LR. We used all code-level optimization (Engstrom et al., 2020) needed for PPO. The implementation is based on the source code from Otto et al. (2021).

	Di-SkilL	**BBRL**	**LinDi-SkilL**
critic activation	tanh	tanh	tanh
hidden sizes critic	[8,8]	[32, 32]	[8,8]
initialization	orthogonal	orthogonal	orthogonal
lr critic	0.0003	0.0003	0.0003
optimizer critic	adam	adam	adam
ciritc epochs	100	100	100
activation context distr.	tanh	–	tanh
epochs context distr.	100	–	100
hidden sizes context distr.	[16,16]	–	[16, 16
initialization	orthogonal	–	orthogonal
lr context distr.	0.0001	–	0.0001
optimizer context distr.	adam	–	adam
batch size per expert	50	209	50
nr. samples from env. distr.	5000	–	5000
number samples per expert	50	209	50
normalize advantages	True	True	True
expert activateion	tanh	tanh	–
epochs	100	100	100
hidden sizes expert	[128]	[32,32]	–
lr policy	0.0003	0.0003	0.0003
covariance type	full	full	full
alpha	0.001	–	0.001
beta	0.5	–	0.5
number experts	10	–	10
covariance bound	0.005	0.0005	0.001
mean bound	0.05	0.05	0.05
projection type	KL	KL	KL
trust region coefficient	100	25	100

Table B.4.: Hyperparameters for Di-SkilL, BBRL, and LinDi-SkilL for the Hard Table Tennis Task (TT-H).

	Di-SkilL	**BBRL**	**LinDi-SkilL**
critic activation	tanh	tanh	tanh
hidden sizes critic	[64,64]	[64, 64]	[64,64]
initialization	orthogonal	orthogonal	orthogonal
lr critic	0.0001	0.0001	0.0001
optimizer critic	adam	adam	adam
ciritc epochs	100	100	100
activation context distr.	tanh	–	tanh
epochs context distr.	100	–	100
hidden sizes context distr.	[16,16]	–	[16, 16]
initialization	orthogonal	–	orthogonal
lr context distr.	0.0001	–	0.0001
optimizer context distr.	adam	–	adam
batch size per expert	80	200	80
number samples from environment distr.	1000	–	1000
number samples per expert	80	200	80
normalize advantages	True	True	True
expert activateion	tanh	tanh	–
epochs	100	100	100
hidden sizes expert	[32, 32]	[32,32]	–
lr policy	0.0003	0.0003	0.0003
covariance type	full	full	full
alpha	0.01	–	0.01
beta	8	–	8
number experts	3	–	3
covariance bound	0.005	0.05	0.005
mean bound	0.05	0.1	0.05
projection type	KL	KL	KL
trust region coefficient	100	25	100

Table B.5.: Hyperparameters for Di-SkilL, BBRL, and LinDi-SkilL for the Hopper Jump Task (HJ).

	Di-SkilL	**BBRL**	**LinDi-SkilL**	**PPO**
critic activation	tanh	tanh	tanh	tanh
hidden sizes critic	[32,32]	[32, 32]	[32, 32]	[256, 256]
initialization	orthogonal	orthogonal	orthogonal	orthogonal
lr critic	0.0003	0.0003	0.0003	0.0001
optimizer critic	adam	adam	adam	adam
ciritc epochs	100	100	100	10
activation context distr.	tanh	–	tanh	–
epochs context distr.	100	–	100	–
hidden sizes context distr.	[16,16]	–	[16, 16]	–
initialization	orthogonal	–	orthogonal	–
lr context distr.	0.0001	–	0.0001	–
optimizer context distr.	adam	–	adam	–
batch size per expert	50	500	50	410
nr. mini batches	–	–	–	40
nr. samples from env. distr.	5000	–	5000	–
number samples per expert	50	500	50	16384
normalize advantages	True	True	True	True
expert activateion	tanh	tanh	–	tanh
epochs	100	100	100	10
hidden sizes expert	[64,64]	[64,64]	–	[256, 256]
lr policy	0.0003	0.0003	0.0003	0.0001
covariance type	full	full	full	diagonal
alpha	0.01	–	0.0001	–
beta	64	–	64	–
number experts	10	–	10	–
covariance bound	0.005	0.0005	0.001	–
mean bound	0.05	0.05	0.05	–
projection type	KL	KL	KL	–
trust region coefficient	100	25	100	–
discount factor	1	1	1	1

Table B.6.: Hyperparameters for Di-SkilL, BBRL, LinDi-SkilL, and PPO for Box Pushing Obstacle task (BPO). We used all code-level optimization (Engstrom et al., 2020) needed for PPO. The implementation is based on the source code from Otto et al. (2021).

