The Dissertation Committee for Zizhao Wang
certifies that this is the approved version of the following dissertation:

Causality-Inspired Reinforcement Learning: State Abstractions, Exploration, and Representations

Committee:

Peter Stone, Supervisor

Amy Zhang

Roberto Martín-Martín

Sandeep Chinchali

Alessandro Lazaric

Causality-Inspired Reinforcement Learning: State Abstractions, Exploration, and Representations

by

Zizhao Wang

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2026

Acknowledgments

I would like to thank many people for their support, encouragement, and guidance throughout my PhD journey.

First and foremost, I am deeply grateful to my PhD advisor, Peter Stone, for his mentorship and unwavering support during my years at UT Austin. In my early years, Peter offers patient guidance and consistently makes time for detailed discussions, helping me polish vague ideas into concrete research directions. He also gives me remarkable freedom to explore, encouraging me to pursue the questions I feel most excited about and to take ownership of my work. I especially appreciate how he regularly points me to foundational and early literature, which helps me understand the history and depth of the field and broadens my perspective beyond immediate trends. Beyond research, Peter sets a personal example of what it looks like to lead with kindness and discipline: he manages his time extraordinarily well, supports students generously, and still maintains his passions, from playing the violin to staying impressively active in soccer. I am also very thankful for his advice and steady encouragement throughout my job search, and for the confidence he shows in me at each step.

I would also like to express my heartfelt thanks to my thesis committee, Amy Zhang, Roberto Martín-Martín, Alessandro Lazaric, and Sandeep Chinchali, for their time, guidance, and thoughtful feedback that sharpen my thinking and strengthen this thesis, as well as for their support for my career development.

I am grateful to my collaborators, labmates, and friends at UT Austin for making my PhD experience both productive and enjoyable. In particular, I would like to thank Xuesu Xiao, who helps me quickly find my footing during my first year and makes a huge difference in helping me get on track. I am also very thankful to Prof. Yuke Zhu for his valuable suggestions, insightful discussions, and contributions to my first several projects, shaping how I approach research questions. I would like to thank my other collaborators and friends as well: Professor Scott Niekum, Professor Ufuk Topcu, Brad Knox, Josiah Hanna, Sanmit Narvekar, Ishan Durugkar, Haresh Karnan, William Macke, Yuqian Jiang, Jin Soo Park, Yu-Sian Jiang, Yifeng Zhu, Bo Liu, Jiaxun Cui, Caroline

Wang, Zifan Xu, Jiaheng Hu, Siddhant Agarwal, Michael Munje, Zhihan Wang, Yoonwoo Kim, Lingyun Xiao, Steven Shi, Hao Fu, Claas Voelcker, Junhong Xu, Nathan Tsoi, Shuijing Liu, Arrasy Rahman, Chen Tang, Alexander Levine, Yoonchang Sung, Rohan Chandra, Shahaf Shperberg, Yulin Zhang, Reuth Mirsky, Harel Yedidsion, Ian Fasel, Kevin Han, Hender Lin, Liza Pavlova, Kevin Rohling, Daniel Kim, Garrett Warnell, Abitpal Gyawali, Alex Nettekoven, Corrie Van Sice, Justin Hart, Mauricio Tec, Nick Pape, Nikunj Parmar, Stephen Chen, Xixi Hu, Caleb Chuck, Harshit Sikchi, Ruohan Zhang, Zhaoyuan He, Haoran Xu, Max Rudolph, Carl Qi, Chang Shi, Dongsu Lee, Zichao Hu, Zhenyu Jiang, Huihan Liu, Mingyo Seo, Rutav Shah, Soroush Nasiriany, Shuozhe Li, Tongzheng Ren, Jake Grigsby, Yu Lei, Siqi Shang, Zhendong Wang, Shenghui Chen, Tianlong Chen, Zhiyun Deng, Bowen Jiang, Shentao Yang. I feel fortunate to be surrounded by such supportive people, and I appreciate the many conversations, brainstorming sessions, and moments of encouragement along the way.

I would also like to thank my internship mentors and collaborators, Aolin Zhang, Teruhisa Misu, Zhimin Chen, Yuxiang Guo, Tianyu Li, Jiang Bian, Li Zhao, Kaixin Wang, Dingcheng Li, Lukas Rutishauser, Vaishakh Keshava, Phillip Wallis, and Ananth Balashankar, for their guidance, feedback, and support. I am grateful for the opportunity to learn from you, and for the mentorship and collaboration that help me grow as a researcher and engineer.

I would like to express my heartfelt thanks to my master's and bachelor's supervisors, Prof. Peter Allen, Prof. Itsik Pe'er, Prof. Gabor Orosz, Prof. Necmiye Ozay, Iretiayo Akinola, Antonio Khalil Moretti, and Wubing Qin, whose guidance helps pave my way into research. I am grateful for the training, encouragement, and trust you give me early on, and for your support during my PhD applications.

Finally, I want to thank my girlfriend, Yiren Wang, for her love, patience, constant support and for not giving up teaching me how to create beautiful posters and slides. I am also deeply grateful to my parents, Zhenguo Wang and Tongrang Fan, for their unconditional love and belief and for always being my foundation throughout the years, and to my cat, Momoko, for the companionship and the many moments of much-needed joy and distraction during long days of debugging. I could not have completed this journey without all of you.

Abstract

Causality-Inspired Reinforcement Learning: State Abstractions, Exploration, and Representations

Zizhao Wang, PhD
The University of Texas at Austin, 2026

SUPERVISOR: Peter Stone

Reinforcement learning (RL) offers a versatile paradigm for developing autonomous decision making agents. Yet, many of today's RL algorithms still suffer from two major drawbacks — high data requirements and poor generalization. A key reason is the limitation of correlation-based learning. Most existing deep RL methods learn policies end-to-end by correlating all input state factors with actions using deep neural networks. When learning from such an unnecessarily large input space, these algorithms not only incur high sample complexity but also tend to capture spurious correlations, leading to poor generalization in new situations.

To overcome these drawbacks, this thesis provides a new perspective of facilitating the generalization and sample efficiency of RL algorithms by leveraging causality. Causality characterizes whether and how variables influence one another in the data generation process. Through the lens of causality, we can reason about dependencies that matter for decision making, particularly in dynamics (which actions and state factors affect the transition of each factor) and in reward functions (which state factors determine task success). These causal structures enable RL agents to learn accurate dynamics and reward models, and to design hierarchical policies for solving complex, long-horizon tasks sample-efficiently while transferring effectively to new environments and previously unseen tasks.

Specifically, this thesis makes four closely connected contributions: (1) a theoretically

grounded method that reasons about causal relationships in dynamics and rewards via conditional independence tests, enabling the derivation of minimal state abstractions that facilitate generalization; (2) an intrinsic reward function based on local causal dependencies that guides exploration and learning in sparse-reward tasks; (3) a method for discovering a library of skills for generating causal interactions that can be composed to solve long-horizon tasks; and (4) for domains where high-level state factors are not available, representation learning methods that extract such factors from low-level observations.

By integrating causal reasoning into RL, these contributions work together to enable an agent to infer the causes and consequences of actions, generalize robustly to unseen states, explore new environments strategically, and learn new tasks sample-efficiently with limited data.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

A longstanding goal of Artificial Intelligence and Intelligent Robotics is to develop a generalist agent capable of mastering a diverse set of tasks that it encounters over the course of its autonomous existence. Deep reinforcement learning (RL) is a general paradigm that enables autonomous decision-making, but many existing RL algorithms suffer from two common deficiencies limiting their applicability in open worlds: (1) *high data requirements* — RL algorithms typically require millions (if not billions) of transitions for convergence, and (2) *poor generalization* — RL agents are typically fragile when faced with distribution shifts during testing, rendering them vulnerable to complete failure even with minor environmental changes.

One of the major causes of these limitations is that most existing algorithms learn based on arbitrary *correlations*: the policy networks predict actions based on *all of the state factors*. However, in many tasks, agents only need to consider a subset of state factors when making decisions. For instance, an autonomous vehicle may benefit from paying attention to driving-related objects, such as surrounding vehicles, pedestrians, and traffic lights, while disregarding irrelevant objects like birds in the sky or advertisements on billboards. The correlation-based training method has two main drawbacks. First, learning from an unnecessarily large input space inevitably slows down learning, as the agent must determine through trial and error whether each state factor is relevant to decision-making. This process requires a significant amount of exploration and can impede efficient learning. Second, even after learning from a large amount of data, correlation-based models often retain spurious correlations present in data noise (as it is challenging for a network to assign an exactly zero weight), leading to poor generalization. In the autonomous vehicle example, if the agent's learned behavior depends even slightly on billboards, an unfamiliar (or even adversarial) advertisement may lead the vehicle to execute a dangerous maneuver.

The sample efficiency of RL algorithms is also hampered by the vast exploration space they encounter. This exploration space is characterized by two significant factors: (1) *sparse reward space* — for many real-world tasks, defining a dense reward function that gives meaningful feedback on each individual action is non-trivial, while only a sparse reward function based on the success

or failure of the task as a whole is directly available. However, such sparse rewards offer limited learning signals, hindering the agent's ability to navigate the environment and discover optimal strategies effectively; and (2) *large temporal space* — tasks with long time horizons require the completion of numerous actions. As the horizon increases, the number of possible action sequences grows exponentially. Consequently, finding an ordering of actions that accomplishes a particular task becomes increasingly difficult with longer-horizon tasks.

To deploy RL agents successfully in open worlds, it is critical for them to overcome these challenges in order to exhibit robust generalization to unseen or adversarial states, adapt swiftly to new environments, and learn new tasks efficiently using limited samples.

With this motivation in mind, we propose to facilitate RL generalization and sample efficiency by leveraging the concept of **causality**. Causality provides insight into whether and how each variable is influenced by other variables in the data generation process. Through the lens of causality, we can better understand the dependencies between variables in the RL setup, particularly in terms of dynamics (which actions and state factors affect transitions of each factor) and reward function (which state factors determine task success).

1.1 Contributions

In summary, we focus on addressing the following question.

> **Thesis Question:** How can a reinforcement learning agent reason about and leverage causal relationships in the environment's dynamics and reward function to learn generalizable policies sample-efficiently?

To address this question, this thesis presents the following contributions.

- **(Contribution 1) State Space – Minimal State Abstraction**: We introduce algorithms for identifying causal relationships in the dynamics and reward, enabling agents to derive a *state abstraction* that retains only task-relevant state factors. This abstraction facilitates generalization and exploration by reducing the dimensionality of the state space and focusing on factors that are crucial for decision-making.

- **(Contribution 2) Reward Space – Intrinsic Reward Function**: We introduce an algorithm for designing an *intrinsic reward function* that augments sparse task rewards based on induced causal relationships. This intrinsic reward function guides exploration and facilitates sample efficiency.

- **(Contribution 3) Action Space – Unsupervised Skill Discovery**: We introduce an algorithm that leverages causal relationships to learn a set of *compositional skills* that can be combined to solve long-horizon tasks. We show that such skills allow the agent to reason about and plan actions at different time scales, leading to efficient learning.

- **(Contribution 4) Observation Space – State Factor Extraction**: The three directions above assume that high-level state factors are given, but in the real world we typically only have access to low-level observations such as images. To address this limitation, we present algorithms for extracting state factors from observations in two forms: object-centric representations and action-grouped representations.

As indicated by their names, the first three contributions each target a major source of poor generalization and low sample efficiency discussed above: spurious correlations induced by irrelevant state factors, sparse rewards, and long time horizons. The fourth contribution extends the first three to environments where only observations are available. Finally, the strategies developed from these four contributions are complementary and can, in principle, be used simultaneously.

1.2 Dissertation Overview

While this dissertation is written to be read sequentially from beginning to end, readers do not necessarily need to follow this order. To accommodate selective reading, we provide a visualization of chapter dependencies in Figure 1.1. The dissertation is structured as follows.

- Chapter 2 presents a formulation of RL and causality, as well as background on state abstractions, intrinsic reward functions, unsupervised skill discovery, and representation learning, which are necessary for understanding the rest of the dissertation.

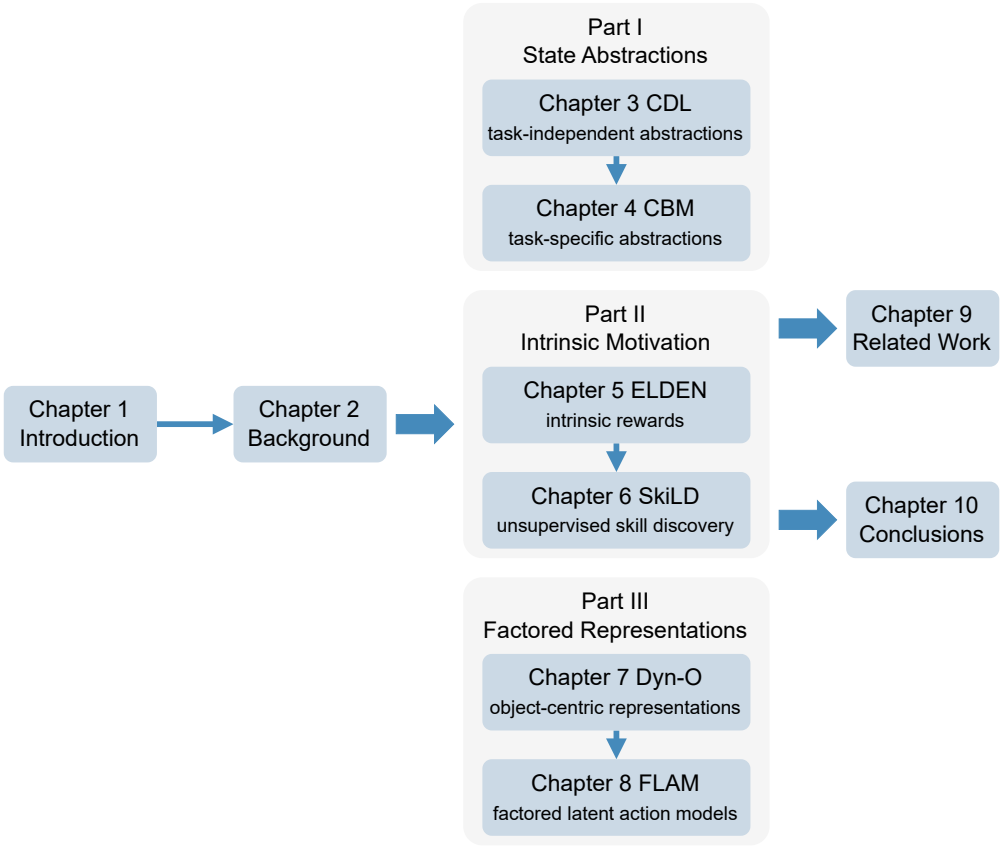


Figure 1.1: Overview of the chapter dependencies. An arrow connection means that one chapter (or a group of chapters) should be read before another. The chapters from different parts can be read independently.

- Chapter 3 introduces Causal Dynamics Learning (CDL), a theoretically grounded method that reasons about causal relationships in dynamics via conditional independence tests, where a causal dependency is considered to exist if including an input state factor improves dynamics prediction than excluding it (Wang et al., 2022b). By retaining only necessary dependencies and eliminating spurious correlations, CDL improves generalization of learned dynamics models compared to dense models that ignore dynamical causal relationships.

- Chapter 4 introduces Causal Bisimulation Modeling (CBM), which extends CDL by additionally reasoning about causal relationships in the reward function to identify which state factors are relevant to the task Wang et al. (2024b). Compared to learning with all state factors, the

resulting state abstractions facilitate sample efficiency and generalization of RL agents.

- Chapter 5 introduces Exploration via Local Dependencies (ELDEN), an intrinsic reward function based on local causal dependencies (Wang et al., 2023). CDL and CBM consider the global causal dependencies between state factors across the state space, but many dependencies are only active in a small part of it, e.g., a robot cannot manipulate an object unless they are in contact. We therefore introduce an approach that captures fine-grained *local causality*, identifying which dynamical dependencies between state factors are active at a particular state. By assessing uncertainty in these local causal relationships, we derive an intrinsic reward signal that encourages exploration of novel object interactions, leading to more efficient exploration and learning than relying on task reward alone.

- Chapter 6 introduces Skill Discovery from Local Dependencies (SkiLD), an unsupervised skill discovery method that learns reusable skills to induce novel local causal dependencies (Wang et al., 2024a). Existing skill learning methods (Nasiriany et al., 2022; Pertsch et al., 2021) typically require substantial expert input to design skill preconditions and goals. In contrast, local causality characterizes interactions between state factors and thus naturally provides a goal specification for learning semantically meaningful skills. We introduce a hierarchical approach that enables an agent to learn a set of such skills without reward signals or human inputs. Specifically, the high-level policy is rewarded for discovering novel local dependencies, and does so by strategically exploring dependencies that are likely to expose new ones (e.g., holding a vacuum increases opportunities to use it to clean the floor). Meanwhile, the low-level policy is rewarded for inducing the local interactions specified by the high-level policy (e.g., holding the vacuum). Over the course of learning, the high-level policy explores local dependencies from those that are easy to induce (e.g., the robot moving itself) to harder ones with complex preconditions (e.g., using a vacuum to clean the floor requires the robot to hold the vacuum and plug it in). This progression naturally forms a curriculum for low-level policy learning, guiding it to acquire diverse skills from simple ones (e.g., navigation, tool picking) to complex ones (e.g., applying tools to other objects). During task learning, these skills can then be composed to solve long-horizon tasks.

- Chapter 7 introduces Object-Centric Representation for Structured World Models (Dyn-O), a method for extracting state factors from observations using object-centric representations (Wang et al., 2025). In the previous chapters, we assume that high-level state factors are given, but in many real-world settings we only have access to low-level observations such as images. To address this limitation, we adopt object-centric representations that encode each object into a separate feature and treat these features as state factors. Compared to prior work, we further disentangle each object's feature into two components, one modeling dynamical attributes (e.g., velocity) and the other modeling static attributes (e.g., color). This disentanglement enables data augmentation for policy learning: we resample static attributes to create data with the same object trajectories but different appearances, improving sample efficiency and generalization compared to training without such augmentation.

- Chapter 8 introduces Factored Latent Action Models (FLAM), which extracts state factors using latent action models by identifying factors associated with independent actions. Existing latent action model methods (Schmidt and Jiang, 2024; Bruce et al., 2024) typically use a monolithic inverse dynamics model (IDM) to encode all environmental changes into a single latent action, which a forward dynamics model (FDM) then uses to predict the next observation. However, in scenarios where multiple entities act independently, encoding all action combinations into a single latent action space is challenging. Instead, we propose a factored IDM and FDM in which the latent state is decomposed into multiple factors, each independently inferring its latent action and predicting its next state. We show that this structured approach better captures multi-entity dynamics and generates more accurate predictions than monolithic models.

- Chapter 9 provides an overview of related work.

- Chapter 10 summarizes the thesis contributions and outlines future research directions.

Chapter 2: Background

This chapter provides the background and notation necessary to understand the thesis contributions introduced in the following chapters. While this chapter provides overview of prior work that following chapters build on or compare against, Chapter 9 provides a comprehensive overview of related work. I discuss the problem setup for RL in Sections 2.1 and causality in Section 2.2. Then I review several RL subfields, including model-based RL (Sections 2.3), state abstractions (Sections 2.4), intrinsic reward functions (Sections 2.5), unsupervised skill discovery (Sections 2.6), and how causality is relevant to them. Finally, I discuss two lines of work that are useful to extract representations for causality, specifically, object-centric representations (Sections 2.7) and latent action models (Section 2.8).

2.1 Markov Decision Processes (MDPs)

Following common practice, we model environments as factored Markov decision processes (MDPs), $\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where

- $\mathcal{S}$ is the state space, consisting of $d_{\mathcal{S}}$ state variables (factors), i.e., $\mathcal{S} = \mathcal{S}^1 \times \cdots \times \mathcal{S}^{d_{\mathcal{S}}}$;
- $\mathcal{O}$ is the observation space;
- $\mathcal{A} \subseteq \mathbb{R}^{d_{\mathcal{A}}}$ is the $d_{\mathcal{A}}$-dimensional action space;
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability, which we also refer to as the *transition dynamics* or simply the *dynamics*;
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function; and
- $\gamma \in [0, 1]$ is the discount factor.

Given an environment $\mathcal{M}$ and its task (specified by $\mathcal{R}$), we seek a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected discounted return $\mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \right]$.

For ease of reference, we summarize the following chapter-specific assumptions regarding $\mathcal{M}$.

- $\mathcal{S}$ and $\mathcal{O}$: Chapters 3-6 assume that $\mathcal{S}$ is given. In contrast, Chapters 7-8 assume that we only have access to observations $o \in \mathcal{O}$ and must infer states $s \in \mathcal{S}$ from them.
- $\mathcal{A}$: Chapters 3-7 assume that $\mathcal{A}$ is given. In contrast, Chapter 8 assumes that we only have access to observations $o \in \mathcal{O}$ and must infer a latent action from a pair or sequence of observations.
- $\mathcal{P}$: All chapters assume that $\mathcal{P}$ is unknown and must be learned from data.
- $\mathcal{R}$: Chapters 3 and 7 consider a single-task setting. Chapter 4 considers multiple tasks that share the same state space, action space, and transition probability, where each task has its own reward function. Chapters 5 and 6 focus on unsupervised RL where $\mathcal{R}$ is the intrinsic reward function instead of a given task reward function, and Chapter 8 does not involve the reward function.

For simplicity, we additionally introduce the following notation.

- We use x_t to denote all state factors and the action at time t, i.e., $x_t = \{s_t^1, \cdots, s_t^{d_S}, a_t\}$, and use $x_t^{\neg i}$ to denote all variables in x_t except s_t^i, i.e., $x_t^{\neg i} = x_t \setminus \{s_t^i\}$.
- For a generic variable y, we use $y_{1:t}$ to denote the set of variables $\{y_1, \ldots, y_t\}$. Similarly, $y^{1:N}$ denotes $\{y^1, \ldots, y^N\}$ for a generic N.

2.2 Causality

In addition to RL, causality is another central theme of this thesis. In this section, we introduce causal graphical models in Section 2.2.1, distinguish between general and actual causal dependencies in Section 2.2.2, and review causal discovery methods for identifying these dependencies from data in Sections 2.2.3 and 2.2.4.

2.2.1 Causal Graphical Models (CGMs)

A causal graphical model (Pearl, 2009) is defined by a distribution $p(X)$ over d random variables $X^1, \ldots, X^d$ and a directed acyclic graph (DAG) $\mathcal{G} = (V, E)$. Each node $X^i \in V$ corresponds to a random variable X^i, and each directed edge $(X^i \to X^j) \in E$ indicates that X^i is a **direct cause** of X^j, i.e., X^j depends on X^i in the data generation process. The joint distribution then factorizes as

$$p(x^1, \ldots, x^d) = \prod_{i=1}^{d} p(x^i \mid \mathbf{PA}(X^i)), \tag{2.1}$$

where $\mathbf{PA}(X^i)$ denotes the set of parents of node X^i in the graph $\mathcal{G}$.

This thesis is themed around formalizing MDPs using the language of CGMs and exploring how such formalizations can be leveraged. Specifically, in an MDP, the following components can be modeled as CGMs:

- **Transition probability ($\mathcal{P}$).** Viewing the transition at time step t as a CGM, the variables consist of $(\mathcal{S}_t, \mathcal{A}_t, \mathcal{S}_{t+1})$, and the graph $\mathcal{G}$ specifies which state factors at time t and the action $\mathcal{A}_t$ can affect each state factor at time $t+1$. Assuming that the transitions of each state factor $\mathcal{S}^i$ are independent, the transition probability can be decomposed as $\mathcal{P}(s_{t+1} \mid s_t, a_t) = \prod_{i=1}^{d_S} p(s^i_{t+1} \mid \mathbf{PA}(\mathcal{S}^i_{t+1}))$.
- **Reward function ($\mathcal{R}$).** For the reward function, the variables consist of $(\mathcal{S}_t, \mathcal{A}_t, \mathcal{R}_t)$ and the graph $\mathcal{G}$ specifies which state factors and whether the action affect the reward. The reward distribution can similarly be written as $\mathcal{R}(r_t \mid s_t, a_t) = p(r_t \mid \mathbf{PA}(\mathcal{R}_t))$, where the reward generation process depends only on its parents rather than on all state factors (we slightly overload notation by using $\mathcal{R}_t$ to denote the reward variable).

2.2.2 General and Actual Causality

In this thesis, we distinguish between *general causality* (the focus of Chapters 3 and 4) and *actual causality* (the focus of Chapters 5 and 6).

General causality refers to the causal relationships encoded by the underlying data generation process, i.e., the dependencies captured by the causal graph $\mathcal{G}$ in Eq. 2.1. In this sense, an edge

$X^i \to X^j$ indicates that X^i is a direct cause of X^j across the population distribution $p(X)$. However, while general causality describes relationships of the form "X^i can affect X^j," it does not determine whether this relationship is *active* in a particular observed sample $X = x$. For example, a robot is able to pick up a cube in general, but when the cube is out of the robot's reach, the robot cannot affect the cube. This example illustrates that although the general causal dependency "robot $\to$ cube" holds, the corresponding actual dependency is inactive at that specific robot and object state.

We take a causality-inspired approach to defining actual causal dependencies (Bontempi and Flauder, 2015; Seitzer et al., 2021). Formally, for an *event of interest* $Y = y$ and its potential causes $X = (X^1, \ldots, X^d)$, given the value $X = x$, actual causal dependencies identify which variables X^i are state-specific causes of the outcome event $Y = y$. We denote the general data generation process of Y as $p : X \to Y$, and the data generation process when Y is *only influenced* by a subset of variables as $p_{\bar{X}} : \bar{X} \to Y$, where $\bar{X} \subseteq X$. Given $X^1 = x^1, \cdots, X^d = x^d$ and $Y = y$, we say that Y actually depends on $\bar{X}$ if $\bar{X}$ is the *minimal* subset of X such that the probability of the outcome under $p_{\bar{X}}$ matches that under the full process p, i.e.,

$$\arg\min_{\bar{X} \subseteq X} |\bar{X}| \quad \text{s.t.} \quad p_{\bar{X}}(Y = y \mid \bar{X} = \bar{x}) = p(Y = y \mid X = x), \tag{2.2}$$

where $\bar{x}$ denotes the restriction of x to the variables in $\bar{X}$, and $|\bar{X}|$ is the number of variables in $\bar{X}$. For example, suppose that a robot opens a refrigerator door in a particular transition. The event of interest $Y = y$ is the refrigerator door becoming open, and it actually depends on two factors: the robot and the refrigerator door, while other state factors such as objects inside the refrigerator do not actually influence Y.

Overall, general causality is a property of the data generation process $p(X)$ (what can cause what in principle) and provides a *global* causal structure, whereas actual causality is a property of an individual trajectory or sample (what did cause what in this particular case) and provides *local* causal dependencies grounded in counterfactual reasoning. For this reason, throughout the thesis we use the terms *general* and *global* dependencies interchangeably, and likewise use the terms *actual* and *local* dependencies interchangeably.

2.2.3 General Causal Discovery

In many settings, the causal graph $\mathcal{G}$ is unknown and must be inferred from observational samples of X. This subsection summarizes a standard constraint-based approach for recovering *general* (global) causal dependencies from purely observational data: the Peter-Clark (PC) algorithm (Spirtes et al., 2000). From a high-level perspective, the PC algorithm leverages conditional independence tests (CITs) to estimate the (in)dependence structure of $p(X)$, and then translates these relationships into graphical constraints via d-separation (see definitions below). Although observational data alone cannot identify all causal dependencies in general, under the assumptions of the causal Markov condition, causal faithfulness, and the absence of unobserved confounders (see definitions below), the pattern of conditional independencies in $p(X)$ constrains the underlying DAG up to its Markov equivalence class (i.e., the set of DAGs that entail the same collection of conditional independence relations), making partial causal discovery possible from observations. Furthermore, as will be detailed in Chapters 3 and 4, by utilizing properties of MDPs (such as $\mathcal{S}_{t+1}$ cannot affect $\mathcal{S}_t$), the PC algorithm can additionally enable full causal discovery for the transition probability and reward functions.

Before describing the PC algorithm, we first introduce the following prerequisite definitions and assumptions.

Definition 2.1 (d-separation (Pearl, 2009))**.** In a directed acyclic graph (DAG) $\mathcal{G}$, a path between nodes X_1 and X_k is blocked by a set $\bar{X}$ (with neither X_1 nor X_k in $\bar{X}$) whenever there is a node $X_i, i = 2, \ldots, k-1$, such that one of the following two possibilities holds:
(i) $X_i \in \bar{X}$ and $X_{i-1} \to X_i \to X_{i+1}$ or $X_{i-1} \leftarrow X_i \leftarrow X_{i+1}$ or $X_{i-1} \leftarrow X_i \to X_{i+1}$.
(ii) Neither X_i nor any of its descendants is in $\bar{X}$ and $X_{i-1} \to X_i \leftarrow X_{i+1}$.

In a DAG $\mathcal{G}$, we say that two nodes A and B are d-separated by a third node C if every path between nodes A and B is blocked by C, denoted as $A \perp\!\!\!\perp_{\mathcal{G}} B \mid C$.

Definition 2.2. (Causal Markov Condition (Spirtes et al., 2000)) Let $\mathcal{G}$ be a causal graph with node set V and p be a probability distribution over the nodes in V generated by the causal structure represented by $\mathcal{G}$. $\mathcal{G}$ and p satisfy the Causal Markov Condition if and only if for every node v in V,

v is independent of $V \setminus (\mathbf{Desc}(v) \cup \mathbf{PA}(v))$ given $\mathbf{PA}(v)$, where $\mathbf{Desc}(v)$ denotes the descendants of v.

Here we use the global version of the Markov condition, which reads: if $A \perp\!\!\!\perp_{\mathcal{G}} B \mid C$ then $A \perp\!\!\!\perp B \mid C$ for all disjoint vertex sets A, B, C (where $\perp\!\!\!\perp_{\mathcal{G}}$ denotes d-separation as defined above, and $\perp\!\!\!\perp$ denotes conditional independence in p).

Definition 2.3. (Causal Faithfulness). A distribution p is faithful to a DAG $\mathcal{G}$ if no conditional independence relations other than the ones entailed by the Markov property are present.

PC algorithm Let $X^1, \ldots, X^d$ denote the d observed variables. The PC algorithm proceeds in two stages: (i) learn an undirected *skeleton* by removing edges that are rejected by CITs, and (ii) orient as many remaining edges as possible using orientation rules.

During skeleton discovery, the PC algorithm first initializes a complete undirected graph over the variables. It then iteratively increases the size of conditioning sets and performs CITs of the form $X^i \perp\!\!\!\perp X^j \mid \bar{X}$ for subsets $\bar{X}$ drawn from the current neighbors of X^i (or X^j). If a set $\bar{X}$ is found such that $X^i \perp\!\!\!\perp X^j \mid \bar{X}$, the edge between X^i and X^j is removed, and the separating set is recorded as $\mathbf{Sep}(X^i, X^j) \leftarrow \bar{X}$. The process terminates when no more edges can be removed given the current neighborhood sizes.

During edge orientation, given the learned skeleton and separating sets, the PC algorithm first identifies and orients *v-structures*. Specifically, for any triple (X^i, X^j, X^k) such that X^i and X^k are non-adjacent but both are adjacent to X^j, if $X^j \notin \mathbf{Sep}(X^i, X^k)$, then the PC algorithm orients $X^i \rightarrow X^j \leftarrow X^k$. Finally, the PC algorithm repeatedly applies a set of orientation propagation rules, e.g., Meek rules (Meek, 1995), to orient additional edges whenever doing so does not introduce a cycle or a new v-structure that contradicts the CITs. The output is a Markov equivalence class, i.e., a set of DAGs encoding all and only those edge orientations that are identifiable from CITs.

Practical considerations In practice, causal discovery can be inaccurate for several reasons. First, finite data may have limited coverage of the data distribution, so some causal dependencies are

rarely activated or are confounded with other factors, making them difficult to detect from CITs alone. Second, CITs can be statistically noisy with limited samples, and this noise can lead to inaccurate edge removal or retainment. Finally, in this thesis, we implement CITs by estimating conditional mutual information (CMI), where conditional distributions are modeled by neural networks. Approximation and optimization errors in these learned models further affect the CIT accuracy and can lead to incorrect causal dependencies.

2.2.4 Actual Causal Discovery

As discussed in Section 2.2.2, for an event of interest $Y = y$ and its potential causes $X = (X^1, \ldots, X^d)$, we are interested in identifying actual causal dependencies given the value of $X = x$ and the outcome event $Y = y$. To do so, one can conduct a conditional independence test $Y \perp\!\!\!\perp X^i \mid \{X/X^i\}$ to examine whether X^i is necessary for predicting the outcome event. In prior work (Lizier and Prokopenko, 2010), one form of this test is to examine whether the pointwise conditional mutual information (pCMI) is greater than 0,

$$\text{pCMI}(y; x^i \mid \{x/x^i\}) = \log \frac{p(y \mid x)}{p^{\{X/X^i\}}(y \mid \{x/x^i\})} > 0. \tag{2.3}$$

If so, then knowing $X^i = x^i$ provides additional information about $Y = y$ that is not present in $\{X/X^i\}$, suggesting that Y locally depends on X^i at x. As the data generation processes are generally unknown, they can be approximated with learned models. Recent work has considered various approximations, such as attention-based estimators (Pitis et al., 2020) and Granger-causality-based estimators (Chuck et al., 2023).

In Chapters 5 and 6, we discuss a novel algorithm to identify actual causal relationships and how the identified relationships can be used to construct intrinsic rewards and to learn skills that enhance RL sample efficiency.

2.2.5 Causality and Compression

There are several connections between causality and compression, largely through the view that a "good" causal model *compresses* raw observations into variables that make prediction and intervention simple.

Causal models can be viewed as compressed descriptions of the data generating process: instead of representing high-dimensional observations directly, a structural causal model summarizes them with a smaller set of variables and sparse, modular mechanisms (e.g., each variable depends only on its parents rather than on all other variables). This modularity also yields compression across environments. If most mechanisms remain invariant across contexts and only a subset changes, then a single shared causal model plus a small amount of context specific adaptation can explain many different environments more compactly than learning separate models for each setting.

While causality can be understood as a particular form of compression, compression alone does not lead to causal models. Compression can produce representations that are predictive in the training distribution but still rely on spurious correlations. What distinguishes causal structure is intervention robustness. The compressed variables and mechanisms are meant to remain valid for predicting the effects of interventions (or distribution shifts induced by changed policies or environments), not merely to encode regularities observed under one fixed dataset.

2.3 Model-Based Reinforcement Learning (MBRL)

Model-based reinforcement learning (MBRL) aims to facilitate the sample efficiency and generalization of RL by explicitly learning a model of the environment and using it to plan, generate additional training data, or learn useful representations for policy learning.

A standard MBRL agent learns one or more predictive models from experience tuples $(s_t, a_t, r_t, \tau_t, s_{t+1})$: a dynamics model $p(\hat{s}_{t+1} \mid s_t, a_t)$ that predicts the next-state distribution given the current state and action, a reward model $p(\hat{r}_t \mid s_t, a_t)$ that predicts the immediate reward, and a termination model $p(\hat{\tau}_t \mid s_t, a_t)$ that predicts whether the episode terminates, where we denote termination by τ. These models are typically trained by minimizing a negative log-likelihood objective,

$$\mathcal{L} = \mathbb{E}\left[-\log p(\hat{s}_{t+1} \mid s_t, a_t) - \log p(\hat{r}_t \mid s_t, a_t) - \log p(\hat{\tau}_t \mid s_t, a_t)\right],$$

with appropriate parameterizations for continuous or discrete variables. In practice, different losses and architectures may be used for each prediction model, and many methods also use ensembles or

uncertainty estimates to generate robust predictions.

Once learned, the model can be used in several ways.

- The model enables *planning* by searching over action sequences using simulated rollouts under the learned dynamics and selecting actions that maximize predicted return. This planning can be implemented via sampling-based methods, e.g., the cross-entropy method (Rubinstein, 1997), gradient-based optimization when the model is differentiable (Srinivas et al., 2018), or approximate dynamic programming within the model.
- The model can be used for *data augmentation* by generating synthetic transitions. These imagined rollouts can be used to train the RL policy, improving sample efficiency by reusing the model to create additional training examples compared to only using real interactions. A common approach is to start from states observed in the real dataset and sample actions from the policy to generate short-horizon synthetic trajectories for policy learning.
- The representations learned by the model can capture dynamics information that facilitates credit assignment, enabling efficient policy learning. Specifically, predicting future states or rewards can shape representations to encode task-relevant dynamics, and multi-step prediction losses can encourage temporal consistency. More broadly, model learning and policy learning can be combined either sequentially (learning a model and then planning) or jointly (alternating between optimizing the model and optimizing the policy).

A central challenge in MBRL is compounding error: small one-step prediction errors can accumulate over long rollouts, which may mislead planning or policy optimization. In practice, these errors are often exacerbated by *dense* dynamics models in which the prediction of each state factor depends on the action and all factors in the current state. Such unnecessary dependencies make the model vulnerable to spurious correlations: when irrelevant factors take values outside the training distribution, the model may still incorrectly use them to predict other factors, leading to inaccurate predictions during planning or data augmentation. Motivated by the observation that unnecessary dependencies are a key source of poor generalization, Chapters 3 and 4 develop causal

discovery methods that explicitly identify which factors and actions affect each factor's next state and the reward, thereby reducing spurious correlations and improving generalization to novel states compared to dense models.

2.4 State Abstraction

In many MDPs, the state space can be high-dimensional, where many state variables may be irrelevant or redundant for decision making. State abstraction aims to reduce this complexity by mapping the original state to a lower-dimensional representation that preserves the information required for planning or policy learning. Formally, a state abstraction is a function

$$\phi : \mathcal{S} \to \bar{\mathcal{S}},$$

where $\mathcal{S}$ is the original state space and $\bar{\mathcal{S}}$ is an abstract state space. We write $\bar{s} = \phi(s)$ for the abstract state corresponding to s. An abstraction is useful if planning or policy learning in the abstract space yields a policy that performs as well (or comparably well) as one learned in the original MDP, ideally with improved sample efficiency and generalization.

Different abstractions formalize what to preserve in different ways. In general, ϕ is chosen so that states mapped to the same abstract state are interchangeable for the purpose of downstream learning. This interchangeability may be defined with respect to the model (i.e., transitions and rewards) (Givan et al., 2003), the Q-values of any policy (Boutilier et al., 2000b), the optimal Q-values (McCallum, 1996), the optimal policies (Jong and Stone, 2005), or some notion of predictive equivalence. As a result, state abstraction definitions differ in which aspects of the MDP are required to be preserved.

In this thesis, we specifically focus on bisimulation, a notion of state abstraction that preserves transition and reward equivalence. Formally, bisimulation considers two states, s_1 and s_2, to be equivalent if, for any action $a \in \mathcal{A}$ and any abstract state $\bar{s}$ (here we overload notations: s_1

and s_2 denote two arbitrary states, not the states at timestamps $t = 1$ and $t = 2$),

$$\mathcal{R}(s_1, a) = \mathcal{R}(s_2, a),$$
$$\int_{s \in \phi^{-1}(\bar{s})} \mathcal{P}(s \mid s_1, a) = \int_{s \in \phi^{-1}(\bar{s})} \mathcal{P}(s \mid s_2, a),$$

where $s \in \phi^{-1}(\bar{s})$ denotes the set of states such that $\phi(s) = \bar{s}$. In other words, bisimulation maps two states to the same abstract state if their immediate rewards are identical and their next-state distributions match when viewed at the level of abstract states. Intuitively, bisimulation preserves the full control-relevant structure of the MDP: if two states are bisimilar, no policy can distinguish them in terms of achievable returns.

In practice, exact bisimulation is often too strict, especially with continuous states or noisy observations. This limitation motivates approximate bisimulation (Dean et al., 1997b) and bisimulation metrics (Ferns et al., 2004), which replace exact equalities with distances that quantify how different two states are in terms of reward and transition behavior. Modern representation-learning methods use these ideas to learn continuous abstractions by encouraging states with similar rewards and similar successor representations (under the same actions) to have similar embeddings $\phi(s)$.

In contrast, Chapters 3 and 4 adopt a different approach to identifying bisimulation in the continuous state space, through the lens of transition and reward causal relationships. By understanding which state factors affect which other factors and the reward, these chapters introduce algorithms to identify task-irrelevant state factors and derive state abstractions that ignore them.

2.5 Intrinsic Reward

In reinforcement learning, the reward function specifies the learning objective, but in many settings external rewards can be sparse, delayed, or even absent. Intrinsic rewards address this challenge by providing an additional learning signal that encourages exploration and representation learning beyond what is directly incentivized by the task reward (Barto et al., 2004), which is the focus of Chapter 5. This idea is closely related to computational models of *curiosity* and *intrinsic motivation*, where agents are driven to seek novelty, such as unpredictable situations, in the absence

of extrinsic goals (Oudeyer et al., 2007). Formally, regarding intrinsic reward, we consider an augmented reward

$$r_t = r_{\text{task},t} + \beta \cdot r_{\text{intrinsic},t},$$

where $r_{\text{task},t}$ is the provided task reward, $r_{\text{intrinsic},t}$ is an intrinsic reward computed by the agent, and $\beta \geq 0$ controls its relative strength. Intrinsic reward functions are typically designed to promote behaviors such as novelty seeking, state coverage, or information gathering, thereby enhancing sample efficiency in sparse reward tasks.

Count-based and pseudo-count-based intrinsic reward functions A classical approach is to reward the agent for visiting rarely encountered states. In tabular settings, this reward can be implemented via count-based bonuses such as $r_{\text{intrinsic},t} \propto 1/\sqrt{C(s_t)}$, where $C(s)$ counts visits to s. In high-dimensional or continuous state spaces, pseudo-count methods extend this idea by approximating visitation counts using density models over observations or learned representations, and defining intrinsic rewards based on the change in estimated density after observing a state (Bellemare et al., 2016; Tang et al., 2017), distances to previously visited embeddings (Savinov et al., 2019; Badia et al., 2020), or representation prediction errors (Burda et al., 2019b), which makes explicit that exploration depends not only on which states are visited but also on how similarity between states is represented.

Intrinsic rewards from transition prediction error Another popular family of intrinsic reward functions uses a learned model to quantify how novel a transition is under the agent's current understanding. Such methods learn a dynamics model that predicts the next state or a distribution over the next state, denoted as $\hat{s}_{t+1} = f(s_t, a_t)$. An intrinsic reward can be computed based on the prediction error as

$$r_{\text{intrinsic},t} = \|s_{t+1} - \hat{s}_{t+1}\|,$$

where $\|\cdot\|$ is a discrepancy measure such as mean squared error or negative log-likelihood under a probabilistic model. Intuitively, transitions that the model predicts poorly are treated as novel, and thus the reward encourages the agent to visit states where its model is inaccurate.

In practice, pure prediction error can over-reward inherently stochastic or noisy states of the environment, leading the agent to seek "unpredictable" outcomes rather than meaningful novelty, known as the "noisy TV" problem. To address this issue, many methods measure uncertainty using ensembles or Bayesian approximations and define intrinsic reward based on epistemic uncertainty. For example, with an ensemble of dynamics models $\{f^i\}_{i=1}^N$, one may reward disagreement among predictions,

$$r_{\text{intrinsic},t} = \mathrm{Var}\left(\{f^i(s_t, a_t)\}_{i=1}^N\right),$$

which better reflects where the agent lacks knowledge and can improve the model through data collection than the prediction error of a single model.

Empowerment-based intrinsic reward functions Empowerment-based objectives encourage states in which the agent has high control, often formalized via mutual information between actions and future states, and recent work further refines this perspective by measuring the causal influence of actions on the environment via conditional mutual information and using it as an intrinsic reward to guide exploration and enhance sample efficiency (Seitzer et al., 2021).

2.6 Unsupervised Skill Discovery (USD)

As the focus of Chapter 6, unsupervised skill discovery studies how an agent can acquire a diverse set of reusable behaviors without task rewards, with the goal that these skills can later be reused or adapted for downstream tasks. Most prior methods introduce a latent skill variable z and learn a skill-conditioned policy $\pi(a \mid s, z)$ using an intrinsic reward function that encourages different values of z to induce diverse behaviors. A common theme is to formalize "diversity" in terms of state-space coverage, discriminability between skills, or the ability of z to control future states. Once learned, during task learning, these skills can serve as temporally extended actions in hierarchical RL, where a high-level policy selects skills over longer time horizons, while the low-level skill policy executes primitive actions, reducing the effective planning horizon and improving exploration and sample efficiency in downstream tasks.