	Di-SkilL	**BBRL**	**LinDi-SkilL**
critic activation	tanh	tanh	tanh
hidden sizes critic	[32,32]	[32, 32]	[32, 32]
initialization	orthogonal	orthogonal	orthogonal
lr critic	0.0003	0.0003	0.0003
optimizer critic	adam	adam	adam
ciritc epochs	100	100	100
activation context distr.	tanh	–	tanh
epochs context distr.	100	–	100
hidden sizes context distr.	[16,16]	–	[16, 16]
initialization	orthogonal	–	orthogonal
lr context distr.	0.0001	–	0.0001
optimizer context distr	adam	–	adam
batch size per expert	50	500	50
nr. samples from env. distr.	5000	–	5000
number samples per expert	50	500	50
normalize advantages	True	True	True
expert activateion	tanh	tanh	–
epochs	100	100	100
hidden sizes expert	[64,64]	[128,128]	–
lr policy	0.0003	0.0003	0.0003
covariance type	full	full	full
alpha	0.0001	–	0.0001
beta	1	–	1
number experts	10	–	10
covariance bound	0.005	0.001	0.001
mean bound	0.05	0.05	0.01
projection type	KL	KL	KL
trust region coefficient	100	25	100

Table B.7.: Hyperparameters for Di-SkilL, BBRL, and LinDi-SkilL for the mini golf task.

C. Appendix for Chapter 5

C.1. Derivations

Lower-Bound Derivation. $\mathcal{H}(\pi_0(\mathrm{a}_0|\mathrm{s})) \geq \ell_{\overleftarrow{\pi}}(\mathrm{a}^0, \mathrm{s})$

$$\begin{aligned}
\mathcal{H}(\pi_0(\mathrm{a}_0|\mathrm{s})) &= -\mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\overleftarrow{\pi}_{0:N}(\mathrm{a}^{0:N}|\mathrm{s})}{\overleftarrow{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0)}\right] &\text{(C.1)}\\
&= -\mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\overleftarrow{\pi}_{0:N}(\mathrm{a}^{0:N}|\mathrm{s})\vec{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0)}{\overleftarrow{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0)\vec{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0)}\right]\\
&= \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\vec{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0)}{\overleftarrow{\pi}_{0:N}(\mathrm{a}^{0:N}|\mathrm{s})}\right] + \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\overleftarrow{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0)}{\vec{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0)}\right]\\
&= \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\vec{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0)}{\overleftarrow{\pi}_{0:N}(\mathrm{a}^{0:N}|\mathrm{s})}\right] \dots\\
&\dots + \mathbb{E}_{\pi_0}\left[D_{\mathrm{KL}}\left(\overleftarrow{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0) \,\|\, \vec{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0)\right)\right]\\
&\geq \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\vec{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0)}{\overleftarrow{\pi}_{0:N}(\mathrm{a}^{0:N}|\mathrm{s})}\right], &\text{(C.2)}
\end{aligned}$$

where we have used the relation

$$\pi_0(\mathrm{a}_0|\mathrm{s}) = \frac{\overleftarrow{\pi}_{0:N}(\mathrm{a}^{0:N}|\mathrm{s})}{\overleftarrow{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{s}, \mathrm{a}^0)} \tag{C.3}$$

and the fact that the KL divergence is always non-negative

Approximate Inference Formulation. Recall the definition of the Q-function

$$Q^{\overleftarrow{\pi}}(\mathrm{s}_t, \mathrm{a}_t^0) = r_t + \sum_{l=1} \gamma^l \mathbb{E}_{\rho_\pi}\left[r_{t+l} + \ell_{\overleftarrow{\pi}}(\mathrm{a}_{t+l}^0, \mathrm{s}_{t+l})\right] \tag{C.4}$$

and

$$\ell_{\overleftarrow{\pi}}(\mathrm{a}^0, \mathrm{s}) = \mathbb{E}_{\overleftarrow{\pi}_{0:N}}\left[\log \frac{\vec{\pi}_{1:N|0}(\mathrm{a}^{1:N}|\mathrm{a}^0, \mathrm{s})}{\overleftarrow{\pi}_{0:N}(\mathrm{a}^{0:N}|\mathrm{s})}\right]. \tag{C.5}$$

We start reformulating the objective

$$J(\overleftarrow{\pi}) \geq \bar{J}(\overleftarrow{\pi}) = \sum_{t=l}^{\infty} \gamma^{t-l} \mathbb{E}_{\rho_\pi} \left[r_t + \ell_{\overleftarrow{\pi}}(\mathbf{a}_t^0, \mathbf{s}_t) \right] \tag{C.6}$$

$$= \sum_{t=l+1}^{\infty} \gamma^{t-l} \mathbb{E}_{\rho_\pi} \left[r_t + \ell_{\overleftarrow{\pi}}(\mathbf{a}_t^0, \mathbf{s}_t) \right] + \mathbb{E}_{\rho^\pi} \left[r_l + \ell_{\overleftarrow{\pi}}(\mathbf{a}_l^0, \mathbf{s}_l) \right] \tag{C.7}$$

$$= \mathbb{E}_{\rho^\pi} \left[Q^{\overleftarrow{\pi}}(\mathbf{s}_t, \mathbf{a}_t^0) \right] + \mathbb{E}_{\rho^\pi} \left[\ell_{\overleftarrow{\pi}}(\mathbf{a}_l^0, \mathbf{s}_l) \right] \tag{C.8}$$

$$= \mathbb{E}_{\rho^\pi} \left[Q^{\overleftarrow{\pi}}(\mathbf{s}_t, \mathbf{a}_t^0) + \ell_{\overleftarrow{\pi}}(\mathbf{a}_l^0, \mathbf{s}_l) \right] \tag{C.9}$$

$$= \mathbb{E}_{\rho^\pi, \overleftarrow{\pi}_{0:N}} \left[Q^{\overleftarrow{\pi}}(\mathbf{s}_t, \mathbf{a}_t^0) + \log \frac{\vec{\pi}_{1:N|0}(\mathbf{a}^{1:N}|\mathbf{a}^0, \mathbf{s})}{\overleftarrow{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s})} \right] \tag{C.10}$$

$$= -\mathbb{E}_{\rho^\pi} \left[D_{\mathrm{KL}} \left(\overleftarrow{\pi}(\mathbf{a}^{0:N}|\mathbf{s}) \| \, \vec{\pi}(\mathbf{a}^{0:N}|\mathbf{s}) \right) - \log \mathcal{Z}^{\overleftarrow{\pi}}(\mathbf{s}) \right], \tag{C.11}$$

where we used

$$\vec{\pi}_0(\mathbf{a}^0|\mathbf{s}) = \frac{\exp Q^{\overleftarrow{\pi}}(\mathbf{s}, \mathbf{a}^0)}{\mathcal{Z}^{\overleftarrow{\pi}}(\mathbf{s})} \tag{C.12}$$

in the last step. When minimizing, the negative sign in front of the KL vanishes. Please note that the expectation over the marginal state distribution was ommited in the main text to avoid cluttered notation.

C.2. Proofs

Proof of Proposition 5.4.1 (Policy Evaluation). Let's define the entropy-augmented reward of a diffusion policy as

$$r_{\overleftarrow{\pi}}(\mathbf{s}_t, \mathbf{a}_t^0) \triangleq r_t(\mathbf{s}_t, \mathbf{a}_t^0) + \mathbb{E}_{\overleftarrow{\pi}_{0:N}} \left[\log \frac{\vec{\pi}_{1:N|0}(\mathbf{a}^{1:N}|\mathbf{a}^0, \mathbf{s})}{\overleftarrow{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s})} \right] \tag{C.13}$$

and the update rule for the Q-function as

$$Q(\mathbf{s}_t, \mathbf{a}_t^0) \leftarrow r_{\overleftarrow{\pi}}(\mathbf{s}_t, \mathbf{a}_t^0) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p, \mathbf{a}_{t+1}^0 \sim \overleftarrow{\pi}} \left[Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}^0) \right]. \tag{C.14}$$

This formulation allows us to apply the standard convergence results for policy evaluation as stated in Sutton and Barto (2018).