Mutual information-based skill discovery A classical family of USD methods adopts mutual information objectives that encourage different skills to induce distinguishable outcomes. Diversity is All You Need (DIAYN) maximizes the mutual information between the skill variable z and states visited by the skill-conditioned policy, which can be implemented by training a discriminator $q(z \mid s)$ and using the discriminability term $\log q(z \mid s)$ as an intrinsic reward (Eysenbach et al., 2019). Intuitively, this objective encourages different skills to visit different regions of the state space that are easy to identify, yielding a set of diverse skills without task rewards. Related formulations in the empowerment and intrinsic control literature similarly maximize mutual information between options (or skills) and their outcomes (Gregor et al., 2017), and dynamics-aware variants maximize mutual information between z and state transitions to discover skills with predictable effects (Sharma et al., 2020; Laskin et al., 2022).

State coverage methods Another line of prior work emphasizes explicit *coverage* of the state space, motivated by the observation that purely discriminability-driven objectives can collapse to "easy" skills that diversify locally but fail to explore globally. Controllability-Aware Skill Discovery (CSD) encourages the discovery of complex, hard-to-control skills, which empirically improves coverage and prevents the skill set from focusing only on simple behaviors (Park et al., 2023). More recently, Metric-Aware Abstraction (METRA) studies scalable unsupervised RL through metric-aware abstractions, using geometric structure in representation space to improve exploration and coverage, especially in challenging high-dimensional domains (Park et al., 2024b).

Goal-conditioned reinforcement learning Another important line of prior work focuses on learning goal-reaching behaviors. Goal-conditioned RL learns a policy $\pi(a \mid s, g)$ that reaches a goal g using a goal-reaching reward (e.g., based on distance in state or representation space), and unsupervised skill discovery arises by automatically generating and learning to reach diverse goals. For example, RL with Imagined Goals (RIG) combines unsupervised representation learning with goal-conditioned RL to learn image-based goal-reaching skills, enabling flexible reuse at test time by specifying new goals in the learned latent space (Nair et al., 2018). Skew-Fit (Pong et al., 2020) further improves state coverage by learning a goal distribution that biases practice

toward underrepresented states, driving the agent to cover the space of achievable goals. This goal-conditioned viewpoint provides a natural interface to hierarchical RL, where downstream tasks can be solved by choosing goals at a higher level, while the learned goal-reaching controller executes low-level actions to achieve them.

2.7 Object-Centric Representations

In many visual domains, scenes are naturally composed of multiple entities (objects). Object-centric representation learning aims to decompose an observation (e.g., an image) into a set of *slots*, where each slot is intended to represent one entity. This decomposition is appealing for model-based RL or policy learning because it provides a structured, factorized state space that can support combinatorial generalization and modular dynamics modeling, as will be discussed in Chapter 7. From a causal perspective, when only low-level observation is available, such slots also provide a set of causal variables: treating each entity-level slot as a causal variable can enable us to conduct causal discovery between entities, as discussed in Section 2.2.

A broad class of object-centric methods learns such decompositions without object labels by combining an encoder that maps an image to a set of feature vectors with a grouping mechanism that assigns features to slots. Among these grouping mechanisms, *Slot Attention* (Locatello et al., 2020) is widely used due to its simplicity, permutation invariance, and compatibility with end-to-end training. Specifically, given an observation o (e.g., an image), we first extract n_e patch-level features using an encoder,

$$e^{1:n_e} = \texttt{Encoder}(o), \qquad e^i \in \mathbb{R}^{d_e},$$

where d_e is the feature dimension and $e^{1:n_e}$ can be viewed as a set of tokens (e.g., flattened patch-level features). Slot Attention then produces K slots $\xi^{1:K}$, where each slot is a vector in $\mathbb{R}^{d_\xi}$ intended to summarize one entity,

$$\xi^{1:K} = \texttt{Slot-Attn}(e^{1:n_e}),$$

where its computation is detailed as follows.

Slots are typically initialized as learned embeddings or as samples from a learned Gaussian, denoted by $\{\xi_0^i\}_{i=1}^K$. The module then iteratively refines these slots for T_{SA} steps. At each refinement step t, Slot Attention performs cross-attention with slots as queries and patch-level features as keys and values. It first projects slots and features to d_{SA}-dimensional queries, keys, and values (in this section, we overload notations: the subscripts (e.g., ξ_0^i) and t denote the slot attention steps, not the timestamps),

$$query_t^i = W_q \xi_t^i, \qquad key^j = W_k e^j, \qquad value^j = W_v e^j,$$

where W_q, W_k, and W_v are learned matrices. Attention logits are computed as dot products between queries and keys, and then normalized across slots for each feature,

$$\ell_t^{ij} = \frac{\langle query_t^i, key^j \rangle}{\sqrt{d_{\text{SA}}}},$$
$$\mathbf{W}_t^{ij} = \frac{\exp(\ell_t^{ij})}{\sum_{k=1}^K \exp(\ell_{kj}^t)}.$$

To update each slot, these weights are further normalized across input features, and each slot receives a weighted sum of values as an update. For simplicity, we omit the update refinement based on the gated recurrent unit (GRU) and multi-layer perceptron (MLP) in the standard implementation,

$$\bar{\mathbf{W}}_t^{ij} = \frac{\mathbf{W}_t^{ij}}{\sum_{k=1}^{n_e} \mathbf{W}_t^{ik}},$$
$$\xi_{t+1}^i = \xi_t^i + \sum_{j=1}^{n_e} \bar{\mathbf{W}}_t^{ij} \cdot value^j.$$

After T_{SA} iterations, the final slots $\{\xi_{T_{\text{SA}}}^i\}_{i=1}^K$ are used as an object-centric representation.

Slot Attention is commonly trained end-to-end via reconstruction. A decoder maps each slot to an image component and an (unnormalized) mask logit, and combines them via a weighted sum,

$$\hat{o}^i, \kappa^i = \texttt{Decoder}(\xi_{T_{\text{SA}}}^i),$$
$$\rho^i = \frac{\exp(\kappa^i)}{\sum_{j=1}^K \exp(\kappa^j)},$$
$$\hat{o} = \sum_{i=1}^K \rho^i \cdot \hat{o}^i,$$

where ρ^i represents how much the i-th slot contributes to the reconstruction of each pixel, and the reconstruction loss provides the learning signal.

Slot Attention encourages different slots to specialize to different regions/entities because overlapping explanations are penalized by the competitive slot-to-patch attention and the mask normalization.

2.8 Latent Action Models (LAMs)

Given a video dataset, we aim to model dynamics from observations alone, without any action labels. A Latent Action Model (LAM) learns an inverse dynamics model (IDM) to infer latent actions and a forward dynamics model (FDM) to predict the next observation (Schmidt and Jiang, 2024). The inferred latent action captures the most essential changes during a transition that cannot be predicted from the current observation alone, and thus usually correspond to entities' underlying actions. As a result, although they may not align exactly with real, physically meaningful actions, these latent actions can still provide useful information for downstream policy learning.

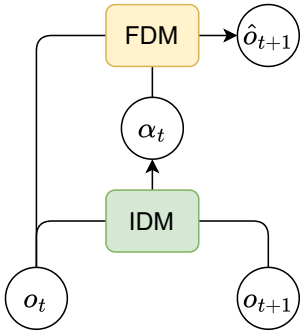


Figure 2.1: Latent action model.

As shown in Figure 2.1, both the IDM and the FDM observe o_t, but only the IDM observes o_{t+1}. Therefore, to accurately predict o_{t+1}, the IDM must extract useful transition information through the latent action α_t. However, without an explicit bottleneck on α_t, the model can collapse to a trivial solution where α_t simply copies o_{t+1}. To prevent this shortcut, prior work typically constrains the capacity of α_t, for example by discretizing it with vector quantization into a small

set of codes or by regularizing it toward a prior using a VAE-style objective (Kingma and Welling, 2014; van den Oord et al., 2017).

While latent action models provide a general framework for learning controllable world models from videos alone, they typically use a single latent action to represent the change of an entire scene, which becomes restrictive when multiple entities act independently. For example, in crowded intersections where many cars and pedestrians are driven by independent actions, learning to compress all entities' actions into one latent action can be challenging. In Chapter 8, we will discuss how factorization of the latent state space and latent action space can address this issue, and how it provides an alternative for extracting state factors from observations.

Summary This chapter introduced the RL problem setup, causality, and relevant subfields (model-based RL, state abstractions, intrinsic reward functions, unsupervised skill discovery, object-centric representations, and latent action models), which collectively equip the reader to understand the contributions described in Chapters 3-8. A more complete discussion of related work is provided in Chapter 9.

Chapter 3: Causal Dynamics Learning for Task-Independent Abstractions

The first challenge to improving RL generalization and sample efficiency is, when the state space consist of many state factors, how to enable agents to focus on task-relevant state factors through state abstraction $\phi : \mathcal{S} \to \bar{\mathcal{S}}$. In contrast to using all state factors, agents can learn more efficiently in a reduced state space and generalize better by disregarding irrelevant factors. To determine the set of task-relevant factors, it is essential to reason about the causal relationships within the transition dynamics (understanding how factors affect each other during state transitions).

This chapter is structured as follows. Section 3.1 first introduces the motivation behind state abstraction and limitations in previous work. Section 3.2 presents the proposed algorithm Causal Dynamics Learning (CDL) that aims to infer general causal relationships between state factors, assuming that state factorization is given. Then it derives a task-independent state abstraction from these causal relationships. Section 3.3 presents the empirical evaluation of CDL, covering its performance in causal discovery, dynamics learning, and task learning. Finally, Section 3.4 discusses limitations of CDL and how following chapters address them.

Contribution: In addition to my advisor, CDL is joint work with Xuesu Xiao, Zifan Xu, and Yuke Zhu. My contributions are the algorithm design and its implementation. Meanwhile, Xiao and Zhu provided technical advice and Xu helped with some of the experiments.

3.1 Motivation

Model-based Reinforcement Learning (MBRL) enables an agent to predict what would happen if it executed various actions, thus allowing it to learn from imagined experience (Hafner et al., 2019). However, such an approach relies on the model being learned accurately. Many model-based approaches rely on *dense* models that predict the next step value of each factor based on the action and all factors in the current state, as shown in Figure 3.1 (a). Such dense models are sensitive to spurious correlations which lead to poor generalization. For example, when door B is

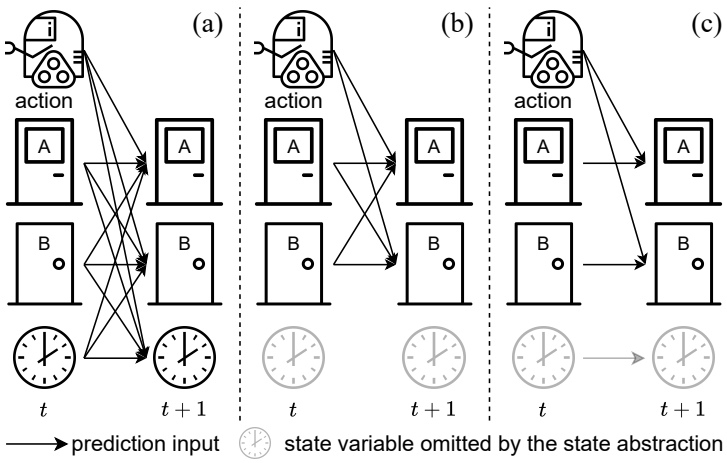


Figure 3.1: With two doors that the robot can open and go through and a clock on the wall, (a) *dense* dynamics models predict dynamics of each state factor unnecessarily using all factors; (b) based on a pre-defined reward (e.g., for navigation), existing state abstractions learn to omit the clock but still use a dense model for the remaining factors; (c) our *causal* models reason about and only keep necessary dependencies (i.e., doors A and B depend on the action individually) and derives a state abstraction independent from any reward function.

at angles unseen during training or the clock is at unseen times, the prediction of door A can be inaccurate due to unnecessary dependence on those factors in the model.

Observing that unnecessary dependencies are the source of poor generalization, existing state abstractions mitigate this problem by removing some of those dependencies. Specifically, state abstractions group many states into an abstract state by omitting some state factors (Chapman and Kaelbling, 1991; McCallum, 1996; Jong and Stone, 2005). For example, bisimulation (Zhang et al., 2021a), which is particularly related to this chapter, omits factors irrelevant to a pre-defined reward function. Though doing so removes unnecessary dependence on omitted factors, as shown in Figure 3.1 (b), the same generalization issues persist as dense models are still used for the remaining factors, leaving unnecessary dependencies in the abstract state. Furthermore, despite bisimulation improving sample efficiency of task learning by reducing the learning space, such methods only find problem-dependent abstractions: they may omit factors that are irrelevant to the current task but that the agent can control and utilize for future tasks.

Given that good generalization requires only keeping necessary dependencies, this chapter introduces Causal Dynamics Learning (CDL) for Task-Independent State Abstraction which learns

a *causal* model that explicitly reasons about which actions and factors affect which factors from collected data, as shown in Figure 3.1 (c). For example, by not depending on door B and the clock, the causal model's predictions about door A are unaffected by those factors and likely to be more accurate than dense models. Specifically, we prove that each factor's dependence on other factors (or actions) can be determined by a single conditional independence test. Such a test is then carried out by a novel architecture which estimates the conditional mutual information while learning the dynamics model.

Furthermore, by revealing all unnecessary dependencies, certain state factors which no other factors depend on (e.g., the clock) can be omitted for planning, forming a new form of state abstraction. Specifically, our model partitions state factors into those that it can change (*controllable factors*, e.g., doors A and B) with its actions, those that it cannot change but that influence actions' results on those that it can (*action-relevant factors*, e.g., an obstacle that may block door A's motion), and the remainder (*action-irrelevant factors*, e.g., the clock) which have no influence on others and thus can be omitted during planning. Also, in the abstract state, the dynamics model is still free of unnecessary dependencies and exhibits the same generalization benefits. Derived purely from dynamics, our state abstraction includes all controllable factors that the agent can use in the future, enabling it to solve a wider range of tasks than bisimulation which only retains factors specific to a single task.

CDL is compared against state-of-the-art dense models in two simulated environments: a chemical environment with causal relationships of different complexities, and a table-top manipulation environment with challenging rigid body dynamics. We find that CDL learns causal relationships accurately and retains similar prediction accuracy on unseen states to the accuracy on seen states, while the prediction accuracies of dense models drop $60 \sim 90\%$ in some complex environments. When applied to downstream tasks, policies with the proposed causal state abstraction learn with higher sample efficiency and also generalize better than those with dense models.

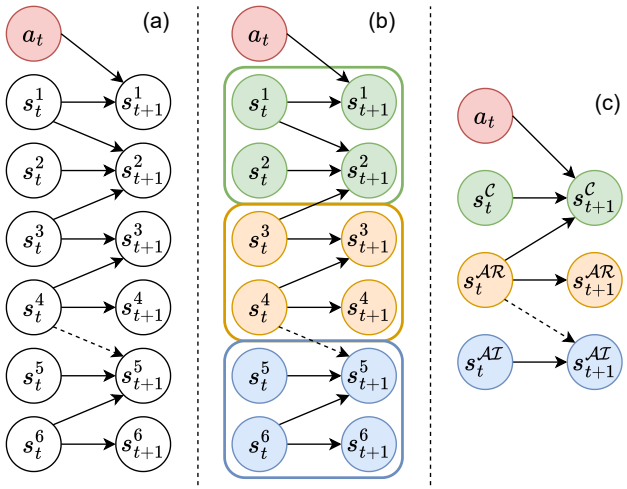


Figure 3.2: **(a)** an example causal dynamics model. **(b)** state factors can be split into three types: controllable (green), action-relevant (orange), and action-irrelevant (blue). The dashed arrow represents whether it exists does not affect $\mathcal{S}^5$ to be action-irrelevant. **(c)** the causal graph can be split into three subgraphs, one for each type of state factor.

3.2 Causal Dynamics Learning (CDL)

In this section, we introduce CDL, which learns a causal dynamics model and then derives a state abstraction from the learned causal relationships that supports learning a wide range of tasks.

3.2.1 Problem Definition

We begin with a formal definition of controllable, action-relevant, and action-irrelevant state factors. If we are given the causal graphical model of a Markov Process with $V = \{\mathcal{S}_t^{1:d_\mathcal{S}}, \mathcal{A}_t, \mathcal{S}_{t+1}^{1:d_\mathcal{S}}\}$ as nodes and E as edges describing causal relationships from $(\mathcal{S}_t^{1:d_\mathcal{S}}, \mathcal{A}_t)$ to $\mathcal{S}_{t+1}^{1:d_\mathcal{S}}$, ancestors of a state factor $\mathcal{S}^i$ are defined as all nodes that have a directed path leading to node $\mathcal{S}^i$ (not necessarily from the immediate previous time step but can be from any previous step). For example, for the causal dynamics model shown in Figure 3.2 (a), $\mathcal{S}^4$ is an ancestor of $\mathcal{S}^2$ as there is a path of $\mathcal{S}_t^4 \to \mathcal{S}_{t+1}^3 \to \mathcal{S}_{t+2}^2$. Descendants of nodes are defined in the same way but in the opposite direction. Then we have:

Definition 3.1 (Controllable State factors)**.** $\mathcal{S}^\mathcal{C}$ are the descendants of the action $\mathcal{A}_t$.

Definition 3.2 (Action-Relevant State factors)**.** $\mathcal{S}^{\mathcal{AR}}$ are ancestors of controllable state factors, excluding those already belonging to $\mathcal{S}^{\mathcal{C}}$.

Definition 3.3 (Action-Irrelevant State factors)**.** $\mathcal{S}^{\mathcal{AI}}$ are those factors that belong to neither $\mathcal{S}^{\mathcal{C}}$ nor $\mathcal{S}^{\mathcal{AR}}$.

In the above definitions, $\mathcal{C}, \mathcal{AR}$, and $\mathcal{AI}$ are the set of state dimension indices for controllable, action-relevant, and action-irrelevant state factors respectively.

Given these definitions, the type of each state factor in the example causal dynamics model is shown in Figure 3.2 (b). Further in (c), one may notice that the causal graph can be split into three parts, allowing us to rewrite the transition probabilities as $p(\mathcal{S}_{t+1} \mid \mathcal{S}_t, \mathcal{A}_t) = p(\mathcal{S}^{\mathcal{C}}_{t+1} \mid \mathcal{S}^{\mathcal{C}}_t, \mathcal{S}^{\mathcal{AR}}_t, \mathcal{A}_t) \cdot p(\mathcal{S}^{\mathcal{AR}}_{t+1} \mid \mathcal{S}^{\mathcal{AR}}_t) \cdot p(\mathcal{S}^{\mathcal{AI}}_{t+1} \mid \mathcal{S}^{\mathcal{AR}}_t, \mathcal{S}^{\mathcal{AI}}_t)$, where the edge from $\mathcal{S}^{\mathcal{AR}}_t$ to $\mathcal{S}^{\mathcal{AI}}_{t+1}$ is optional as it does not change the split of state factors ($\mathcal{S}^{\mathcal{AI}}$ are still neither $\mathcal{A}_t$'s descendants nor $\mathcal{S}^{\mathcal{C}}$'s ancestors).

Then CDL forms the state abstraction ϕ by omitting action-irrelevant state factors, i.e., $\phi(\mathcal{S}_t) = (\mathcal{S}^{\mathcal{C}}_t, \mathcal{S}^{\mathcal{AR}}_t)$, and the dynamics in the abstract space can be expressed by removing the subgraph involving action-irrelevant state factors, remaining a causal dynamics model itself, as follows:

$$p(\phi(\mathcal{S}_{t+1}) | \phi(\mathcal{S}_t), \mathcal{A}_t) = p(\mathcal{S}^{\mathcal{C}}_{t+1} \mid \mathcal{S}^{\mathcal{C}}_t, \mathcal{S}^{\mathcal{AR}}_t, \mathcal{A}_t) \cdot p(\mathcal{S}^{\mathcal{AR}}_{t+1} \mid \mathcal{S}^{\mathcal{AR}}_t). \tag{3.1}$$

This state abstraction ϕ can be used to solve any *actively-accomplishable* downstream task. Here, downstream tasks are tasks defined in the same Markov Process so that the agent can use ϕ to solve the task by learning the provided rewards. Meanwhile, actively-accomplishable means that the reward function of the task only depends on controllable and action-related state factors $(\mathcal{S}^{\mathcal{C}}, \mathcal{S}^{\mathcal{AR}})$. Additionally, if the task involves extra variables, e.g., a goal value for certain controllable state factors to reach, those factors should also be provided so that the agent can learn the reward accurately. Notice that actively-accomplishable tasks do not cover those involving action-irrelevant state factors (e.g., for the example in Figure 3.1, a reward of $+1$ for opening door A when the clock shows 1 pm and 0 otherwise). However, as the reward function of a task can be arbitrarily designed using any state factor, being able to solve all tasks means that no state factor can be omitted (i.e., no

state abstraction). We assume that in practice, it is relatively uncommon for a task's reward function to involve action-irrelevant state factors, making ϕ fairly generally applicable.

So far, we have defined three types of state factors and the derived state abstraction for a known causal dynamics model. However, for real-world problems, such a model is usually not accessible. Instead, agents can only collect transition data via its interactions with the environment. Hence, as presented in Algorithm 1, this chapter introduces CDL, a novel method that: (1) learns a **causal** dynamics model $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, (2) learns a data collection policy $\pi : \mathcal{S} \to \mathcal{A}$ to learn f efficiently, (3) derives the state abstraction $\phi : \mathcal{S} \to \mathcal{S}^{\mathcal{C}} \times \mathcal{S}^{\mathcal{AR}}$ and dynamics f^{ϕ} in the abstract space, (4) learns a reward predictor for any actively-accomplishable task in the abstract space $\hat{\mathcal{R}} : \phi(\mathcal{S}) \times \mathcal{A} \to \mathbb{R}$, and (5) uses planning methods to solve the task with the learned f^{ϕ} and $\hat{\mathcal{R}}$.

3.2.2 Causal Dynamics Model Learning

The key challenge when learning a causal dynamics model is to determine whether a causal edge exists between two state factors, i.e., $\mathcal{S}_t^i \to \mathcal{S}_{t+1}^j$. First, adapting the work from Mastakouri et al. (2021), we present a method for inferring whether such a causal relationship exists, based on the following assumptions about the ground truth dynamics model:

Assumption 1. *The ground-truth transition dynamics and reward function are causal Markovian (Definition 2.2) and faithful (Definition 2.3).*

Assumption 2. *The state is fully observable and the state factorization is given.*

Assumption 3. *The ground-truth transition dynamics and reward function are Markovian (i.e., $\mathcal{S}_{t+1}$ and $\mathcal{R}_t$ only depend on $(\mathcal{S}_t, \mathcal{A}_t)$, and they do not depend on historical states $\mathcal{S}_{<t}$ nor historical actions $\mathcal{A}_{<t}$).*

Assumption 4. *The causal relationship $\mathcal{S}_t^i \to \mathcal{S}_{t+1}^i$ exists for all state factors $\mathcal{S}^i$.*

Assumption 5. *There is no simultaneous causal relationship, i.e., $\forall i, j, \mathcal{S}_t^i \not\to \mathcal{S}_t^j$.*

Assumption 6. *The transitions for each state factor are independent, i.e., $p(\mathcal{S}_{t+1} \mid \mathcal{S}_t, \mathcal{A}_t) = \prod_{i=1}^{d_{\mathcal{S}}} p(\mathcal{S}_{t+1}^i \mid \mathcal{S}_t, \mathcal{A}_t)$.*

Except for Assumption 5, which requires no redundant information in state factors (e.g., state factors that include both joint angles and the end-effector of a robot arm), and Assumption 6, which does not necessarily hold for rich observation spaces (e.g., images), these assumptions are commonly made for causal inference and dynamical systems. Moreover, for partially observable or high-dimensional state spaces, low-dimensional disentangled representations that encode the space can be learned to adhere to Assumption 2 and Assumption 6.

With the above assumptions, we prove that the dynamics causal relationship $\mathcal{S}_t^i \to \mathcal{S}_{t+1}^j$ can be evaluated by one CIT.

Theorem 3.1. *With Assumptions 1-6, denoting $X_t^{\neg i} := \{\mathcal{A}_t, \mathcal{S}_t \setminus \mathcal{S}_t^i\}$, for any two state factors $\mathcal{S}^i$ and $\mathcal{S}^j$, if $\mathcal{S}_t^i \not\perp\!\!\!\perp \mathcal{S}_{t+1}^j \,|X_t^{\neg i}$, then $\mathcal{S}_t^i \to \mathcal{S}_{t+1}^j$. Similarly, if $\mathcal{S}_t^i \not\perp\!\!\!\perp \mathcal{R}_t \,|X_t^{\neg i}$, then $\mathcal{S}_t^i \to \mathcal{R}_t$.*

Proof. We need to show that if $\mathcal{S}^i$ is not a direct cause of $\mathcal{S}_{t+1}^j$, then the conditional independence $\mathcal{S}_t^i \perp\!\!\!\perp \mathcal{S}_{t+1}^j \,|X_t^{\neg i}$ holds or the assumption is violated.

Assume there is no direct edge from $\mathcal{S}_t^i$ to $\mathcal{S}_{t+1}^j$ but they are dependent, i.e., $\mathcal{S}_t^i \not\to \mathcal{S}_{t+1}^j$ and $\mathcal{S}_t^i \not\perp\!\!\!\perp \mathcal{S}_{t+1}^j$, then there is a confounder $Q_{t'}$ with the confounding path $\mathcal{S}_t^i \dashleftarrow Q_{t'} \dashrightarrow \mathcal{S}_{t+1}^j$, where $t' < t$ as we assume there is no simultaneous edge. There are two cases for $Q_{t'}$:
(1) If $Q_{t'}$ is observed, i.e., $Q_{t'}$ is one of the state factors, then it violates the condition because we condition on $X_t^{\neg i}$ which blocks all possible paths for $Q_{t'} \dashrightarrow \mathcal{S}_{t+1}^j$ and thus d-separates $\mathcal{S}_t^i$ and $\mathcal{S}_{t+1}^j$.
(2) If $Q_{t'}$ is unobserved, then it must be memoryless (i.e., $Q_t \not\to Q_{t+1}$) to adhere to Assumption 3 (the state is fully observable and the dynamics is Markovian). In that case, there exists such a path $\mathcal{S}_{t-1}^i \to \mathcal{S}_t^i \dashleftarrow Q_{t'} \dashrightarrow \mathcal{S}_{t+1}^j$ given Assumption 4 ($\mathcal{S}_t^i \to \mathcal{S}_{t+1}^i$ exists for all state factors $\mathcal{S}^i$). This path suggests $\mathcal{S}_{t-1}^i \not\perp\!\!\!\perp \mathcal{S}_{t+1}^j \,|\, \mathcal{S}_t^i$ where $\mathcal{S}_t^i$ serves as the collider connecting $\mathcal{S}_{t-1}^i$ and $\mathcal{S}_{t+1}^j$. If we further condition the dependence on all other state factors at t to block all potential paths $\mathcal{S}_{t-1}^i \to \mathcal{S}_t^k \to \mathcal{S}_{t+1}^j$, we have $\mathcal{S}_{t-1}^i \not\perp\!\!\!\perp \mathcal{S}_{t+1}^j \,|\, \mathcal{S}_t$. However this conditional dependence violates Assumption 3, i.e., Markovian property of the dynamics $\mathcal{S}_{t-1} \perp\!\!\!\perp \mathcal{S}_{t+1} \,|\, \mathcal{S}_t$.

In both cases either the condition or Assumption 3 is violated. Therefore we show that if $\mathcal{S}_t^i \not\perp\!\!\!\perp \mathcal{S}_{t+1}^j \,|X_t^{\neg i}$, then $\mathcal{S}_t^i \to \mathcal{S}_{t+1}^j$. □

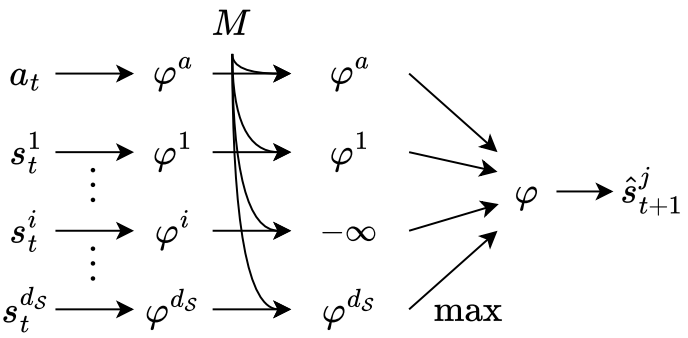


Figure 3.3: The predictive model for the state factor $\mathcal{S}_{t+1}^j$. Different conditional densities can be represented by applying different masks M. $\hat{p}(\mathcal{S}_{t+1}^j \,|X_t^{\neg i})$ is shown as an example in the figure.

For simplicity, in the remainder of the chapter, we will only describe the CIT between two state factors, $\mathcal{S}_t^i \not\perp\!\!\!\perp \mathcal{S}_{t+1}^j \,|X_t^{\neg i}$. For the test between the action and the state factor, $\mathcal{A}_t \not\perp\!\!\!\perp \mathcal{S}_{t+1}^j \,|\, \mathcal{S}_t$, the same method applies by changing $\mathcal{S}_t^i$ to $\mathcal{A}_t$ and the conditioning set according to Theorem 3.1.

The CIT between $\mathcal{S}_t^i$ and $\mathcal{S}_{t+1}^j$ can be made by measuring the Conditional Mutual Information, CMI^{ij}:

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\mathcal{S}_t,\mathcal{A}_t,\mathcal{S}_{t+1}^j}\left[\log \frac{p(\mathcal{S}_t^i, \mathcal{S}_{t+1}^j \,|X_t^{\neg i})}{p(\mathcal{S}_t^i \,|X_t^{\neg i})\; p(\mathcal{S}_{t+1}^j \,|X_t^{\neg i})}\right] \\
&= \mathop{\mathbb{E}}_{\mathcal{S}_t,\mathcal{A}_t,\mathcal{S}_{t+1}^j}\left[\log \frac{p(\mathcal{S}_t^i \,|X_t^{\neg i})\; p(\mathcal{S}_{t+1}^j \,|\{\mathcal{A}_t, \mathcal{S}_t\})}{p(\mathcal{S}_t^i \,|X_t^{\neg i})\; p(\mathcal{S}_{t+1}^j \,|X_t^{\neg i})}\right] && \text{// by expanding } p(\mathcal{S}_t^i, \mathcal{S}_{t+1}^j \,|X_t^{\neg i}) \\
&= \mathop{\mathbb{E}}_{\mathcal{S}_t,\mathcal{A}_t,\mathcal{S}_{t+1}^j}\left[\log \frac{p(\mathcal{S}_{t+1}^j \,|\, \mathcal{A}_t, \mathcal{S}_t)}{p(\mathcal{S}_{t+1}^j \,|X_t^{\neg i})}\right]. && \text{// by canceling out } p(\mathcal{S}_t^i \,|X_t^{\neg i}) \qquad (3.2)
\end{aligned}
$$

where the expectation is over the joint distribution of $\{\mathcal{S}_t, \mathcal{A}_t, \mathcal{S}_{t+1}^j\}$ and all p are the ground truth conditional densities. If $\text{CMI}^{ij} \geq \epsilon_{\mathcal{G}}$ where $\epsilon_{\mathcal{G}}$ is a pre-defined threshold, it suggests that $\mathcal{S}_t^i$ is necessary to predict $\mathcal{S}_{t+1}^j$ accurately, $\mathcal{S}_t^i \not\perp\!\!\!\perp \mathcal{S}_{t+1}^j \,|X_t^{\neg i}$ is true, and the causal edge $\mathcal{S}_t^i \to \mathcal{S}_{t+1}^j$ exists.

In practice, as the ground truth joint distribution and conditional densities are unknown, the expectation is computed over the collected transition data and CDL learns predictive models $\hat{p}(\mathcal{S}_{t+1}^j \,|\, \mathcal{A}_t, \mathcal{S}_t), \hat{p}(\mathcal{S}_{t+1}^j \,|X_t^{\neg i})$ to approximate conditional densities.

However, computing CMI^{ij} for all state factor pairs requires training $(d_{\mathcal{S}})^2$ predictive models, which is intractable. Instead, we develop a novel architecture and training paradigm which reduces

the requirement to $d_{\mathcal{S}}$ models, where each model learns to predict of state factor s^j_{t+1}. For simplicity, throughout this section, when it is clear that the symbols pertain to the model predicting s^j_{t+1}, we omit the subscript j for brevity. As shown in Figure 3.3, to predict each state factor s^j_{t+1}: (1) the action and all current state factors are individually mapped by a separate network to features $\varphi^a, \varphi^1, \ldots, \varphi^{d_{\mathcal{S}}}$ where all features share the same size; (2) certain features are masked to $-\infty$ according to a binary mask $M \in \{0,1\}^{d_{\mathcal{S}}+1}$; (3) the aggregated feature φ is computed by taking the element-wise max of all features; and (4) a predictive network takes φ as input and predicts the distribution of s^j_{t+1}. This architecture can represent conditional densities with different masks M: (1) for $\hat{p}(\mathcal{S}^j_{t+1} \,|\, \mathcal{A}_t, \mathcal{S}_t)$, none of the features is masked; (2) for $\hat{p}(\mathcal{S}^j_{t+1} \,|X^{\neg i}_t)$, the feature φ^i is masked to $-\infty$ so that it will not be used to predict s^j_{t+1}, as shown in Figure 3.3; and (3) for $\hat{p}(\mathcal{S}^j_{t+1} \,|\mathbf{PA}(\mathcal{S}^j))$, after deriving the parents of $\mathcal{S}^j_{t+1}$ from $\{\text{CMI}^{ij}\}_{i=1}^{d_{\mathcal{S}}}$, all non-parent features are masked to $-\infty$.

The architecture described above predicts one state factor $\mathcal{S}^j_{t+1}$, and the whole causal dynamics model f consists of $d_{\mathcal{S}}$ such models. To train f, we maximize the following log-likelihood:

$$L_f = \sum_{j=1}^{d_{\mathcal{S}}} \big[\log \hat{p}(s^j_{t+1}|a_t, s_t) + \log \hat{p}(s^j_{t+1}|x^{\neg i}_t) + \log \hat{p}(s^j_{t+1}|\mathbf{PA}(\overset{j}{\mathcal{S}})) \big], \tag{3.3}$$

where the state factor to be masked i is uniformly sampled from $\{1, \ldots, d_{\mathcal{S}}\}$ for each j, and $\mathbf{PA}(\mathcal{S}^j)$ are inferred from $\{\text{CMI}^{ij}\}_{i=1}^{d_{\mathcal{S}}}$ learned so far. In Eq. 3.3, the first two predictive likelihoods train the models necessary to evaluate CMI^{ij}, and the last prediction likelihood finetunes the performance of the inferred causal dynamics model $\hat{p}(\mathcal{S}^j_{t+1} \,|\mathbf{PA}(\mathcal{S}^j))$. We split the collected transition data into the training part used to learn f and the validation part for evaluating CMI.

After training, CDL evaluates the causal graph by checking if each $\text{CMI}^{ij} \geq \epsilon_{\mathcal{G}}$ and derives the learned state abstraction $\phi(\mathcal{S}) = (\mathcal{S}^{\mathcal{C}}, \mathcal{S}^{\mathcal{AR}})$ from the learned causal graph, according to Definitions 3.1-3.3. The dynamics in the abstract space f^ϕ can also be derived by omitting the prediction networks for action-irrelevant state factors $\hat{s}^{\mathcal{AI}}$.

Algorithm 1 Causal Dynamics Learning (CDL)

1: Initialize the dynamics model f, the data collection policy π, and the reward predictor $\hat{\mathcal{R}}$.
2: // Learn the causal dynamics model and derive the state abstraction.
3: **for** number of training steps **do**
4: Collect (s_t, a_t, s_{t+1}) with $a_t \sim \pi$.
5: Update f with the prediction loss in Eq. 3.3.
6: Update π to maximize the reward in Eq. 3.4.
7: **end for**
8: Evaluate dynamics causal graph following Eq. 3.2 and derive the state abstraction following Section 3.2.1.
9: // Learn downstream tasks.
10: **for** number of training steps **do**
11: Collect (s_t, a_t, s_{t+1}) with a_t from the planning algorithm.
12: Update $\hat{\mathcal{R}}$ with the prediction loss in Eq. 3.5.
13: **end for**

3.2.3 Policy Learning for Data Collection

The collected transition data are important to learning accurate causal dynamics models, as they are used in predictive model training (Eq. 3.3) and CMI evaluations (Eq. 3.2). An ideal data collection policy π would cover all state-action pairs to expose causal relationships thoroughly and actively explore states where the causal dynamics model is not accurate.

To this end, in Algorithm 1 line 6, an RL agent is used for data collection with a reward measuring the prediction difference between the dense predictor and the causal predictor learned so far:

$$r_t = \tanh\left(\eta \cdot \sum_{j=1}^{d_{\mathcal{S}}} \log \frac{\hat{p}(s_{t+1}^j | s_t, a_t)}{\hat{p}(s_{t+1}^j | \mathbf{PA}(\mathcal{S}^j))}\right), \tag{3.4}$$

where η is the scaling factor and $\tanh$ is used to keep the reward bounded. This reward motivates taking transitions where the dense predictor is better than the causal predictor, which usually suggests the learned causal graph is inaccurate.

3.2.4 Policy Learning for Downstream Tasks

When solving actively-accomplishable downstream tasks, in Algorithm 1 lines 10 -13, like many MBRL algorithms, CDL simultaneously (1) learns a reward predictor with the abstract state, action, and any provided extra factors as input, $\hat{\mathcal{R}} : \phi(\mathcal{S}) \times \mathcal{A} \to \mathbb{R}$, and (2) uses a planning algorithm with the learned dynamics and reward predictor for action selection. As the reward predictor is learned in an abstract space rather than the full state space, it is more sample-efficient and less vulnerable to spurious correlations brought about by excessive information (i.e., action-irrelevant state factors). Meanwhile, planning in the abstract space also reduces the computation cost by relieving the need to roll out action-irrelevant state factors.

The reward predictor is modeled as a neural network and trained by minimizing the following prediction loss,

$$L_{\hat{\mathcal{R}}} = \mathbb{E}_{(s_t,a_t,r_t)} \|\hat{\mathcal{R}}(\phi(s_t), a_t), r_t\|, \tag{3.5}$$

where $\|\cdot\|$ can take any loss function (we experiment with the absolute value of the prediction error).

For planning, we use the cross entropy method (CEM), a population-based optimization algorithm (Rubinstein, 1997), to search for the best action with the learned dynamics and reward predictor. For each time step t, depending on whether the action is continuous or discrete, CEM initializes a time-dependent belief over the optimal action sequence $a_{t:t+L} \sim \mathcal{N}(\mu_{t:t+L}, \sigma^2_{t:t+L})$ or Categorical$(\omega_{t:t+L})$ where L is the planning length. Starting from a unit normal distribution for $\mathcal{N}$ or a uniform distribution for Categorical, it repeatedly samples multiple candidate action sequences, evaluates them based on cumulative rewards, and updates the action belief to the distribution of the top candidates. After several iterations, the planner returns μ_t or $\arg\max \omega_t$ as the current optimal action.

3.3 Experiments

In this section, we provide empirical evaluations pertaining to the following questions that support the indicted answers.

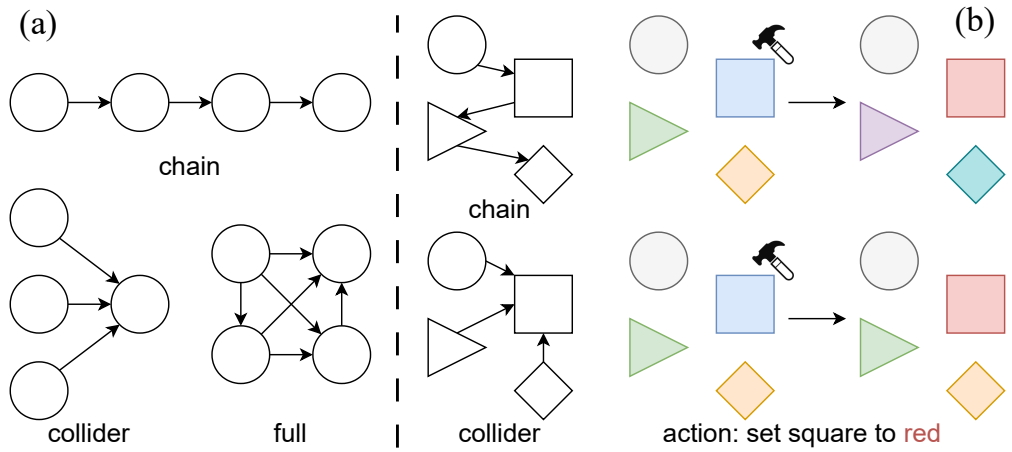


Figure 3.4: **(a)** different types of causal graphs. **(b)** illustration of the chemical environment (left: ground truth causal graphs, right: transitions after applying the action).

- **Q1)** Can CDL infer the causal graph, with less tuning and better accuracy than a regularization-based method (Wang et al., 2021c)? Yes (Section 3.3.3).
- **Q2)** Can causal models learned by CDL generalize better than dense models in both dynamics and downstream task learning on unseen states? Yes (Section 3.3.3).
- **Q3)** Can the transition collection policy learned by CDL improves the accuracy of causal dynamics learning compared to a uniform policy or the policy learned from the curiosity reward (Pathak et al., 2017)? Yes. (Section 3.3.4).
- **Q4)** Can state abstractions derived by CDL improve sample-efficiency when learning downstream tasks compared to dense models which use the full state space? Yes (Section 3.3.5).
- **Q5)** Can policies learned by CDL for downstream tasks generalize better than those learned by dense models? Yes (Section 3.3.5).

3.3.1 Environments

We use (1) the Chemical environment modified from Ke et al. (2021) (Figure 3.4 (b)) to examine CDL's performance on causal graphs with different complexities, and (2) a manipulation environment (Figure 3.5) to validate CDL with complex rigid-body dynamics.

In the chemical environment modified from Ke et al. (2021), there are 10 objects and each of them has its different x, y position (two continuous factors) and color (a categorical factor out of 5 possible colors), forming 30 state factors in total. At each step, the x, y positions of objects are sampled from a normal distribution $\mathcal{N}(0, 1e^{-4})$. Meanwhile, the action changes a chosen object to a specified new color (a categorical factor out of 50 possible actions). After the chosen object changes its color, all its decadents in the underlying graph will change their colors in the breadth-first order. When changing to its new color, each descendant follows a pre-defined conditional probability table modeled as a randomly initialized multi-layer perceptron (MLP). The MLP takes the newest color of each parent as inputs and outputs a categorical distribution over 5 colors out of which one color is sampled for the object. For example, in the chain graph shown in Figure 3.4 (a) top, if the color of the leftmost object is changed by the action, then the second left object will change its color given the new color of the leftmost object changes to, then the third object changes, etc. We consider three graph structures shown in Figure 3.4 (a): full, chain, and collider.

In the ground truth causal graph for the chemical environment, each object's x position or y position only depends on itself and thus is action-irrelevant, while the color depends on the action and the colors of its parents and thus is controllable. However, if an object only has one parent, then its color doesn't depend on the color of that parent, as whenever its color needs to change, it's either (1) because the color is directly set by the action where the parent color has no effect or; (2) because that its only parent changes the color, and this new parent color that it is going to depend on is not included in s_t (s_t only includes the parent color before the change). As a result, there is no dependence from the parent color at t to its new color at $t + 1$.

In the manipulation environment implemented with the `robosuite` simulation framework (Zhu et al., 2020b), there are three objects that the robot can interact with: a *movable* red cube that the robot can manipulate (denoted as mov), an *unmovable* green cube that is fixed on the table (unm), and a *randomly moving* blue cube that also cannot be moved ($rand$). The purpose of the randomly moving object is to test whether CDL can learn that its motion is not caused by the robot's actions. There are also three randomly moving non-interactable black markers (mkr^1, mkr^2, and mkr^3) on the table, as distractors to introduce potential spurious correlations. The state space consists of the robot end-effector (EEF) location ($\mathbb{R}^3$), gripper (grp) joint angles ($\mathbb{R}^2$), and

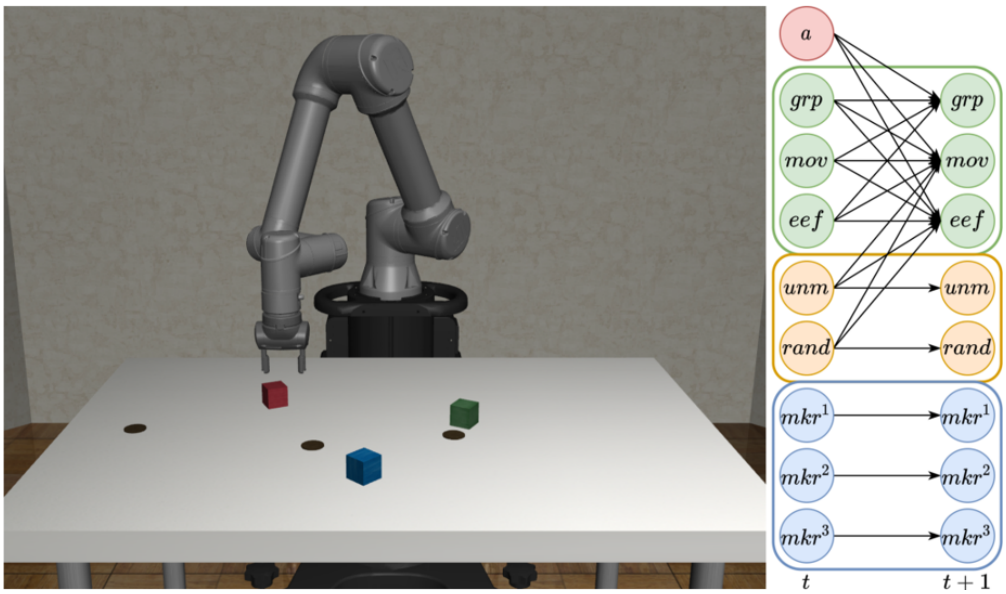


Figure 3.5: **(Left)** the manipulation environment. **(Right)** the learned causal graph in the object level which recovers the ground truth relationship between the state factors and the action.

locations of objects and markers ($6 \times \mathbb{R}^3$). The action space includes EEF location displacement ($\mathbb{R}^3$) and the degree to which the gripper is opened ($[0, 1]$). In each episode, the objects and markers are reset to randomly sampled poses on the table.