Proof of Proposition 5.4.2 (Policy Improvement). It holds that

$$\overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s) = \frac{\exp Q^{\pi^{(i)}}(\mathbf{s}, \mathbf{a}^N)}{Z^{\pi^{(i)}}(s)} \vec{\pi}^{(i)}(\mathbf{a}^{0:N-1}|\mathbf{a}^N, \mathbf{s}). \tag{C.15}$$

Moreover, using the fact that the KL divergence is always non-negative, we obtain

$$0 = D_{\mathrm{KL}} \left(\overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s) \| \overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s) \right) \leq D_{\mathrm{KL}} \left(\overleftarrow{\pi}^{(i)}(\mathbf{a}^{0:N}|s) \| \overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s) \right). \tag{C.16}$$

Rewriting the KL divergences yields

$$\mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[\log \frac{\overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s)}{\overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s)}\right] \leq \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[\log \frac{\overleftarrow{\pi}^{(i)}(\mathbf{a}^{0:N}|s)}{\overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s)}\right] \tag{C.17}$$

$$\begin{aligned}
\iff \quad & \mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[\log \overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s)\right] - \mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[\log \overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s)\right] \\
& \leq \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[\log \overleftarrow{\pi}^{(i)}(\mathbf{a}^{0:N}|s)\right] - \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[\log \overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s)\right]
\end{aligned} \tag{C.18}$$

$$\begin{aligned}
\iff \quad & \mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[\log \overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s)\right] - \mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[\log \frac{\exp Q^{\pi^{(i)}}(\mathbf{s},\mathbf{a}^N)}{Z^{\pi^{(i)}}(\mathbf{s})}\overrightarrow{\pi}^{(i)}(\mathbf{a}^{0:N-1}|\mathbf{a}^N,\mathbf{s})\right]
\end{aligned} \tag{C.19}$$

$$\leq \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[\log \overleftarrow{\pi}^{(i)}(\mathbf{a}^{0:N}|s)\right] - \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[\log \frac{\exp Q^{\pi^{(i)}}(\mathbf{s},\mathbf{a}^N)}{Z^{\pi^{(i)}}(\mathbf{s})}\overrightarrow{\pi}^{(i)}(\mathbf{a}^{0:N-1}|\mathbf{a}^N,\mathbf{s})\right]$$

$$\begin{aligned}
\iff \quad & \mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[Q^{\pi^{(i)}}(\mathbf{s},\mathbf{a}^N)\right] + \mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[\log \frac{\overrightarrow{\pi}^{(i)}(\mathbf{a}^{0:N-1}|\mathbf{a}^N,\mathbf{s})}{\overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s)}\right] \\
& \geq \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[Q^{\pi^{(i)}}(\mathbf{s},\mathbf{a}^N)\right] + \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[\log \frac{\overrightarrow{\pi}^{(i)}(\mathbf{a}^{0:N-1}|\mathbf{a}^N,\mathbf{s})}{\overleftarrow{\pi}^{(i)}(\mathbf{a}^{0:N}|s)}\right].
\end{aligned} \tag{C.20}$$

To keep the notation uncluttered we use

$$d^{(i+1)}(\mathbf{s},\mathbf{a}^N) = \mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[\log \frac{\overrightarrow{\pi}^{(i)}(\mathbf{a}^{0:N-1}|\mathbf{a}^N,\mathbf{s})}{\overleftarrow{\pi}^{(i+1)}(\mathbf{a}^{0:N}|s)}\right] \quad \text{and} \tag{C.21}$$

$$d^{(i)}(\mathbf{s},\mathbf{a}^N) = \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[\log \frac{\overrightarrow{\pi}^{(i)}(\mathbf{a}^{0:N-1}|\mathbf{a}^N,\mathbf{s})}{\overleftarrow{\pi}^{(i)}(\mathbf{a}^{0:N}|s)}\right] \tag{C.22}$$

$$Q^{\pi^{(i)}}(\mathbf{s},\mathbf{a}^N) = r_0 + \mathbb{E}\left[\gamma\left(d^{(i)}(\mathbf{s}_1,\mathbf{a}_1^N) + \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[Q^{\pi^{(i)}}(\mathbf{s}_1,\mathbf{a}_1^N)\right]\right)\right] \tag{C.23}$$

$$\leq r_0 + \mathbb{E}\left[\gamma\left(d^{(i+1)}(\mathbf{s}_1,\mathbf{a}_1^N) + \mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[Q^{\pi^{(i)}}(\mathbf{s}_1,\mathbf{a}_1^N)\right]\right)\right] \tag{C.24}$$

$$= r_0 + \mathbb{E}\left[\gamma\left(d^{(i+1)}(\mathbf{s}_1,\mathbf{a}_1^N) + r_1\right) + \gamma^2\left(d^{(i)}(\mathbf{s}_2,\mathbf{a}_2^N) + \mathbb{E}_{\overleftarrow{\pi}^{(i)}}\left[Q^{\pi^{(i)}}(\mathbf{s}_2,\mathbf{a}_2^N)\right]\right)\right] \tag{C.25}$$

$$\leq r_0 + \mathbb{E}\left[\gamma\left(d^{(i+1)}(\mathbf{s}_1,\mathbf{a}_1^N) + r_1\right) + \gamma^2\left(d^{(i+1)}(\mathbf{s}_2,\mathbf{a}_2^N) + \mathbb{E}_{\overleftarrow{\pi}^{(i+1)}}\left[Q^{\pi^{(i)}}(\mathbf{s}_2,\mathbf{a}_2^N)\right]\right)\right] \tag{C.26}$$

$$\vdots \tag{C.27}$$

$$\leq r_0 + \mathbb{E}\left[\sum_{t=1}^{\infty}\gamma^t\left(d^{(i+1)}(\mathbf{s}_t,\mathbf{a}_t^N) + r_t\right)\right] = Q^{\pi^{(i+1)}}(\mathbf{s},\mathbf{a}^N) \tag{C.28}$$

Since Q improves monotonically, we eventually reach a fixed point $Q^{(i+1)} = Q^{(i)} = Q^*$.

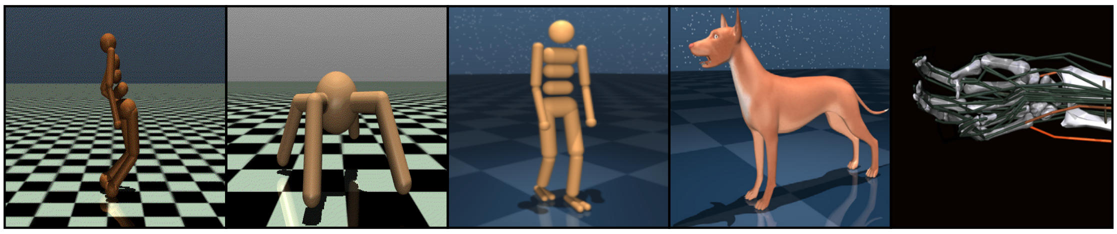


Figure C.1.: Considered environments. The *Humanoid-v3* and the *Ant-v3* are environments from the mujoco gym benchmark (Brockman et al., 2016). The three environments*humanoid-run*,*humanoid-walk* and *humanoid-stand* are from the deepmind control suite (DMC) benchmark (Tunyasuvunakool et al., 2020). The dog environments consist of *dog-run*, *dog-walk*, *dog-stand*, *dog-trot* and are also from the DMC sutie benchmark. Finally, the myo suite hand environments *object-hold-hard*,*reach-hard*, *key-turn-hard*, *pen-twirl-hard* are from the myo suite (Caggiano et al., 2022).

Proof of Proposition 5.4.3 (Policy Iteration). From Proposition 5.4.2 it follows that $Q^{\overleftarrow{\pi}^{i+1}}(\mathbf{s}, \mathbf{a}) \geq Q^{\overleftarrow{\pi}^{i}}(\mathbf{s}, \mathbf{a})$. If for $\lim_{k\to\infty} \overleftarrow{\pi}^{k} = \overleftarrow{\pi}^{*}$, then it must hold that $Q^{\overleftarrow{\pi}^{*}(\mathbf{s},\mathbf{a})} \geq Q^{\overleftarrow{\pi}}(\mathbf{s}, \mathbf{a})$ for all $\overleftarrow{\pi} \in \overleftarrow{\Pi}$ which is guaranteed by Proposition 5.4.2.