For the ground truth causal graph of the manipulation environment, the controllable state factors include EEF position, gripper joint angles, and movable object positions; the action-relevant state factors are positions of the unmovable object and the randomly moving object as they can block the motion of the EEF, gripper, and the movable object (when pushed or held by the gripper); and action-irrelevant state factors include positions of all non-interactable markers.

3.3.2 Implementation Details

Baselines We compare CDL with the following baselines:

- **Reg**: regularization-based causal dynamics learning (Wang et al., 2021c), where the causal mask M is learned as trainable parameters rather than inferred from CMI.
- **Modular**: using a separate network to predict each state factor (Ke et al., 2021), which can

be represented by our method when none of the features are masked.

- **GNN**: a graph neural network, following (Ke et al., 2021), we consider the C-SWM model (Kipf et al., 2020) which assumes dense dependence for each state factor.
- **Monolithic**: a multi-layer perceptron (MLP) network that takes all state factors and actions as inputs and predicts the next step values of all state factors, which is commonly used in MBRL.

For a fair comparison, we use the same architecture for CDL, Reg, and Modular. For GNN and Monolithic whose architectures are different from the others, we try our best to give them the same number of parameters as others.

Causal Dynamics Learning We use MLP networks to extract features from the action and state factors $\varphi^a, \varphi^1, \ldots, \varphi^{d_S}$ and to predict the distribution of s^j_{t+1}. For continuous state factor s^j_{t+1}, the network predicts the mean and standard deviation of a normal distribution. For discrete state factor, it outputs the logits of a categorical distribution.

For CDL, after training the dynamics model and evaluating CMI, we derive the causal graph by examining whether CMI $\geq \epsilon_{\mathcal{G}}$ for each edge. The threshold $\epsilon_{\mathcal{G}}$ is an important hyperparameter that determines the accuracy of the derived causal graph and thus the accuracy of the dynamics model. Even though we can use the same value used for training, as the best threshold is unknown before training, that value is just roughly estimated from several runs and may not be the best threshold. As a result, to select the best threshold, we conduct a linear search and select the one that gives the best F1 score. For a fair comparison, we apply the same threshold selection method to the regularization-based causal dynamics learning (Reg): Reg learns each edge of the causal graph as an individual Bernoulli distribution where its success probability is the learnable parameter. During training, as a rule-of-thumb, we use 0.5 as the threshold when judging whether an edge exists for each sample. After training and when evaluating the causal graph based on the learned success probability, we conduct a linear search and again select the threshold that gives the best F1 score.

Meanwhile, during CDL training, we find that reducing the frequency of optimizing causal prediction likelihood (i.e., $\log \hat{p}(s_{t+1}^j | \mathbf{PA}(\mathcal{S}^j))$ in Eq. 3.3) can stabilize training, so we only include this term once every 10 updates of the dynamics model according to Eq. 3.3 .

Data Collection Policy For the simple chemical environment, we use a random policy to collect data as it is enough to cover most of the state-action pairs. For the manipulation environment, we use Proximal Policy Optimization (PPO) as our RL agent (Schulman et al., 2017). Meanwhile, as exploring the whole state space for manipulation domains is still a challenging research topic and not the focus of this chapter, to reduce the complexity of exploration, we follow Nasiriany et al. (2022) and modify PPO's action space to long-horizon skills, such as position reaching and object grasping/pushing. Notice that dynamics learning still uses low-level raw motor actions described in Section 3.3.1.

3.3.3 Causal Dynamics Learning Evaluation

After training all methods on the same set of collected transition data, we evaluate the quality of their learned dynamics in the following aspects:

- *Causal Relationship Inference*. We compare the causal graph inferred by CDL and Reg with the ground truth graph, in terms of edge accuracy.
- *Predicting Future States*. Given the current state and a sequence of actions, we evaluate the accuracy of each method's multi-step prediction, for states both in and out of the training distribution (denoted as ID and OOD states). For the chemical environment, we evaluate color prediction in terms of mean accuracy (MA). For manipulation, log-likelihoods is used.

In the remainder of the chapter, the performances shown for each method are mean and standard deviation computed from 3 models with different seeds.

Causal Relationship Inference After deriving the causal graph for each run with its best threshold, we evaluate the performance of CDL and Reg on causal graph learning, in terms of accuracy, recall,

Table 3.1: Performance on Causal Graph Learning for CDL and Reg on All Environments

Metrics	Method	Environments			
		Chemical (collider)	Chemical (chain)	Chemical (full)	Manipulation
Accuracy (↑)	CDL	**1.000** ± 0.000	**1.000** ± 0.001	**0.991** ± 0.001	**0.902** ± 0.003
	Reg	0.994 ± 0.004	0.997 ± 0.001	0.977 ± 0.004	0.844 ± 0.005
Recall (↑)	CDL	**1.000** ± 0.000	**0.992** ± 0.012	**0.917** ± 0.016	**0.612** ± 0.009
	Reg	0.891 ± 0.086	0.942 ± 0.012	0.794 ± 0.006	0.459 ± 0.048
Precision (↑)	CDL	1.000 ± 0.000	1.000 ± 0.000	**0.968** ± 0.016	**0.980** ± 0.006
	Reg	0.993 ± 0.010	0.991 ± 0.012	0.917 ± 0.051	0.839 ± 0.068
F1 Score (↑)	CDL	**1.000** ± 0.000	**0.996** ± 0.006	**0.941** ± 0.009	**0.754** ± 0.008
	Reg	0.937 ± 0.046	0.966 ± 0.006	0.850 ± 0.019	0.589 ± 0.025
ROC AUC (↑)	CDL	**1.000** ± 0.000	0.996 ± 0.006	0.958 ± 0.008	0.805 ± 0.005
	Reg	0.932 ± 0.042	0.990 ± 0.009	0.956 ± 0.008	**0.893** ± 0.002

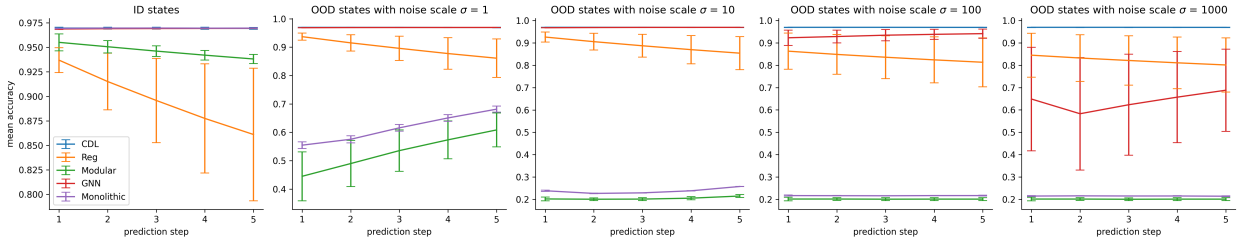


Figure 3.6: Multi-step prediction performance for the chemical environment with the **collider** graph. **(leftmost)** prediction on ID states. **(others)** prediction on OOD states with increasing noise scale σ.

precision, F1 score, and area under the receiver operating characteristic curve (ROC AUC). As shown in Table 3.1, in all environments, CDL learns the causal graph more accurately than Reg. Other baselines are not compared as they do not identify causal relationships, thus answering Question **Q1** in the affirmative. Figure 3.5 right shows the causal graph learned by CDL for the manipulation environment in the object level.

Predicting Future States We evaluate each method for $1 \sim 5$-step prediction on 5K transitions, for both ID and OOD states. To create OOD states, we change object positions in the chemical environment and marker positions in the manipulation environment to unseen values, and the

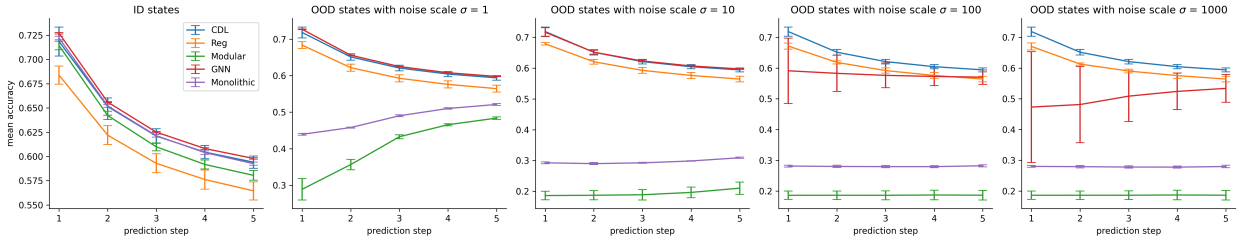


Figure 3.7: Multi-step prediction performance for the chemical environment with the **chain** graph. **(leftmost)** prediction on ID states. **(others)** prediction on OOD states with increasing noise scale σ.

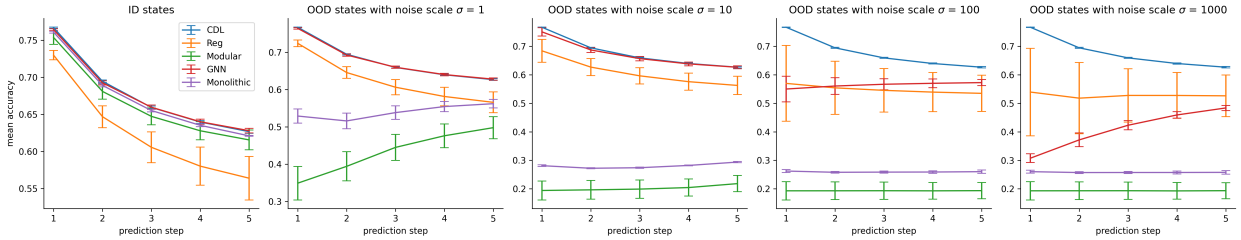


Figure 3.8: Multi-step prediction performance for the chemical environment with the **full** graph. **(leftmost)** prediction on ID states. **(others)** prediction on OOD states with increasing noise scale σ.

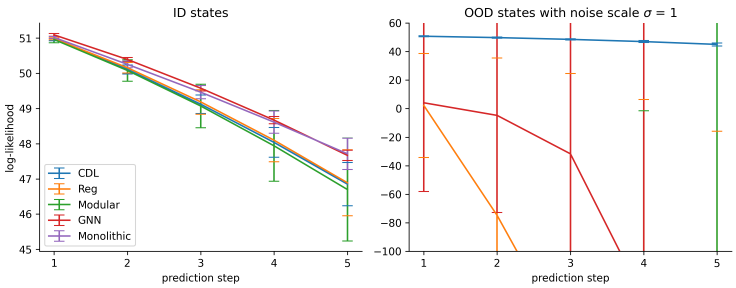


Figure 3.9: Multi-step prediction performance for the manipulation environment. **(leftmost)** prediction on ID states. **(others)** prediction on OOD states with noise scale $\sigma = 1$.

Table 3.2: Causal Graph Accuracy for CDL with Different Transition Collection Policies

Method	**PredDiff** (Ours)	**Uniform**	**Curiosity**
Accuracy $(\%)$	**90.2** $\pm$ 0.3	89.1 $\pm$ 0.6	88.9 $\pm$ 0.2

prediction is measured on other unchanged state factors. The 1-step prediction performance for each method is shown in Figure 3.6-3.9 (the Modular and Monolithic in Figure 3.9 are cropped due to their bad performances). For ID states, all methods show similar prediction performance. However, for OOD states, the performance of dense models drops significantly as they suffer from spurious correlations while CDL and Reg mask those out with the learned dependencies. However, Reg also performs badly in the manipulation domain, because the trade-off between prediction accuracy and regularization in its learning objective leads to an inaccurate causal graph. Combined with results in Section 3.3.5, results from Figures 3.6-3.9 answers Question **Q2** in the affirmative.

3.3.4 Transition Collection Policy Learning Evaluation

To evaluate the benefits of the proposed transition collection policy (denoted as **PredDiff**), we compare its performance with the following methods, in terms of causal graph accuracy in the manipulation environment:

- **Uniform**: a fixed policy that uniformly selects between skills and skill parameters instead of active exploration,
- **Curiosity**: a policy that uses the prediction error of the causal predictor $\hat{p}(s^j_{t+1}|\mathbf{PA}(\mathcal{S}^j))$ as rewards which encourages transitions where prediction errors are high (Pathak et al., 2017);

Table 3.2 shows the accuracies of the causal graphs learned by CDL from the same amount of data (1.8M transitions) but collected by those three different policies, measured in the same way as described in Section 3.3.3. Our PredDiff policy learns the graph with the highest accuracy, thus answering Question **Q3** in the affirmative. Though the accuracy differences between methods seem small, in a causal graph representing complex rigid body dynamics with 23×24 potential dependencies, 1% accuracy difference means prediction errors for $5 \sim 6$ edges, which may severely

harm the generalizability of the learned causal dynamics model. Specifically, causal graphs learned by Uniform policy have more spurious correlations as it doesn't actively explore transitions where the causal graph is not learned well, and the Curiosity policy suffers from procrastination (i.e., repeats the transition with large stochasticity to maximize rewards) and fails to expose other causal dependencies.

3.3.5 Downstream Tasks Learning Evaluation

We evaluate each method with the following downstream tasks in the chemical environment (C) and in the manipulation environment (M):

- **Match** (C): match the object colors with goal colors individually,
- **Reach** (M): move the end-effector to the goal position,
- **Pick** (M): pick the movable object to the goal position,
- **Stack** (M): stack the movable object on the top of the unmovable object.

The tasks' reward functions are unknown to the agent. During training, the dynamics model is frozen and only the reward predictor is learned as described in Section 3.2.4.

Compared to dense baselines, CDL and Reg also learn a causal graph so that they can learn tasks with the state abstraction described in Section 3.2.1. The training curves for all tasks are shown in Figure 3.10, answering Question **Q4** in the affirmative. Specifically, the pick and stack tasks, CDL learns tasks the most sample-efficiently with the state abstraction. Though Reg also uses abstraction, it learns more slowly as some action-irrelevant state factors are also included in the abstraction because of the inaccurately learned causal graphs. Compared to pick and stack, the sample efficiency advantages of the derived are less obvious for match and reach tasks, because those tasks are relatively simple and have dense reward signals that are easy to learn.

Furthermore, to test the generalization of the learned policies, we evaluate their performance for 100 episodes on both ID and OOD states, as shown in Figure 3.11. Again, CDL generalizes

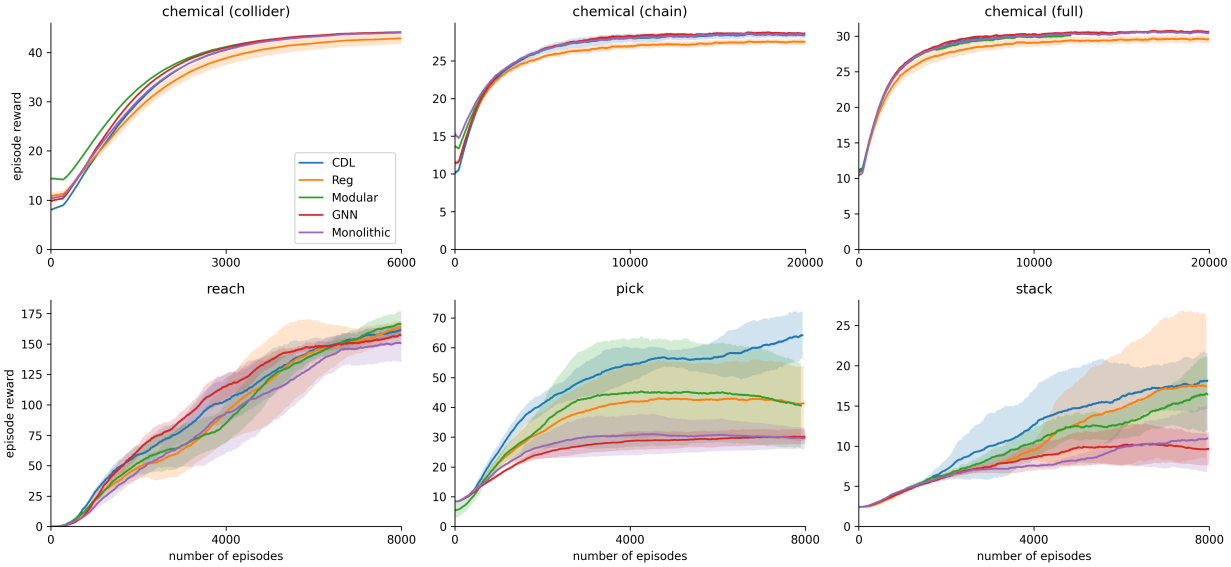


Figure 3.10: Traninig curve for all downstream tasks.

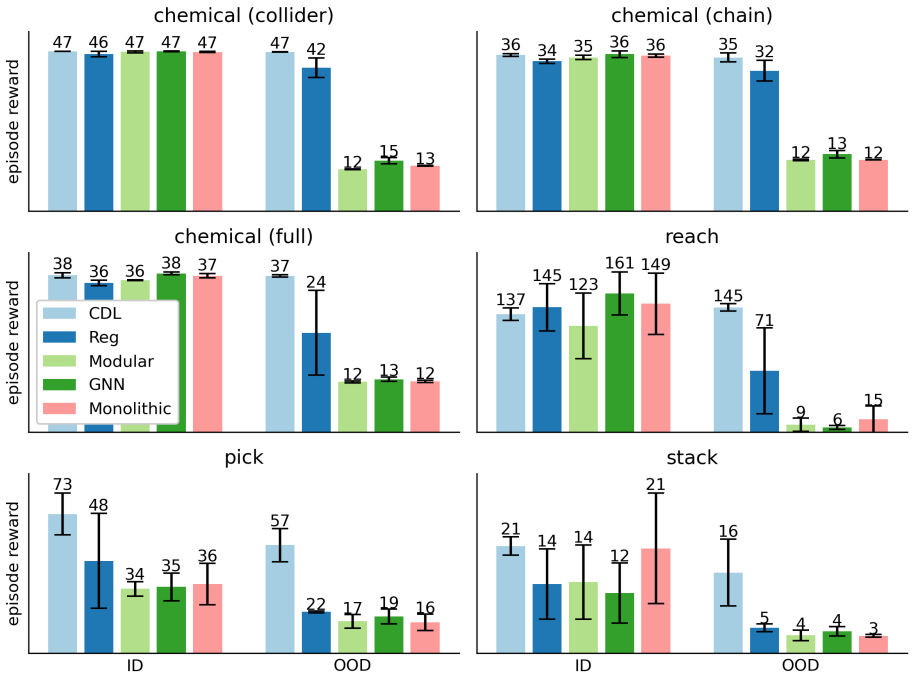


Figure 3.11: Episode reward of learned policies on ID and OOD states.

the best across all methods on OOD states. Combined with results in Section 3.3.3, results from Figure 3.11 answers Question **Q2** in the affirmative.

3.4 Summary

This chapter introduces CDL, Causal Dynamics Learning for Task-Independent State Abstraction. In contrast to most existing MBRL methods that learn a dense dynamics model, CDL learns a model that only keeps necessary dependencies on other factors and actions, thus generalizing better to unseen states than dense models. Furthermore, CDL derives a state abstraction from the learned causal relationships, which can be applied to a wider range of tasks than existing state abstraction methods that only retain state factors specific to one task. Our experiments demonstrate CDL's generalization both on the learned dynamics models and the policies for downstream tasks, as well as its sample efficiency resulting from its state abstraction.

While CDL improves sample efficiency and generalization compared to dense models, it has several limitations.

- First, while CDL's state abstractions can then be applied to many downstream task, it may not be minimal for a significant number of downstream tasks, since many tasks require manipulating only a subset of controllable factors. In such cases, CDL's abstraction could be further reduced to improve sample efficiency and generalization. Furthermore, ignoring state factors that are action-irrelevant limits CDL's application to many tasks. For example, for an autonomous vehicle, CDL's abstraction will ignore the traffic light, as the traffic light cannot be affected by the vehicle. In such cases, CDL's abstraction will cause the vehicle to be unable to follow traffic rules. Chapter 4 introduces a novel approach to address this issue by analyzing causal relationships in the reward function and deriving a minimal state abstractions.

- Furthermore, CDL focuses on identifying general causality $\mathcal{G}$, where the causal relationship $\mathcal{S}_t^i \to \mathcal{S}_{t+1}^j$ is considered to exist whenever it is possible for $\mathcal{S}^j$ to depend on $\mathcal{S}^i$, no matter how rarely it happens. However, most pairs of factors in real-world systems can be deemed

independent in most states, even though they may interact in some cases. For example, we can only move a mouse when we hold it, and the mouse is independent of our hand otherwise. Motivated by this insight, Chapter 5 seeks to infer *actual* causal dependencies specific to (s_t, a_t), represented by a local causal graph that is minimal by removing inactive edges in $\mathcal{G}$, and I investigate how to leverage it to derive a novel intrinsic reward function.

- Finally, the assumptions that the state space is fully observable and the factorization is known may not hold in all environment, such as those where only image observations are available. To address this limitation, Chapters 7 and 8 present two novel approaches to extract state factors through representation learning.

Chapter 4: Building Minimal and Reusable Causal State Abstractions

Although CDL, as introduced in Chapter 3, improves RL generalization and sample efficiency by deriving a task-independent state abstraction, its abstraction is often non-minimal for many downstream tasks and can omit action-irrelevant but task-critical factors (e.g., ignoring traffic lights for an autonomous vehicle). This chapter introduces a novel algorithm Causal Bisimulation Modeling (CBM) that identifies a minimal state abstraction for all tasks while further improving dynamics accuracy with implicit models.

This chapter is structured as follows. Section 4.1 first introduces the drawbacks of CDL in details and motivates CBM. Section 4.2 presents the proposed algorithm CBM that aims to derive a minimal state abstraction by identifying causal relationships in the reward function. Section 4.3 presents the empirical evaluation of CBM, covering its performance in dynamics learning, state-abstraction derivation, and task learning. Finally, Section 4.4 discusses limitations of CBM and how following chapters address them.

Contribution: In addition to my advisor, CBM is joint work with Caroline Wang, Xuesu Xiao, and Yuke Zhu. My contributions are the algorithm design and its implementation. Meanwhile, Wang helped with some of the experiments, while Xiao and Zhu provided technical advice.

4.1 Motivation

As mentioned in Section 3.1, a common deficiency of deep RL algorithms is their sample inefficiency and lack of generalization to unseen states, thus limiting their applicability in data-expensive or safety-critical tasks. One way to improve sample efficiency and generalization is to learn a *state abstraction* which reduces the task learning space and eliminates unnecessary information that may lead to spurious correlations. Determining the optimal abstraction necessitates understanding which state factors affect the reward and how those factors are influenced by others during state transitions.

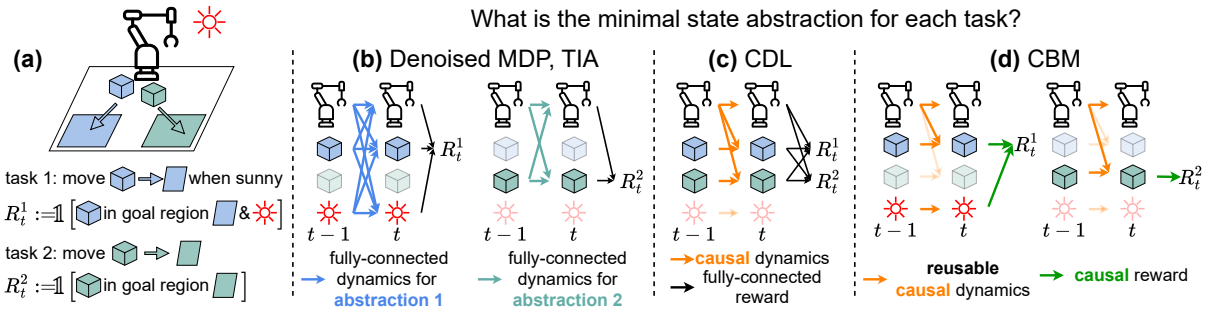


Figure 4.1: **(a)** Two tasks are defined by rewards R^1, R^2, and consist respectively of moving the blue and green blocks to their goal regions. Task 1 additionally requires moving the block only when it is sunny. Factors that are ignored by a state abstraction are marked semi-transparent. **(b)** TIA and Denoised MDP (Fu et al., 2021; Wang et al., 2022a) can learn concise abstractions (minimal in this example), but they require training fully-connected dynamics from scratch for each task. **(c)** CDL (Wang et al., 2022b) learns causal dependencies in the dynamics, but its derived state abstraction keeps *all* controllable state factors and ignores action-irrelevant ones, and thus the abstraction is non-minimal for task 2 and cannot learn task 1 due to its omission of the sun. **(d)** In addition to the implicit *causal dynamics* that can be *reused* for all tasks, CBM identifies which factors affect the reward and derives a minimal state abstraction from the *causal reward* models.

Prior methods obtain the desired abstractions by searching for the smallest subset of state factors that can predict the reward accurately while ensuring the subset is self-predictable in dynamics. Yet, their *dense* dynamics models are specific to the subset and thus have to be learned from scratch for each new task (Fu et al., 2021; Wang et al., 2022a; Zhang et al., 2021a), as shown in Figure 4.1 (b). Such approaches overlook a key characteristic of realistic problems: we often wish to build agents that solve multiple instances of tasks in the same environment, e.g. different cooking skills in a kitchen. To learn multi-task dynamics models, recent works seek to learn *causal* dynamical dependencies between state factors, from which they derive a task-*independent* abstraction (Ding et al., 2022; Wang et al., 2021c, 2022b). A notable example is CDL introduced in the previous chapter which identifies all state factors that can be affected by the action and retains them in the abstraction.

While dynamics-based state abstractions only need to be learned once for the environment and can then be applied to any downstream task, we observe three weaknesses. First, the abstraction may not be minimal for a significant number of downstream tasks, since many tasks require manipulating only a subset of controllable factors. In such cases, CDL's abstraction could be further

reduced to improve sample efficiency and generalization, as shown in Figure 4.1 (b). Second, ignoring state factors that are action-irrelevant limits CDL's application to many tasks. For example, for an autonomous vehicle, CDL's abstraction will ignore the traffic light, as it cannot be affected by the vehicle. Consequently, CDL's abstraction will cause the vehicle to be unable to follow traffic rules. Third, CDL employs explicit modeling of dynamics, directly predicting the next state as $\hat{s}_{t+1} = f^{\text{expl}}(s_t, a_t)$ where f^{expl} is a generic function parameterized by neural networks. However, prior work has shown *implicit modeling* ($\hat{s}_{t+1} = \arg\max_{s_{t+1}} f^{\text{impl}}(s_{t+1}; s_t, a_t)$ where f^{impl} is a critic function, see Section 4.2.2 for details) generally achieves higher accuracy in model learning, particularly for non-smooth dynamics in real-world physical systems (Florence et al., 2022; Song and Kingma, 2021). For instance, in robot manipulation, the object cannot be moved by the robot until they are in contact. In such environments, we show that inaccuracies of explicit modeling will lead to incorrect dynamical dependencies and thus non-minimal or incorrect state abstractions.

To address these weaknesses of CDL, we introduce Causal Bisimulation Modeling (CBM), a method that (1) learns **shared** task-agnostic dynamics between tasks while recovering a minimal, **task-specific** state abstraction, and (2) models causal dynamics dependencies with implicit models. Regarding the first contribution, in addition to causal relationships in the transition dynamics, CBM further infers which state factors affect the reward function with a causal reward model. In this way, CBM identifies state factors relevant to each task to further refine the state abstractions. The resultant causal abstraction is equivalent to bisimulation, the minimal abstraction that allows an agent to learn the optimal value function (Ferns et al., 2011). Regarding the second contribution, to the best of our knowledge, CBM is the first work that recovers causal dependencies with implicit models. To this end, CBM identifies and addresses two key problems of estimating conditional mutual information (CMI) with implicit models, allowing them to surpass explicit ones in both predictive accuracy and the identification of causal dependencies.

We validate CBM in robotic manipulation and Deepmind Control Suite, showing that (1) implicit models learn dynamical relationships and state abstractions more accurately compared to the explicit ones, and (2) CBM's task-specific state abstractions significantly improve sample efficiency and generalization compared to task-independent ones.

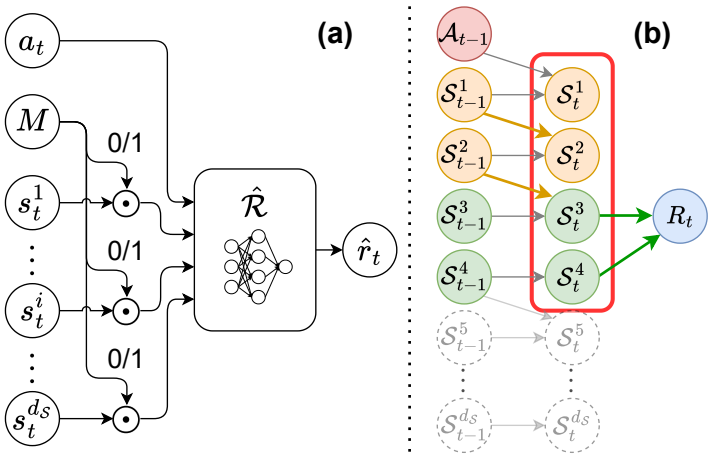


Figure 4.2: **(a)** The reward predictor architecture which can represent $\hat{\mathcal{R}}(M \odot \mathcal{S}_t, \mathcal{A}_t)$ conditioning any subsets of inputs by setting the binary mask M. **(b)** After CBM identifies the causal dependencies in dynamics and reward, its minimal state abstraction (marked by the red box) consists of (1) green factors that affect the reward, and (2) orange factors that influence green ones via dynamics. The semi-transparent state factors are ignored by the abstraction.

4.2 Causal Bisimulation Modeling (CBM)

This section describes two main contributions of CBM: obtaining a task-specific state abstraction by augmenting causal dynamics models with causal reward modeling, and recovering accurate causal dynamics when using implicit models.

4.2.1 Causal Reward Model for Task-Specific Abstraction

Though CDL recovers the causal relationships in dynamics, it still uses a dense model for reward learning. Consequently, without knowing which state factors are causal parents of the reward (i.e., reward-relevant), there is no direct way to remove irrelevant factors to improve sample efficiency.

To resolve these issues, CBM learns a causal reward model following a similar strategy to what CDL uses to learn dynamics, with the following assumption (for notational simplicity, we omit the task index k for the reminder of the chapter).

Assumption 7. *All state factors affect the reward for task $\mathcal{R}_t$ independently.*

Specifically, CBM examines the causal relationship between the state factor $\mathcal{S}_t^i$ and the reward $\mathcal{R}_t$ by learning two predictive models for the reward: (1) $\hat{\mathcal{R}}(\mathcal{S}_t, \mathcal{A}_t)$, which uses all state factors for prediction, and (2) $\hat{\mathcal{R}}(X_t^{\neg i})$, which ignores $\mathcal{S}_t^i$ when predicting. Intuitively, if the prediction performance when ignoring $\mathcal{S}_t^i$ is significantly lower than when including it, then the causal dependency $\mathcal{S}_t^i \to \mathcal{R}_t$ exists. More precisely, CBM evaluates whether the Conditional Mutual Information (CMI) is larger than a predefined threshold, i.e., $\text{CMI}^{i,\mathcal{R}} := \mathop{\mathbb{E}}_{s_t,a_t,r_t}\left[\log \frac{\hat{\mathcal{R}}(\mathcal{S}_t,\mathcal{A}_t)}{\hat{\mathcal{R}}(X_t^{\neg i})}\right] \geq \epsilon_{\mathcal{G}}$.

As shown in Figure 4.2 (a), to make the method scalable, rather than training $\hat{\mathcal{R}}_t(X_t^{\neg i})$ for each $i \in \{1, \ldots, d_{\mathcal{S}}\}$, CBM combines all predictive models into one network $\hat{\mathcal{R}}(M \odot \mathcal{S}_t, \mathcal{A}_t)$, where M is a manually defined binary mask used to ignore some input factors when predicting r_t. During training, CBM maximizes the prediction likelihood of both $\hat{\mathcal{R}}(\mathcal{S}_t, \mathcal{A}_t)$ where M uses all inputs (i.e., all entries are set to 1), and $\hat{\mathcal{R}}(X_t^{\neg i})$ where the entry for $\mathcal{S}_t^i$ is set to 0 in M.

After recovering causal parents of $\mathcal{R}_t$, CBM derives a bisimulation by combining the causal reward model with the causal dynamics model, based on the following assumptions that are the same as in the previous chapter.

Assumption 2. *The state is fully observable and the state factorization is given.*

Assumption 3. *The ground-truth transition dynamics and reward function are Markovian (i.e., $\mathcal{S}_{t+1}$ and $\mathcal{R}_t$ only depend on $(\mathcal{S}_t, \mathcal{A}_t)$, and they do not depend on historical states $\mathcal{S}_{<t}$ nor historical actions $\mathcal{A}_{<t}$).*

With the above assumptions, adapting the work from Zhang et al. (2020), we prove that the bisimulation can be derived from the inferred causal relationships as follows.

Theorem 4.1 (Bisimulation Derivation, Theorem 1 in Zhang et al. (2020))**.** *Consider an MDP $\mathcal{M}$ that satisfies Assumptions 2 and 3. Let $\boldsymbol{PA}(\mathcal{R}) \subseteq \{\mathcal{S}^1, \ldots, \mathcal{S}^{d_{\mathcal{S}}}, \mathcal{A}_t\}$ be the set of factors such that the reward $\mathcal{R}(\mathcal{S}, \mathcal{A})$ is a function only of $\boldsymbol{PA}(\mathcal{R})$. Let $\boldsymbol{AN}(\mathcal{R})$ denote the ancestors of $\boldsymbol{PA}(\mathcal{R})$ in the causal graph corresponding to the transition dynamics of $\mathcal{M}$. Then the state abstraction $\phi(s) = \boldsymbol{AN}(\mathcal{R})$ is a bisimulation abstraction for reward $\mathcal{R}$.*

As illustrated in Figure 4.2 (b), CBM's abstraction is selected as the union of (1) all $\mathcal{S}^j$ that $\mathcal{R}$ depends on, i.e., $\mathbf{PA}(\mathcal{R})$, and (2) all other state factors that can affect $\mathbf{PA}(\mathcal{R})$ via dynamics and not already included in $\mathbf{PA}(\mathcal{R})$. In other words, this union corresponds to $\mathcal{R}$'s causal ancestors (i.e., $\mathbf{AN}(\mathcal{R})$) in the learned causal dynamics and reward graph, and thus being equivalent to bisimulation — the *minimal state abstraction* that allows an agent to learn the optimal value function (Dean and Givan, 1997; Ferns et al., 2011).

4.2.2 Causal Discovery with Implicit Dynamics Models

Implicit models have been shown to learn dynamics more accurately than explicit models (Florence et al., 2022). Motivated by this finding, the goal of CBM's dynamics learning module is to recover causal relationships between states and action factors via an *implicit* modeling approach. As in CBM's reward-learning approach, causal dependencies will also be detected by measuring the conditional mutual information CMI^{ij} between $\mathcal{S}_t^i$ and $\mathcal{S}_{t+1}^j$ for each i, j pair. In this section, first, we introduce the implicit dynamics model. Second, we describe how CBM estimates CMI from the implicit dynamics with a prior method by Sordoni et al. (2021). Third, we discuss two CMI overestimation issues of the prior method and how CBM solves them.

Implicit Dynamics Models For the transition of each state factor $\mathcal{S}^j$, we would like to learn an implicit model $f^{\text{impl,i}} : \mathcal{S}_{t+1}^j \times \mathcal{S}_t \times \mathcal{A}_t \to \mathbb{R}$ to assign a high score to the label s_{t+1}^j from the ground truth distribution, and low values to other labels. During dynamics prediction, the model selects the s_{t+1} that maximizes the total score over all state factors: $\hat{s}_{t+1} = \arg\max_{s_{t+1}^j \in \mathcal{S}^i} \sum_{i=1}^{d_\mathcal{S}} f^{\text{impl,i}}(s_{t+1}^j; x_t)$. For notational simplicity, in the remainder of Section 4.2.2, we omit the state factor index i on $f^{\text{impl,i}}$ as it is clear from the input factor s_{t+1}^j. An implicit dynamics model can be trained by minimizing the contrastive InfoNCE loss (van den Oord et al., 2018),

$$L_{\text{NCE}}(f^{\text{impl}}) = -\log \frac{\exp(f^{\text{impl}}(s_{t+1}^j; x_t))}{\exp(f^{\text{impl}}(s_{t+1}^j; x_t)) + \sum_{n=1}^{N} \exp(f^{\text{impl}}(\tilde{s}_{t+1}^{j,n}; x_t))}. \tag{4.1}$$

Minimizing the InfoNCE loss requires the ground truth label s_{t+1}^j, as well as N negative examples $\{\tilde{s}_{t+1}^{i,n}\}_{n=1}^N \sim p(s_{t+1}^j)$ sampled the value range S_{t+1}^j. This loss encourages f^{impl} to

distinguish the label s^j_{t+1} from negative samples, i.e., extract information about s^j_{t+1} from x_t. The remainder of the chapter uses $L_{\text{NCE}}(F(x;y))$ to denote the InfoNCE loss that encourages a generic model F to extract information about x from y.

CMI Estimation with Implicit Dynamics Models

Observation 1 van den Oord et al. (2018) show that, for a generic model $F(x,y)$ that minimizes L_{NCE}, the minimized L_{NCE} approximates the mutual information between x and y:

$$\mathbb{E}\left[\log(N+1) - L_{\text{NCE}}(F(x;y))\right] \approx \mathbb{E}\left[\frac{p(y|x)}{p(y)}\right] = I(x;y).$$

Inspired by this observation, Sordoni et al. (2021) propose to estimate CMI^{ij} using a conditional implicit model:

$$\text{CMI}^{ij} = \mathbb{E}\left[\log \frac{(N+1)\exp(\phi(s^j_{t+1}; s^i_t|x_t^{\neg i}))}{\exp(\phi(s^j_{t+1}; s^i_t|x_t^{\neg i})) + \sum_{n=1}^{N}\exp(\phi(\tilde{s}^{j,n}_{t+1}; s^i_t|x_t^{\neg i}))}\right],$$
$$\text{where } \tilde{s}^{j,n}_{t+1} \sim p(s^j_{t+1}|x_t^{\neg i}). \tag{4.2}$$

Since CMI^{ij} measures how using s^i_t could additionally contribute to predicting s^j_{t+1} given the other state and action factors, $x_t^{\neg i}$, ϕ is a **conditioned** model trained to capture the additional information about s^j_{t+1} in s^i_t that is not present in $x_t^{\neg i}$. However, training ϕ and estimating CMI^{ij} with Eq. 4.2 both require negative samples from $p(s^j_{t+1}|x_t^{\neg i})$, which are not readily accessible, as the data can only be collected from the full state transition distribution $p(s_{t+1}|x_t)$.

Observation 2 To tackle this issue, CBM uses the importance sampling approximation proposed by Sordoni et al. (2021) to compute $p(s^j_{t+1}|x_t^{\neg i})$ with samples from the marginal distribution $p(s^j_{t+1})$. Thus, CMI^{ij} may be approximated as,

$$\mathbb{E}\left[\log \frac{(N+1)\exp(\phi(s^j_{t+1}; s^i_t|x_t^{\neg i}))}{\exp(\phi(s^j_{t+1}; s^i_t|x_t^{\neg i})) + N\sum_{n=1}^{N} w_n \exp(\phi(\tilde{s}^{j,n}_{t+1}; s^i_t|x_t^{\neg i}))}\right],$$
$$w_n = \frac{\exp(\psi(\tilde{s}^{j,n}_{t+1}; x_t^{\neg i}))}{\sum_{k=1}^{N}\exp(\psi(\tilde{s}^{j,k}_{t+1}; x_t^{\neg i}))} \approx \frac{p(s^j_{t+1}|x_t^{\neg i})}{p(s^j_{t+1})}. \tag{4.3}$$

where $\tilde{s}_{t+1}^{j,n} \sim p(s_{t+1}^j)$, and ψ is trained to extract information about s_{t+1}^j from $x_t^{\neg i}$ by minimizing $L_{\text{NCE}}(\psi(s_{t+1}^j; x_t^{\neg i}))$ and is used to compute importance weights w_n in a self-normalized manner.

After learning ψ^*, one only needs ϕ to estimate CMI with Eq. 4.3. To this end, Sordoni et al. (2021) train ϕ by minimizing $L_{\text{NCE}}(\phi(s_{t+1}^j; s_t^i | x_t^{\neg i}) + \psi^*(s_{t+1}^j; x_t^{\neg i}))$ while keeping ψ^* frozen, so ϕ learns to capture the *additional* information about s_{t+1}^j from s_t^i that is not present in $x_t^{\neg i}$ (i.e., absent in ψ^*). See pseudo-code in the Algorithm 2 for details.

Algorithm 2 ϕ learned by Sordoni et al. (2021)

Initialize the ψ and ϕ.
repeat
    Sample s_{t+1}^j and $x_t^{\neg i}$ from data; Sample $\{\tilde{s}_{t+1}^{j,n}\}_{n=1}^N$ from $\mathcal{S}^i$.
    Optimize ψ with $L_{\text{NCE}}(\psi(s_{t+1}^j; x_t^{\neg i}))$.
until ψ converges to ψ^*
repeat
    Sample s_{t+1}^j, s_t^i, $x_t^{\neg i}$ from data; Sample $\{\tilde{s}_{t+1}^{j,n}\}_{n=1}^N$ from $\mathcal{S}^i$.
    Optimize ϕ with the following loss while **keeping** ψ^* **frozen**

$$L_{\text{NCE}}(\phi + \psi^*) = -\log \frac{\exp((\phi + \psi^*)(s_{t+1}^j))}{\exp((\phi + \psi^*)(s_{t+1}^j)) + \sum_{n=1}^N \exp((\phi + \psi^*)(\tilde{s}_{t+1}^{j,n}))},$$
$$\text{where } (\phi + \psi^*)(\cdot) = \phi(\cdot; s_t^i | x_t^{\neg i}) + \psi^*(\cdot; x_t^{\neg i}).$$

until ϕ converges

Inaccurate CMI Estimation and Solutions In practice, the method proposed by Sordoni et al. (2021) often yields inaccurate CMI estimations and thus leads to incorrect causal dependencies. We discovered two reasons for the inaccuracy and proposed corresponding solutions.