C.3. Environments

All environments are visualized in Fig. C.1. We consider the *Ant-v3* and the *Humanoid-v3* environments from mujoco gym (Brockman et al., 2016). The *humanoid-stand*, *humanoid-walk* , *humanoid-run*, *dog-stand*, *dog-walk*, *dog-trot* and *dog-run* environments from the deepmind control suite (DMC) (Tunyasuvunakool et al., 2020). The hand environments from myo suite are the *object-hold-random*,*reach-random*, *key-turn-random* and *pen-twirl-random* environments (Caggiano et al., 2022). The action and observation spaces of the respective environments are shown in Table C.1.

C.4. Implementation Details

We consider a score network with *3* layers and a *256* dimensional hidden layer with gelu activation function. We use Fourier features to encode the timestep and scale the embedding using a feed-forward neural network with two layers, with a hidden dimension of 256. For the diffusion coefficient, we use a cosine schedule and additionally optimize a scaling parameter for the diffusion coefficient per dimension end-to-end (i.e,. we learn the parameter β (please see Appendix C.6).

We employ distributional Q following Bellemare et al. (2017), where the Q-model outputs probabilities q over b bins. Using the bellman backup operator for diffusion models from Eq. 5.21 and the bin values b we follow Bellemare et al. (2017) and calculate the target probabilities q_{target}. Using the entropy-regularized cross-entropy loss $\mathcal{L}(\boldsymbol{\phi}) = -\sum q_{target} \log q_{\boldsymbol{\phi}} - 0.005 \sum q_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}$ we update the parameters $\boldsymbol{\phi}$ of the Q-function. Please

Training Environment	**Observation Space Dim.**	**Action Space Dim.**
Ant-v3	111	8
Humanoid-v3	376	17
dog-run	223	38
dog-walk	223	38
dog-trot	223	38
dog-stand	223	38
humanoid-run	67	24
humanoid-walk	67	24
humanoid-stand	67	24
myoHandObjHoldRandom-v0	91	39
myoHandReachRandom-v0	115	39
myoHandKeyTurnRandom-v0	93	39
myoHandPenTwirlRandom-v0	83	39

Table C.1.: Observation and Action Space Dimensions for Various Training Environments.

note that the entropy regularization was not proposed in the original paper from Bellemare et al. (2017), however, we noticed that a small regularization helps improve the performance in the early learning stages but does not change the asymptotic performance. Additionally, we follow Nauman et al. (2024) and use the *mean* of the two Q-values instead of the *min* as it has usually been used in RL so far.

The expected Q-values for updating the actor can be easily calculated using the expectation $Q(\mathbf{s}, \mathbf{a}_t^0) = \sum_i q_i(\mathbf{s}_t, \mathbf{a}_t^0) b_i$

Action Scaling. Practical applications have a bounded action space that can usually be scaled to a fixed range. However, the action range of the diffusion policy $\overleftarrow{\pi}$ is unbounded. Therefore, we follow recent works (Haarnoja et al., 2018b) and propose applying the change of variables with a *tanh* squashing function at the last diffusion step $n = 0$. For the backward process $\overleftarrow{q}_{0:N}(u^{0:N}|\mathbf{s})$ with unbounded action space $u \in \mathbb{R}^D$ we can squash the action $\mathbf{a}^0 = \tanh u^0$ such that $\mathbf{a}^0 \in (-1, 1)$ and its density is given by

$$\overleftarrow{\pi}_{0:N}(\mathbf{a}^{0:N}|\mathbf{s}) = \overleftarrow{q}_{0:N}(u^{0:N}|\mathbf{s}) \det \left| \left(\frac{\mathrm{d}\mathbf{a}^0}{\mathrm{d}u^0} \right) \right|^{-1}, \tag{C.29}$$

with the corresponding log-likelihood

$$\log \overleftarrow{\pi}_{0:N}(a^{0:N}|\mathbf{s}) = \log \overleftarrow{q}_N(u^N) + \sum_{n=1}^{N} \log \overleftarrow{q}_{n-1}(u^{n-1}|u^n, \mathbf{s}) - \sum_{i=1}^{D} \log \left(1 - \tanh^2 \left(u_i^N\right)\right). \tag{C.30}$$

This means that the Gaussian kernels of the diffusion chain have the same log probabilities except for the correction term of the last step at $n = 0$.