Reason 1 – Overfitted Conditional Models In theory, when ϕ is trained as described above, it should condition on ψ and capture the *additional* information only. However, in practice, such trained ϕ still uses $x_t^{\neg i}$ to predict s_{t+1}^j directly rather than conditioning on it to estimate the additional contribution of s_t^i. Figure 4.3 left shows an example where knowing s_t^i does not provide any additional information about s_{t+1}^j. In such a case, the ground truth conditional model should

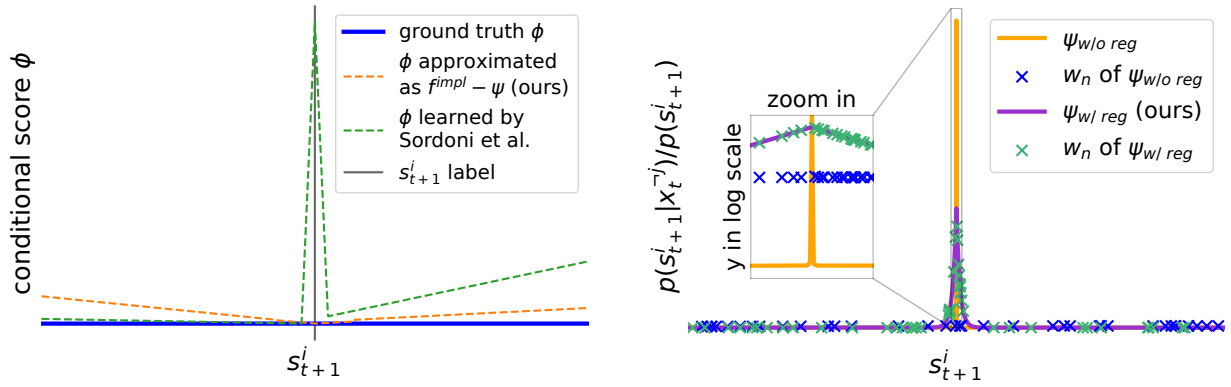


Figure 4.3: Two sources of inaccurate CMI estimation: (**left**) Overfitted conditional model: learned ϕ overestimate conditional information while the $f^{\text{impl}} - \psi$ approximation is closer to the ground truth (0 for all s^i_{t+1} values), especially when close to the ground truth label $s^i_{t+1} = 0$. (**right**) Inaccurate Importance Sampling: without regularization, the likelihood ratio of $p(s^j_{t+1}|x^{\neg i}_t)/p(s^j_{t+1})$ computed by ψ can be peaked at the label $s^i_{t+1} = 0$ and is therefore challenging to approximate by self-normalized importance sampling. In comparison, regularized ψ has a flatter likelihood ratio landscape and is easier to approximate with samples.

output the same score for all s^j_{t+1} values. In contrast, the scores output by ϕ are still peaked at the label of s^j_{t+1}, and using such an overfitted model in Eq. 4.3 would overestimate CMI.

To solve this issue, rather than using a learned ϕ, we use the approximation $\phi = f^{\text{impl}} - \psi$. The motivation is as follows: as f^{impl} is trained to use x_t to estimate the score of s^j_{t+1} and ψ estimates with all factors except for s^i_t, the difference of their estimated scores should reflect the additional information from s^i_t. In practice, as shown in Figure 4.3 top, the conditional score estimated by this approximation is closer to the ground truth than the score of ϕ learned by Sordoni et al. (2021), especially in the neighbor of the ground truth label where the accuracy of conditional scores significantly influences CMI.

Reason 2 – Inaccurate Importance Sampling Meanwhile, when s^j_{t+1} has an almost deterministic transition (which is common in many environments, e.g., objects will not move unless manipulated by the robot), the importance sampling approximation in Eq. 4.3 could be inaccurate.

In detail, as shown in Figure 4.3 bottom, for such transitions, the score estimated by ψ tends to have extremely sharp maxima — it is high only when s^j_{t+1} is very close to the ground truth

labels. As a result, even with many negative samples from $p(s^j_{t+1})$, it is likely that none of them are similar enough to samples from $p(s^j_{t+1}|x_t^{\neg i})$. Therefore, since the importance weight w_n in Eq. 4.3 is self-normalized among all negative samples, samples that are not from $p(s^j_{t+1}|x_t^{\neg i})$ still have large weights (rather than near-zero weights), thus leading to inaccurate CMI estimation.

To mitigate this issue, when training f^{impl} and ψ for the prediction of s^j_{t+1}, beyond the InfoNCE loss, we regularize them to have flatter score landscapes with L2 penalty on their computed scores and the partial derivative of scores as follows,

$$L_{\text{dyn}} = L_{\text{reg}}(f^{\text{impl}}) + L_{\text{reg}}(\psi) \text{ where for generic critic model } f, \tag{4.4}$$

$$L_{\text{reg}}(f) = L_{\text{NCE}}(f(s^j_{t+1};\cdot)) + \sum_{\tilde{s}^j_{t+1}} \left(\lambda_1 \|f(\tilde{s}^j_{t+1};\cdot)\|^2 + \lambda_2 \| \frac{\partial f(\tilde{s}^j_{t+1};\cdot)}{\partial \tilde{s}^j_{t+1}} \|^2 \right). \tag{4.5}$$

The regularization is applied to both the label and all negative samples (i.e., $\tilde{s}^j_{t+1} \in \{s^j_{t+1}, \tilde{s}^{j,n}_{t+1}\}$), and λ_1, λ_2 are the weights of the regularization terms. As shown in Figure 4.3 bottom, ψ trained with regularization has flatter output landscape than ψ trained without regularization. As a result, with the same number of samples, the importance weights w_n approximate the likelihood ratio of regularized ψ more accurately than the unregularized ψ. In Algorithm 3, we provide a summary for how CBM learn the conditional implicit model ϕ.

Algorithm 3 ϕ learned by CBM

Initialize the f^{impl} and ψ.
repeat
    Sample s^j_{t+1} and x_t from data; Sample $\{\tilde{s}^{j,n}_{t+1}\}_{n=1}^N$ from $\mathcal{S}^i$.
    Optimize f^{impl} with $L_{\text{reg}}(f^{\text{impl}})$ in Eq. 4.5.
until f^{impl} converges
repeat
    Sample s^j_{t+1} and $x_t^{\neg i}$ from data; Sample $\{\tilde{s}^{j,n}_{t+1}\}_{n=1}^N$ from $\mathcal{S}^i$.
    Optimize ψ with $L_{\text{reg}}(\psi)$ in Eq. 4.5.
until ψ converges
return $\phi(s^j_{t+1}; s^i_t | x_t^{\neg i}) = f^{\text{impl}}(s^j_{t+1}; x_t) - \psi(s^j_{t+1}; x_t^{\neg i})$

After learning ϕ, CBM computes CMI^{ij} as in Algorithm 4, following Sordoni et al. (2021).

Algorithm 4 CMI^{ij} computation using learned ψ and ϕ

1: Sample $\{\tilde{s}_{t+1}^{j,n}\}_{n=1}^{N}$ from $\mathcal{S}^i$.
2: Compute the self-normalized importance weight for each negative sample $\tilde{s}_{t+1}^{j,n}$ as

$$w_n = \frac{\exp(\psi(\tilde{s}_{t+1}^{j,n}; x_t^{\neg i}))}{\sum_{k=1}^{N} \exp(\psi(\tilde{s}_{t+1}^{j,k}; x_t^{\neg i}))} \approx \frac{p(s_{t+1}^j | x_t^{\neg i})}{p(s_{t+1}^j)}.$$

3: Compute the conditional mutual information as

$$\text{CMI}^{ij} = \mathbb{E}\left[\log \frac{(N+1)\exp(\phi(s_{t+1}^j; s_t^i | x_t^{\neg i}))}{\exp(\phi(s_{t+1}^j; s_t^i | x_t^{\neg i})) + N \sum_{n=1}^{N} w_n \exp(\phi(\tilde{s}_{t+1}^{j,n}; s_t^i | x_t^{\neg i}))}\right].$$

To make the dynamics model scalable, we use the same masking technique as in Section 4.2.1 to combine $f^{\text{impl,j}}$ and ψ^i into one network. We use f^{impl} to denote the parameters of $d_{\mathcal{S}}$ such networks, each modeling the dynamics of state factor $\mathcal{S}_{t+1}^j$.

4.2.3 CBM for Task Learning

Algorithm 5 Causal Bisimulation Modeling (CBM)

1: Initialize the dynamics model f^{impl}.
2: (Optional) Pretrain f^{impl} (Eq. 4.4) with offline data.
3: **for** all tasks **do**
4: Initialize the reward model $\hat{\mathcal{R}}$ and the policy π.
5: **for** T training steps **do**
6: Collect (s_t, a_t, r_t, s_{t+1}) with $a_t \sim \pi$.
7: Update f^{impl} (Eq. 4.4, optional) and $\hat{\mathcal{R}}$ (Section 4.2.1).
8: Evaluate dynamical and reward dependencies (Eq. 4.3);
9: Update the state abstraction for π (Figure 4.2 (b)).
10: Update π with SAC losses.
11: **end for**
12: **end for**

As in Algorithm 5, for tasks with the same dynamics, CBM's dynamics model is shared across tasks. The dynamics model can either learn from offline data (line 2), or from transitions collected during task learning (line 7), or both.

When solving each task, CBM interleaves reward learning (line 7) with policy learning. The policy is trained via Soft Actor Critic (SAC), an off-policy reinforcement learning algorithm (Haarnoja et al., 2018). The reward model is combined with the pre-trained dynamics to generate the task-specific abstraction (line 8). CBM applies the state abstraction to the policy π as a binary mask that zeros out ignored factors. During task learning, as the policy explores and learns, we expect it to gradually expose causal relationships that are necessary to solve the task, and, in return, the updated state abstractions reduce the learning space of the policy, making its learning sample efficient.

4.3 Experiments

In this section, we provide empirical evaluations pertaining to the following questions that support the indicted answers.

- **Q1)** Can implicit models learn dynamical dependencies and dynamics models more accurately than explicit models? Yes (Section 4.3.1).
- **Q2)** Compared to CDL's task-independent state abstraction and prior task-specific abstraction works, can CBM learn a more concise state abstraction? Yes (Section 4.3.2).
- **Q3)** Compared to baselines, can CBM achieve better sample efficiency and generalization during task learning? Yes (Section 4.3.2).

Environments To test CBM, we use two manipulation environments implemented with Robosuite (Zhu et al., 2020b), shown in Figure 4.7 left, and two tasks from the DeepMind Control Suite (DMC) (Tunyasuvunakool et al., 2020). In the block environment (b), there are multiple movable and unmovable blocks. The tasks in this environment include Pick and Stack. In the tool-use environment, we consider a challenging long-horizon task Series: the agent needs to use an L-shaped tool to move a faraway block within reach, pick it up, and place it within the box. In the DMC, we consider the Cheetah and Walker tasks, two high-dimensional continuous control tasks. In all environments, as controllable distractors (cd), 20 factors whose values are random projections

	block			tool-use	
	causal graph	pick	stack	causal graph	series
explicit	87.5 ± 0.1	53.2 ± 4.6	59.6 ± 4.6	82.6 ± 0.2	80.0 ± 1.5
implicit (ours)	**90.5** ± 0.4	**95.7** ± 6.0	**95.7** ± 6.0	**85.5** ± 0.1	**98.8** ± 1.3

Table 4.1: Mean $\pm$ std. error of accuracy ($\uparrow$) for learned dynamics causal graphs and task abstractions.

of the action are added to the state space. We also add 20 uncontrollable distractors (ud) whose values are uniformly sampled from $[-1, 1]$. The distractors have no interaction with other state factors.

Baselines For the dynamics learning experiments, we compare the implicit dynamics model against the explicit model. All methods are trained and evaluated with 3 random seeds. For the state abstraction and task learning experiments, we compare CBM against the closely related methods of CDL, which uses a task-independent abstraction; TIA (Fu et al., 2021) and Denoised MDP (Wang et al., 2022a), which both learn task-specific state abstractions. To contextualize these methods, we also compare against an Oracle that learns with the ground-truth minimal abstractions, and reinforcement learning over the Full state space (no state abstraction). All methods are trained and evaluated with 5 random seeds.

State Abstractions for Task Learning To fairly compare the effects of the various state abstractions for task learning, all methods use **implicit** dynamics models, including CDL, which originally used the inferior explicit models. All methods learn tasks via Soft Actor Critic (Haarnoja et al., 2018). The dynamics model is pretrained in Pick and Stack tasks, and it is learned jointly with the policy *from scratch* in all other tasks.

4.3.1 Dynamics and Causal Graph Learning

This section compares implicit and explicit dynamics models, learned from the same offline data, in terms of how accurately they recover dynamics causal graphs and task abstractions.

Experiments are conducted on the Robosuite environments[1], and results are shown in Table 4.1. For causal graphs, we compare the learned dynamics dependencies with the ground truth and measure the *causal graph accuracy* as the # correctly learned edges / total edges in the graph. The accuracy of implicit models is 3% higher than that of explicit ones, which corresponds to causal relationships between 67 pairs of state factors in the block environment, and 48 pairs in tool-use. As a result, when being used to derive a *task-specific* abstraction (we use the ground truth reward dependencies in this experiment only to avoid the interference from reward learning), implicit models derive more accurate state abstractions ($\geq 20\%$ difference in accuracy on all tasks) than explicit models. Note that the *state abstraction accuracy* is measured as the # of correctly classified state factors / total state factors.

We also evaluate CBM using implicit dynamics and CDL in terms of prediction accuracy. Specifically, given s_t and $a_{t:t+19}$, we use them to generate 20-step predictions, i.e., $s_{t+1:t+20}$, on both in-distribution (ID) and out-of-distribution (OOD) s_t for the block and tool-use environments. For OOD states, distractor values are replaced with random values sampled from $\mathcal{N}(0, 100)$. The results are again measured on 10K transitions for each method. As shown in Figure 4.4, when measuring the prediction error of all state factors, CBM has lower prediction error than CDL on both ID and OOD states. Especially on OOD states, as CBM learns fewer spurious correlations than CDL, it keeps similar performance while CDL's errors increase significantly compared to ID states.

Together, results in Table 4.1 and Figure 4.4 answer Question **Q1** in the affirmative.

4.3.2 Task Learning with State Abstractions

This section compares policy learning with different state abstractions on various tasks in DMC, the block environment (b), and the tool-use environment (t).

State Abstraction Accuracy Figure 4.5 shows the abstraction learned by each method for the Stack task on the object level. State factor-level abstractions are shown in Figure 4.6. Among all

[1]DMC tasks are not suitable for these experiments, as the ground truth causal graphs are not clear.

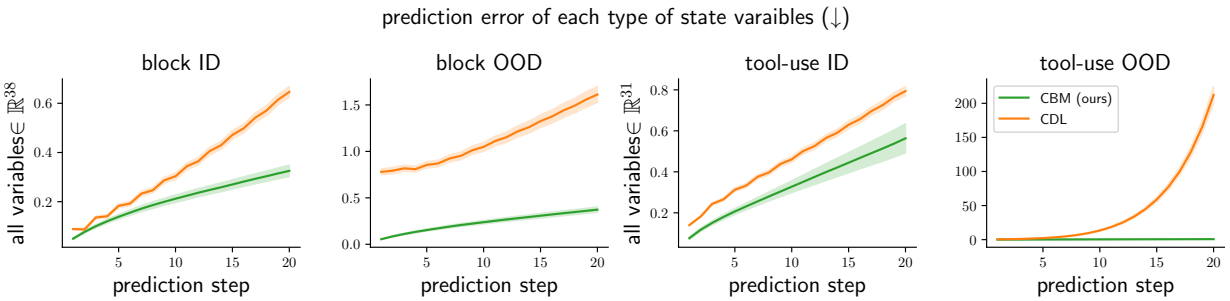


Figure 4.4: Dynamics prediction error in the block and tool-use environments.

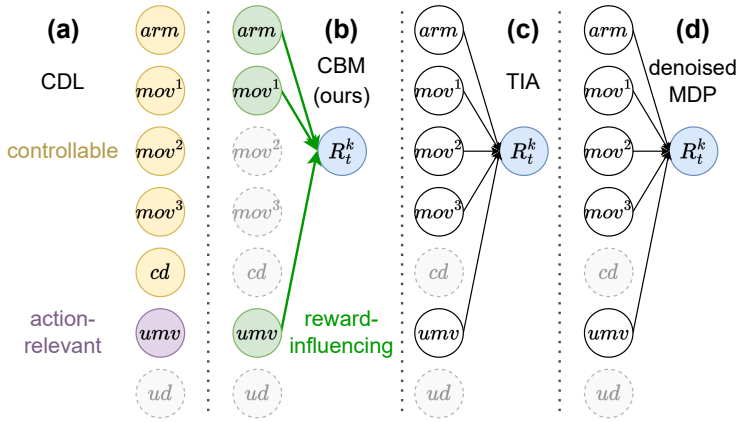


Figure 4.5: Object-level state abstractions for the Stack task learned by each method; semitransparent factors are excluded. **(a)** Though mov^2, mov^3, and controllable distractos cd are unnecessary, CDL still keeps them as they are controllable. **(b)** Compared to CDL, CBM (ours) successfully learns the minimal abstraction by further reasoning about which state factors influence the reward. **(c)** TIA and **(d)** Denoised MDP fail to learn meaningful abstractions when their assumptions on the dynamics do not hold.

methods, only CBM learns minimal abstraction. CDL keeps all controllable factors and thus uses a non-minimal abstraction. Meanwhile, TIA and Denoised MDP assume that state factors can be segregated into several dynamically independent components, and their abstractions keep only the task-relevant component. However, though mov^2 and mov^3 are task-irrelevant, their dynamics still depend on the task-relevant part (the end-effector and gripper). As a result, when their assumptions do not hold, TIA and Denoised MDP learn that (almost) all factors belong to the same component. Then, depending on their definitions of task relevance, all such factors are either included (TIA) or ignored (Denoised MDP) by the abstractions. The results in Figure 4.5 and Figure 4.6 thus answer Question **Q2** in the affirmative.

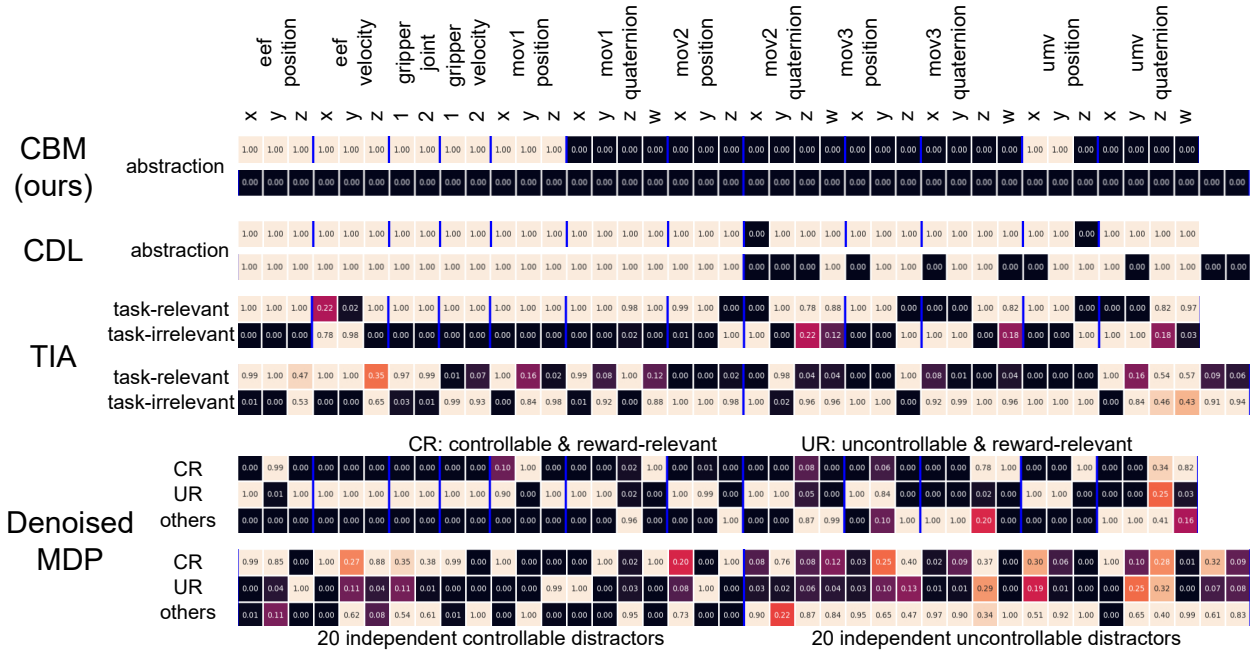


Figure 4.6: Factor-level state abstractions learned by CBM, TIA, and Denoised MDP for the Stack task.

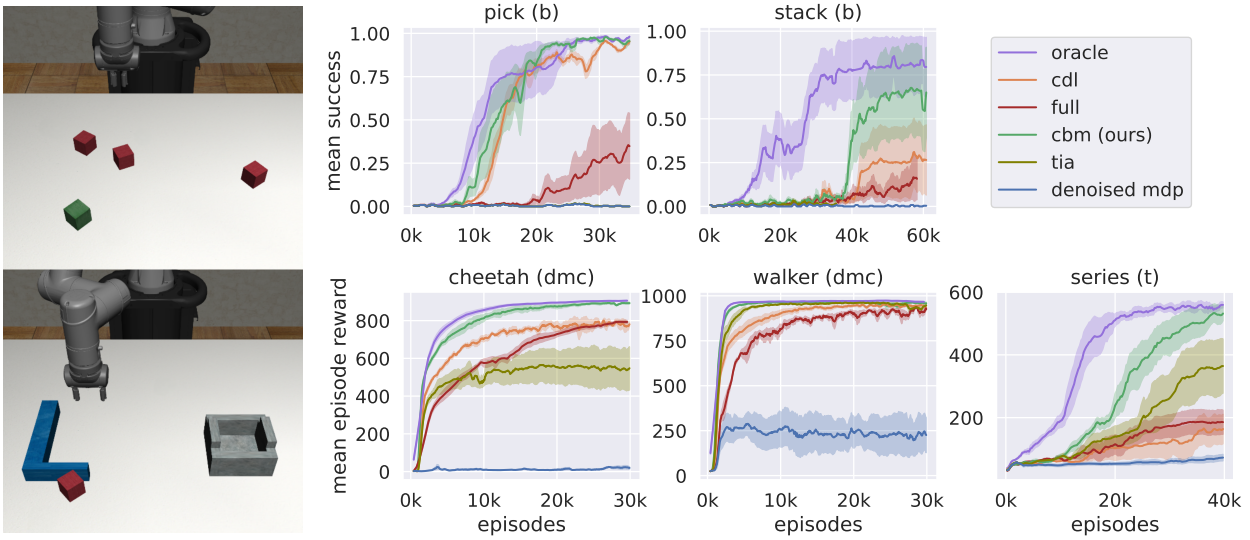


Figure 4.7: (**left**) top: Block environment with three *movable* blocks ($mov^{1\sim3}$), and an *unmovable* (unm) block fixed on the table. bottom: Tool-Use environment with the block, the L-shaped tool, and the box. (**right**) Learning curves of CBM (ours) compared to baseline methods and RL with the Oracle abstraction in five tasks. Each learning curve is generated from independent runs using 5 different random seeds, with mean and std. error computed across 50 test episodes per point on the learning curve. CBM is among the most sample-efficient methods, even approaching the efficiency of the Oracle on Pick, Cheetah, and Walker.

CBM is Sample-Efficient For task performance, the metric for Pick and Stack tasks is the mean success rate at accomplishing the task, and the metric for Series and DMC tasks is the mean episode reward, evaluated over 50 test episodes and plotted with respect to the number of episodes.[2] The learning curves for all tasks are shown in Figure 4.7 right. Recall that Oracle learns with the ground-truth state abstraction, whereas Full learns with no state abstraction. For all tasks, Oracle learns the fastest, demonstrating the possible gain in sample efficiency with an ideal state abstraction.

Overall, we find that CBM matches or improves in sample efficiency over the CDL, TIA, and Denoised MDP baselines in all settings. In all tasks, CBM is among the closest to the Oracle in sample efficiency, showing the benefit of the learned, near-minimal state abstraction. We observe that the higher the difficulty level of the task, the greater the benefit of learning a small task abstraction. For instance, Walker is a simple task, where almost all methods converge rapidly to an episode reward of 1000 by 20k episodes. The only exception is Denoised MDP, where the learned abstraction masks out some key factors among robot joint angles and angular velocities. On the other hand, the Series task is much more challenging, requiring a sequence of successful behaviors (reach tool, use tool to move block closer, pick and place block). Learning without any abstraction (Full) only learns to grasp the tool and achieves an episode reward of 200 over 40k training episodes, while CBM learns with much greater sample efficiency. We observe a similarly large gap between CBM and other methods on Stack (B), another complex manipulation task. Combined with results in the next paragraph, Figure 4.7 answers Question **Q3** in the affirmative.

CBM is Generalizable As shown in Figure 4.8, we further measure each method's generalizability to unseen states on Pick and Stack (TIA and Denoised MDP fail to learn the tasks and thus are not evaluated). In addition to in-distribution (ID) states, we also evaluate the learned policies on out-of-distribution (OOD) states where the values of all task-irrelevant state factors are set to noise sampled from $\mathcal{N}(0, 1)$. For both tasks, only Oracle and CBM keep similar performance across ID and OOD states, as their minimal task-specific abstractions eliminate the influence of OOD factors. In contrast, though CDL can be robust against factors ignored by its abstraction, it still fails to

[2]As each episode is a fixed number of steps, episodes have a linear relationship with training steps.

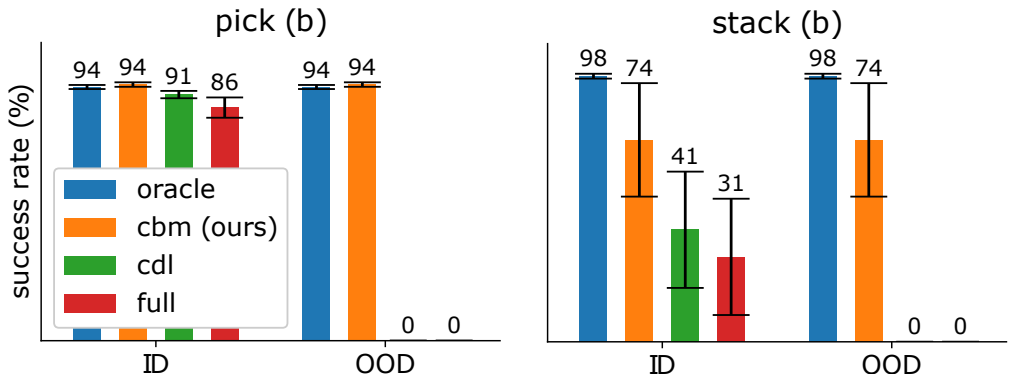


Figure 4.8: Performance of policies with different state abstractions on both ID and OOD states, in terms of mean and std. error of success rates (↑) in the block environment.

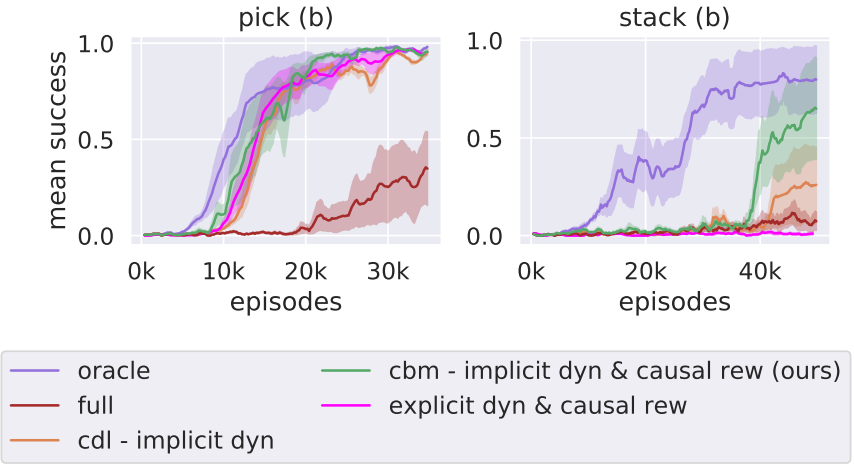


Figure 4.9: Learning curves of CBM and CDL (which both use implicit dynamics in the main chapter), and an ablation of CBM that uses explicit dynamics on Pick and Stack tasks. We observe that the ablation has much worse sample efficiency on Stack.

generalize when redundant factors in its non-minimal abstractions have unseen values. The results in Figure 4.8 thus answer Question **Q3** in the affirmative.

4.3.3 Task Learning Ablation

CBM learns implicit dynamics and a causal reward function. In Section 4.3.1, we show that implicit dynamics models surpass explicit models in terms of causal graph accuracy and state abstraction accuracy. Figure 4.9 shows an ablation of CBM which uses explicit dynamics instead of implicit dynamics. We observe that on the Pick task, the difference from CBM is not significant as Pick is relatively easy. On the more challenging Stack task where accurate abstraction plays a more important role, the performance of explicit dynamics is much worse than CBM which uses implicit dynamics.

4.4 Summary

This chapter introduces Causal Bisimulation Modeling (CBM), an algorithm that (1) learns a minimal state abstraction via causal reward learning, and (2) learns an implicit causal dynamics model. The experiments demonstrate that CBM learns more accurate and concise state abstractions, which leads to improved sample efficiency on downstream tasks compared to related methods. Furthermore, the implicit dynamics model introduced by CBM improves over explicit dynamics models in terms of prediction error and causal graph accuracy.

While CBM improves state abstraction accuracy compared to CDL, it still has following drawbacks.

- Similar to CDL, CBM focuses on identifying general causality $\mathcal{G}$, assuming that the causal relationship $\mathcal{S}_t^i \rightarrow \mathcal{S}_{t+1}^j$ exists whenever it is possible for $\mathcal{S}^j$ to depend on $\mathcal{S}^i$, no matter how rarely it happens. However, most pairs of factors in real-world systems can be deemed independent in most states, even though they may interact in some cases. For example, we can only move a mouse when we hold it, and the mouse is independent of our hand otherwise. Motivated by this insight, Chapter 5 seeks to infer *actual* causal dependencies specific to

(s_t, a_t), represented by a local causal graph that is minimal by removing inactive edges in $\mathcal{G}$, and it investigates how to leverage this graph to derive a novel intrinsic reward function.

- Finally, the assumptions that the state space is fully observable and the factorization is known may not hold in all environment, such as those where only image observations are available. To address this limitation, Chapters 7 and 8 present two novel approaches to extract state factors through representation learning.

Together, the results in Chapter 3 and this chapter represent the completion of our efforts to derive state abstractions and facilitate RL generalization and sample efficiency (Chapter 1.1 Contribution 1).

Chapter 5: Intrinsic Reward Functions from Actual Causality

The previous two chapters focused on enhancing RL sample efficiency from a state-space perspective, developing methods that rely on state abstraction to ignore irrelevant factors. Instead, this chapter considers the sample efficiency from a reward-function perspective. Specifically, the second challenge in sample efficiency lies in exploration in environments with sparse rewards. To overcome the limited learning signals that hinder the agent from traversing the state space effectively, this chapter introduces a novel algorithm Exploration via Local DepENdencies (ELDEN). ELDEN augments the sparse reward with an intrinsic reward that motivates the agent to discover novel local causal dependencies between state factors, where it utilizes the partial derivative of the learned dynamics to model the local dependencies between factors accurately and computationally efficiently.

This chapter is structured as follows. Section 5.1 first introduces the motivation behind intrinsic reward function and limitations in previous work. Section 5.2 presents the proposed algorithm ELDEN – specifically, how it identifies local causal relationships and derive the intrinsic reward function. Section 5.3 presents the empirical evaluation of ELDEN, covering its performance in identifying local causal dependencies and faciliatating task learning. Finally, Section 5.4 discusses limitations of ELDEN and how following chapters address them.

Contribution: ELDEN is joint work with Jiaheng Hu and Roberto Martín-Martín. My contributions are the identification of local causal relationships. Meanwhile, Hu proposed how to derive the intrinsic reward function from the identified relationships, while Martín-Martín provided technical advice.

5.1 Motivation

While the previous two chapters focused on enhancing RL sample efficiency by finding a state abstraction to reduce the learning space, this chapter focuses on addressing the challenge of sparse rewards. Specifically, for many real-world RL tasks, defining a dense reward function is

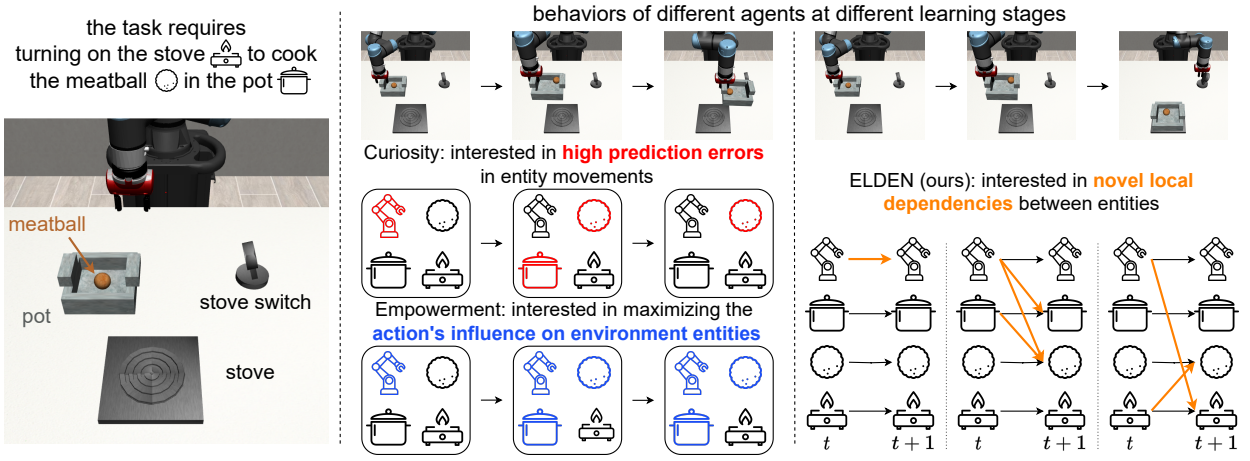


Figure 5.1: **(Left)** In a kitchen task with multiple potential agent-object and object-object interactions, **(Middle)** for a curiosity-based agent interested in hard-to-predict factor motion, it will initially focus on exploring arm movement, then on pot and meatball manipulation, and finally keep rolling the meatball whose outcomes are challenging to predict. On the other hand, for an empowerment-based agent interested in maximizing the action's influence, it begins with controlling the arm and then learning to move the pot and meatball simultaneously, but it ignores the potential interaction between the stove and the meatball. **(Right)** ELDEN avoids those issues by identifying whether dependencies between factors happen and focusing the exploration on novel ones. After the agent learns that it can control the pot and meatball, it will move on to explore other potential interactions, e.g., whether the stove can influence the meatball. Hence it has a larger opportunity to learn this task, compared with a curiosity or empowerment-based agent.

non-trivial, yet a sparse reward function based on success or failure is directly available. For such reward functions, learning good policies is often challenging, as it requires efficient exploration of the state space.

To address this challenge, RL researchers introduced the use of an *intrinsic reward function*, an additional task-agnostic signal given to the agent for visiting *interesting states*. Intrinsic reward functions can be roughly classified into two main paradigms: curiosity (Pathak et al., 2017; Schmidhuber, 1991; Burda et al., 2019b) and empowerment (Stadie et al., 2015; Seitzer et al., 2021; Klyubin et al., 2005a) , where the agent is rewarded either for visiting novel states or for obtaining maximal control over the environment, respectively.

While these methods significantly improve exploration in some domains, there are cases where the aforementioned methods fail. Consider, for example, a kitchen environment with several objects where there are multiple potential agent-object and object-object interactions, and an agent

is tasked with putting a meatball in a pot and cooking it on the stove (Figure 5.1). On the one hand, curiosity-driven methods will encourage the agent to explore the environment by visiting states where the exact outcome of an action is uncertain. Consequently, for each interactable object, the agent will exhaust any possible interaction until it can accurately predict every change in the object's state. As a result, such an exploration strategy can be inefficient, especially for environments with many objects. In the kitchen example, it is hard to predict how the meatball rolls in the pot, and thus the curiosity-driven agent would keep rolling it. On the other hand, for empowerment methods, the agent is encouraged to remain in states where it can influence as many states (objects) simultaneously as possible (e.g. holding the pot with the meatball inside). By doing so, however, it ignores object-object interactions that the action cannot directly control but indirectly induce, which can be the key to task completion. In the kitchen case, an empowerment-driven agent will therefore not be interested in placing the pot and the meatball on the stove, as it forfeits control of them by doing so, even though this action enables the stove to heat the meatball. Our main insight is that, in this type of environment, an intelligently exploring agent should be able to learn that it can use the pot to move the meatball after a few trials. Then, instead of spending time learning the complex meatball movement or different styles to manipulate the pot, it would move on to explore other modes of interacting with other objects, e.g., putting the pot on the stove and switching on the stove.

Following this motivation, we introduce a new definition of interesting states — focusing on *whether* the environment factors (consisting of the agents and objects) can interact, rather than *how* exactly they interact. We present ELDEN, **E**xploration via **L**ocal **D**ep**EN**dencies, a novel intrinsic reward function that models the local dependencies between factors in the scene (agent-object, object-object) and uses the uncertainty about the dependencies to guide exploration. By relaxing the curiosity signal from dynamics prediction to dependencies prediction, ELDEN implicitly biases the exploration toward states where novel interaction modes happen rather than states where the state value is novel but dependencies remain the same. Specifically, ELDEN trains an ensemble of dynamics models. In each model, the local dependencies between objects are modeled by the partial derivatives of state predictions w.r.t. the current state and action. Then, the local dependency uncertainty is measured as the variance across all dynamic models.

We evaluate ELDEN on discrete and continuous domains with multiple objects leading

to many interaction modes and tasks with chained dependencies. Our results show that using a partial derivative-based extractor on dynamic models training allows us to accurately identify local connectivities between environment factors. Furthermore, the intrinsic reward function derived from the identified local connectivities allows ELDEN to outperform state-of-the-art exploration methods (curiosity and empowerment-driven) on the tested domains.

5.2 Exploration via Local Dependencies (ELDEN)

In the section, we introduce ELDEN, which infers the local dependencies between environment entities and uses the uncertainty of dependencies as an intrinsic reward function for tackling hard-exploration problems. In Section 5.2.1, we formally define the problem setup of ELDEN. In Section 5.2.2, we discuss how ELDEN uncovers local dependencies. In Section 5.2.3, we describe how ELDEN facilitates exploration with the intrinsic reward function.

5.2.1 Problem Setup

In this chapter, we focus on the factored MDP setup with the following assumptions that are the same as in the previous two chapters.

Assumption 2. *The state is fully observable and the state factorization is given.*

Assumption 5. *There is no simultaneous causal relationship, i.e.,* $\forall i, j, \mathcal{S}_t^i \not\to \mathcal{S}_t^j$.

Assumption 6. *The transitions for each state factor are independent, i.e.,* $p(\mathcal{S}_{t+1} \mid \mathcal{S}_t, \mathcal{A}_t) = \prod_{i=1}^{d_\mathcal{S}} p(\mathcal{S}_{t+1}^i \mid \mathcal{S}_t, \mathcal{A}_t)$.

As described in Section 3.2.2, for Assumption 2, a factored state space is commonly employed in the causality literature and applies to many simulated or robotics environments. In cases where low-level observations or partial observability are present, disentangled representation or causal representation methods can be utilized to learn a factored state space, as will be discussed in Chapters 7 and 8. When a factored state space is available, Assumptions 5 and 6 generally hold.

In contrast to the previous two chapters that identify general causality, this chapter focuses on identifying local causal dependencies that are specific to a given transition (in this chapter, we use "actual dependencies" and "local dependencies" interchangeably). The reason to focus on local causality is that, for many environments, the general causal dependency graph $\mathcal{G}$ can be dense or even fully connected, because whenever it is possible for $\mathcal{S}^j$ to depend on $\mathcal{S}^i$, no matter how unlikely, it is necessary to include the edge $\mathcal{S}^i \to \mathcal{S}^j$. However, in the real world, even if possible to interact, most entities are independent of each other most of the time. Following this observation, we are interested in inferring *local* dependencies that are specific to (s_t, a_t), represented by a local causal graph $\mathcal{G}_t$ that is minimal by removing inactive edges in $\mathcal{G}$.

5.2.2 Identifying Local Dependencies with Dynamics Partial Derivatives

Based on the definition in Section 5.2.1, the key component of ELDEN is to accurately evaluate which potential dependencies between environment entities are locally active, i.e., identify the local causal graph $\mathcal{G}_t$ given (s_t, a_t). This identification requires answering a challenging question — whether factor i's value, $\mathcal{S}_t^i = s_t^i$ is the cause of factor j to have value $\mathcal{S}_{t+1}^j = s_{t+1}^j$. ELDEN approaches it with the inspiration from the *but-for* test: the local dependency exists if $\mathcal{S}_{t+1}^j = s_{t+1}^j$ would not happen but for $\mathcal{S}_t^i = s_t^i$. In other words, we assess whether $\mathcal{S}_{t+1}^j$ would change if $\mathcal{S}_t^i$ has a different value. Notice that, since we focus on *local* dependencies, we only want to vary $\mathcal{S}_t^i$ near its actual value s_t^i rather than trying all its possible values.

To this end, ELDEN utilizes partial derivatives to identify local dependencies, as they naturally capture the extent of change in $\mathcal{S}_{t+1}^j$ with respect to $\mathcal{S}_t^i$. Specifically, assuming the access of ground truth transition probability p (which we will relax later), ELDEN considers $\mathcal{S}_{t+1}^j = s_{t+1}^j$ locally depends on $\mathcal{S}_t^i = s_t^i$ if

$$\left| \frac{\partial p(s_{t+1}^j | s_t, a_t)}{\partial s_t^i} \right| \geq \epsilon_\partial, \tag{5.1}$$

where ϵ_∂ is a predefined threshold. A large partial derivative indicates that a slight change in $\mathcal{S}_t^i$ will lead to a substantial change in $\mathcal{S}_{t+1}^j$, thus satisfying the but-for test.

To evaluate partial derivatives without the ground truth transition probability, ELDEN approximates p with a learned dynamics model f parameterized by a neural network $\hat{p}(s_{t+1}^j) =$

$f(s_t, a_t)$ and trains f by maximizing the log-likelihood of $\hat{p}(s^j_{t+1})$ (for notational simplicity, we omit the conditionals s_t, a_t in $\hat{p}$ in this section). Due to limited data and training errors, even when two environment entities are not locally dependent, there occasionally exists a large partial derivative between them. To reduce such false positives, ELDEN further applies regularization to suppress partial derivatives w.r.t. inputs that are not necessary for predicting s^j_{t+1}. The overall loss of dynamics training is

$$L_f = -\log \hat{p}(s^j_{t+1}) + \lambda \sum_{i,j} \left| \frac{\partial \hat{p}(s^j_{t+1})}{\partial s^i_t} \right|, \tag{5.2}$$

where λ is the regularization coefficient.

The training loss for the dynamics model is:

$$L = -\log \prod_{j=1}^{N} \hat{p}(s^j_{t+1}|s_t, a_t) + \lambda \sum_{i,j} \left| \frac{\partial \hat{p}(s^j_{t+1})}{\partial s^i_t} \right|, \tag{5.3}$$

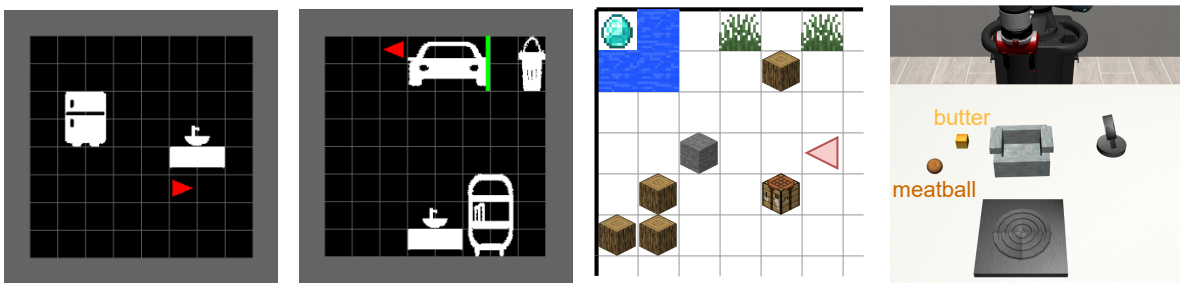
where λ is the coefficient for partial derivative regularization.

5.2.3 ELDEN Policy Learning

Algorithm 6 Training of ELDEN (on-policy)

1: Initialize the dynamics ensemble $\{f^i\}_{i=1}^N$, the policy π, and the replay buffer.
2: **for** number of training iterations **do**
3: Collect an environment transition $(s_t, a_t, r_{t,\text{task}}, s_{t+1})$ with current policy π.
4: $\mathcal{G}^i_t = f^i.\text{compute\_}\mathcal{G}_t(s_t, a_t)$ (Section 5.2.2). ▷ Compute the local dependency graph
5: $r_{t,\text{intrinsic}} = \text{variance}(\{\mathcal{G}^i_t\}_{i=1}^N)$.
6: $r_t = r_{t,\text{task}} + \beta \cdot r_{t,\text{intrinsic}}$.
7: Add (s_t, a_t, r_t, s_{t+1}) to the replay buffer.
8: Update the policy π with (s_t, a_t, r_t, s_{t+1}) (Section 5.2.3).
9: Update the dynamics ensemble $\{f^i\}_{i=1}^N$ with (s_t, a_t, s_{t+1}) sampled from the buffer (Section 5.2.2).
10: **end for**

ELDEN utilizes the local dependency identification described in Section 5.2.2 to improve exploration for model-free RL. The key idea behind ELDEN is to encourage an agent to visit states where new local dependencies are likely to emerge. When the ground truth local dependencies are available, the novelty of local dependencies between state factors can be measured by the magnitude



(a) Thawing (b) CarWash (c) Minecraft 2D (d) Kitchen

Figure 5.2: We test ELDEN on three domains and four environments. **(a) (b)** Mini-behavior and **(c)** Minecraft 2D with discrete state spaces, where the agent has to achieve a series of temporally extended tasks with complex object interactions. **(d)** Robosuite, a robot table-top manipulation simulation environment with continuous state spaces, where the robot needs to perform multiple interdependent subtasks to finish the cooking task.

of error of the dependencies identified by our method. Unfortunately, in many cases, it is hard to manually specify the ground truth local dependencies. Instead, ELDEN trains an ensemble of dynamics models, and measures dependency novelty by the variance of the local dependency graphs extracted independently from each of the dynamics models in the ensemble. Specifically, ELDEN first computes the variance of each edge in the graph and then uses the mean of the edge variance as the graph variance. Finally, the calculated variance is used as the intrinsic reward and is scaled with a coefficient β that controls the magnitude of the exploration bonus, and added to the task reward r_{task}.

ELDEN is applicable to both on-policy and off-policy settings. We show the pseudo-code for on-policy ELDEN in Algorithm 6. The off-policy version of ELDEN can be easily derived by updating the policy with transitions sampled from the replay buffer in line 8 of Algorithm 6. Importantly, any model-free RL algorithm can be used for the policy update step.

5.3 Experiments

In this section, we provide empirical evaluations pertaining to the following questions that support the indicted answers.

- **Q1)** Is ELDEN able to accurately detect local dependencies between environment factors in factored state spaces? Yes (Section 5.3.1).
- **Q2)** Does ELDEN improve the performance of RL algorithms in sparse-reward environments? Yes (Section 5.3.2).
- **Q3)** How do different design choices affect ELDEN's local dependency and task learning? (Section 5.3.3).

Environments As shown in Figure 5.2, we evaluate ELDEN in four simulated environments with different objects that have complex and chained dependencies: (1) **CarWash**, (2) **Thawing**, (3) **2D Minecraft** and (4) **Kitchen**. Both CarWash and Thawing are long-horizon household tasks in discrete gridworld from the Mini-BEHAVIOR Benchmark Jin et al. (2023). 2D Minecraft is an environment modified from the one used by Andreas et al. (2017), where the agent needs to master a complex technology tree to finish the task. Kitchen is a continuous robot table-top manipulation domain implemented in RoboSuite (Zhu et al., 2020b). To complete tasks in these environments and receive the sparse reward, the agent has to conduct a series of actions that change not only the state of the interacted factors but also induce further interaction between interacted factors and others (e.g., interacting with the stove switch that enact interaction between the stove and the cooking the meatball). The agents in all environments can select between a set of action primitives, a set of discrete actions that can be applied on each object, e.g., `goTo(obj)` or `pick(obj)`. Notice that even with the action primitives, these domains are still very hard to solve due to the presence of many interaction modes and the difficulty in finding the correct (potentially long) sequence of interactions among many options that will lead to task success.

Implementation Details For discrete state or action spaces, the partial derivatives w.r.t. s_t/a_t are undefined. To address this issue, we use Mixup (Zhang et al., 2018a) to create synthetic inputs and labels by linearly combining pairs of inputs and labels, thus approximately changing the input space to be continuous. Compared to learning from discrete inputs only, dynamics models trained on such data generate partial derivatives that better reflect local dependencies, as shown in Sec 5.3.3.

For the 2D Minecraft and Kitchen environments where some local dependencies have complex preconditions and thus are hard to induce, we apply sample prioritization to dynamics learning, where the priority is measured as prediction error. In this way, the dynamics model gets aware of unknown interactions faster and guides the exploration more efficiently than not using prioritization.

Network Architecture In Figure 5.3(a), the architecture of ELDEN for predicting each state factor s_{t+1}^j is illustrated. The process consists of the following steps.

1. Feature Extraction: For each input state factor s_t^i, ELDEN utilizes a separate multi-layer perception (MLP) to extract its corresponding feature φ^i.

2. Entity Interaction: ELDEN employs a multi-head self-attention module to model factor interactions and generates a set of transformed features ϑ^i that incorporate information from other state factors.

3. Prediction using Multi-Head Attention: With ϑ^j as the query, ELDEN utilizes a multi-head attention module to compute the prediction $\hat{p}(s_{t+1}^j|s_t, a_t)$ for each state factor. For continuous state factor, $\hat{p}(s_{t+1}^j)$ is modeled as a normal distribution with the mean computed by the network and a fixed variance equal to 1. For discrete factor, $\hat{p}(s_{t+1}^j)$ is a categorical distribution with network outputs as class probabilities.

Throughout the prediction process, there are a total of $d_{\mathcal{S}}$ such networks in ELDEN, with each network responsible for predicting a separate state factor s_{t+1}^j.

5.3.1 Evaluating the Detection of Local Dependencies

We compare the local dependencies extracted by ELDEN with the following baselines.

- **pCMI** (point-wise conditional mutual information): it considers that the local dependency $s_t^i \to s_{t+1}^j$ exists if their point-wise conditional mutual information is greater than a predefined threshold, i.e., $\text{pCMI}^{i,j} := \log \frac{p(s_{t+1}^j|s_t,a_t)}{p(s_{t+1}^j|x_t^{\neg i})} \geq \epsilon_\partial$. As shown in Figure 5.3(b), to compute $\text{pCMI}^{i,j}$, we use the same architecture as CDL introduced in Chapter 3, where a manually defined binary mask $M \in [0,1]^{d_{\mathcal{S}}}$ decides which inputs are ignored when predicting s_{t+1}^j: (1) to compute

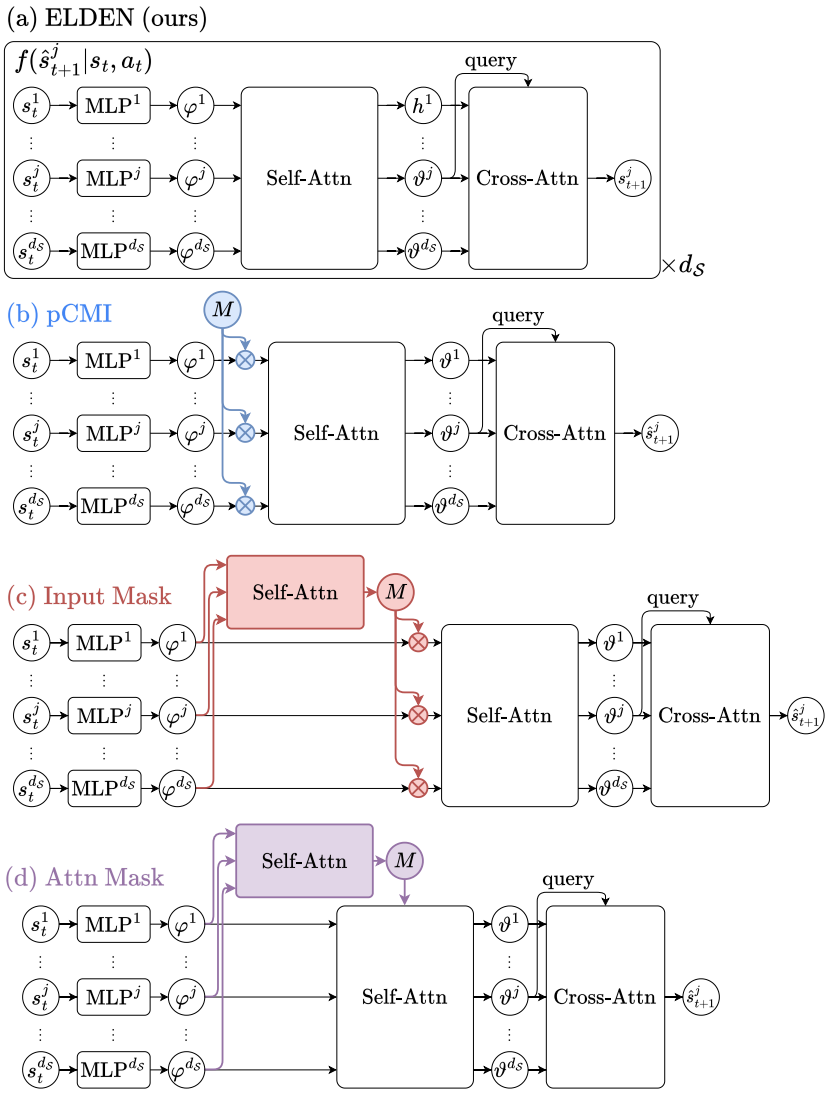


Figure 5.3: The dynamics model of each local dependency detection method. **(a)** The dynamics model of ELDEN for predicting s^j_{t+1}. Notice that each network predicts s^j_{t+1} only, and there are d_S such networks in total, each responsible for predicting one state factor in s_{t+1}. For visual simplicity, the "$\times d_S$" symbol is only shown in (a). **(b)** pCMI computes $p(s^j_{t+1}|s_t, a_t)$ and $p(s^j_{t+1}|x_t^{\neg i})$ by manually setting the binary mask M to different values, where $\otimes$ represents element-wise multiplication. **(c)** For Input Mask, M is learned to condition on (s_t, a_t) and is regularized to use as few inputs as possible. **(d)** For Attn Mask, M also conditions on (s_t, a_t) but is applied to the attention score in the self-attention module.

$p(s^j_{t+1}|s_t, a_t)$, M uses all inputs (all its entries are set to 1), and (2) to compute $p(s^j_{t+1}|x_t^{\neg i})$,

Table 5.1: Mean ± std. error of ROC AUC (↑) and F1 (↑) of local dependency prediction

	Thawing		CarWash		Kitchen	
	ROC AUC	F1	ROC AUC	F1	ROC AUC	F1
ELDEN	**0.71** ± 0.01	0.57 ± 0.00	**0.78** ± 0.02	0.66 ± 0.02	**0.66** ± 0.01	0.25 ± 0.01
pCMI	0.55 ± 0.01	0.60 ± 0.00	0.73 ± 0.02	**0.78** ± 0.01	0.60 ± 0.00	**0.28** ± 0.00
Attn	0.65 ± 0.04	**0.63** ± 0.01	0.66 ± 0.01	0.55 ± 0.03	0.51 ± 0.01	0.22 ± 0.02
Input Mask	0.50 ± 0.00	0.40 ± 0.00	0.50 ± 0.00	0.32 ± 0.01	0.50 ± 0.00	0.08 ± 0.00
Attn Mask	0.45 ± 0.03	0.47 ± 0.02	0.47 ± 0.07	0.43 ± 0.03	0.52 ± 0.01	0.13 ± 0.01

the entry for φ^i is set to 0. When evaluating the local dependency, pCMI needs to compute $p(s^j_{t+1}|x^{\neg i}_t)$ for every i, and thus its computation cost is $d_{\mathcal{S}}$ times larger than ELDEN. We also computes pCMI following Seitzer et al. (2021), which yields similar performance but is even more computationally expensive compared to CDL.

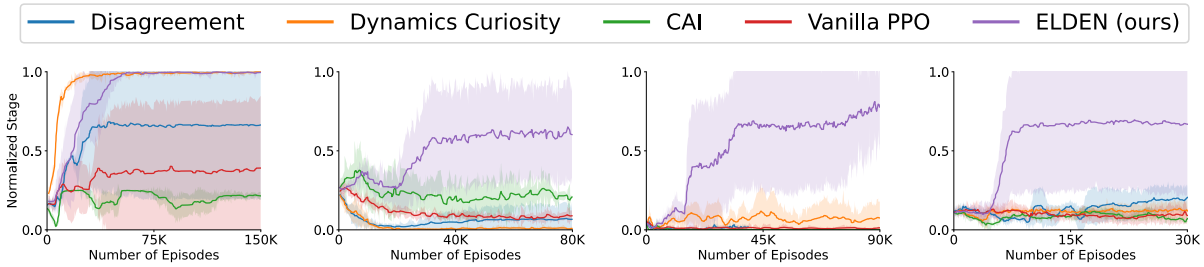
- **Attn** (attention): it uses the same architecture as ELDEN that is shown in Figure 5.3(a). When computing the overall attention score, it averages the attention score across all heads in each module, then computes the likelihood of dependency $s^i_t \to s^j_{t+1}$ as $\sum_{k=1}^{d_{\mathcal{S}}} \text{attn}(\varphi^i, \vartheta^k) \cdot \text{attn}(\vartheta^k, s^j_{t+1})$ where $\text{attn}(x, y)$ is the averaged score between the input x and the output y.

- **Input Mask**: as shown in Figure 5.3(c), it also uses a binary mask M except that M is computed from (s_t, a_t). During training, to only use necessary inputs for s^j_{t+1} prediction, M is regularized with the L1 norm on its number of non-zero entries. The Gumbel reparameterization is used to compute the gradient for the binary M (Jang et al., 2017).

- **Attn Mask**: as shown in Figure 5.3(d), similar to Input Mask, a mask M of size $d_{\mathcal{S}} \times d_{\mathcal{S}}$ is computed from (s_t, a_t), but it is applied to the attention score. The mask is regularized with Stochastic Kernel Modulated Dot-Product (SKMDP) proposed by Weiss et al. (2022).

For modules that are shared by all methods, we use the same architecture for a fair comparison.

We train the dynamics model of each method with three random seeds on pre-collected transition data and evaluate their performance by predicting the local causal graph $\mathcal{G}_t$ for 50 unseen episodes based on the state-action pair (s_t, a_t). We compare their predictions with the ground truth local dependencies extracted from the simulator. In the three environments, many potential local



(a) Thawing (b) CarWash (c) Craft (d) Kitchen

Figure 5.4: Learning curve of ELDEN (ours) compared to baseline approaches. Each method uses three random seeds, and we show the mean $\pm$ std dev of the number of stages completed toward task success. The stage count is normalized to [0, 1], where 1 corresponds to task completion. ELDEN learns successful policies in all four test environments, and is the only method that succeeds in the *CarWash*, *2D Minecraft*, and *Kitchen* environments with complex chained dependencies.

dependencies are inactive most of the time, and thus only a small portion ($\leq 3\%$) of the ground truth labels indicate the existence of local dependencies for a given factor pair. To account for such imbalance, we use the area under the receiver operating characteristic curve (ROC-AUC) and the best achievable F-score (F1) as evaluation metrics.

The results of the evaluation on the detection of local dependencies are summarized in Table 5.1. ELDEN outperforms all baselines in terms of ROC-AUC consistently across all environments. For the F1 score, pCMI performs best in most environments (especially in the more complex CarWash and Kitchen), but ELDEN performs comparably or achieves the second-best F1 scores with much less computation: pCMI computation cost is $d_{\mathcal{S}}$ times higher than ELDEN, where $d_{\mathcal{S}}$ is the number of environment factors, and thus pCMI scales badly to environments with a large number of objects. Hence, results in Table 5.1 answer Question **Q1** in the affirmative.

5.3.2 Evaluating Exploration in Sparse-Reward RL Tasks

The ultimate goal of our method is to improve exploration for RL in sparse-reward setups. In the second evaluation, we compare the performance of ELDEN against several state-of-the-art intrinsic-motivation exploration algorithms in reinforcement learning, including:

- **Disagreement** (Pathak et al., 2019): the intrinsic reward is computed based on the variance of the

predictions from an ensemble of forward dynamics models.

- **Dynamics Curiosity** (Burda et al., 2019a): intrinsic reward is computed based on the prediction error of a trained forward dynamics model.
- **CAI** (Causal Influence Detection) (Seitzer et al., 2021): an empowerment-based method, where the agent is given intrinsic reward for maximizing the number of state factors that depend on its action.
- **Vanilla PPO** (Schulman et al., 2017): baseline without intrinsic reward that serves as control signal.

While ELDEN can be used with any RL algorithm, in our experiments we use proximal policy optimization (PPO), as well as with the baselines (Schulman et al., 2017). To facilitate the introspection of the results, we define manually a set of semantic stages representing internal progress toward task completion. Notice that these stages are not used by the agents during training and do not provide any additional reward; they are only used to facilitate the analysis of the results.

Figure 5.4 depicts the count of reached stages per episode during training for each task, normalized by the number of stages to complete the task. In the normalized stage count, a value of 1 corresponds to successfully completing the task and it is the only stage where the learning agents receive sparse task reward (not intrinsic). Figure 5.4 indicates that ELDEN is able to learn successful policies in all four environments. Importantly, in CarWash, 2D Minecraft and Kitchen, ELDEN is the only method that successfully learns to complete the task, demonstrating the advantage of ELDEN over the baseline algorithms in tackling tasks with complex chained dependencies, thus answering Question **Q2** in the affirmative.

The normalized stage count of ELDEN in CarWash, 2D Minecraft and Kitchen does not converge to 1 (completing the entire task in all episodes) mainly due to two reasons: First, in both tasks, the locations of the objects are randomly initialized at the start of each episode. For some initialization (e.g. a target object is blocked by unmovable obstacles), the task is impossible to solve. Second, in both tasks, two out of the three ELDEN training procedures with different random seeds converge to succeeding most of the time, but the training process with one seed fails to find a good

Table 5.2: Ablation of ELDEN on local dependency prediction (mean ± std. error of ROC AUC and F1)

	Thawing		CarWash		Kitchen	
	ROC AUC (↑)	F1 (↑)	ROC AUC (↑)	F1 (↑)	ROC AUC (↑)	F1 (↑)
no Mixup & no Reg	0.48 ± 0.01	0.42 ± 0.01	0.44 ± 0.00	0.27 ± 0.01	N/A	N/A
no Reg, i.e., $\lambda = 0$	0.57 ± 0.01	0.52 ± 0.01	0.54 ± 0.01	0.42 ± 0.01	0.64 ± 0.01	0.24 ± 0.01
$\lambda = 10^{-1}$	0.68 ± 0.00	**0.57** ± 0.00	0.73 ± 0.01	0.58 ± 0.02	0.55 ± 0.00	0.14 ± 0.00
$\lambda = 10^{-2}$	**0.71** ± 0.01	**0.57** ± 0.00	0.76 ± 0.01	0.60 ± 0.00	0.60 ± 0.01	0.21 ± 0.01
$\lambda = 10^{-3}$	0.64 ± 0.01	0.55 ± 0.01	**0.78** ± 0.02	**0.66** ± 0.02	0.65 ± 0.00	0.24 ± 0.01
$\lambda = 10^{-4}$	0.65 ± 0.02	0.55 ± 0.01	0.75 ± 0.01	0.60 ± 0.01	**0.66** ± 0.00	**0.25** ± 0.01
$\lambda = 10^{-5}$	0.63 ± 0.00	0.53 ± 0.01	0.72 ± 0.00	0.57 ± 0.00	0.65 ± 0.01	0.24 ± 0.01

policy, dragging down the mean value of the normalized stage count. This large variance in success is a current limitation of ELDEN.

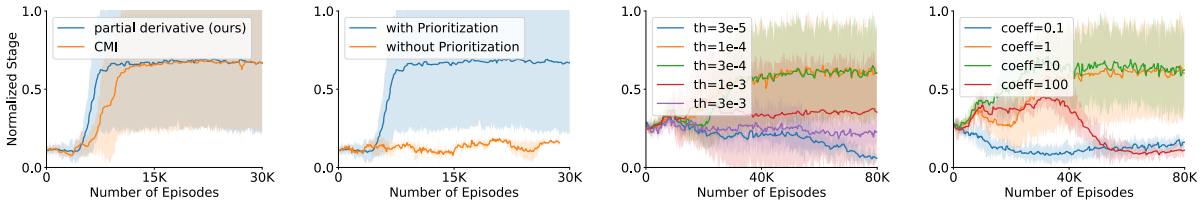
In the relatively simple Thawing environment, we found ELDEN does not provide a significant advantage over the other baseline methods. The **Dynamics Curiosity** baseline learns faster to achieve the task indicating a better sample efficiency. This was rather expected: as with any exploration heuristic, ELDEN is not universally better than previous intrinsic reward methods — instead, it is better suited for a specific type of environment, where there are many complex and chained object dependencies, not the case for Thawing.

5.3.3 Ablation Studies

We ablate different components of ELDEN to examine their importance to the overall methods.

Ablations for Local Dependency Detection In our ablation study on ELDEN for local dependency detection, we investigate the impact of each component with the following variations.

- No Mixup & No Reg: We disable the use of Mixup for discrete space prediction, and no partial derivative regularization is applied in this case.
- Different partial derivative regularization coefficients: we test with different λ values in $\{0, 10^{-1},$



(a) Dependency detection (b) Sample prioritization (c) Gradient thresholds (d) Reward coefficients

Figure 5.5: Ablation of ELDEN on task learning. Each curve uses three random seeds and shows the mean $\pm$ std dev of the normalized stages. We found ELDEN to have moderate tolerance towards hyperparameters. We found sample prioritization in dynamics learning to be crucial to the performance of ELDEN.

$10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.

As shown in Table 5.2, in the Thawing and CarWash environments, partial derivative regularization with appropriate coefficients significantly improves ELDEN's detection of local dependencies, compared to no regularization (i.e., $\lambda = 0$) or inappropriate λ values. Furthermore, in discrete-state environments, Mixup smooths the landscape of partial derivatives by providing synthesized continuous inputs as exemplified in Figure 1(b) of Zhang et al. (2018a), thus facilitating local dependency prediction — even when compared to using Mixup without any regularization, not using Mixup leads to a noticeable degradation in the prediction performance.

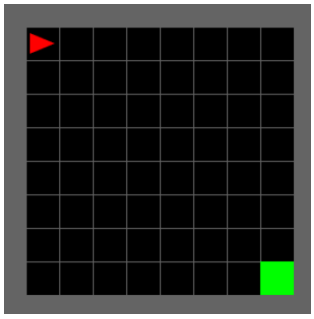
Ablations for Task Learning Next, we examine how different components and hyperparameters of ELDEN affect task learning:

- **Local Dependency Metrics**: We compare the exploration performance when using different local dependency detection methods. Specifically, we compare with pCMI as it achieves the best local dependency detection in Section 5.3.1. We present the comparison results between ELDEN and pCMI in the Kitchen environment in Figure 5.5(a) where both methods successfully learn to solve the task. However, it is important to notice that the computation cost of pCMI is $d_{\mathcal{S}}$ times more than that of ELDEN (where $d_{\mathcal{S}}$ is the number of environment factors), and thus may not scale to environments with a large number of factors.

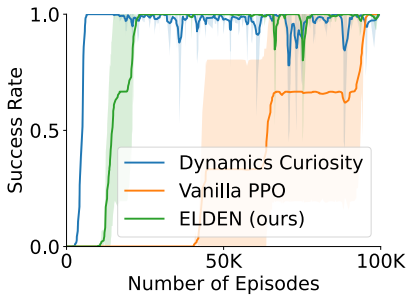
- **Dynamics Sample Prioritization**: We study the effectiveness of applying sample prioritization in dynamics model training. Specifically, we test ELDEN with and without prioritization in the Kitchen environment, and show the result in Figure 5.5(b). We can see that ELDEN without prioritization fails to learn a useful policy. The reason is that some key factor interactions occur rather rarely before the agent masters them, e.g., frying meatball with butter. In such cases, the dynamics model needs to quickly learn that unknown dependencies appear so that it can bias the exploration toward reproducing such dependencies. Sample prioritization helps the dynamics model learn such infrequent dependencies quickly, making it critical in environments with novel and hard-to-induce local dependencies.

- **Partial Derivative Threshold**: The partial derivative threshold ϵ_∂ determines the dependency predictions. A threshold that is too large / too small will make all dependency predictions negative / positive respectively, leading to deteriorated performance. In this section, we examine whether our method is sensitive to the choice of threshold in the CarWash environment, where the results are presented in Figure 5.5(c). We observe that our method is relatively sensitive to the choice of threshold, and an inappropriate threshold could cause catastrophic failure. A potential next step for ELDEN is to automatically determine the partial derivative threshold.

- **Intrinsic Reward Coefficient**: The intrinsic reward coefficient controls the scale of the intrinsic reward relative to the task reward. We examine the effect of this coefficient by experimenting with different values in the CarWash environment, where the results are presented in Figure 5.5(d). We find that our methods work well in a large range of the intrinsic reward coefficients (1 - 10), since the task only gives sparse rewards and the intrinsic rewards are the only learning signal most of the time. The only exceptions are (1) when the intrinsic reward coefficient is too large (e.g., 100), the intrinsic reward significantly surpasses the task reward, and (2) when the coefficient is too small (e.g., 0.1), the episode intrinsic reward is too small (e.g., 0.03) for PPO to learn any useful policy.



(a) a navigation task with the goal in green



(b) ELDEN performs worse than Dynamics Curiosity

Figure 5.6: We demonstrate a failure mode of our method on a navigation task.

5.3.4 Failure Modes of ELDEN

Intrinsic rewards inherently encode specific exploratory biases, and no single mechanism, including ELDEN, can be expected to outperform others across all environments. In particular, ELDEN may have limited advantages for tasks that require precise control of a specific environment factor. One such example is navigation, where the agent needs to reach a very specific point in space that has no particular semantic meaning. We empirically examine this statement in the Minigrid environment (Chevalier-Boisvert et al., 2019), where the agent needs to navigate to the green goal point in an empty room through primitive actions (turn left, turn right, and move forward), as shown in Figure 5.6(a). We compare ELDEN against Dynamics Curiosity and Vanilla PPO, and present the result in Figure 5.6(b). Since this environment is relatively simple, all three methods are eventually able to solve the task. However, the Dynamics Curiosity converges faster than ELDEN, showing that ELDEN is indeed not as capable as curiosity-driven explorations in tasks that focus on precise control rather than exploring dependencies between environment factors. The Vanilla PPO converges slowest, indicating that even in the Empty environment, ELDEN still has advantages over purely random exploration.

5.4 Summary

This chapter introduces ELDEN, a method for improving exploration in sparse reward reinforcement learning tasks. ELDEN identifies local dependencies between environment entities

and uses the uncertainty about such dependencies as an intrinsic reward function to facilitate exploration. Experiments demonstrate that ELDEN uncovers local dependencies more accurately compared to related methods, and significantly outperforms previous exploration methods in tasks with complex chained dependencies.

While ELDEN improves sample efficiency compared to relying only on the sparse reward, it has several limitations.

- First, ELDEN intentionally biases exploration towards "covering up the possible interactions between objects" rather than "becoming an expert at manipulating a particular object". While such an inductive bias works well in many practical domains, it may fail when facing tasks that require a precise object interaction (e.g., rotating the meatball in the pot to a specific orientation). A future direction to alleviate this problem and expand the scope of solvable tasks is to combine ELDEN with dynamics curiosity and formulate a composite intrinsic reward function.

- Second, ELDEN utilizes the intrinsic reward solely as a reward augmentation signal for a single task. As a result, whenever the task changes, the agent typically learns it from scratch and discards the knowledge of how to induce certain interactions between state factors, leading to inefficiencies and poor reuse of the acquired exploratory behaviors. A more scalable alternative is to treat these interaction-inducing behaviors as reusable skills, instead of discarding them across tasks. Motivated by this insight, Chapter 6 seeks to utilize the intrinsic reward to learn a library of skills, each focusing on inducing a specific type of state factor interaction, allowing us to reapply learned skills across many downstream tasks.

- Finally, the assumptions that the state space is fully observable and the factorization is known may not hold in all environment, such as those where only image observations are available. To address this limitation, Chapters 7 and 8 introduce two novel approaches to extract state factors through representation learning.

Overall, the results in this chapter represent the completion of our efforts to design an

intrinsic reward function to augment the sparse task reward and facilitate RL sample efficiency (Chapter 1.1 Contribution 2).

Chapter 6: Unsupervised Skill Discovery from Actual Causality

As introduced in Chapter 5, ELDEN proposed an intrinsic reward function that encourages the agent to induce novel local dependencies between state factors. However, ELDEN employed this signal solely as a reward augmentation for a single task, and the interaction-inducing behaviors are discarded once that task is learned. Instead, this chapter introduces a method for leveraging this intrinsic reward function to learn a library of skills, each inducing a specific type of state factor interaction. Unlike in ELDEN, these skills can be reused across multiple downstream tasks. Furthermore, during downstream task learning, using these learned skills in place of primitive actions shortens the effective exploration horizon and improves RL sample efficiency from an action-space perspective.

This chapter is structured as follows. Section 6.1 first introduces the motivation behind skill discovery and limitations in previous work. Section 6.2 presents the proposed algorithm SkiLD and describes how it leverages the intrinsic reward function introduced in the previous chapter to learn skills in a hierarchical way. Section 6.3 presents the empirical evaluation of SkiLD, covering its performance in learning diverse skills and downstream tasks. Finally, Section 6.4 discusses limitations of SkiLD and how following chapters address them.

Contribution: In addition to my advisor, SkiLD is joint work with Caleb Chuck, Jiaheng Hu, Stephen Chen, Roberto Martín-Martín, Amy Zhang, and Scott Niekum. My contributions are the identification of local causal relationships. Meanwhile, Chuck designed the training framework for the high-level policy that selects which skill to execute, Hu designed low-level skill policy learning, and Chen helped with experiments, while Martín-Martín, Zhang and Niekum provided technical advice.

6.1 Motivation

While the previous chapter utilizes the intrinsic reward to augment sparse task reward, this chapter focuses on leveraging it to learn a library of reusable skills, each inducing a specific type

of state factor interaction. Specifically, humans and other intelligent creatures can learn, without external reward supervision, behaviors that produce repeatable and predictable changes in the environment (Du et al., 2023). These behaviors, which we call *skills*, can be later repurposed to solve downstream tasks efficiently. One of the promises of this form of unsupervised RL is to endow artificial agents with similar capabilities to discover reusable skills without explicit rewards.

One predominant strategy of prior skill discovery methods focuses on training skills to reach diverse states while being distinguishable (Eysenbach et al., 2019; Song et al., 2023; Park et al., 2023). However, in complex environments that contain many *state factors*—distinct elements such as individual objects in a household, the exponential number of distinct states makes it impossible to learn skills that cover every state. Consequently, these methods typically result in simple skills that only change the easy-to-control factors (e.g., in a manipulation task moving the agent itself to diverse positions or manipulating each factor independently), and fail to cover other desirable but challenging behaviors. Meanwhile, in a factored state space, many downstream tasks require inducing interactions between state factors, e.g., cooking requires using a knife to cut the ingredients and cooking them in a pan, etc. Unsurprisingly, these simple skills often struggle to solve such tasks, resulting in poor downstream performance.

Our key insight is to utilize interactions between state factors as a powerful inductive bias for learning useful skills. In factored state spaces and their downstream tasks, there usually exist bottleneck states that an agent must pass through to explore different regions of the environment, and many of them can be characterized by interactions between state factors. For example, in a household environment, a robot must first grasp the knife before moving it to different locations, with the bottleneck being the interaction between the robot and the knife. In environments that have a large state space due to many state factors, rather than inefficiently relying on randomly visiting different states to reach such bottlenecks, we introduce a method for training the agent to actively induce these critical interactions.

To this end, we introduce Skill Discovery from Local Dependencies (SkiLD), a novel skill discovery method that explicitly learns skills that induce diverse interactions. Specifically, SkiLD models the interactions between state factors using the framework of *local dependencies* and

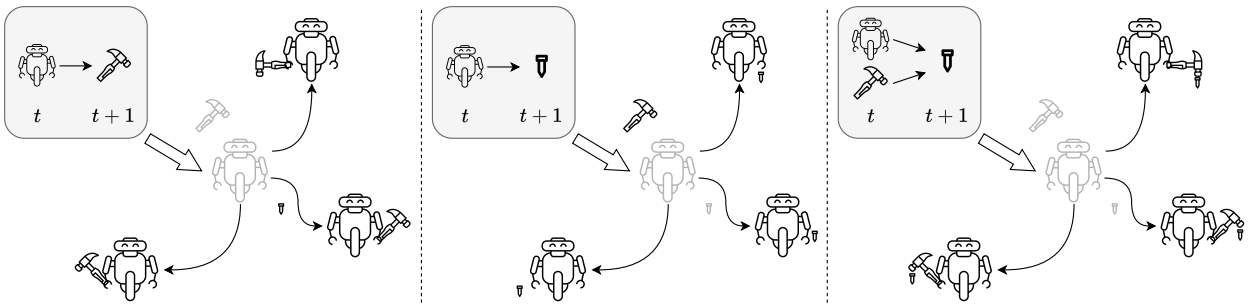


Figure 6.1: **Skill Discovery from Local Dependencies** (SkiLD) describes skills that encode interactions (i.e., local dependencies) between state factors. In contrast to prior diversity-based methods that can easily get stuck by moving the robot to diverse, but non-interactive states, and factor-based methods that are trained to manipulate the hammer and nail, but not their interactions, SkiLD not only manipulate each object (left, middle) but also induce interactions between them (right), by specifying different local dependencies. These skills are often more useful than the "easy" skill learned by previous methods for downstream task-solving.

proposes a novel intrinsic reward that 1) encourages the agent to induce specified interactions, and 2) encourages the agent to discover diverse ways of inducing specified interaction, as visualized in Figure 6.1. During skill learning, SkiLD gradually discovers new interactions and learns to induce them, based on the skills that it already mastered, resulting in a diverse set of interaction-inducing behaviors that can be readily repurposed for downstream tasks. During task learning, the skill policy is reused, and a task-specific policy is learned to select (a sequence of) skills to maximize task rewards efficiently.

We evaluate the performance of SkiLD on factor-rich environments with 10 downstream tasks against existing unsupervised reinforcement learning methods. Our experiments indicate that SkiLD learns to induce diverse interactions and outperforms other methods on most of the examined tasks.

6.2 Skill Discovery from Local Dependencies (SkiLD)

In this section, we describe SkiLD, which enhances skills using local dependencies. SkiLD represents local dependencies as *state-specific dependency graphs*, defined in Section 2.2.2, and learns to induce different dependency graphs in the environment for different skills. To intelligently

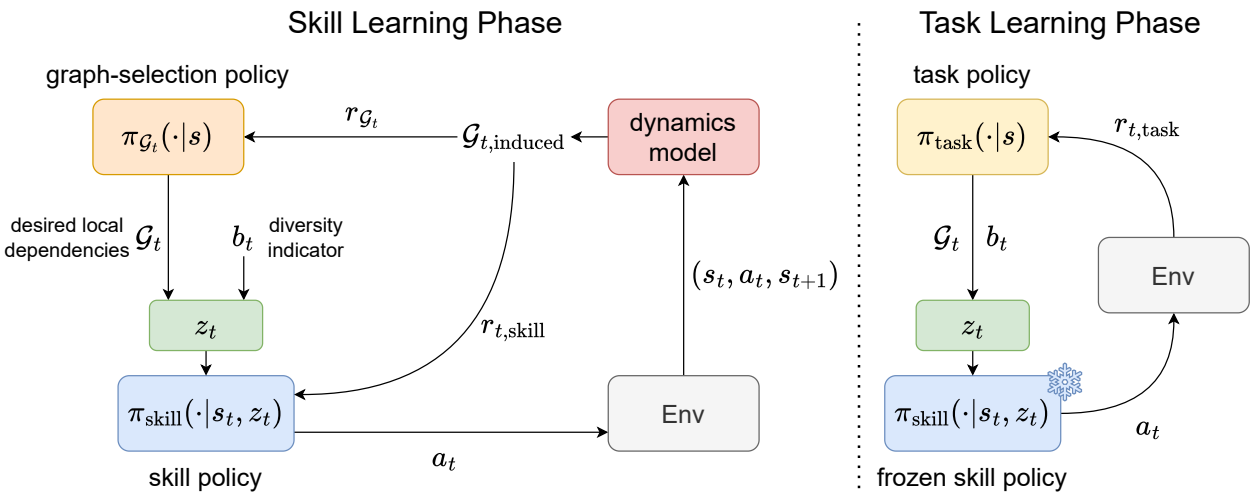


Figure 6.2: During **skill learning** of SkiLD, the graph-selection policy specifies desired local dependencies for the skill policy to induce, and the induced dependency graph is identified by the dynamics model and used to update both policies. During **task learning** (right), the skill policy is kept frozen and a task policy is trained to select skills to maximize task reward.

generate target dependency graphs during training, SkiLD frames unsupervised skill discovery as a hierarchical RL problem described in Figure 6.2 and Algorithm 7, where a high-level graph selection policy chooses target local dependencies to guide exploration and skill learning, and a graph-conditioned skill policy learns to induce the specified local dependencies using primitive actions.

This framework requires formalizing three components: (1) the skill representation $\mathcal{Z}$, presented in Section 6.2.1, (2) the graph selection policy $\pi_{\mathcal{G}_t}(z|s)$ and its reward function $\mathcal{R}_{\mathcal{G}_t}$, presented in Section 6.2.2, and (3) the skill policy $\pi_{\text{skill}}(a|s,z)$ and its corresponding reward function $\mathcal{R}_{\text{skill}}$, presented in Section 6.2.3.

6.2.1 Skill Representation

Prior unsupervised skill discovery methods usually focus skill learning on changing the state or each factor diversely, which is inefficient when there exist bottleneck states for explorations. Consequently, they are can be limited to learning simple skills, for example, only changing the easiest-to-control factor in the state (i.e., the agent itself). To address this problem, SkiLD not only

focuses on changing the state but also considers the interactions between state factors.

Skill Representation. SkiLD represents the skill space as the combination of two components: $\mathcal{Z} = \mathbf{G} \times \mathcal{B}$, where $\mathcal{G}_t \in \mathbf{G}$ is a state-specific dependency graph that specifies the *desired* local dependencies between state factors (e.g., hammering the nail), and $b \in \mathcal{B}$ is a diversity indicator the same as that used in Eysenbach et al. (2019). While the agent inducing particular local dependencies $\mathcal{G}_t$, we use b to further encourage it to visit distinguishable states (e.g., under different b values, training the agent to hammer the nail into different locations). Specifically, the dependency graph is represented as a binary matrix $\mathcal{G}_t \in \{0, 1\}^{N \times (N+1)}$. As described in Section 2.2.2, each edge $\mathcal{G}_t^{ij}$ denotes, during the transition (s_t, a_t, s_{t+1}), whether the state factor s_{t+1}^j locally depends on s_t^i. The diversity indicator $\mathcal{B}$ can be discrete or continuous. In this chapter, without loss of generality, we follow the procedure of Eysenbach et al. (2019) and use a discrete b sampled uniformly from $\{1, \ldots, K\}$, where K is a predefined number.

Given this skill space, SkiLD learns the skills as a skill-conditioned policy $\pi_{\text{skill}} : \mathcal{S} \times \mathcal{Z} \to \mathcal{A}$, where π_{skill} is trained to reach diverse states while ensuring that local dependencies specified by the graph $\mathcal{G}_t$ are induced. Before we describe π_{skill} training in Section 6.2.3, we first discuss how to select the skill z for π_{skill} to follow during the skill learning stage.

6.2.2 High-Level Graph-Selection Policy

For completeness and readability, this section covers the architecture and training of the high-level policy, although these components are not my individual contributions.

As described in Wang et al. (2024a), to acquire skills that are useful for downstream tasks, the skill policy π_{skill} needs to learn to induce a wide range of local dependencies *sample-efficiently*. To this end, we propose to learn a graph-selection policy $\pi_{\mathcal{G}_t} : \mathcal{S} \to \mathbf{G}$ to guide the training of π_{skill}. Specifically, training π_{skill} requires a wise selection of graphs — as graph space $\mathbf{G}$ increases super-exponentially in the number of state factors $d_{\mathcal{S}}$, many graphs are not inducible. To this end, we only select target graphs for the skill policy from a history of all seen graphs. As the agent learns to induce existing graphs in diverse ways, new graphs may be encountered, gradually expanding the set of seen graphs.

Algorithm 7 SkiLD Skill Discovery

Initialize the high-level graph-selection policy $\pi_{\mathcal{G}_t} : \mathcal{S} \to \mathbf{G}$, the low-level skill policy $\pi_{\text{skill}} : \mathcal{S} \times \mathcal{Z} \to \mathcal{A}$, the diversity indicator discriminator $q : \mathcal{S} \times \mathbf{G} \to p(\mathcal{B})$, and the dynamics model $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, graph selection interval L.
for each skill training timestep i **do**
    // data collection
    if $i \,\%\, L == 0$ **then**
        Sample the target dependency graph $\mathcal{G}_t \sim \pi_{\mathcal{G}_t}(s_t)$.
        Sample the diversity indicator from uniform distribution $b_t \sim \text{Uniform}(\mathcal{B})$.
        Compose the skill variable $z_t = (\mathcal{G}_t, b_t)$.
    end if
    Collect state transitions (s_t, z_t, a_t, s_{t+1}) with actions from $\pi_{\text{skill}}(a_t|s_t, z_t)$.
    Infer the induced dependency graph $\mathcal{G}_{t,\text{induced}}$ using the dynamics model f (Section 6.2.1).
    Update the history of the seen graphs with $\mathcal{G}_{t,\text{induced}}$.
    // training
    Sample a batch of (s_t, z_t, a_t, s_{t+1}) from the replay buffer.
    Update the dynamics model f by minimize the prediction error.
    Update the high-level policy with reward $\mathcal{R}_{\mathcal{G}_t}$ (Eq. 6.1) with the history of seen graphs.
    Update the discriminator q with the discrimination (cross-entropy) loss.
    Infer the induced dependency graph $\mathcal{G}_{t,\text{induced}}$ using the dynamics model f (Section 6.2.1).
    Infer the diversity reward $\mathcal{R}_{\text{diversity}} = \log q(b_t|s_t, \mathcal{G}_t)$.
    Update π_{skill} with $\mathcal{R}_{\text{skill}} = \mathbb{1}[\mathcal{G}_{t,\text{induced}} = \mathcal{G}_t] \cdot (1 + \lambda_b \mathcal{R}_{\text{diversity}})$ (Eq. 6.2).
end for

However, though this history guarantees graph inducibility, two challenges still remain: (1) How to efficiently explore novel local dependencies, especially hard-to-visit ones? (2) For all seen graphs, which one should π_{skill} learn next to maximize training efficiency? We address these challenges based on the following insight — compared to well-learned skills, π_{skill} should focus its training on underdeveloped skills. Meanwhile, learning new skills opens up the possibility of visiting novel local dependencies, e.g., learning to grasp the hammer makes it possible for the robot to hammer the nail.

According to this idea, we learn a graph-selection policy $\pi_{\mathcal{G}_t}$ that guides the exploration and training of the skill policy π_{skill}. Specifically, $\pi_{\mathcal{G}_t} : \mathcal{S} \to \mathbf{G}$ selects a new dependency graph the skill policy should induce for the next L time steps. To increase the likelihood of visiting hard graphs, $\pi_{\mathcal{G}_t}$ is trained to maximize the following graph novelty reward

$$\mathcal{R}_{\mathcal{G}_t} = \frac{1}{\sqrt{C(\mathcal{G}_{t,\text{induced}})}}, \tag{6.1}$$

where $C(\mathcal{G}_{t,\text{induced}})$ is the number of times that we have seen the graph in the collected transition.

6.2.3 Low-Level Skill Policy

Given the skill parameter z from the graph-selection policy, SkiLD learns skills as a skill-conditioned policy $\pi_{\text{skill}} : \mathcal{S} \times \mathcal{Z} \to \mathcal{A}$, where π_{skill} learns to reach diverse states while ensuring that the local dependencies specified by $\mathcal{G}_t$ are induced. During skill learning, we select actions by iteratively calling the skill policy π_{skill}, and we denote $\mathcal{G}_{t,\text{induced}}$ as the graph that describes the local dependencies induced in a transition (s_t, a_t, s_{t+1}) when executing a selected action a_t. We design the reward function of the skill policy as:

$$\mathcal{R}_{\text{skill}} = \mathbb{1}[\mathcal{G}_{t,\text{induced}}(s_t, a_t, s_{t+1}) = \mathcal{G}_t] \cdot (1 + \lambda_b \cdot \mathcal{R}_{\text{diversity}}), \tag{6.2}$$

where $\mathbb{1}[\mathcal{G}_{t,\text{induced}} = \mathcal{G}_t]$ measures whether the induced dependency graph matches the desired graph, $\mathcal{R}_{\text{diversity}}$ is the weighted diversity reward that further encourages visiting diverse states when the desired graph is induced, and λ_b is the coefficient of diversity reward. In the following paragraphs, we describe how we infer $\mathcal{G}_{t,\text{induced}}$ and estimate $\mathcal{R}_{\text{diversity}}$ for each transition.

Inferring Induced Graphs. To infer the induced graph for a transition (s_t, a_t, s_{t+1}), we need to determine, for each state factor $\mathcal{S}^j_{t+1}$, whether it locally depends on each factor $\mathcal{S}^i_t$ and the action $\mathcal{A}_t$. Following Section 2.2.2, we evaluate the conditional dependency $s^j_{t+1} \not\perp\!\!\!\perp s^i_t | x^{\neg i}_t$ by examining whether their pointwise conditional mutual information (pCMI) is greater than a predefined threshold $\epsilon_{\mathcal{G}_t}$. If $\text{pCMI}^{ij} = \frac{p(s^j_{t+1}|s,a)}{p(s^j_{t+1}|x^{\neg i}_t)} \geq \epsilon_{\mathcal{G}_t}$, it suggests that s^i_t is necessary to predict s^j_{t+1} and thus the local dependency exists. Meanwhile, as the transition probability p is unknown, we approximate it with a learned dynamics model f that is trained to minimize prediction error.