Algorithm 4 DIME: Diffusion-Based Maximum Entropy Reinforcement Learning

```
Input: Initialized parameters $\theta, \phi, \alpha$, learningrates $\lambda$
 1: for $k = 1$ to M do
 2:     if k % UTD then
 3:         $\mathbf{a}_t^{0:T} \sim \pi_{0:N}^{\theta}(\mathbf{a}^{0:N}|\mathbf{s}_t)$
 4:         $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{a}_t^0, \mathbf{s}_t)$
 5:         $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{\mathbf{s}_t, \mathbf{a}_t^0, r_t, \mathbf{s}_{t+1}\}$
 6:     end if
 7:     $\phi \leftarrow \phi - \lambda_\phi \nabla_\phi J_Q(\phi)$ (Eq. 5.24)
 8:     if k % POLICYDELAY then
 9:         $\theta \leftarrow \theta - \lambda_\theta \nabla_\theta \mathcal{L}(\theta)$ (Eq. 5.25)
10:         $\alpha \leftarrow \alpha - \lambda_\alpha J(\alpha)$ (Eq. 5.28)
11:     end if
12: end for
```

Algorithm 4 shows the learning procedure of DIME. Note that policy delay refers to the number of delayed updates of the policy compared to the critic. UTD is the update to data ratio.

C.5. List of Hyperparameters

Please note that we have used the official code releases of the respective baseline methods for training. For BRO and BRO Fast we used the provided learning curves

DIME. For DIME, we use distributional Q, where the maximum and minimum values for the bins have been chosen per benchmark suite. We have used $v_{min} = -1600$ and $v_{max} = 1600$ for the gym environments, $v_{min} = -200$ and $v_{max} = 200$ for the DMC suite and $v_{min} = -3600$ and $v_{max} = 3600$ for the myo suite.

QSM. In certain environments, we observed that QSM with default hyperparameters performed poorly, particularly in several DMC tasks and the Gym Ant-v3 tasks. To address this, we fine-tuned the hyperparameters for QSM in each of these underperforming tasks. For the DMC tasks, we found that QSM often requires an α value—representing the alignment factor between the score and the Q-function (Psenka et al., 2024)—in the range of 100-200, rather than the default value of 50 reported in QSM's original implementation. In the Ant-v3 task, we determined that α needs to be set to 1. In the original implementation, the number of diffusion steps is set to be 5, however, we found using more steps, such as 10 and 15, can significantly improve the performance in these under performed tasks.

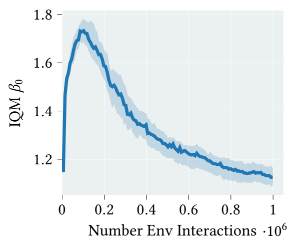
CrossQ. We used the hyperparameters from the original paper (Bhatt et al., 2024) for the gym benchmark suite. However, we used a different set of hyperparameters for the DMC and MYO suites for improved performance. More precisely, we increased the policy size to *3 layers* with *256 hidden size*. Additionally, we reduced the learning rate to *7e-4*.

	DIME	QSM	Diff-QL	Consistency-AC	DIPO	DACER	QVPO
Update-to-data ratio	2	1	1	1	1	1	1
Discount	0.99	0.99	0.99	0.99	0.99	0.99	0.99
batch size	256	256	256	256	256	256	256
Buffer size	1e6	1e6	1e5	1e5	1e6	1e6	1e6
$\mathcal{H}_{target}$	$4\text{dim}(\mathcal{A})$	N/A	N/A	N/A	N/A	$-0.9\text{dim}\mathcal{A}$	N/A
Critic hidden depth	2	2	2	3	3	3	2
Critic hidden size	2048	2048	256	256	256	256	256
Actor/Score depth	3	3	4	4	4	3	2
Actor/Score size	256	256	256	256	256	256	256
Num. Bins/Quantiles	100	N/A	N/A	N/A	N/A	2	N/A
Temp. Learn. Rate	1e-3	N/A	N/A	N/A	N/A	3e-2	N/A
Learn. Rate Critic	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4
Learn. Rate Actor/Score	3e-4	3e-4	1e-5	1e-5	3e-4	3e-4	3e-4
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Diffusion Steps	16	15	5	N/A	100	20	20
Prior Distr.	$\mathcal{N}(0, 2.5)$	$\mathcal{N}(0, 1)$	N/A	N/A	N/A	$\mathcal{N}(0, 1)$	$\mathcal{N}(0, 1)$
Exploration Steps	5000	1e4	1e4	1e4	1e4	1e4	1e4
Score-Q align. factor	N/A	50	N/A	N/A	N/A	N/A	N/A

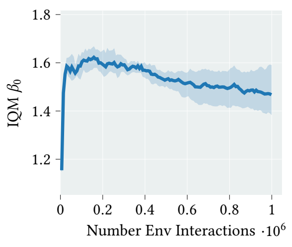
Table C.2.: Hyperparameters of DIME and all diffusion-based algorithms for all benchmark suits. Varying hyperparameters for different benchmark suits are described in the text.

	DIME	BRO	BRO Fast	CrossQ
Polyak weight	N/A	0.005	0.005	N/A
Update-to-data ratio	2	10	2	2
Discount	0.99	0.99	0.99	0.99
batch size	256	128	128	256
Buffer size	1e6	1e6	1e6	1e6
$\mathcal{H}_{target}$	$4\text{dim}(\mathcal{A})$	$\text{dim}(\mathcal{A})/2$	$\text{dim}(\mathcal{A})/2$	$\text{dim}(\mathcal{A})$
Critic hidden depth	2	BRONET	BRONET	2
Critic hidden size	2048	512	512	2048
Actor/Score depth	3	BRONET	BRONET	3
Actor/Score size	256	256	256	256
Num. Bins/Quantiles	100	100	100	N/A
Temp. Learn. Rate	1e-3	3e-4	3e-4	3e-4
Learn. Rate Critic	3e-4	3e-4	3e-4	7e-4
Learn. Rate Actor/Score	3e-4	3e-4	3e-4	7e-4
Optimizer	Adam	AdamW	AdamW	Adam
Diffusion Steps	16	N/A	N/A	N/A
Prior Distr.	$\mathcal{N}(0, 2.5)$	N/A	N/A	N/A
Exploration Steps	5000	2500	2500	5000

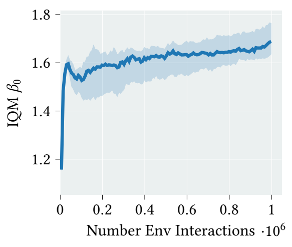
Table C.3.: Hyperparameters of DIME and Gaussian-based algorithms for all benchmark suits. Varying hyperparameters for different benchmark suits are described in the text.



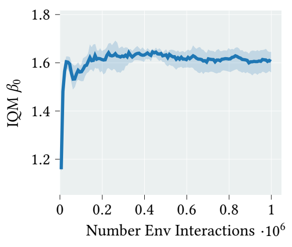
(a) β_0 on the Dog Run



(b) β_{10} on the Dog Run



(c) β_{20} on the Dog Run



(d) β_{30} on the Dog Run

Figure C.2.: Learned β parameters. DIME's policy improvement objective (Eq. 5.27) allows to train various parameters end-to-end, such as the scaling for the diffusion coefficient β. More concretely, we train a scaling parameter β_k per dimension k, that scales the cosine schedule. We visualize the adaptation of the parameter for the dimension $k = 0, 10, 20, 30$ over the training, averaged over 10 seeds for the dog-run task. Clearly, DIME first increases the parameter at the beginning of the training phase. Depending on the dimension, it either converges to a rather high value ($k = 20$ and $k = 30$), or keeps being reduced for other dimensions $k = 0$ and $k = 10$.

C.6. Additional Experiments

End-To-End Learning of β. We visualize the adaptation of the scaling for the diffusion coefficient β in Fig. C.2 during learning on DMC's dog-run environment.