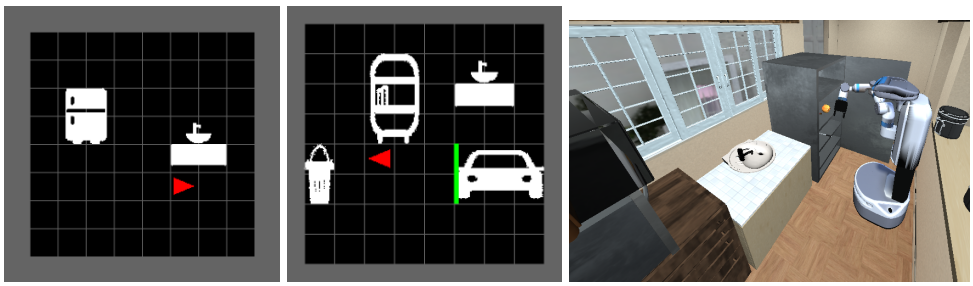
Finally, after obtaining the induced dependency graph, we evaluate $\mathbb{1}[\mathcal{G}_{t,\text{induced}} = \mathcal{G}_t]$ by examining whether each edge $\mathcal{G}^{ij}_{t,\text{induced}}$ matches the corresponding edge in the desired graph $\mathcal{G}^{ij}_t$. As $\mathcal{R}_{\text{skill}}$ only provides sparse rewards to the skill policy when the desired graph is induced, we use hindsight experience replay (Andrychowicz et al., 2017) to enrich learning signals, by relabelling induced graphs as desired graphs in some episodes.

Diversity Rewards. When the skill policy induces the desired graph, $\mathcal{R}_{\text{diversity}}$ further encourages it to visit different distinguishable states under different diversity indicators b, e.g., hammering the nail to different locations. This diversity enhances the applicability of learned skills. To this end, we design the diversity reward $\mathcal{R}_{\text{diversity}}$ as the forward mutual information between visited states and the diversity indicator $I(s_t; b_t)$, following DIAYN. To estimate the mutual information, we approximate it with a variational lower bound $I(s_t; b_t) \geq \mathbb{E} \log q(b_t|s_t)$, where $q(b_t|s_t)$ is a neural network discriminator trained to predict the diversity indicator b_t from the visited state.

In practice, rather than learning a single low-level skill to handle all graphs, SkiLD utilizes a factorized lower-level policy. When the target dependency graph is specified, SkiLD identifies which state factor should be influenced and uses its corresponding policy to sample primitive actions.

6.2.4 Downstream Task Learning

In SkiLD, after the skill learning stage, we utilize hierarchical RL to solve reward-supervised downstream tasks with the discovered skills. The skill policy, π_{skill} acts as the low-level policy while



(a) Thawing (b) Cleaning Car (c) Interactive Gibson

Figure 6.3: **Evaluation environments**: Mini-behavior: Installing Printer, Thawing and Cleaning Car, and iGibson.

a task policy, $\pi_{\text{task}} : \mathcal{S} \rightarrow \mathcal{Z}$, is learned to select which skill $z_t = (\mathcal{G}_t, b_t)$ to execute for L steps. Compared to diversity-based skills that are limited to simple behaviors, our local-dependency-based skills enable a wide range of interactions between state factors, leading to more efficient exploration and superior performance of downstream task learning.

6.3 Experiments

In this section, we provide empirical evaluations pertaining to the following questions that support the indicted answers.

- **Q1)** Do the skills learned by SkiLD induce a diverse set of interactions among state factors? Yes (Section 6.3.1).
- **Q2)** Do the skills learned by SkiLD enable more efficient downstream task learning compared to other unsupervised reinforcement learning methods? Yes (Section 6.3.2).
- **Q3)** How does each component in the skill representation affect SkiLD's task learning? (Section 6.3.3).

Environments In this chapter, we focus on addressing the challenge of vast state space brought by the number of state factors. Hence, we evaluate our method on two challenging *object-rich* embodied AI benchmarks: Mini-behavior (Jin et al., 2023) and Interactive Gibson (Li et al., 2022).

The **Mini-behavior (Mini-BH) environment** (Jin et al., 2023) (Figure 6.3a) contains a set of gridworld environments where an agent can move around and interact with a variety of objects to accomplish certain household tasks. While conceptually simple, due to highly **sequentially interdependent** state factors, this environment has been shown to be extremely challenging for the agent's exploration ability, especially under sparse reward (Jin et al., 2023). Each Mini-BH environment contains different objects and different success criteria. We tested on three particular environments in Mini-behavior, including:

- **Installing Printer**: A relatively simple environment with three state factors: the agent, a table, and a printer that can be installed.
- **Thawing**: An environment with lots of movable objects. The state factors include the agent, a sink, a fridge that can be opened, and three objects that can be picked up and thawed in the sink: fish, olive, and a date.
- **Cleaning Car**: An environment where the objects have rich and complex interactions. The state factors include the agent, a toggleable sink, a piece of rag that can be soaked in the sink, a car that the rag can clean, a soap and a bucket which can together be used to clean the rag.

The **Interactive Gibson (iGibson)** environment (Li et al., 2022) (Figure 6.3b) contains a realistic simulated Fetch Robot that operates in a kitchen environment with a refrigerator, sink, knife, and peach. The peach can be washed or cut. This environment is very difficult especially when using low-level motor commands because much of the environment is free space, meaning that only a minute fraction of action sequences will manipulate the objects meaningfully.

Both Mini-BH and iGibson require learning long-horizon policies spanning many low-level actions from sparse reward, making these challenging environments.

Baselines Before evaluating the empirical questions, we provide a brief description of the baselines. These baselines include unsupervised skill learning, and causal and hierarchical methods.

- **Diversity is all you need** (DIAYN): This method learns unsupervised state-covering skills using a mutual information objective (Eysenbach et al., 2019). SkiLD utilizes a version of this for

state-diversity skills modulated by a desired dependency graph. This baseline determines how incorporating graph information affects the algorithm.

- **Controllability-Aware Skill Discovery** (CSD): Extends DIAYN with a factorization based on controllability. This baseline is a comparable skill learning method that leverages state factorization but does not encode local dependencies (Park et al., 2023).
- **Exploration via Local Dependencies** (ELDEN, Chapter 5): This method utilizes gradient-based techniques to infer local dependencies for exploration. However, without a skill learning component, it can struggle to chain together complex behavior.
- **Chain of Interaction Skills** (COInS): This is a hierarchical algorithm that constructs a chain of skills using Granger-causality to identify local dependencies (Chuck et al., 2023). Because it is restricted to pairwise interactions, it struggles to represent the rich policies necessary for these tasks.
- **Vanilla RL**: This baseline uses PPO (Schulman et al., 2017) to directly train an agent with the extrinsic reward. Unlike other baselines, this method does not have a pertaining phase. Since all the task rewards are sparse and the tasks are often long horizon, vanilla RL often struggles.

6.3.1 Interaction Graph Diversity

We first evaluate whether SkiLD is indeed capable of achieving complex interaction graphs, comparing against two strong skill discovery baselines introduced earlier: DIAYN and CSD.

Each of these methods is trained for 10 M steps without having access to any reward. Then to evaluate their learned skills, we unroll each of them for 500 episodes with randomly sampled skills z and examine the diversity of the interaction graphs they can induce. Figure 6.4 illustrates the percentages of episodes where some hard local dependencies have been induced at least once, in Mini-BH Cleaning Car. We find that DIAYN and CSD are limited to skills that only manipulate one object individually, for example, picking up the rag (agent, rag, action $\rightarrow$ rag) or the soap (agent, soap, action $\rightarrow$ soap). By contrast, SkiLD learns to induce more complicated causal interactions, such as soaking the rag in the sink (sink, rag $\rightarrow$ rag) and cleaning the car with the soaked mug (car, rag $\rightarrow$ car). Thus, the results in Figure 6.4 answer Question **Q1** in the affirmative.

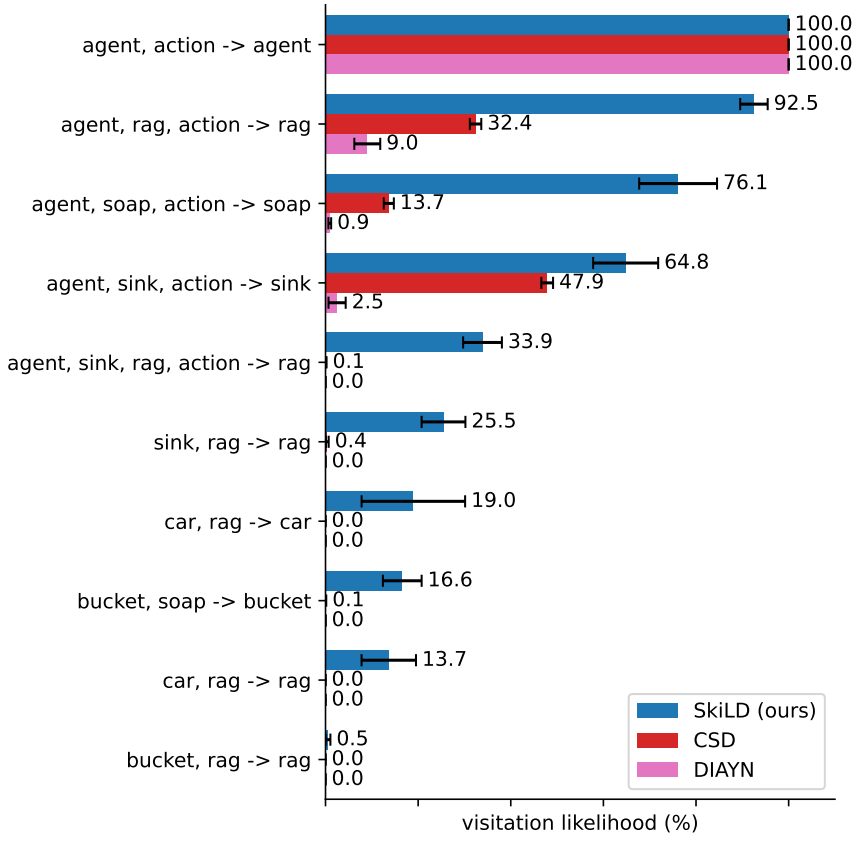


Figure 6.4: The percentage of episodes where a dependency graph is induced through random skill sampling. Standard deviation is calculated across five random seeds.

6.3.2 Sample Efficiency and Performance

Next, we evaluate whether the local dependency coverage provided by SkiLD leads to a performance boost in downstream task learning under the same number of environment interactions. We follow the evaluation setup in the unsupervised reinforcement learning benchmark (Laskin et al., 2021), where for a given environment, an agent is first pre-trained without access to task reward for K_{pt} steps, and then finetuned for K_{ft} steps. Importantly, the same pre-trained skills are reused on multiple distinct downstream tasks within the same environment, so that only the upper-level skill-selection policy is task-specific. We have $K_{\text{pt}} = 2M$, $K_{\text{ft}} = 1M$ for installing printer, $K_{\text{pt}} = 10M$, $K_{\text{ft}} = 5M$ for thawing and cleaning car, and $K_{\text{pt}} = 4M$, $K_{\text{ft}} = 2M$ for iGibson, and evaluate each method for each task across 5 random seeds. Specifically, we evaluate

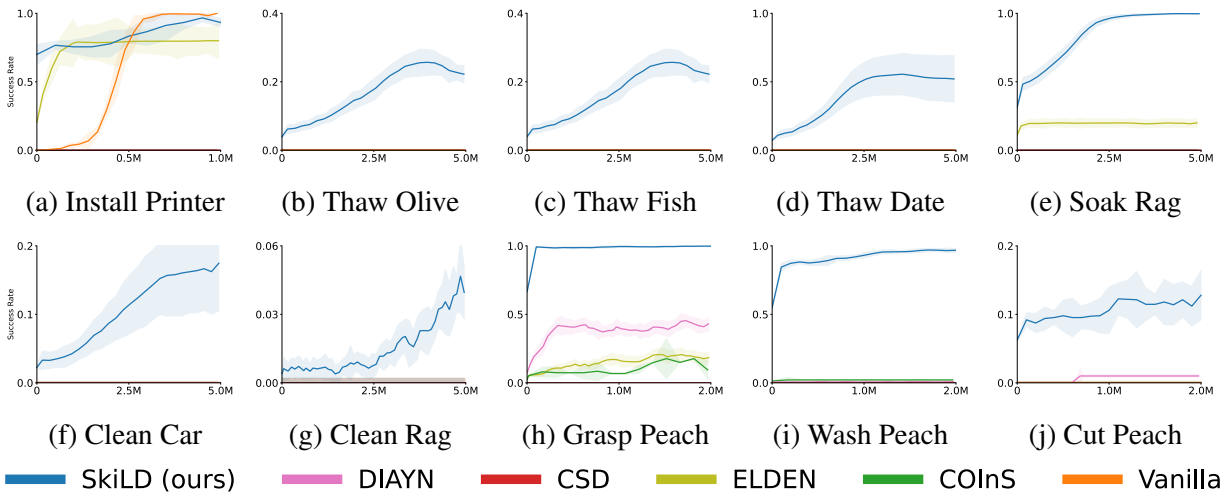


Figure 6.5: Training curves of SkiLD and baselines on multiple downstream tasks (reward supervised second phase). Each curve depicts the mean and standard deviation of the success rate over 5 random seeds. SkiLD outperforms all baselines for most tasks, converging faster and to higher returns.

on the following downstream tasks.

- **Installing Printer**: We have a single downstream task in this environment, where the agent needs to pick up the printer, put it on the table, and turn it on.
- **Thawing**: We have three downstream tasks: thawing the fish or the olive or the date.
- **Cleaning Car**: We consider three downstream tasks, where each task is a pre-requisite of the following one. The tasks are: soak the rag in the sink; clean the car with the rag; and clean the dirty rag using the soap in the bucket.
- **IGibson**: The tasks for this environment are: grasping the peach, washing the peach in the sink, and cutting the peach with a knife.

After skill learning, we train a new upper-level policy that uses z as actions and is trained with extrinsic reward, as described in Section 6.2.4. Figure 6.5 illustrates the improvement of SkiLD as compared to other methods. Without combining dependency graphs with skill learning, other methods struggle with any but the simpler tasks. COInS performs poorly because of its chain structure, which restricts the agent controlling policy from picking up objects. ELDEN's exploration

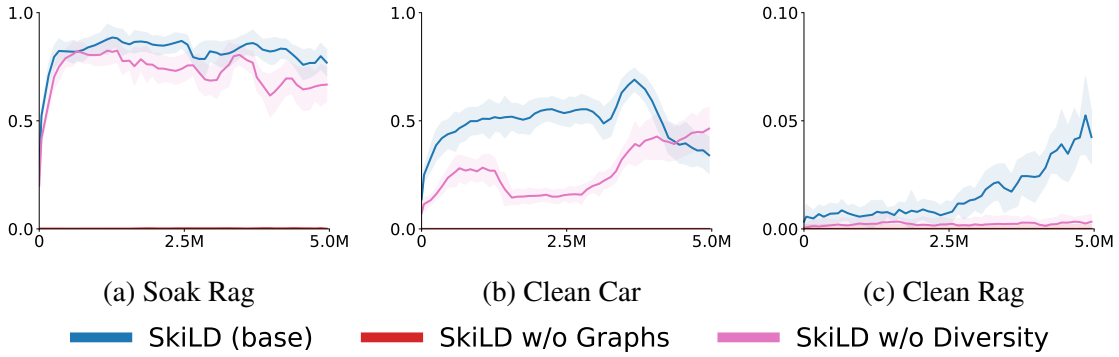
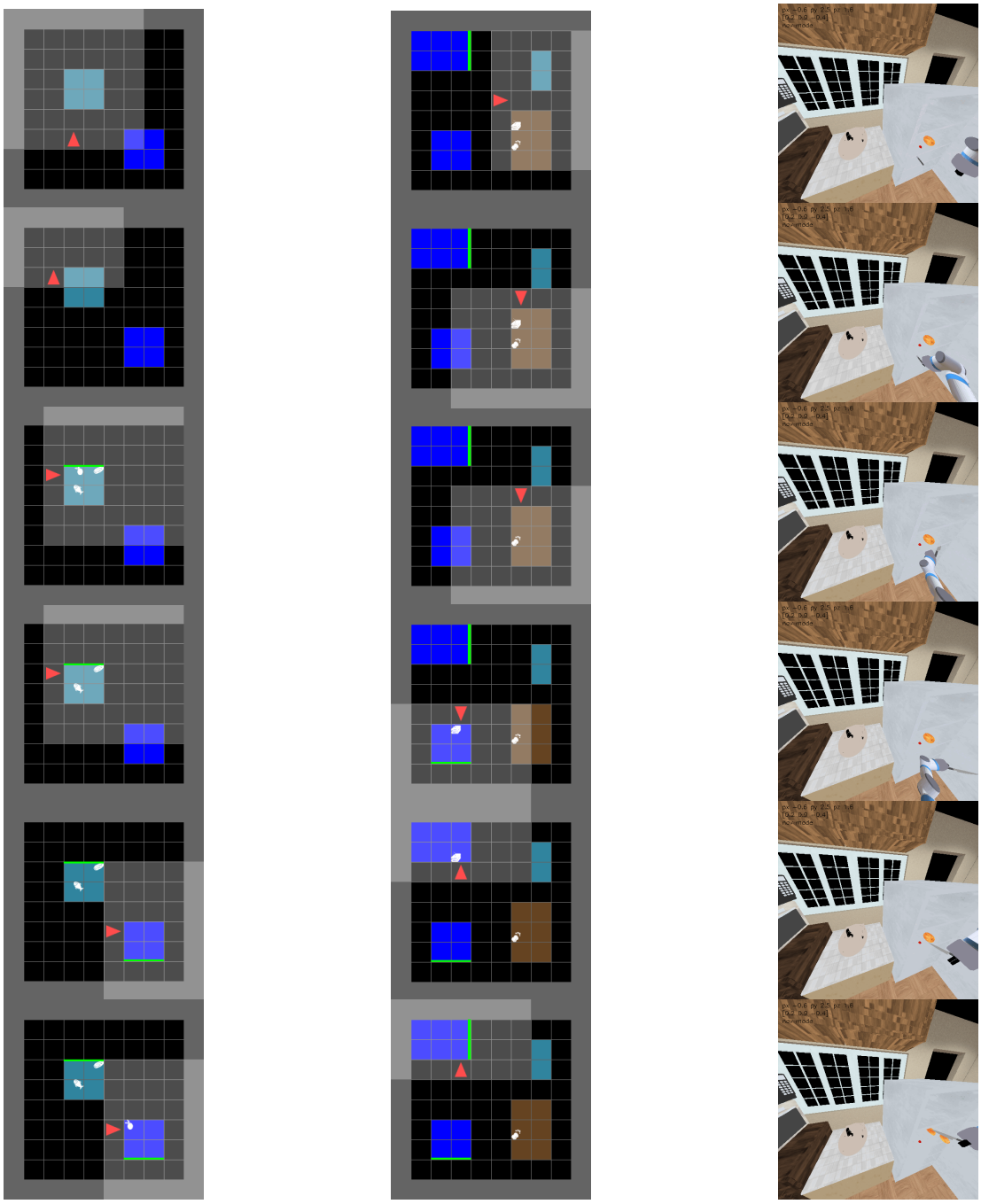


Figure 6.6: A figure illustrating the ablative performance of SkiLD without diversity or without graphs. Each curve depicts the mean and standard deviation of the success rate over 5 random seeds. Without graphs, the method collapses completely, while removing diversity results in a noticeable reduction in downstream performance.

reaches graphs, but without skills struggles to utilize that information in downstream tasks. DIAYN learns skills, but few manipulate the objects, so a downstream model struggles to utilize those skills to achieve meaningful rewards. By comparison, SkiLD achieves superior performance on 9 of the 10 downstream tasks evaluated. In the two hardest tasks which require a very long sequence of precise controls, Clean Rag and Cut Peach, SkiLD is the only method that can achieve a non-zero success rate (although still far from fully mastering the tasks), showcasing the potential of local dependencies for skill learning. Together, the results in Figure 6.5 answer Question **Q2** in the affirmative.

6.3.3 Graph and Diversity Ablations

We also explore the functionality of the graph and diversity components of the skill parameter z by assessing the downstream performance of SkiLD without these components. This produces two ablative versions of SkiLD: SkiLD without diversity and SkiLD without dependency graphs. To isolate learning from the effect of learned local dependencies, we use ground truth dependency graphs for ablative evaluations where relevant. In Figure 6.6, learning without graphs results in zero performance, consistent with DIAYN results. In addition, removing diversity produces a notable decline in performance, especially on more challenging tasks like clearning the rag. These



(a) Thaw Olive Skill (b) Clean Car Skill (c) Cut Fruit Skill

Figure 6.7: Policy rollouts for learned policies that achieve long horizon tasks **(a)** Mini-BH thaw olive, **(b)** Mini-BH clean car, **(b)** iGibson cut peach.

evaluations demonstrate that SkiLD benefits from both the incorporation of dependency graphs and diversity.

6.3.4 Skill Visualizations

In Figure 6.7 we visualize three challenging long-horizon skills learned by SkiLD: thawing the olive, cleaning the car, and cutting the peach. All of these skills require a sequence of interactions that is difficult to recover without directed behavior. Thus, comparable baselines do not learn skills of similar complexity. More skill visualizations can be found at: `https://wangzizhao.github.io/SkiLD/`.

6.4 Summary

This chapter introduces SkiLD, a method for unsupervised skill discovery. SkiLD utilizes state-specific dependency graphs, identified using learned pointwise conditional mutual information models, to guide skill discovery. The framework of defining skills according to a dependency graph and diversity goal, combined with a learned sampling scheme, learns to solve difficult downstream tasks. In domains where hand-coded primitive skills are typically given to the agent, like Mini-behavior and Interactive Gibson, SkiLD can achieve high performance without requiring explicit domain knowledge. These results arise intuitively from incorporating local dependencies as skill targets, illuminating a meaningful direction for unsupervised skill learning to be applied to a wider array of environments.

While SkiLD improves sample efficiency compared to prior unsupervised skill discovery work, it has several limitations.

- First, the assumptions that the state space is fully observable and the factorization is known may not hold in all environment, such as those where only image observations are available. To address this limitation, Chapters 7 and 8 introduce two novel approaches to extract state factors through representation learning.
- Second, SkiLD requires accurate detection of local dependencies. While off-the-shelf meth-

ods (Wang et al., 2023; Seitzer et al., 2021) work well for detecting local dependencies in our experiments, future works that can more accurately detect local dependencies will be beneficial to the performance of SkiLD.

Overall, the results in this chapter represent the completion of our efforts to learn a set skills to facilitate RL exploration and sample efficiency (Chapter 1.1 Contribution 3).

Chapter 7: Object-Centric Representation for Structured World Models (Dyn-O)

Methods introduced in Chapters 3-6 assume that the state factorization is given. However, in many environments, only low-level observations, such as images, are accessible. To address this limitation, this chapter introduces a method for extracting state factors using object-centric representations. Object-centric representations decompose observations into a set of latent factors, referred to as *slots*, where each slot encodes the properties of an individual object in the scene. Prior work learns such representations by iteratively refining a set of slots that attend to and bind different parts of an observation, and then decoding these slots back into the full scene (Locatello et al., 2020). After extracting these latent factors, a dynamics model is learned on top of them, enabling the generation of imagined rollouts for policy learning. In addition, this chapter introduces a method for decomposing each slot into a static, time-invariant component (e.g., color) and a dynamic, time-variant component (e.g., position). This decomposition allows static features to be modified while preserving the underlying dynamics, which is crucial for generating diverse novel scenarios and improving policy generalization.

This chapter is structured as follows. Section 7.1 first introduces the motivation behind object-centric world model and limitations in previous work. Section 7.2 presents the proposed algorithm Dyn-O and describes how it learns better object-centric representations than prior work by distilling from segmentation masks and how it learns accurate world models on top of object-centric representations. Section 7.3 presents the empirical evaluation of Dyn-O, covering its performance in representation and world model learning. Finally, Section 7.4 discusses limitations of Dyn-O and how the following chapter addresses them.

Contribution: In addition to my advisor, Dyn-O is joint work with Kaixin Wang, Li Zhao, and Jiang Bian. My contributions are the representation learning and world model design and its implementation. Meanwhile, Wang proposed the annealing for SAM segmentation mask and helped with some of the experiments, while Zhao and Bian provided technical advice.

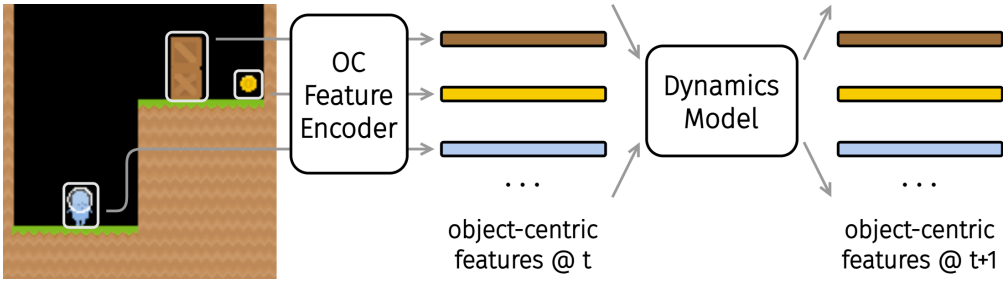


Figure 7.1: A high-level overview of the object-centric (OC) world model framework. The latent features are not monolithic or patch-based, but instead are bound to the objects present in the scene.

7.1 Motivation

While the previous chapters assume that the state factorization is given, this chapter focuses on the setting where the factorization is unknown, and investigates how to extract state factors from image observations using object-centric representations, as well as how to learn a world model on top of them. Specifically, this chapter builds on the framework of world models, which have emerged as powerful tools for simulating environment dynamics and enabling agents to predict and plan for the future (Sutton, 1990; Ha and Schmidhuber, 2018; Hafner et al., 2020; Hansen et al., 2024; Hafner et al., 2025). A common design in these models is to map high-dimensional observations (e.g., images) into latent features and then model the environment's dynamics in this latent space. However, these latent features are typically monolithic, encoding the entire scene as a whole without accounting for its internal compositional structure. Yet, interactions in most environments are inherently object-centric. This observation motivates the development of object-centric world models, which can offer improved efficiency, interpretability, and compositional generalization.

Building object-centric world models involves two key steps: 1) learning object-centric representations and 2) modeling dynamics on top of them, as illustrated in Figure 7.1. For the former, the goal is to encode features associated with each object present in the observation. For the latter, the dynamics model should account for the different representation space, in contrast to a monolithic representation, the input and the prediction target is a set of object-centric representations. While several prior approaches have explored object-centric world models, they primarily focus on simple

environments consisting of basic shapes (Ferraro et al., 2025; Nakano et al., 2023; Wu et al., 2023b; Baek et al., 2025), or rely on externally provided compositional signals such as language (Zhou et al., 2024). It remains underexplored whether object-centric world models can be learned purely from trajectories to capture complex dynamics in more challenging, complex settings.

As a step toward bridging this gap, we introduce **Dyn-O**, a novel object-centric world model with improved designs for both object-centric representation learning and dynamics modeling over prior work. To handle complex visual observations, we draw inspiration from prior work on learning object-centric representations from real-world videos (e.g., SOLV (Aydemir et al., 2023)) and adopt an autoencoder-style architecture to embed observations into object-associated features, referred to as *slots*. Specifically, we adopt a pre-trained Cosmos encoder (NVIDIA Cosmos Team, 2025), which offers two key advantages compared to the DINO encoder (Oquab et al., 2024) used in SOLV: (1) improved representation quality, and (2) access to a pretrained visual decoder, eliminating the need to train a decoder from scratch for reconstructing pixel observations. To further enhance the quality of the extracted slot features, we incorporate priors from a high-performing pretrained segmentation model, SAM2 (Ravi et al., 2025). While this incorporation significantly improves performance, it also introduces substantial computational overhead, particularly during inference. To mitigate this issue, we use a scheduling strategy that gradually reduces reliance on the segmentation mask during training, enabling the model to maintain performance at inference time without requiring the mask.

The extracted slot features are modular in nature, with each slot associated with a distinct object. To model transition dynamics in this slot space, we aim for the world model to respect this modular structure. To this end, we adopt a state-space model (SSM) based on the Mamba architecture (Gu and Dao, 2024), borrowing the idea from SlotSSM (Jiang et al., 2024). Unlike SlotSSM, which tightly integrates the slot encoder and decoder within each transformer block of the SSM, we use the pretrained slot representation module introduced above and apply the SSM solely for dynamics modeling. This decoupling of representation learning and dynamics modeling aligns with recent trends in the world model literature (Micheli et al., 2023, 2024; Cohen et al., 2024). Additionally, we explore disentangling each object's slot feature into a static component capturing time-invariant properties (e.g., texture) and a dynamic component encoding time-varying properties (e.g., position). This disentanglement enables fine-grained manipulation of slot features.

For instance, we can modify the static feature of one object while keeping its dynamics unchanged, allowing for diverse data generation during world model rollouts.

We evaluate Dyn-O in seven Procgen (Cobbe et al., 2020b) environments. Our experiments indicate that Dyn-O learns high-quality object-centric representations, generalizable world models, and disentangled static and dynamic representations. We summarize our contributions as follows.

- We propose a novel object-centric representation learning method by leveraging segmentation masks with a dropout schedule, enhancing representation quality while keeping efficient inference.
- We propose a novel object-centric world model that uses state-space models as backbones and outperforms monolithic models in both rollout quality and generalization.
- We propose a novel object-centric representation that disentangles each object's representation into static and dynamic features, allowing the generation of diverse rollouts by altering static attributes while preserving dynamic behavior.

7.2 Dyn-O

From a high level perspective, Dyn-O models the environment's dynamics in an object-centric manner, enabled by two learning phases (Figure 7.2).

- **Object-centric representation learning** (Section 7.2.1): Dyn-O learns an object-centric representation in which coherent objects are each encoded into independent features, referred to as *slots*. This factorized representation, in contrast to a monolithic scene-level encoding, leverages the natural compositionality of objects in the world.
- **Dynamics learning** (Section 7.2.2): Dyn-O adopts a State Space Model (SSM) to predict transitions from the current slots to the next-step slots, conditioned on the action. Each object's slot feature is further decoupled into static and dynamic components to enable fine-grained manipulation.

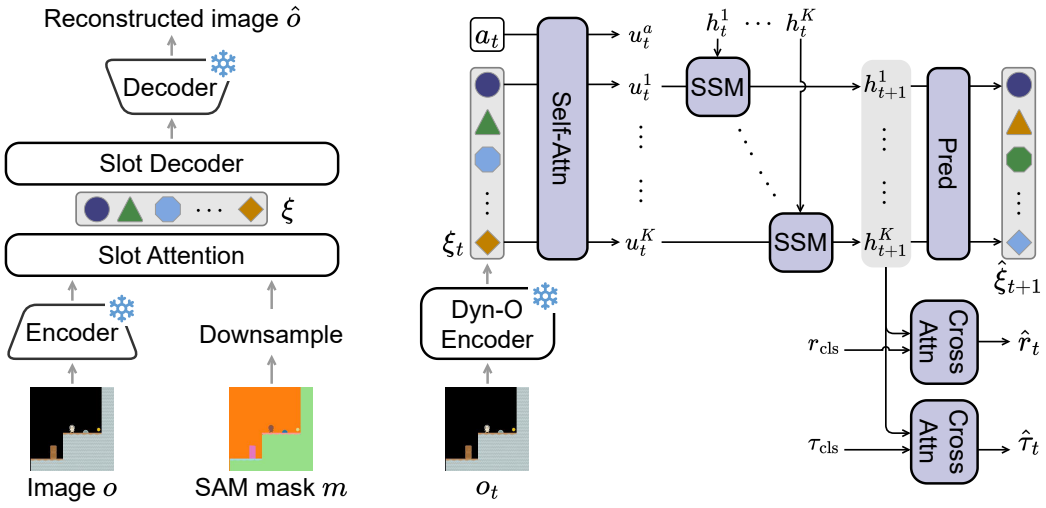


Figure 7.2: Components of Dyn-O: **(a)** object-centric representation learning; **(b)** dynamics learning. Modules marked with ❄ are fixed, while others are learnable. The "Dyn-O Encoder" in (b) corresponds to the lower half of (a), which maps the image o to the latent slot feature ξ. See Section 7.2 for details.

7.2.1 Extracting Object-Centric Representations

In the first learning phase, Dyn-O learns an object-centric representation from image observations. In contrast to a monolithic representation that mixes all objects' information, this factored representation enables Dyn-O to modify each object independently and generate novel object combinations. Formally, as shown in Figure 7.2(a), Dyn-O learns an encoder $\texttt{Enc} : \mathcal{O} \to \xi$ to extract object-centric representations ξ from observations o, where the representations consist of K slots, $\xi = [\xi^1, \ldots, \xi^K]$, each corresponding to an object. Here, K is a predefined hyperparameter, and when the number of objects in a scene is smaller than K, some slots may remain unused.

During encoder learning, inspired by SOLV (Aydemir et al., 2023), instead of training from scratch with raw pixels, Dyn-O learns on top of the high-quality features extracted by the Cosmos tokenizer (NVIDIA Cosmos Team, 2025). Given an input frame $o \in \mathbb{R}^{H \times W \times 3}$, Dyn-O first applies the Cosmos encoder (`CosmosEnc`) to extract patch-level features $e \in \mathbb{R}^{n_e \times d_e}$, where n_e denotes the number of patches in the frame and d_e denotes the feature dimension. It then initializes K slots $\xi_{\text{init}} \in \mathbb{R}^{K \times d_\xi}$ from a set of learnable vectors. These slots compete to bind to objects using

iterative Slot Attention (Locatello et al., 2020), producing $\xi = \texttt{Slot-Attn}(\xi_{\text{init}}, e)$. The learning signals for slot extraction arise from reconstructions – a decoder (`Slot-Dec`) reconstructs features $\hat{e}$ from the slots, which are then used to reconstruct the observation via the Cosmos decoder as $\hat{o} = \texttt{CosmosDec}(\hat{e})$. That is,

$$
\begin{aligned}
e &= \texttt{CosmosEnc}(o), \\
\xi &= \texttt{Slot-Attn}(\xi_{\text{init}}, e), \\
\hat{e} &= \texttt{Slot-Dec}(\xi), \\
\hat{o} &= \texttt{CosmosDec}(\hat{e}).
\end{aligned}
$$

The `Slot-Attn` and `Slot-Dec` modules are trained jointly by minimizing the reconstruction error as follows, while the Cosmos encoder-decoder remains frozen:

$$
\mathcal{L}_{\text{slot}} = \|e - \hat{e}\|^2 + \|o - \hat{o}\|^2. \tag{7.1}
$$

Empirically, we observe that this fully unsupervised training objective usually results in inaccurate object-slot bindings (see examples in Section 7.3.1). To improve the quality of object-centric representations, we leverage the prior provided by foundational segmentation models. Specifically, we use a pre-trained SAM2 model (Ravi et al., 2025) to generate a segmentation mask $m \in \{0, 1\}^{H \times W \times K}$ and use it as an attention mask during slot attention, $\xi = \texttt{Slot-Attn}(\xi_{init}, e, m)$, binding each slot to one segmented object and constraining it to only attend to patches from that object.

While the segmentation mask enhances object-slot binding, it also introduces substantial computational overhead, especially during inference. This dependency could limit the practicality of Dyn-O. For example, when extracting slots for an agent interacting with the environment in an online setting, we would need to run SAM2 inference at every environment step, which would be prohibitively expensive compared to the typical cost per step. To address this issue, during encoder learning, we introduce an annealing schedule that gradually reduces the reliance on the segmentation mask, aiming to eliminate its use by the end of training. Initially, the segmentation mask is always used to guide slot extraction. As training progresses, the probability of not using

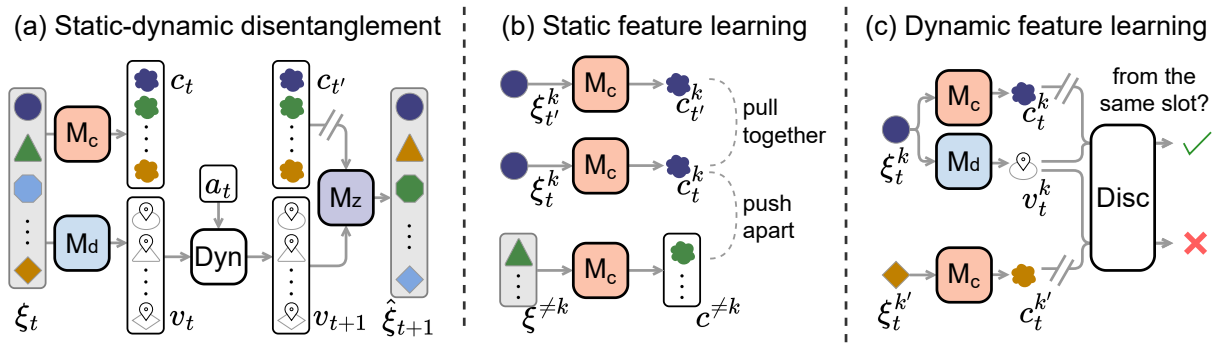


Figure 7.3: Illustration of **(a)** the overall design for disentangling slot features in dynamics modeling, and the training procedures for **(b)** static features and **(c)** dynamic features. // indicates a stop-gradient operation.

the mask increases according to a logarithmic schedule w.r.t. the number of network updates, as $\log(1+\#\text{ updates})/\log(1+\#\text{ total updates})$. This dropout schedule allows the encoder to achieve the best of both worlds – it learns high-quality representations with the initial guidance of segmentation masks, while enabling efficient inference by phasing out the need for them.

7.2.2 World Model with Object-Centric Representations

After learning object-centric representations as described above, the next phase of Dyn-O focuses on training a world model to reason about object interactions. Given the history of slots $\xi_{\leq t}$ and actions $a_{\leq t}$, the world model predicts the next slots, the reward, and whether the episode terminates as $\hat{\xi}_{t+1}, \hat{r}_t, \hat{\tau}_t = \texttt{Dyn}(\xi_{\leq t}, a_{\leq t})$.

World Model For the world model `Dyn`, we adopt state-space models (SSMs) for their strength at capturing long-range temporal dependencies. Meanwhile, the model needs to account for the *permutation equivariance* among slots – if the input slots are randomly permuted, the slots' prediction should follow the same permutation. Therefore, as shown in Figure 7.2(b), we design `Dyn` as follows. We first apply self-attention to extract information about object interactions:

$$u_t^k = \texttt{Self-Attn}(\texttt{query} = \xi_t^k, \texttt{key} = [\xi_t^1, \ldots, \xi_t^K, a_t]),$$

Algorithm 8 Dyn-O World Model Learning

1: Collect a dataset of (o_t, a_t, r_t, o_{t+1}). Initialize object-centric encoder (`Enc`), mappings to static features ($\texttt{M}_c$), dynamic features ($\texttt{M}_v$), and slots ($\texttt{M}_\xi$), the world model (`Dyn`), and the discriminator (`Disc`).
2: Train object-centric encoder `Enc` with Eq. 7.1.
3: Train the world model with joint optimization:
4: Update the world model `Dyn` by minimizing prediction losses in Eq. 7.2.
5: **if** learn static-dynamic disentanglement **then**
6: Update static features from $\texttt{M}_c$ by enforcing time-invariance in Eq. 7.3.
7: Update `Disc` with Eq. 7.4 to estimate mutual information.
8: Update dynamic features from $\texttt{M}_v$ via reconstruction $\texttt{M}_\xi$ and disentanglement loss in Eq. 7.5.

where no position encoding is used to maintain permutation invariance. Next, each slot is processed by a shared SSM to update its hidden state, which is then used by Dyn-O to predict the next slots, reward, and whether the episode terminates using separate prediction networks as follow:

$$h_{t+1}^k = \texttt{SSM}(h_t^k, \xi_k^t), \quad \hat{\xi}_{t+1}^k = \texttt{Pred}(h_{t+1}^k),$$
$$\hat{r}_t = \texttt{Cross-Attn}(\texttt{query} = r_{\text{cls}}, \texttt{key} = [h_{t+1}^1, \ldots, h_{t+1}^K]),$$
$$\hat{\tau}_t = \texttt{Cross-Attn}(\texttt{query} = \tau_{\text{cls}}, \texttt{key} = [h_{t+1}^1, \ldots, h_{t+1}^K]),$$

where h^k is the hidden state of the `SSM` that tracks past information for the k-th slot, and r_{cls} and τ_{cls} learnable query tokens used to extract reward and termination signals from the hidden states. Finally, the world model is optimized by minimizing the following prediction loss (Algorithm 8 line 4).

$$\mathcal{L}_{\text{wm}} = \sum_{t=1}^{T-1} \|\hat{\xi}_{t+1} - \xi_{t+1}\|^2 + \sum_{t=1}^{T} \left(\|\hat{r}_t - r_t\|^2 + \text{CE}(\hat{\tau}_t, \tau_t) \right), \tag{7.2}$$

where CE stands for cross-entropy loss for binary classification.

Static-Dynamic Disentanglement The object-centric representation learned in Section 7.2.1 enables the creation of scenes with novel object combinations. However, when synthesizing data, it is often desirable to only modify object appearance (e.g., colors) while preserving their dynamic

properties (e.g., positions). For example, modifying dynamic information could result in invalid scenes, such as a table being moved while the objects originally on it remain suspended in midair.

To this end, as shown in Figure 7.3 (a), Dyn-O disentangles each slot ξ^k into two components: static features $c^k \in \mathbb{R}^{d_c}$ that captures time-invariant properties and dynamic features $v^k \in \mathbb{R}^{d_v}$ that encodes time-variant properties. This decomposition allows us to modify an object's static features while keeping its dynamic features unchanged, generating novel scenarios with consistent dynamics. Formally, Dyn-O maps slots to static and dynamic features with two separate neural networks: $c^k = \mathtt{M}_c(\xi^k)$ and $v^k = \mathtt{M}_v(\xi^k)$. We describe how these features are learned below.

Static features are intended to capture time-invariant properties. Thus, for an object present at timestep t, its static feature should remain the same across all timesteps. Therefore, a natural training objective is to minimize the difference between static features across all timesep pairs. However, this objective alone admits a degenerate solution where all static features collapse to a constant. To prevent this collapse, Dyn-O incorporates contrastive learning to encourage feature diversity, as illustrated in Figure 7.3(b): static features should be similar if they belong to the same slot, and distinct otherwise. Dyn-O therefore learns static features by optimizing the following loss (Algorithm 8 line 5).

$$\mathcal{L}_{\text{stat}} = \underbrace{\sum_{k,t\neq t'} \|c_t^k - c_{t'}^k\|^2}_{\text{time invariance}} + \underbrace{\sum_{k,t} -\log \frac{\cos(c_t^k, c_{t'}^k)}{\cos(c_t^k, c_{t'}^k) + \sum_{\tilde{c}\in\tilde{C}} \cos(c_t^k, \tilde{c})}}_{\text{contrastive learning}}, \tag{7.3}$$

where $\cos$ denotes cosine similarity, and the second term corresponds to the InfoNCE loss (van den Oord et al., 2018). For each c_t^k, the positive sample $c_{t'}^k$ comes from the same slot at a different timestep, while the negative samples $\tilde{C}$ are drawn from other slots in the batch.

On the other hand, Dyn-O learns dynamic features by reconstructing the slot content while ensuring their disentanglement from static features. Specifically, Dyn-O uses a reconstruction network $\mathtt{M}_\xi$ to output $\hat{\xi}^k = \mathtt{M}_\xi(\mathtt{sg}(c^k), v^k)$, where $\mathtt{sg}$ denotes stop-gradient, preventing the static feature c^k from being updated by the reconstruction loss. (For simplicity, in this paragraph, we omit the subscript t in ξ, c and v). However, minimizing reconstruction loss alone may lead the dynamic features to encode time-invariant information as well, bypassing the need for c^k. To avoid this

effect, Dyn-O promotes disentanglement by minimizing the mutual information $\sum_{t,k} I(c^k, v^k)$ via adversarial training, as illustrated in Figure 7.3 (c). A discriminator `Disc` is trained to distinguish whether a pair of static and dynamic features comes from the same slot. While `Disc` minimizes the discrimination loss, the dynamic feature extractor $\mathtt{M}_v$ is trained to maximize it, thereby reducing the static information encoded in dynamic features (Ganin and Lempitsky, 2015). For stability, we adopt the Wasserstein distance as the discrimination loss and apply LeCam regularization (Tseng et al., 2021) (Algorithm 8, lines 6-7):

$$\mathcal{L}_{\text{dyn}} = \sum_k \underbrace{\|\hat{\xi}^k - \xi^k\|^2}_{\text{reconstruction}} + \underbrace{\mathtt{Disc}(c^k, v^k)}_{\text{disentanglement}}, \tag{7.4}$$

$$\mathcal{L}_{\text{disc}} = \sum_k \underbrace{\Big(-\mathtt{Disc}(c^k, v^k) + \mathtt{Disc}(c^k, v^{k'})\Big)}_{\text{discrimination}} + \underbrace{\text{LeCam}(\mathtt{Disc})}_{\text{regularization}}, \tag{7.5}$$

where (c^k, v^k) are from the same slot, and $(c^k, v^{k'})$ are from different slots.

7.3 Experiments

In this section, we provide empirical evaluations pertaining to the following questions that support the indicted answers.

- **Q1)** Can Dyn-O learn accurate object-centric representations? Yes (Section 7.3.1).
- **Q2)** Can Dyn-O learn accurate world models? Yes (Section 7.3.2).
- **Q3)** Can representations learned by Dyn-O facilitate RL learning? Yes (Section 7.3.3).
- **Q4)** Are static and dynamic features learned by Dyn-O truly disentangled? Yes (Section 7.3.4).

Implementations Our experiments are conducted in Procgen (Cobbe et al., 2020b), a set of procedurally-generated 2D video game environments. We use 7 Procgen environments: bigfish, coinrun, caveflyer, dodgeball, jumper, ninja, and starpilot. In each Procgen environment, a PPG policy (Cobbe et al., 2021) is trained and used to collect an offline dataset of 1M transitions from the first 200 levels for the learning of all methods.

Table 7.1: slot-object binding accuracy, measured by FR-ARI (↑).

Environments	Oracle	Dyn-O (ours)	SOLV
bigfish	0.96	0.80	0.54
coinrun	0.33	0.27	0.10
dogeball	0.79	0.48	0.17
starpilot	0.86	0.47	0.49
average	0.74	0.51	0.33

7.3.1 Evaluating Object-Centric Representation

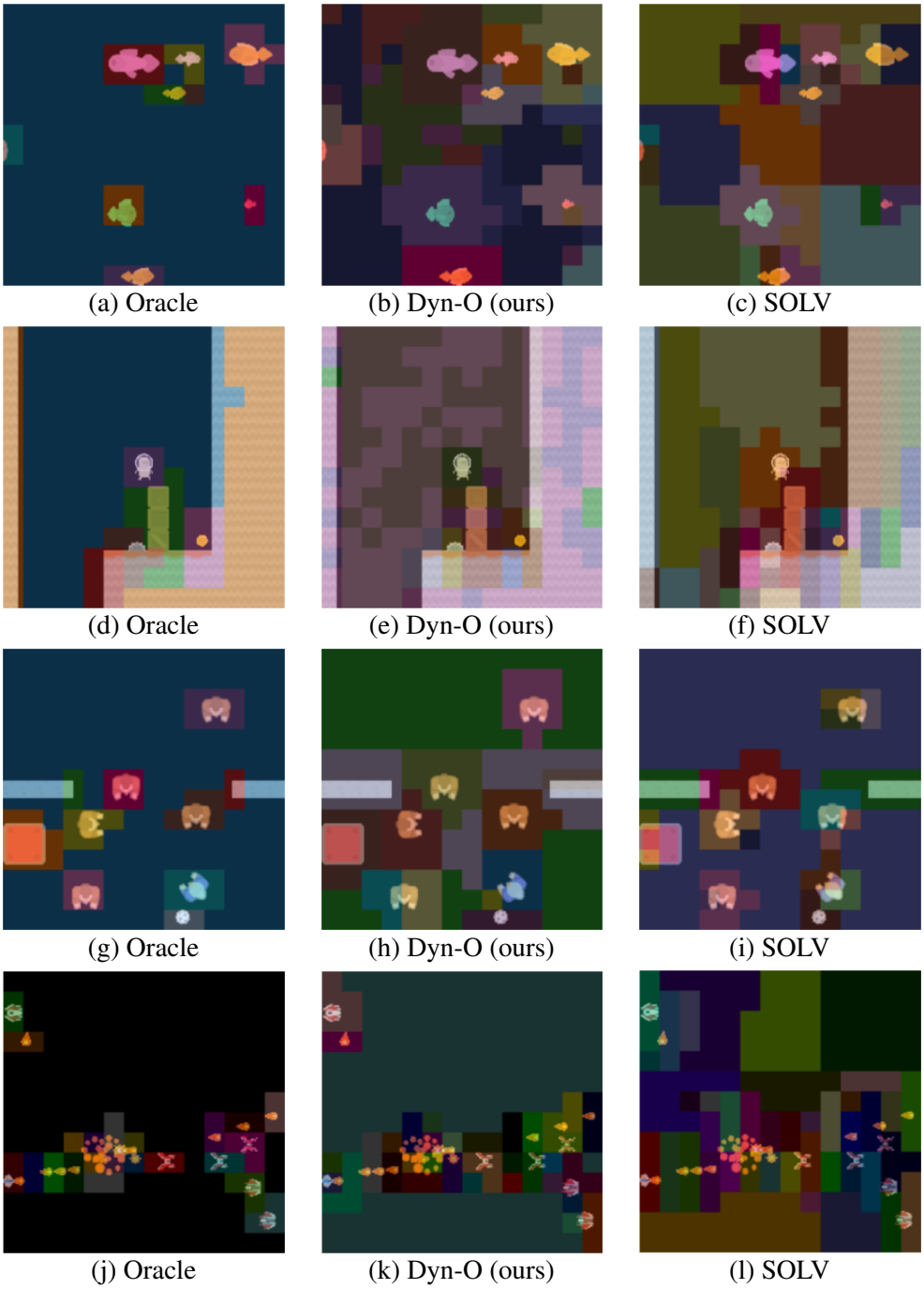
As Dyn-O's world model is learned on top of the object-centric representations, the quality of learned representation is critical to its prediction accuracy. We compare Dyn-O's representation learning against the following baselines.

- **SOLV** (Aydemir et al., 2023): the same as our method but does not use a segmentation mask for training or inference.
- **Oracle**: the same as our method but always uses the segmentation mask for both training and inference.

As shown in Figure 7.4, compared to SOLV that often splits an object into multiple slots, Dyn-O achieves more accurate object-slot binding, assigning each object to a single slot. We also evaluate the foreground adjusted rand index (FG-ARI) which is a widely used metric in the object-centric literature that measures the similarity of the discovered objects masks to ground-truth masks. Again, as shown in Table 7.1, Dyn-O outperforms SOLV, suggesting the benefit of using segmentation masks to guide representation learning. Thus, the results in Figure 7.4 and Table 7.1 answer Question **Q1** in the affirmative.

7.3.2 Evaluating World Model Accuracy

The promise of Dyn-O is to learn accurate and and generalizable world models based on object-centric representations. Therefore, the most critical evaluation of this chapter focuses on the world model quality, and we compare Dyn-O against the following baselines and ablations.



(a) Oracle (b) Dyn-O (ours) (c) SOLV

(d) Oracle (e) Dyn-O (ours) (f) SOLV

(g) Oracle (h) Dyn-O (ours) (i) SOLV

(j) Oracle (k) Dyn-O (ours) (l) SOLV

Figure 7.4: Qualitative evaluation of the object-centric representation learning in **bigfish**, **coinrun**, **dodgeball**, and **starpilot**.

- **DreamerV3** (Hafner et al., 2025): one of the state-of-the-art model-based RL methods.
- **Dreamweaver** (Baek et al., 2025): an object-centric world model designed to discover hierarchical and compositional representations.
- **Dyn-O without object-centric representations** (denoted as **Dyn-O w/o OC**): our method but without object-centric representations. The dynamics model is learned on top of 14x14 patch-level features extracted by Cosmos encoder, where each patch is treated as a "slot".

For a fair comparison, we use the same frozen Cosmos tokenizer for DreamerV3 and Dreamweaver as Dyn-O during world model learning.

To evaluate the generalizability of each method, we use the learned world model to generate 20-step rollouts in 500 unseen levels. The evaluation metric covers pixel-level quality (PSNR) as well as temporal and spatial coherence (FVD, LPIPS, and SSIM). The results are shown in Table 7.2. Dyn-O significantly outperforms dreamer which uses a monolithic representation, demonstrating the advantage of predicting in the object-centric representation space. Meanwhile, in contrast to Dyn-O w/o OC whose latent representation consist of 196 patch-level "slots", Dyn-O only uses 31 or 47 object-level slots (where each object-level slot has the same dimension as a patch-level "slot") and achieves higher performance, demonstrating the superior efficiency of object-centric representations. To complement the analysis with qualitative examples, Figure 7.5 and Figure 7.6 show the generated rollouts of Dyn-O and its comparison with baselines. In addition to Procgen, we further compare Dyn-O against DreamerV3 on the CLEVR dataset (Johnson et al., 2017) and two ALE environments (Bellemare et al., 2013). As shown in Table 7.3, Dyn-O again outperforms the DreamerV3 baseline in terms of prediction accuracy.

Together, the results in Table 7.2, Table 7.3, Figure 7.5, and Figure 7.6 answer Question **Q2** in the affirmative.

7.3.3 Evaluating Representation Effectiveness for Policy Learning

Next, we evaluate whether the learned object-centric representations can facilitate policy learning. To do so, we fix the representation backbone and train only the policy head using the PPG

Table 7.2: Rollout accuracy for each Procgen environment at 20-th timestamp, measured as mean and standard error.

Environment	Metric	DreamerV3	Dreamerwaver	Dyn-O w/o OC	Dyn-O (ours)
bigfish	LPIPS (↓)	0.39 ± 0.01	0.52 ± 0.01	0.36 ± 0.01	**0.25** ± 0.01
	FVD (↓)	567.20 ± 11.18	831.17 ± 15.55	248.94 ± 16.32	**126.88** ± 6.42
	SSIM (↑)	0.73 ± 0.01	0.68 ± 0.00	0.68 ± 0.01	**0.76** ± 0.01
	PSNR (↑)	19.34 ± 0.14	19.09 ± 0.12	**20.16** ± 0.27	**20.28** ± 0.30
caveflyer	LPIPS (↓)	**0.53** ± 0.01	–	0.59 ± 0.01	0.61 ± 0.01
	FVD (↓)	1104.50 ± 18.87	–	966.07 ± 28.71	**877.74** ± 20.99
	SSIM (↑)	**0.44** ± 0.01	–	0.26 ± 0.01	0.26 ± 0.01
	PSNR (↑)	**11.99** ± 0.12	–	10.93 ± 0.19	10.82 ± 0.13
coinrun	LPIPS (↓)	0.34 ± 0.01	–	0.36 ± 0.01	**0.28** ± 0.01
	FVD (↓)	530.31 ± 10.51	–	583.11 ± 16.07	**266.13** ± 6.16
	SSIM (↑)	0.60 ± 0.01	–	0.47 ± 0.01	**0.62** ± 0.01
	PSNR (↑)	**14.22** ± 0.25	–	12.61 ± 0.14	13.56 ± 0.19
dodgeball	LPIPS (↓)	0.42 ± 0.01	–	0.15 ± 0.01	**0.14** ± 0.01
	FVD (↓)	903.51 ± 20.44	–	481.55 ± 20.83	**372.50** ± 15.07
	SSIM (↑)	0.50 ± 0.01	–	0.72 ± 0.01	**0.76** ± 0.01
	PSNR (↑)	16.10 ± 0.05	–	20.33 ± 0.12	**20.88** ± 0.15
ninja	LPIPS (↓)	0.48 ± 0.01	–	0.49 ± 0.01	**0.45** ± 0.01
	FVD (↓)	423.27 ± 14.26	–	521.32 ± 15.85	**313.57** ± 9.73
	SSIM (↑)	0.45 ± 0.01	–	0.33 ± 0.01	**0.54** ± 0.01
	PSNR (↑)	12.58 ± 0.17	–	**12.80** ± 0.10	11.95 ± 0.12
starpilot	LPIPS (↓)	0.37 ± 0.01	0.50 ± 0.01	0.50 ± 0.01	**0.22** ± 0.01
	FVD (↓)	626.47 ± 14.95	735.24 ± 21.48	429.36 ± 15.46	**211.27** ± 12.70
	SSIM (↑)	0.69 ± 0.01	0.68 ± 0.00	0.72 ± 0.01	**0.76** ± 0.01
	PSNR (↑)	19.99 ± 0.08	19.48 ± 0.13	19.76 ± 0.23	**20.55** ± 0.13

algorithm on three Procgen games: Bigfish, Starpilot, and Coinrun, with three random seeds for each game. We use the easy difficulty setting with an unlimited number of levels.

We compare the performance of policies using object-centric representations learned by Dyn-O against policies using representations directly output by the Cosmos encoder, and the policy performance (measured as reward) are shown in Table 7.4. Policies using object-centric representations achieve much higher reward than policies with raw patch-level features, demonstrating the effectiveness of Dyn-O's representations for downstream task learning and thus answering Question

Table 7.3: Rollout accuracy for CLEVR and ALE environments at 20-th timestamp, measured as mean and standard error.

Environment	Metric	DreamerV3	Dyn-O (ours)
CLEVR	LPIPS (↓)	0.34 ± 0.00	**0.31** ± 0.00
	FVD (↓)	1676.18 ± 40.92	**446.57** ± 8.55
	SSIM (↑)	0.86 ± 0.00	**0.88** ± 0.00
	PSNR (↑)	21.37 ± 0.09	**22.63** ± 0.06
Atari Skiing	LPIPS (↓)	0.12 ± 0.00	**0.09** ± 0.01
	FVD (↓)	217.84 ± 3.76	**187.76** ± 8.04
	SSIM (↑)	0.91 ± 0.00	**0.93** ± 0.00
	PSNR (↑)	25.46 ± 0.13	**26.39** ± 0.16
Atari Boxing	LPIPS (↓)	0.04 ± 0.00	**0.02** ± 0.00
	FVD (↓)	237.90 ± 12.15	**218.68** ± 15.68
	SSIM (↑)	0.93 ± 0.00	**0.95** ± 0.00
	PSNR (↑)	**30.00** ± 0.14	29.46 ± 0.09

Q3 in the affirmative..

Table 7.4: Reward of policies using different representations, measured by the mean and standard deviation across three random seeds.

Representation	Bigfish	Starpilot	Coinrun
Cosmos encoder	0.90 ± 0.02	8.80 ± 0.47	8.00 ± 0.18
Dyn-O (ours)	**7.67** ± 4.02	**19.19** ± 0.88	**8.95** ± 0.38

7.3.4 Evaluating Static-Dynamic Disentanglement

To examine whether the static and dynamic features learned by Dyn-O are disentangled, We probe the model with environment-privileged information, evaluating whether static features only capture time-invariant properties and whether dynamic features only capture time-varying properties. Specifically, we use SAM to segment out objects in 10K transitions and extract the following properties for each object as probing targets: area (i.e., the number of pixels), xy positions, and average RGB values across object pixels. Then we compute each object's **slot**, **static** feature, and **dynamic** feature as probing inputs. During probing, we discretize each property into 10 bins

Table 7.5: Probing accuracy (↑), in percentage (%), on environment privilege properties, shown the mean and standard deviation. Static features have much higher prediction accuracy than dynamic features for static properties (i.e., RGB values), while dynamic features have higher accuracy on dynamic properties (i.e., position and area), demonstrating that Dyn-O achieves effective static-dynamic disentanglement

Method	R value	G value	B value	x position	y position	area
			coinrun			
	static	static	static	dynamic	dynamic	dynamic
slots	**83.5** $\pm$ 0.0	**81.7** $\pm$ 0.0	**89.3** $\pm$ 0.0	**91.8** $\pm$ 0.0	**94.5** $\pm$ 0.0	**97.1** $\pm$ 0.0
dynamic features	47.7 $\pm$ 7.3	45.7 $\pm$ 7.3	47.0 $\pm$ 8.9	78.1 $\pm$ 5.1	75.2 $\pm$ 6.6	83.3 $\pm$ 3.1
static features	67.3 $\pm$ 1.4	70.9 $\pm$ 1.9	81.7 $\pm$ 2.2	34.9 $\pm$ 1.2	37.7 $\pm$ 2.0	75.3 $\pm$ 0.4
random features	25.8 $\pm$ 0.0	27.3 $\pm$ 0.0	23.2 $\pm$ 0.0	29.3 $\pm$ 0.0	28.3 $\pm$ 0.0	73.3 $\pm$ 0.0
			bigfish			
	static	static	static	dynamic	static	static
slots	74.7 $\pm$ 0.0	77.9 $\pm$ 0.0	83.8 $\pm$ 0.0	97.7 $\pm$ 0.0	98.3 $\pm$ 0.0	100.0 $\pm$ 0.0
dynamic features	49.3 $\pm$ 3.0	48.4 $\pm$ 2.4	47.4 $\pm$ 3.7	94.0 $\pm$ 1.9	45.2 $\pm$ 5.1	95.9 $\pm$ 1.2
static features	65.8 $\pm$ 4.2	67.6 $\pm$ 4.8	77.5 $\pm$ 1.2	29.3 $\pm$ 0.4	88.8 $\pm$ 0.7	100.0 $\pm$ 0.0
random features	37.6 $\pm$ 0.0	31.8 $\pm$ 0.0	37.3 $\pm$ 0.0	19.2 $\pm$ 0.0	20.8 $\pm$ 0.0	88.6 $\pm$ 0.0
			dodgeball			
	static	static	static	mixed	mixed	static
slots	90.2 $\pm$ 0.0	91.7 $\pm$ 0.0	91.2 $\pm$ 0.0	98.7 $\pm$ 0.0	98.7 $\pm$ 0.0	99.8 $\pm$ 0.0
dynamic features	66.0 $\pm$ 1.6	62.4 $\pm$ 1.2	61.1 $\pm$ 1.8	55.4 $\pm$ 1.8	57.7 $\pm$ 2.5	95.3 $\pm$ 1.7
static features	73.1 $\pm$ 1.2	74.2 $\pm$ 1.4	75.5 $\pm$ 1.7	74.9 $\pm$ 2.4	70.4 $\pm$ 2.9	99.3 $\pm$ 0.0
random features	53.3 $\pm$ 0.0	35.5 $\pm$ 0.0	42.9 $\pm$ 0.0	20.1 $\pm$ 0.0	19.7 $\pm$ 0.0	90.1 $\pm$ 0.0
			ninja			
	static	static	static	dynamic	dynamic	dynamic
slots	87.1 $\pm$ 0.0	80.7 $\pm$ 0.0	86.1 $\pm$ 0.0	95.0 $\pm$ 0.0	96.6 $\pm$ 0.0	97.3 $\pm$ 0.0
dynamic features	60.7 $\pm$ 9.7	51.3 $\pm$ 8.2	60.6 $\pm$ 9.0	86.3 $\pm$ 3.0	88.1 $\pm$ 0.3	87.6 $\pm$ 2.2
static features	77.6 $\pm$ 2.9	62.2 $\pm$ 0.4	79.6 $\pm$ 0.4	35.9 $\pm$ 0.9	37.8 $\pm$ 1.1	82.2 $\pm$ 1.3
random features	32.9 $\pm$ 0.0	25.4 $\pm$ 0.0	31.3 $\pm$ 0.0	25.0 $\pm$ 0.0	19.5 $\pm$ 0.0	80.5 $\pm$ 0.0
			starpilot			
	static	static	static	dynamic	static	static
slots	71.5 $\pm$ 0.0	79.1 $\pm$ 0.0	71.0 $\pm$ 0.0	97.2 $\pm$ 0.0	97.5 $\pm$ 0.0	99.5 $\pm$ 0.0
dynamic features	55.6 $\pm$ 7.2	66.5 $\pm$ 6.4	55.3 $\pm$ 6.2	94.8 $\pm$ 0.9	77.0 $\pm$ 14.2	97.9 $\pm$ 1.7
static features	55.9 $\pm$ 1.1	70.7 $\pm$ 1.1	50.5 $\pm$ 1.8	26.8 $\pm$ 0.7	76.6 $\pm$ 3.2	99.2 $\pm$ 0.0
random features	35.9 $\pm$ 0.0	50.6 $\pm$ 0.0	36.3 $\pm$ 0.0	18.3 $\pm$ 0.0	22.7 $\pm$ 0.0	92.0 $\pm$ 0.0

and use a linear classifier to predict the target. To establish a reference point for interpretation, we also perform the same prediction using randomly initialized features, representing the performance expected when no meaningful information is available in the inputs. Table 7.5 shows the prediction accuracy of all features in procgen environment. Notice that he same privilege information may belong to different properties in different environments, which we label out in the tables. In some environments, the same privilege information can be static properties of some objects and can dynamic for other objects. In such case, we mark such privilege information as "mixed". Dyn-O shows strong disentanglement results, particularly as the prediction accuracy of static features for position and area is close to that of random features, indicating that static features capture little to no information about dynamic properties.

We further illustrate the disentanglement between static and dynamic features with qualitative examples. Figure 7.7 presents rollouts generated after swapping static features between two initial states. As shown, rollouts from the same initial states (the first and third rows) remain nearly identical, except for the color changes in the avatar and floor. The same pattern is observed in the second and fourth rows, demonstrating that Dyn-O preserves dynamics while modifying only static properties.

Together, the results in Table 7.5 and Figure 7.7 answer Question **Q4** in the affirmative.

7.4 Summary

This chapter introduces Dyn-O, a world modeling method that builds on object-centric representations. By leveraging segmentation masks during training with a schedule, Dyn-O learns high-quality object-centric representations while avoiding the overhead of segmentation computation at inference time. Additionally, Dyn-O further disentangles each object feature into static, time-invariant components (e.g., color) and dynamic, time-variant components (e.g., position), allowing it to generate diverse predictions.

While Dyn-O improves representation and prediction quality compared to prior object-centric representation and world model work, it has several limitations.

- First, in some environments, object-centric representations may extract more state factors than necessary. For example, in a parking lot, many cars remain stationary and effectively constitute part of the background, rather than meaningful state factors that should be modeled by the world model. To address this limitation, Chapter 8 introduces a novel approach for extracting state factors through latent action model learning, which focuses on extracting factors associated with independent actions.
- Second, Dyn-O performs prediction in latent space and relies on a pretrained decoder solely for visualization. Exploring more sophisticated decoding methods, such as diffusion models, could further improve the visual fidelity of the generated rollouts.

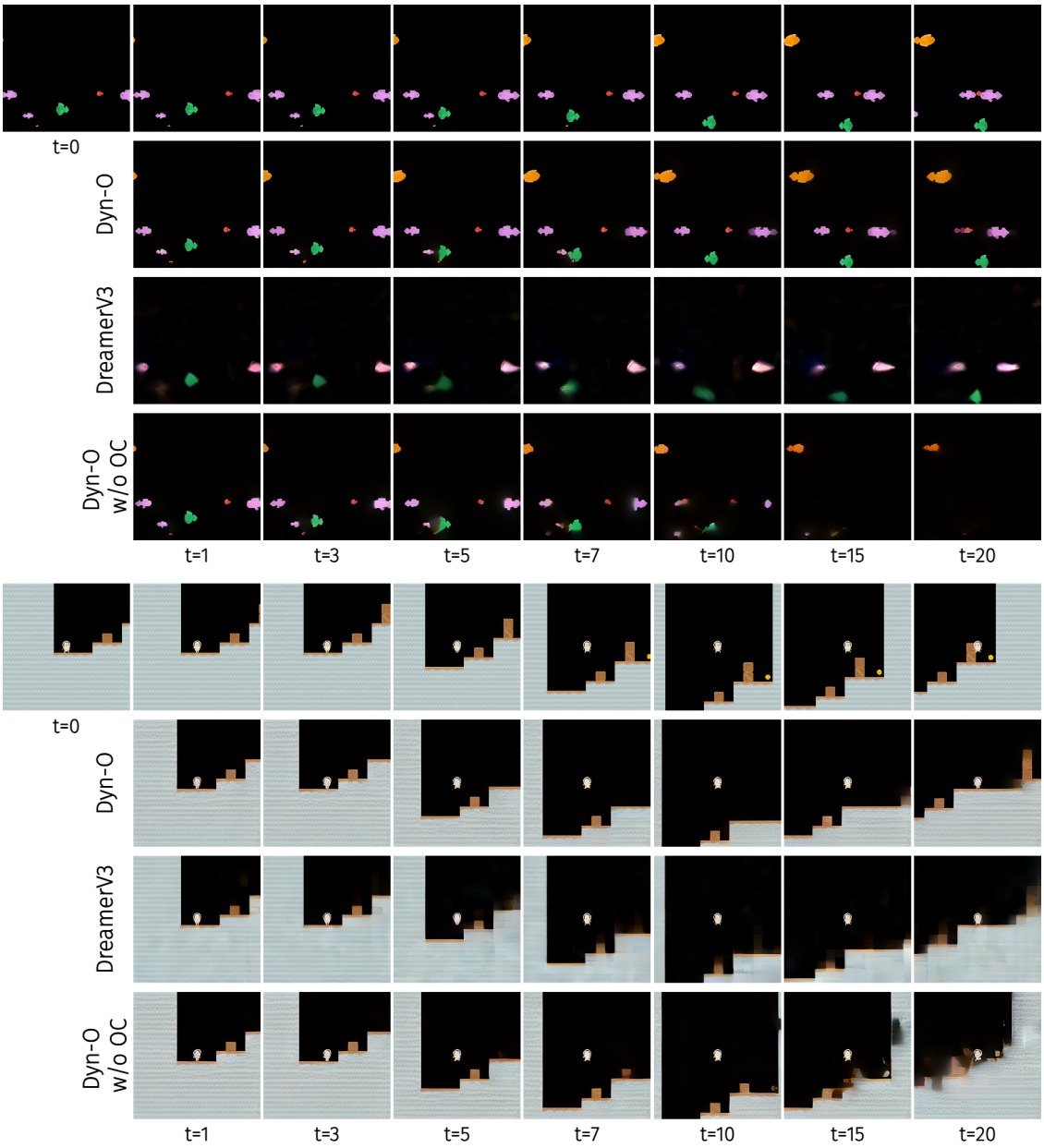


Figure 7.5: Dyn-O generates more accurate rollouts than Dreamer in **bigfish** and **coinrun**. In each environment, the first row displays a trajectory collected in the real environment. The second row depicts the prediction inside the world model by Dyn-O (ours). The third and fourth rows show the prediction of Dreamer and Dyn-O w/o OC respectively. In bigfish, our method keeps consistent prediction for each fish until the 15th step, while baselines lose track of multiple small fish before the 10th step. Similarly, in coinrun, compared to baselines, Dyn-O generates predictions with clearer floor and boxes.

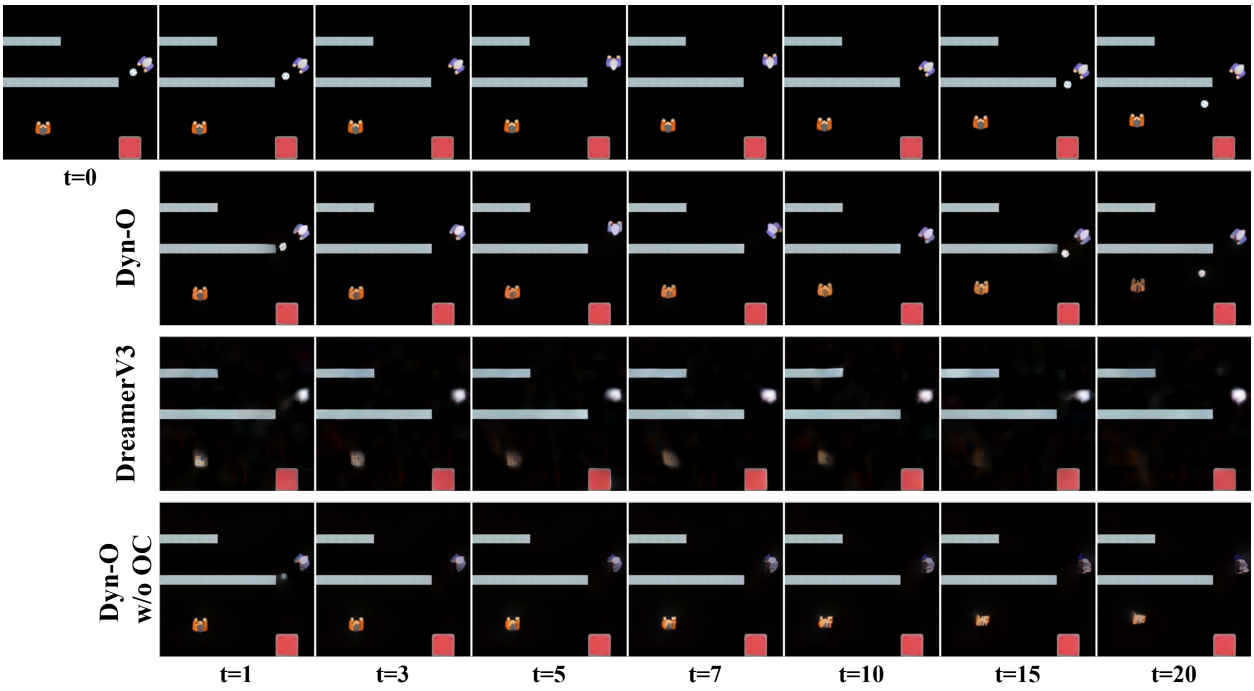


Figure 7.6: 20-step rollouts in **dodgeball**. **1st** row: ground-truth, **2nd** row: Dyn-O (ours), **3rd** row: DreamerV3, and **4th** row: Dyn-O w/o OC. Dyn-O significantly outperforms dreamer, with sharp player shape and accurate predictions of threw balls.

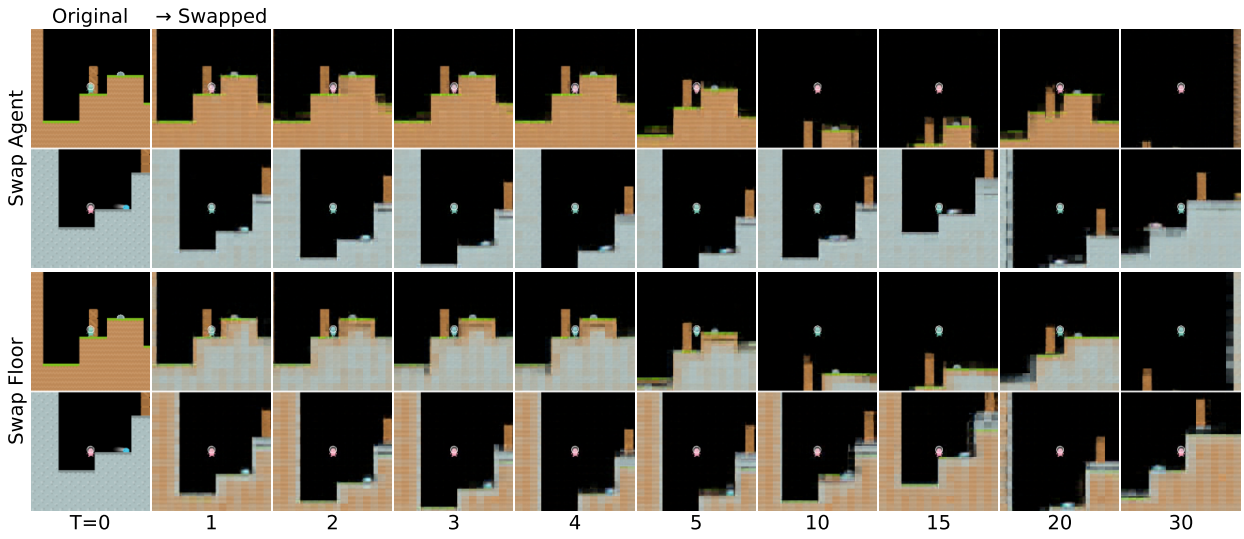


Figure 7.7: Dyn-O generates dynamically consistent rollouts after exchanging static features. In the top two rows, we swap the static features of the avatar (the small agent in the center of the image) between two initial states and generate 30-step rollouts using Dyn-O. The results show that only the avatar's color changes, while all other objects remain unchanged. In the bottom two rows, we exchange the static features of the floor, and Dyn-O consistently swaps their colors while keeping all other objects intact.

Chapter 8: Action-Grouped Representation from Factored Latent Action Models (FLAM)

The methods introduced in Chapters 3-6 assume that the state factorization is given. However, in many environments, only low-level observations, such as images, are accessible. Although Chapter 7 introduced a method for extracting state factors using object-centric representations, such representations may extract more state factors than necessary. For example, in a parking lot, many cars remain stationary and effectively constitute part of the background, rather than meaningful state factors that the world model should explicitly represent. To address this limitation, this chapter introduces a method for extracting state factors using latent action models, which focus on identifying factors associated with independent actions. A Latent Action Model (LAM) learns world models from action-free videos by learning an inverse dynamics model (IDM) to infer latent actions and a forward dynamics model (FDM) to predict the next observation. In-the-wild videos often contain complex scenes where many entities may act simultaneously. However, most existing approaches rely on monolithic inverse and forward dynamics models that learn a single latent action to control the entire scene, and therefore struggle in such environments. To address this issue, this chapter introduces the Factored Latent Action Model (FLAM), a factored dynamics framework that decomposes the scene into independent factors, each inferring its own latent action and predicting its own next-step factor value. This factorized structure enables more accurate modeling of complex multi-entity dynamics and improves video generation quality in action-free video settings compared to monolithic models.

This chapter is structured as follows. Section 8.1 first introduces the motivation behind latent action models and the limitations of previous work. Section 8.2 presents the proposed algorithm FLAM and describes how it learns to factorize the state space and the latent action space. Section 8.3 presents the empirical evaluation of FLAM, covering its performance in representation learning, world model learning, and policy learning. Finally, Section 8.4 discusses limitations of FLAM.

Contribution: In addition to my advisor, FLAM is joint work with Chang Shi, Jiaheng Hu, Kevin Rohling, Roberto Martín-Martín, and Amy Zhang. My contributions are the latent

action model design and its implementation. Meanwhile, Shi proposed how to use extract latent actions to facilitate policy learning, Hu and Rohling helped with some of the experiments, while Martín-Martín and Zhang provided technical advice.

8.1 Motivation

While Chapters 3-6 assume that the state factorization is given, this chapter considers the setting where the factorization is unknown, and investigates how to extract state factors from image observations by leveraging the independent actions of each entity. This chapter builds on the framework of Latent Action Models (LAM) (Schmidt and Jiang, 2024; Bruce et al., 2024), which have unlocked the possibilities of learning world models from action-free videos that are abundant on the web. Specifically, these approaches use an inverse dynamics model to encode environmental changes into a *single* latent action. The latent action is then used to train a forward dynamics model, allowing controllable predictions of future frames from in-the-wild videos.

However, in-the-wild videos often contain complex scenes where many entities may be taking actions simultaneously: for instance, a robot video may include independent arm movements and shifting camera perspectives; while a soccer game involves several players, the ball, and even background audience motion, each of which acts independently. Compressing all these motions into a *single* latent action is challenging, since the complexity of the underlying action space grows exponentially with the number of movable entities (Figure 8.1). Consequently, existing methods struggle with latent action learning in such settings, which severely limits their applicability to diverse scenarios.

This chapter introduces the Factored Latent Action Model (FLAM), where the latent state is decomposed into a set of factors, each independently predicting its latent action and its next-step value via shared factored inverse and forward dynamics models. Compared to prior work that must capture all joint action combinations within a single latent action space, FLAM assumes a shared latent action space across factors and thereby reduces the learning problem to modeling each entity's action patterns within this common space. Additionally, regarding forward dynamics, unlike most prior LAM approaches that use a monolithic scene representation entangling all entities, FLAM

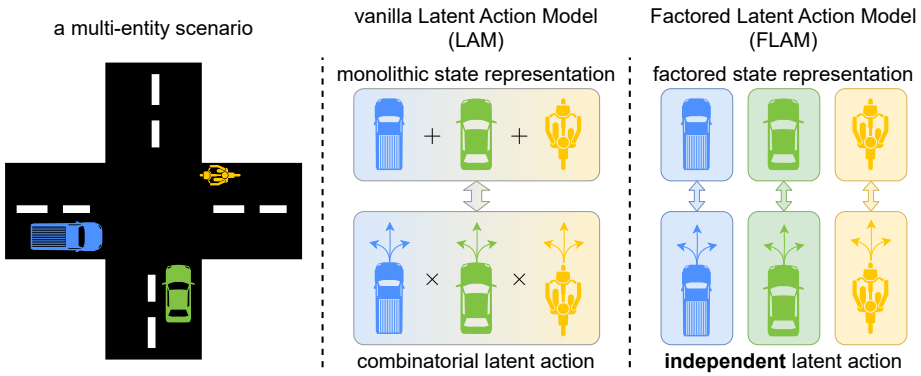


Figure 8.1: In multi-entity scenarios, **(left)** such as an intersection with three road users: **(middle)** a vanilla latent action model encodes the scene change with a *single* latent action of dimension d, which makes learning challenging as this latent action space needs to model all $|\mathcal{A}|^K$ joint action combinations. **(right)** In contrast, FLAM decomposes the state into K factors, each with its own latent action of dimension $\frac{d}{K}$. Additionally, we assume all latent actions share the same space (i.e., with the same prior / codebook), which reduces the learning problem to modeling the $|\mathcal{A}|$ actions per factor rather than their $|\mathcal{A}|^K$ joint combinations.

factorizes the scene into compositional entities with a shared forward dynamics model, inherently supporting permutation invariance and enabling stronger generalization. With next frame prediction as the training objective, FLAM learns factorized state and action representations from action-free video data, leading to more accurate modeling of complex, multi-entity dynamics and improved prediction quality compared to previous work.

Based on experiments on both simulated and real-world multi-entity datasets, including autonomous driving videos, we find that FLAM outperforms prior state-of-the-art methods in both prediction quality and factor-entity correspondence. Moreover, the inferred latent actions capture essential entity behaviors and enable sample-efficient policy learning.

8.2 FLAM

From a high-level perspective, FLAM enables efficient latent action model learning in multi-entity scenarios by inferring a set of factorized latent actions rather than a single latent action between each pair of frames, via the two learning phases shown in Figure 8.2 and Algorithm 9:

- **Encoder learning** (Section 8.2.1): FLAM pre-trains a VQ-VAE to extract high-level features

from pixels, allowing the latent action model to learn in the feature space rather than the pixel space for the purpose of efficient learning.

- **Latent action model (LAM) learning** (Section 8.2.2): Using the extracted features, FLAM decomposes the scene into several independent factors, also referred to as *slots*. For each slot, an inverse dynamics model infers a separate latent action from its current and next-frame values. Then, based on its current value and the corresponding latent action, a forward dynamics model independently predicts the next-frame value for each slot. Finally, all predicted slots are mapped back to the feature space and decoded into the next video frame.

Note that, although we separate encoder learning and LAM learning into two stages, they can in principle be learned jointly in an end-to-end manner. After FLAM learns to model the world from action-free videos, its inferred latent actions can be used for either controllable video generation or policy learning to solve downstream tasks. We refer to this utilization as the third phase, and discuss how to leverage FLAM for both settings in Section 8.2.3.

8.2.1 Pretrained Encoder

As shown in Figure 8.2 (a), FLAM learns a VQ-VAE (van den Oord et al., 2017) to extract features from raw pixels, enabling fast LAM learning. For each frame $o \in \mathbb{R}^{H \times W \times 3}$, a CNN encoder first extracts n_e patch-level features $e \in \mathbb{R}^{n_e \times d_e}$. The features are then quantized from continuous encoder outputs to the nearest entry in a discrete codebook, denoted as $e_q \in \mathbb{R}^{n_e \times d_e}$, using finite scalar quantization `FSQ` (Mentzer et al., 2024), which forces features into a finite set of learned embeddings. Finally, a decoder reconstructs the frame from the quantized feature:

$$
\begin{aligned}
e &= \texttt{Encoder}(o), \\
e_q &= \texttt{FSQ}(e), \\
\hat{o} &= \texttt{Decoder}(e_q).
\end{aligned}
$$

The VQ-VAE is trained by minimizing the following image reconstruction loss:

$$
\mathcal{L}_{\texttt{VQ-VAE}}(o) = ||o - \hat{o}||^2. \tag{8.1}
$$

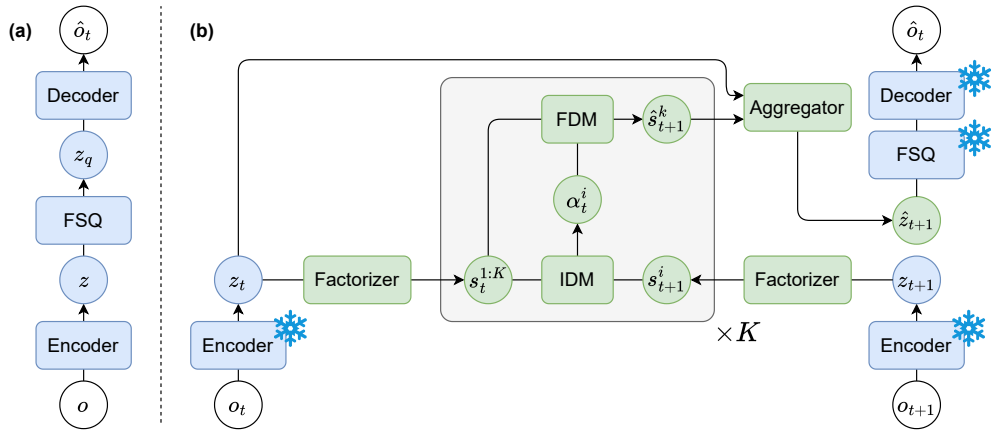


Figure 8.2: Two training stages of FLAM. **(a)** A VQ-VAE is pretrained to extract features for latent action model learning. **(b)** FLAM infers latent actions and makes predictions for each factor independently, with all modules trained jointly to minimize the prediction error.

8.2.2 Factored Latent Action Model (FLAM)

As shown in Figure 8.2 (b), our latent action model contains four key components: 1) a factorizer that decomposes the scene e into a set of independent slots ξ, 2) a shared inverse dynamics model that infers a separate latent action α^i for each slot, 3) a shared forward dynamics model that, given the current slot value and latent action, predicts the next-frame value for each slot, and 4) an aggregator that maps the predicted slots back to the feature space. These four components are jointly trained to minimize the next frame prediction error.

Factorizer To decompose the scene into a set of factors with independent actions, FLAM uses slot attention (Locatello et al., 2020). For each frame, K slots $\xi_t \in \mathbb{R}^{K \times d_\xi}$ are initialized from learned embeddings and then compete to bind to different regions in the frame through iterative slot attention. Meanwhile, to ensure that each slot consistently binds to the same object, we add an additional causal temporal attention layer so that each slot can refer to its values at all previous

timestamps. Overall, the factorizer computes the slots as follows:

$$\text{for each slot attention iteration}$$

$$\xi_t = \texttt{Slot-Attn}(\texttt{query} = \xi_t, \texttt{key} = e_t), \tag{8.2}$$

$$\xi_t^i = \texttt{Self-Attn}(\texttt{query} = \xi_t^i, \texttt{key} = \xi_{1:t}^i). \tag{8.3}$$

Inverse dynamics model (IDM) After extracting independent slots, the IDM aims to infer a latent action α_t^i for each slot i, based on all current slots $\xi_t^{1:K}$ and its next-frame value ξ_{t+1}^i:

$$\alpha_t^i = \texttt{IDM}(\xi_t^{1:K}, \xi_{t+1}^i), \tag{8.4}$$

where for a variable x, $x^{1:K}$ denotes the set $\{x^i\}_{i=1}^K$, and we use $\xi_t^{1:K}$ and ξ_t interchangeably. We use all current slots rather than just ξ_t^i as inputs to account for interactions between factors, enabling more accurate latent action prediction (e.g., a person in a car moves because of the car rather than by themself).

To implement the IDM, we adopt the spatio-temporal model introduced in Genie (Bruce et al., 2024). The spatial block applies self-attention to capture interaction information across $\xi_t^{1:K}$, while the temporal block applies cross-attention to compare ξ_{t+1}^i with its current value and encode the most meaningful changes between them:

$$\tilde{\xi}_t^i = \texttt{Self-Attn}(\texttt{query} = \xi_t^i, \texttt{key} = \xi_t^{1:K}),$$
$$\tilde{\alpha}_t^i = \texttt{Cross-Attn}(\texttt{query} = \xi_{t+1}^i, \texttt{key} = [\tilde{\xi}_t^i, \xi_{t+1}^i]).$$

Note that although we use only the current and next values in the temporal block, the model can be easily extended to condition on the full history $\xi_{1:t}^i$ or a fixed window $\xi_{t-w:t}^i$, where w denotes the window length.

Similar to prior LAM methods, we regularize the capacity of the latent action to prevent it from simply copying the next state ξ_{t+1}^i and thereby bypassing dynamics learning. Concretely, following the variational autoencoder (VAE) framework (Kingma and Welling, 2014), we project

Algorithm 9 FLAM

Prepare a video dataset of $(o_{1:T})$.
Initialize the VQ-VAE (`Encoder`, `FSQ`, `Decoder`) and the latent action model (`Factorizer`, `IDM`, `FDM`, `Aggregator`).
Pretrain the VQ-VAE with the reconstruction loss in Eq. (8.1).
// Train the latent action model
Extract features with the encoder:

$$e_{1:T} = \texttt{Encoder}(o_{1:T}).$$

Extract slots using Eq. (8.2) - (8.3):

$$\xi_{1:T}^{1:K} = \texttt{Factorizer}(e_{1:T}).$$

for slot $i = 1, \ldots, K$ and time $t = 1, \ldots, T-1$ **do**
  Infer the latent action: $\alpha_t^i = \texttt{IDM}(\xi_t^{1:K}, \xi_{t+1}^i)$.
  Predict the next-frame slot: $\hat{\xi}_{t+1}^i = \texttt{FDM}(\xi_t^{1:K}, \alpha_t^i)$.
end for
Map predicted slots back to the feature space:

$$\hat{e}_{2:T} = \texttt{Aggregator}(e_{1:T-1}, \hat{\xi}_{2:T}^{1:K}).$$

Optimize the latent action model with the prediction loss in Eq. (8.6).

$\tilde{\alpha}_t^i$ to a normal posterior distribution $q(\alpha_t^i)$ and impose a KL divergence regularization toward a unit normal prior $p(\alpha_t^i)$. The latent action is then sampled from this posterior as:

$$\alpha_t^i \sim q(\alpha_t^i) = \mathcal{N}\left(\texttt{M}_\mu(\tilde{\alpha}_t^i), \texttt{M}_\sigma(\tilde{\alpha}_t^i)\right),$$

where $\texttt{M}_\mu$ and $\texttt{M}_\sigma$ are the projection networks that map $\tilde{\alpha}_t^i$ to the mean and standard deviation of the posterior distribution.

Forward dynamics model (FDM) To provide the learning signals for the IDM, the forward dynamics model takes all current slots $\xi_t^{1:K}$ together with the latent action α_t^i and predicts the next-frame value for each slot $\hat{\xi}_{t+1}^i$:

$$\hat{\xi}_{t+1}^i = \texttt{FDM}(\xi_t^{1:K}, \alpha_t^i). \tag{8.5}$$

Similarly to the IDM, the same FDM is shared across all slots, and we use all current slots instead of just ξ_t^i as inputs to capture interactions between slots, as they are not intended to be encoded by the latent action. For implementation, we use the same spatio-temporal model as the IDM, except

that the temporal attention uses different queries and keys:

$$\tilde{\xi}_t^i = \texttt{Self-Attn}(\texttt{query} = \xi_t^i, \texttt{key} = \xi_t^{1:K}),$$
$$\hat{\xi}_{t+1}^i = \texttt{Cross-Attn}(\texttt{query} = \xi_t^i, \texttt{key} = [\tilde{\xi}_t^i, \alpha_t^i]).$$

Aggregator Finally, the aggregator maps the predicted slots back into the feature space, enabling decoding into the next-frame prediction. Instead of relying solely on $\hat{\xi}_{t+1}^i$ to predict the feature, the aggregator also conditions on the current feature e_t. This design allows $\hat{\xi}_{t+1}^i$ to focus on capturing changes between current and next time steps, rather than redundantly encoding static visual details. Concretely, the aggregator uses cross-attention to update each patch feature based on the predicted slots:

$$\hat{e}_{t+1} = \texttt{Cross-Attn}(\texttt{query} = e_t, \texttt{key} = \hat{\xi}_{t+1}^{1:K}).$$

Training objective The four components of the latent action model (Factorizer, IDM, FDM, and Aggregator) are trained jointly to minimize the feature prediction error and the KL regularization:

$$\mathcal{L}_{\texttt{LAM}}(e_t, e_{t+1}) = \|e_{t+1} - \hat{e}_{t+1}\|_2^2 + \beta_{KL} \cdot \sum_{i=1}^{K} D_{KL}[q(\alpha_t^i)||p(\alpha_t^i)], \tag{8.6}$$

where β_{KL} is the KL regularization coefficient.

Note that although FLAM adopts components similar to prior work (Villar-Corrales and Behnke, 2025; Klepach et al., 2025), such as slot attention and latent action model, it fundamentally differs in its training paradigm and the resulting inductive bias. Prior work typically follows a two-stage procedure: object-centric representations are first learned in isolation, and a latent action model is then trained on top of these fixed representations. As a result, the learned slots largely mirror conventional object-centric representations and are primarily organized according to visual appearance. In contrast, FLAM jointly optimizes slot extraction and latent action model with the prediction loss, with each slot endowed with an independent latent action in both the IDM and FDM. This end-to-end factorized dynamics learning encourages the model to separate entities based on the independence of their dynamics rather than visual similarity, yielding representations that are action-driven.

8.2.3 Learned Latent Actions Utilization

The latent actions learned by FLAM implicitly capture the controllable dynamics of individual entities in the environment, providing a natural interface for manipulating video generation. In particular, users can specify latent actions by sampling or selecting values from the prior distribution and use them as control variables during generation, resulting in diverse future trajectories. The latent actions can also be integrated with the forward dynamics model to enable model-based policy learning.

In addition to generation, the learned latent actions encode rich information about agent behavior and enable efficient policy learning directly from video. Using a small demonstration dataset with action labels, we first train an action decoder $\texttt{M}_a$ via supervised learning to map latent actions α to real actions. For larger datasets that contain only videos without action annotations, we extract latent actions using FLAM and apply the learned decoder to generate pseudo-labels for the real actions. These inferred action labels are then used to train a behavior cloning policy:

$$\mathcal{L}_\pi = \mathbb{E}_{(o,\alpha)\sim D} \|\pi(o) - \texttt{M}_a(\alpha)\|. \tag{8.7}$$

8.3 Experiments

Our central hypothesis is that the proposed factorized state representation and latent action formulation can better capture the features and dynamics of each entity, leading to more accurate world modeling on multi-entity videos. When used for policy learning, these improved world models should in turn yield higher downstream performance. To test this hypothesis, we evaluate FLAM on both simulation and real-world datasets and provide empirical evaluations pertaining to the following questions that support the indicated answers.

- **Q1)** Can FLAM learn more accurate world models than baselines? **Yes** (Section 8.3.1).
- **Q2)** Does FLAM learn action-driven factorization, grouping entities that share the same action into the same factor while separating entities with different actions? **Yes** (Section 8.3.2).
- **Q3)** Do latent actions extracted by FLAM enable more sample-efficient policy learning than learning without them? **Yes** (Section 8.3.3).

- **Q4)** How do architecture designs and hyperparameters affect performance? (Section 8.3.4).

Datasets We conducted experiments on 4 simulation datasets and 1 real-world dataset. The simulation datasets are collected from the MultiGrid environment (Oguntola et al., 2023) and the Procgen benchmark (Cobbe et al., 2020b). From Procgen, we use 3 environments: Bigfish, Leaper, and Starpilot. For a real-world dataset, we use nuPlan (Karnchanachari et al., 2024), an autonomous driving dataset. We use these datasets because they include multiple independent entities, such as agents, enemies, and cars.

Implementations As introduced in Section 8.2, the encoder is pretrained and frozen during latent action model learning. Rather than training a universal feature extractor across all datasets, we train a separate encoder for each dataset, as our work focuses on demonstrating the benefits of factorization for world models, rather than developing a universal world model. Dataset-specific encoder learning ensures that representation learning is not the performance bottleneck, enabling a meaningful comparison across different methods.

Baselines We compare FLAM against the following baselines.

- **Genie** (Bruce et al., 2024) and **AdaWorld** (Gao et al., 2025) are vanilla latent action models that infer a single scene-level latent action. Genie constrains latent action capacity via vector quantization, while AdaWorld adopts a VAE-style regularization. Since Genie shares a similar implementation with LAPO, we omit LAPO from our comparisons.
- **World Model** uses ground-truth actions instead of inferred latent actions and only learns the forward dynamics model. It is evaluated only on simulation datasets where action labels are available.
- **PlaySlot** (Villar-Corrales and Behnke, 2025) first learns object-centric representations using SAVi (Kipf et al., 2022) and then trains a latent action model on top of the frozen representations.

- **SlotFormer** (Wu et al., 2023b) similarly learns object-centric representations with SAVi, but trains a slot-based prediction model that conditions only on the current slots (i.e., without latent actions).

For a fair comparison, all methods use the same pretrained encoder for feature extraction and, when applicable, the same architectures for the factorizer, IDM, and FDM. In addition, to match latent action capacity across methods, while Genie and AdaWorld use a single latent action of dimension d, FLAM and PlaySlot use K factor-wise latent actions of dimension d/K each.

8.3.1 World Model Accuracy

We first evaluate FLAM's dynamics modeling accuracy, i.e., how accurately the model predicts future frames, and controllability, i.e., how much users can steer video generation via latent actions.

Prediction Accuracy To assess dynamics modeling accuracy, we first infer latent actions $\alpha_{1:T}$ from the *unseen* ground-truth frames $o_{1:T}$ and then generate T-step predictions $\hat{o}_{1:T}$ autoregressively. We measure prediction quality using peak signal-to-noise ratio (PSNR), learned perceptual image patch similarity (LPIPS), structural similarity index measure (SSIM), and Fréchet video distance (FVD). Finally, to contextualize the scores, we also report the reconstruction performance of the pretrained VQ-VAE on the future frames (denoted as **Recon**), which reflects the *best achievable quality* given the frozen encoder-decoder.

As shown in Table 8.1, FLAM achieves the best average performance and outperforms most baselines across datasets, highlighting the effectiveness of factored state representations and latent actions in multi-entity domains. Notably, FLAM even outperforms the World Model baseline. We attribute this result to two factors: in Procgen, only the player action is provided, so inferred latent actions help capture the stochastic motion of other entities (e.g., enemies); in MultiGrid, although actions for all agents are available, we follow the standard practice of treating them as a single monolithic action token. Compared to FLAM, World Model's underperformance further illustrate the limited generalization of monolithic models to state–action pairs that are rare or unseen during

Table 8.1: Prediction performance of all methods on simulation and real-world datasets. **Recon** reports the reconstruction quality of the pretrained VQ-VAE, serving as a reference upper bound for prediction performance.

Dataset	Metric	FLAM (ours)	AdaWorld	Genie	World Model	PlaySlot	SlotFormer	Recon
Average	PSNR (↑)	**34.9**	24.7	28.1	N/A	29.1	19.9	37.1
	SSIM (↑)	**0.890**	0.862	0.859	N/A	0.847	0.814	0.909
	LPIPS (↓)	**0.051**	0.096	0.091	N/A	0.120	0.223	0.035
	FVD (↓)	**419.6**	749.5	717.5	N/A	858.0	1597.7	215.1
MultiGrid	PSNR (↑)	**56.5**	24.0	33.9	56.1	**56.5**	24.9	56.5
	SSIM (↑)	**0.944**	0.909	0.933	0.943	**0.944**	0.912	0.944
	LPIPS (↓)	**3e-4**	0.08	0.014	6e-4	**3e-4**	0.056	3e-4
	FVD (↓)	**2.9**	913.6	177.5	5.5	**2.9**	451.6	2.5
Bigfish	PSNR (↑)	**30.8**	30.3	28.5	23.9	26.0	19.6	34.4
	SSIM (↑)	**0.983**	0.982	0.974	0.949	0.962	0.934	0.992
	LPIPS (↓)	**0.011**	0.012	0.021	0.049	0.038	0.221	0.004
	FVD (↓)	63.6	**63.3**	106.5	168.9	227.4	1455	16.8
Leaper	PSNR (↑)	**38.1**	23.3	35.0	24.7	27.5	21.4	40.3
	SSIM (↑)	**0.976**	0.924	0.973	0.924	0.945	0.913	0.977
	LPIPS (↓)	**0.002**	0.072	0.005	0.044	0.031	0.164	0.001
	FVD (↓)	**12.1**	348.1	27.3	127.3	361.5	1151	6.27
Starpilot	PSNR (↑)	**29.3**	27.8	27.0	25.8	18.3	19.8	32.6
	SSIM (↑)	**0.971**	0.965	0.962	0.959	0.886	0.907	0.980
	LPIPS (↓)	**0.015**	0.023	0.028	0.040	0.187	0.166	0.006
	FVD (↓)	**73.4**	113.4	141.0	194.9	792.0	720.9	24.7
nuPlan	PSNR (↑)	**19.7**	18.1	16.0	N/A	17.2	14.0	21.7
	SSIM (↑)	**0.577**	0.529	0.453	N/A	0.498	0.405	0.650
	LPIPS (↓)	**0.229**	0.292	0.389	N/A	0.344	0.508	0.163
	FVD (↓)	**1946**	2309	3135	N/A	2906	4210	1025

training. Meanwhile, as shown in Figures 8.4-8.8, object-centric baselines such as PlaySlot perform competitively in the simple MultiGrid domain but degrade on datasets with more complex visuals, where they often fail to maintain temporally consistent slots. This inconsistency leads to unreliable latent action inference and inaccurate predictions. These results demonstrate the benefit of jointly learning representations and dynamics, rather than training them in separate stages.

Scaling with the number of entities To investigate whether FLAM facilitates dynamics modeling in complex multi-entity videos, we conduct an ablation study on MultiGrid by varying the number of entities. For a fair comparison, we match the total latent action capacity between FLAM and AdaWorld. For instance, on the 8-agent setting, AdaWorld uses a single 256-dimensional latent action, while FLAM uses eight 32-dimensional latent actions. For Genie, we similarly ensure its latent action capacity by matching the codebook size to the number of joint action combinations. As shown in Figure 8.3, as the number of entities increases, FLAM remains substantially more robust than a non-factored LAM.

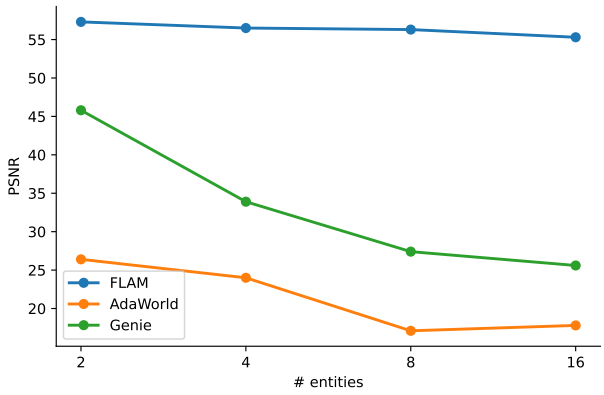


Figure 8.3: Prediction performance variation along with increasing number of entities in the scene.

Controllable video generation The factored latent actions learned by FLAM not only help with accurate world modeling, but can also serve as a manipulation surrogate to guide the video generation. We let human user specify an entity to control, together with a choice from the latent action prior distribution for each time step, and then rollout multiple steps. Meanwhile, the

remaining entities that are not manipulated would follow their original latent actions. An example of generated videos is shown in Figure 8.9. The first row is the original video. Given the first frame of the original video as the initial start, each of the following rows represents the generated video of manipulating one entity. It demonstrates that latent actions can be used as a control variable to generate various video frames even with the same initial frame. This result indicates that factorization offers the freedom of manipulating each entity independently instead of the monolithic scene, therefore leading to more diversity in video generation.

Together, the results in Figures 8.3-8.8, Figure 8.9 and Table 8.1 answer Question **Q1** in the affirmative.

8.3.2 FLAM State Representation

Next, we evaluate whether FLAM factorizes state representations according to entities' actions. Intuitively, the learned factorization is jointly determined by the degree of action independence among entities and the number of factors K. When many entities have similar or correlated actions (e.g., a flock of birds flying together), a smaller K encourages the model to merge them into the same factor, whereas a larger K allows the model to allocate separate factors to individual entities and capture finer-grained action differences. Conversely, when entities act largely independently, FLAM requires a sufficiently large K to model each entity accurately.

Correlated Entities We construct a controlled MultiGrid dataset where two of four agents always execute identical actions at every timestamp (i.e., their action sequences are strongly correlated), while other aspects of the scene (e.g., positions) may vary. Using $K = 3$, we evaluate whether FLAM maps the two correlated agents to the same factor. As shown in Figure 8.10, FLAM consistently assigns the correlated agents into a single factor, demonstrating that FLAM groups and separates entities based on their dynamics rather than visual appearance.

Independent Entities On a separate MultiGrid dataset where four agents act independently, we evaluate whether FLAM separates them into different factors with $K = 4$. We adopt the DCI

Table 8.2: Factor-agent correspondence in the MultiGrid environments, evaluated by DCI (disentanglement, completeness, informativeness) metrics.

	FLAM(ours)	PlaySlot	SlotFormer	Random
D (↑)	**0.91**	0.81	0.85	0.32
C (↑)	**0.91**	0.81	0.85	0.30
I (↑)	**0.93**	0.86	0.90	0.10

(disentanglement, completeness, informativeness) metric (Eastwood and Williams, 2018) by treating each factor as a latent variable and measuring how well it encodes each agent's information with a linear probe. High disentanglement and completeness indicates that each agent is primarily captured by a distinct factor, rather than being spread across multiple factors. High informativeness indicates that the corresponding factor captures a good amount of information about the agent and has good position prediction accuracy. As shown in Table 8.2, FLAM achieves the highest disentanglement and completeness, suggesting that it more reliably assigns each agent to an independent factors than the object-centric baselines. It also attains competitive informativeness, indicating that this improved factor–agent correspondence does not come at the cost of predictive accuracy. For reference, we also report the DCI scores obtained from randomly generated representations, denoted as **Random**.

Number of factors K Finally, on the same domain with four independently acting agents, we study the effect of the number of factors by sweeping $K \in \{2, 4, 8, 16\}$.

As shown in Table 8.3, when K is smaller than the number of entities, prediction performance degrades, as expected. Once K is at least the number of entities, performance becomes stable across different choices of K, with negligible differences due to training stochasticity. Moreover, as shown in Figure 8.11, even with excessive K, FLAM still consistently assigns each entity to a distinct factor across timestamps, demonstrating robustness to over-specifying K.

Therefore, for datasets where the number of entities varies across scenarios, one can choose a sufficiently large K to cover the maximum number of entities, allowing unused factors to remain inactive when fewer entities are present.

Table 8.3: Prediction performance of FLAM on the MultiGrid dataset with 4 agents and different number of factors K.

K	2	4	8	16
PSNR (↑)	53.4	**56.5**	**56.5**	**56.5**
SSIM (↑)	0.943	**0.944**	**0.944**	**0.944**
LPIPS (↓)	4e-4	**3e-4**	**3e-4**	**3e-4**
FVD (↓)	3.62	2.87	2.80	**2.73**

Together, the results in Figures 8.10-8.11 and Tables 8.2 and 8.3 answer Question **Q2** in the affirmative.

8.3.3 Latent Action Policy Learning

We evaluate whether FLAM can facilitate policy learning from video with limited action supervision. First, for each environment, we train an expert policy using Phasic Policy Gradient (Cobbe et al., 2021) and collect an expert demonstration dataset containing 1M frames. Next, we train an action decoder on a small labeled subset of the demonstrations, where ground-truth actions are available, and we vary the labeled subset size (1k and 10k frames) to study label efficiency. We then use learned IDM and action decoder to infer pseudo action labels on the full 1M-frame demonstration dataset, yielding state-action pairs without using the true action labels for training. Finally, we train a behavior cloning policy on these pseudo-labeled data and evaluate its performance in unseen environments. To quantify the benefit of pseudo labels, we compare against a behavior cloning policy trained only on the labeled subset. For a fair comparison, FLAM and the vanilla behavior cloning baseline use the same policy architecture and training setup, and the only difference is that FLAM additionally trains on pseudo action labels inferred from unlabeled videos. For reference, we also report the performance of the expert policy and a random policy.

As shown in Table 8.4, FLAM provides the largest gains at the intermediate label size (10k). With very few action labels (1k), it is challenging to learn a reliable action decoder, so the resulting noisy pseudo labels provide limited benefit. Thus, the results in Table 8.4 answer Question **Q3** in the affirmative.

Table 8.4: Behavior cloning policy performance, evaluated by mean episodic return.

Dataset	FLAM	vanilla BC	random
Bigfish (1k)	**1.8**	1.0	0.9
Bigfish (10k)	**8.8**	3.6	
Starpilot (1k)	1.8	**1.9**	1.1
Starpilot (10k)	**9.1**	6.4	

8.3.4 Ablation Studies

We further evaluate how architecture designs and hyperparameters affect FLAM's prediction performance.

Factorizer architecture As described in Section 8.2.2, a key design of FLAM's factorizer is the causal temporal attention, which allows each slot to attend to its past values across timestamps and improves slot consistency. We ablate this component by removing temporal attention from the factorizer. Instead, the factorizer follows SAVi by computing slots autoregressively and initializing the slots at each timestamp using the slots from the previous timestamp. Then we evaluate its impact on prediction performance and representation quality.

As shown in Table 8.5 and Figure 8.12, removing temporal attention leads to temporally inconsistent factor assignments and degraded predictions. Moreover, Table 8.6 shows that this ablation substantially reduces factor-entity correspondence, demonstrating the importance of temporal attention in maintaining consistent bindings over time.

Latent action model architecture In FLAM, for slot i, the IDM infers the latent action α_t^i from $\xi_t^{1:K}$ and ξ_{t+1}^i, while the FDM predicts ξ_{t+1}^i conditioned on $\xi_t^{1:K}$ and α_t^i. This design encourages per-slot action independence while still accounting for slot interactions. To evaluate this architectural choice, we compare against an ablation, **Global-Coupled FLAM**, where the IDM infers actions from all slots $(\xi_t^{1:K}, \xi_{t+1}^{1:K})$ and the FDM predicts ξ_{t+1}^i conditioned on $(\xi_t^{1:K}, \alpha_t^{1:K})$.

As shown in Table 8.5, this ablation achieves prediction performance similar to the original

Table 8.5: Prediction performance of FLAM and its ablative variants on the MultiGrid dataset.

Metric	FLAM (ours)	FLAM w/o Factorizer Temporal Attention	Global-Coupled FLAM
PSNR (↑)	**56.5**	51.6	56.3
SSIM (↑)	**0.944**	0.941	**0.944**
LPIPS (↓)	**3e-4**	3e-3	4e-4
FVD (↓)	**2.9**	36.6	3.72

Table 8.6: Factor-agent correspondence of FLAM and its ablative variants on the MultiGrid dataset, evaluated by DCI (disentanglement, completeness, informativeness) metrics.

	FLAM(ours)	FLAM w/o Factorizer Temporal Attention	Global-Coupled FLAM	Random
D (↑)	**0.91**	0.33	0.79	0.32
C (↑)	0.91	0.23	**1.00**	0.30
I (↑)	**0.93**	0.20	0.58	0.10

FLAM, as it retains the same overall capacity for encoding transition information. However, because both the IDM and FDM condition on all slots and all latent actions, the model has little incentive to maintain a one-to-one correspondence between entities and factors. In particular, the FDM can attend to α_t^j when predicting ξ_{t+1}^i and still achieve accurate prediction. Consequently, as shown in Table 8.6 and Figure 8.12, Global-Coupled FLAM tends to merge multiple entities into the same factor, yielding less disentangled representations.

KL regularization coefficient β_{KL} controls the trade-off between latent action capacity and prediction quality. We report prediction performance across different β_{KL} values to quantify FLAM's sensitivity to this hyperparameter. As shown in Table 8.7, FLAM is robust across a wide range of β_{KL} values, which makes it easy to tune.

8.4 Summary

This chapter introduces FLAM, a factored latent action model that addresses a limitation of prior work in challenging multi-entity scenarios, where the underlying action space grows exponentially with the number of entities. Specifically, FLAM decomposes both the state and

Table 8.7: Prediction performance of FLAM on the MultiGrid dataset with different KL regularization coefficient β_{KL} values.

Metric	$\beta_{KL} = 1e-5$	$\beta_{KL} = 1e-4$	$\beta_{KL} = 1e-3$	$\beta_{KL} = 1e-2$
PSNR (↑)	**56.5**	**56.5**	56.3	24.9
SSIM (↑)	**0.944**	**0.944**	**0.944**	0.911
LPIPS (↓)	**3e-4**	**3e-4**	4e-4	0.057
FVD (↓)	**2.84**	3.40	3.78	479.9

latent action representations into independent factors, and assumes all factor share in a common latent action space. Compared to prior work that must capture multi-entity behavior within a single monolithic latent action space, FLAM instead models each entity's action patterns within this shared space. This factorized structure enables more accurate modeling of complex multi-entity dynamics and improves both prediction quality and controllability over prior methods.

While FLAM improves representation and prediction quality compared to prior latent action model work, it has two main limitations.

- First, FLAM trains a VQ-VAE from scratch for each dataset. Adopting and fine-tuning pretrained tokenizers is a promising step toward sharing the encoder across datasets and moving toward a universal latent action model.
- Second, FLAM performs prediction in latent space and relies on a pretrained decoder solely for visualization. Exploring more sophisticated decoding methods, such as diffusion models, could further improve the visual fidelity of the generated rollouts.

FLAM constitutes the last technical contribution of this thesis. Together, the results in Chapter 7 and this chapter represent the completion of our efforts to extract state factors from low-level observations (Chapter 1.1 Contribution 4). To conclude the thesis, Chapter 9 provides a comprehensive overview of advancements in the fields that this thesis focuses on, and Chapter 10 summarizes the thesis contributions and outlines several future research directions.

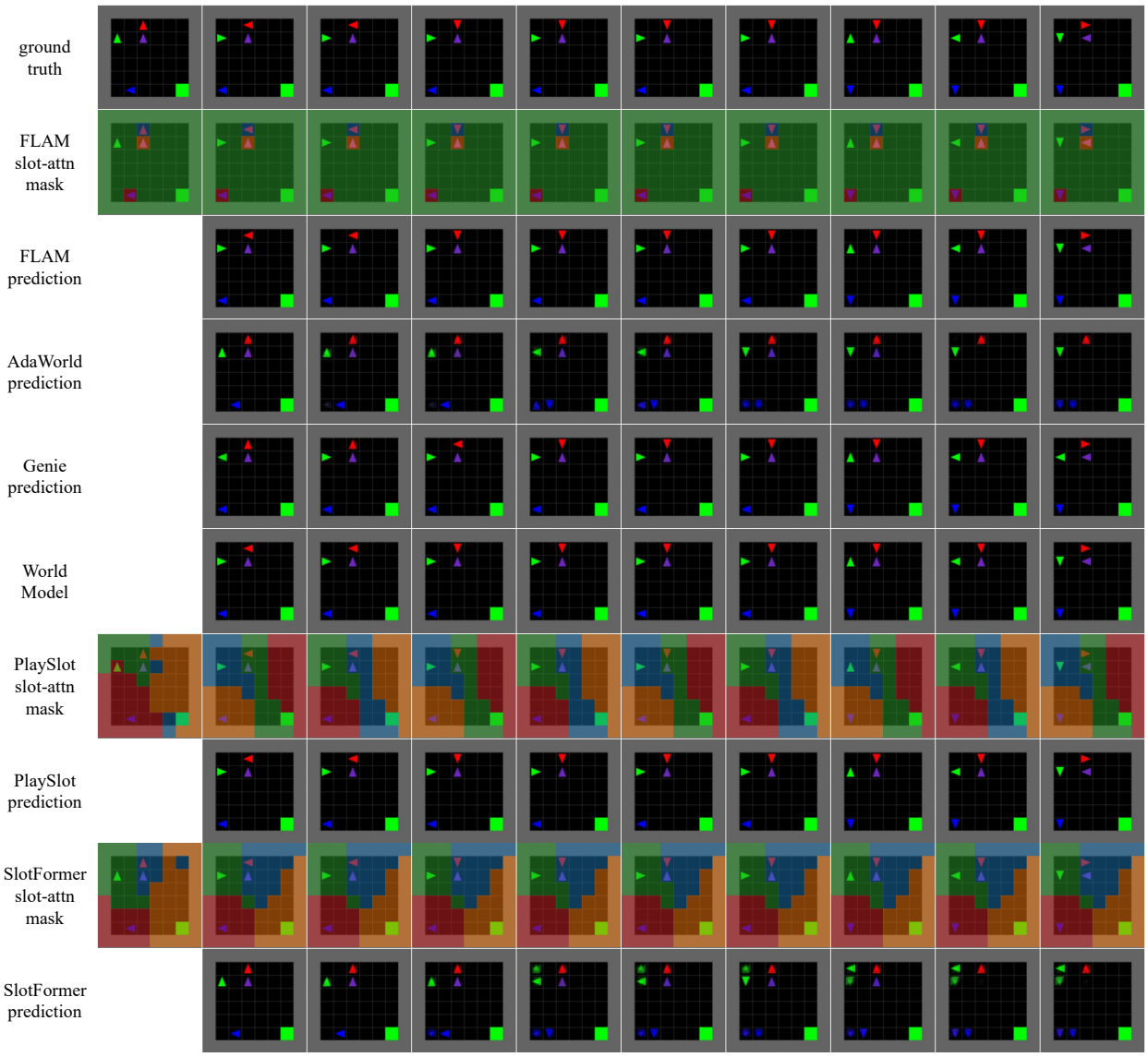


Figure 8.4: Prediction rollouts of all methods on the MultiGrid dataset.

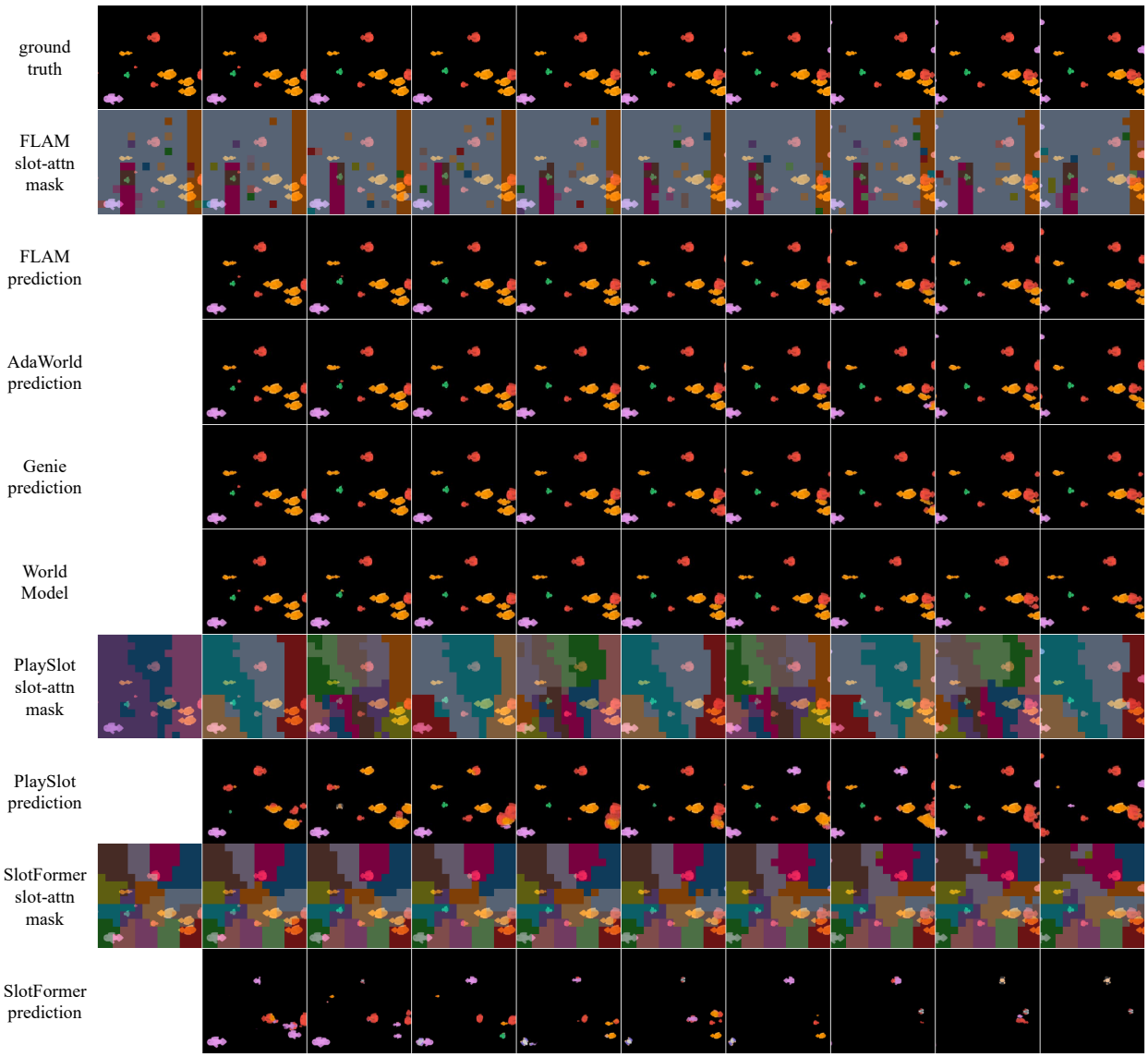


Figure 8.5: Prediction rollouts of all methods on the Bigfish dataset.

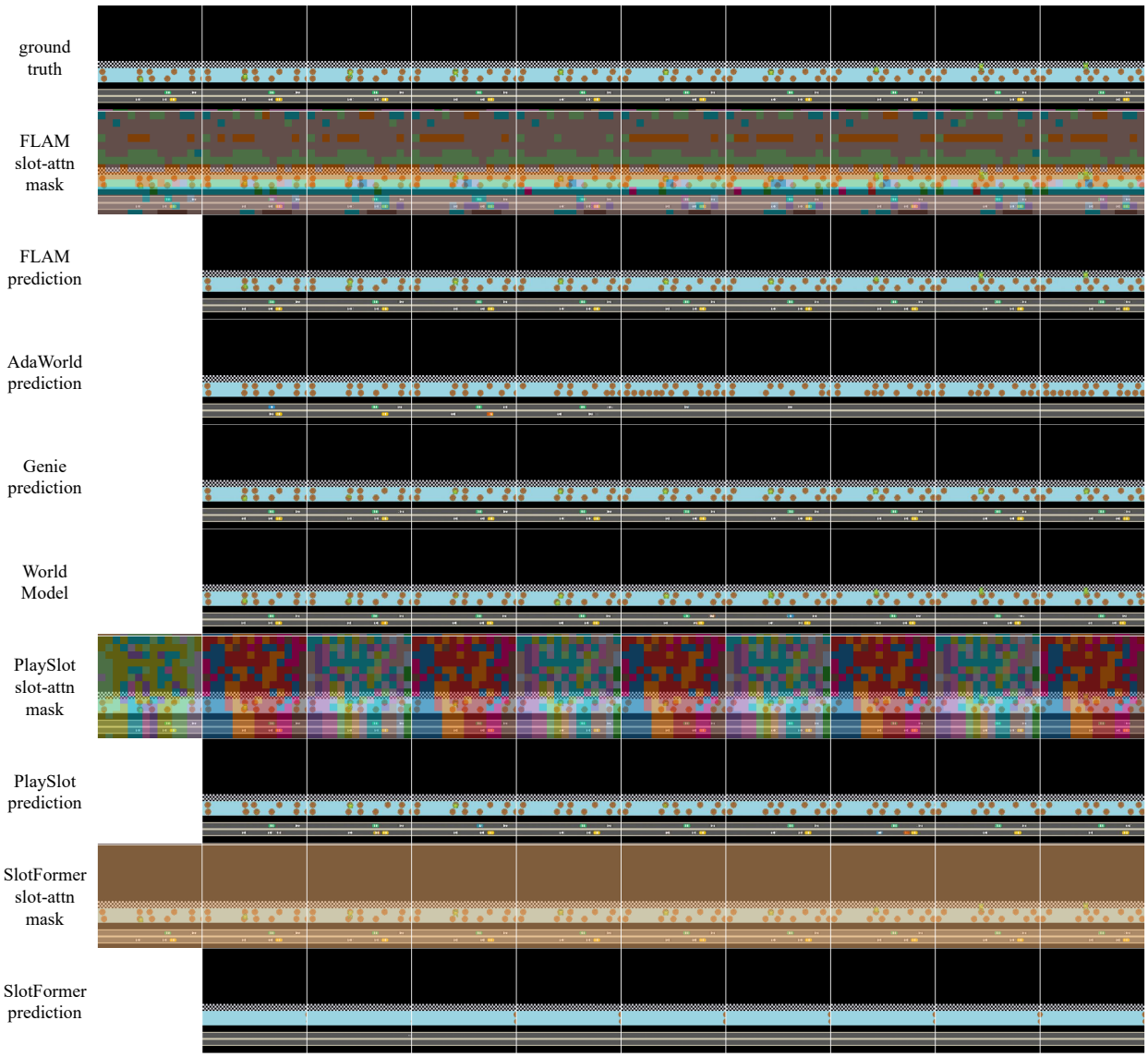


Figure 8.6: Prediction rollouts of all methods on the Leaper dataset.

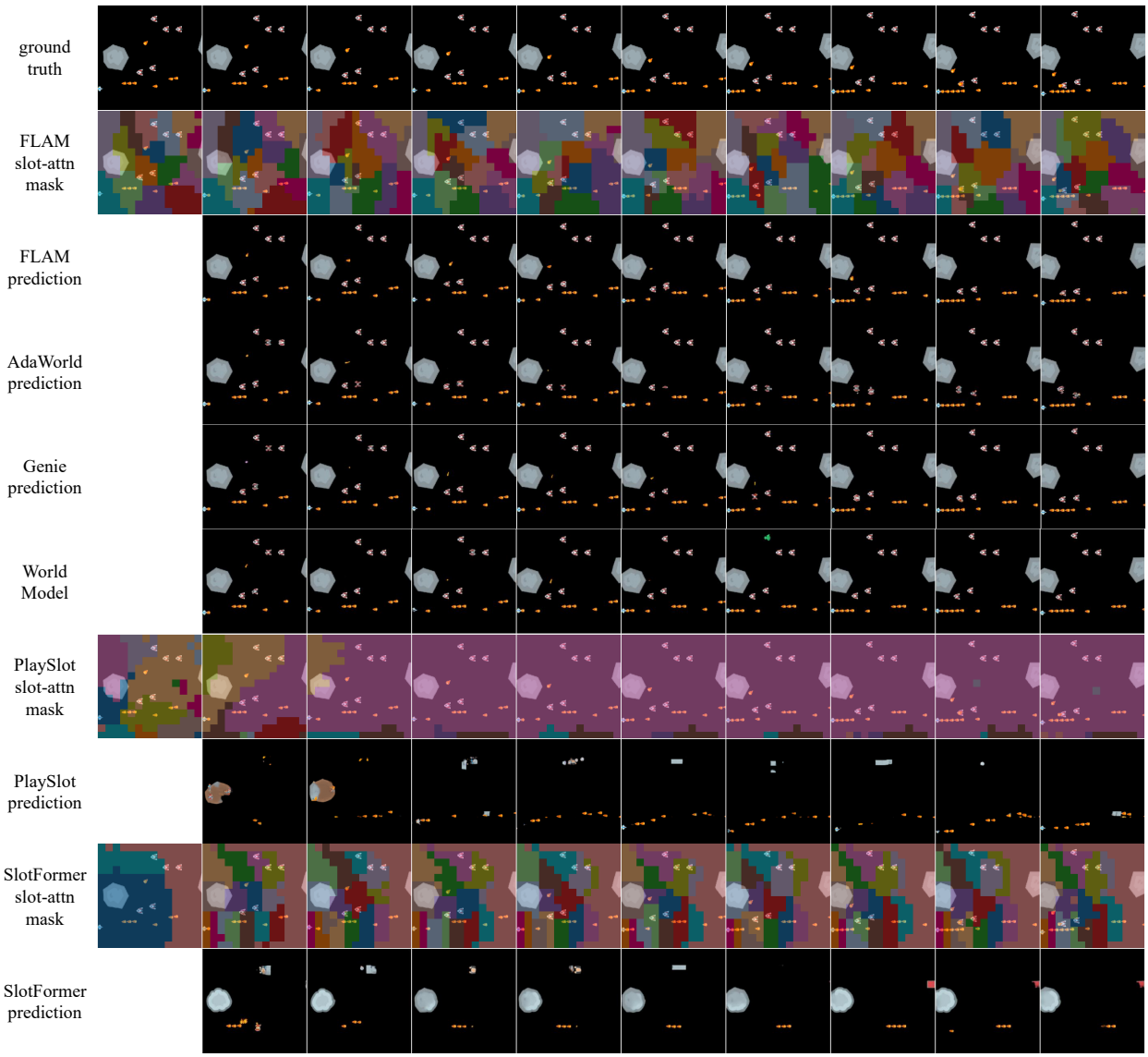


Figure 8.7: Prediction rollouts of all methods on the Starpilot dataset.

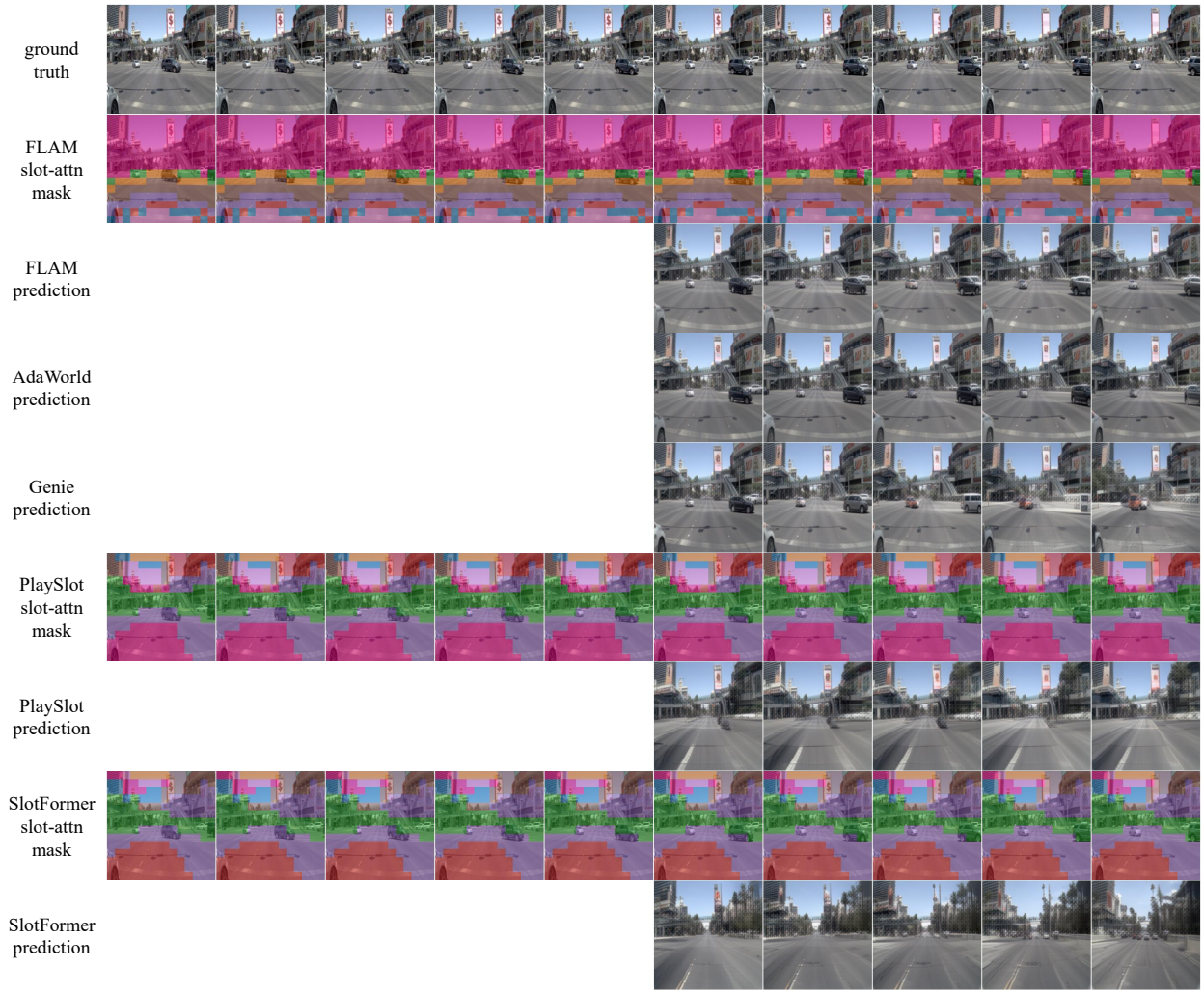


Figure 8.8: Prediction rollouts of all methods on the nuPlan dataset. Model conditioned on history of fixed window size $w = 5$.

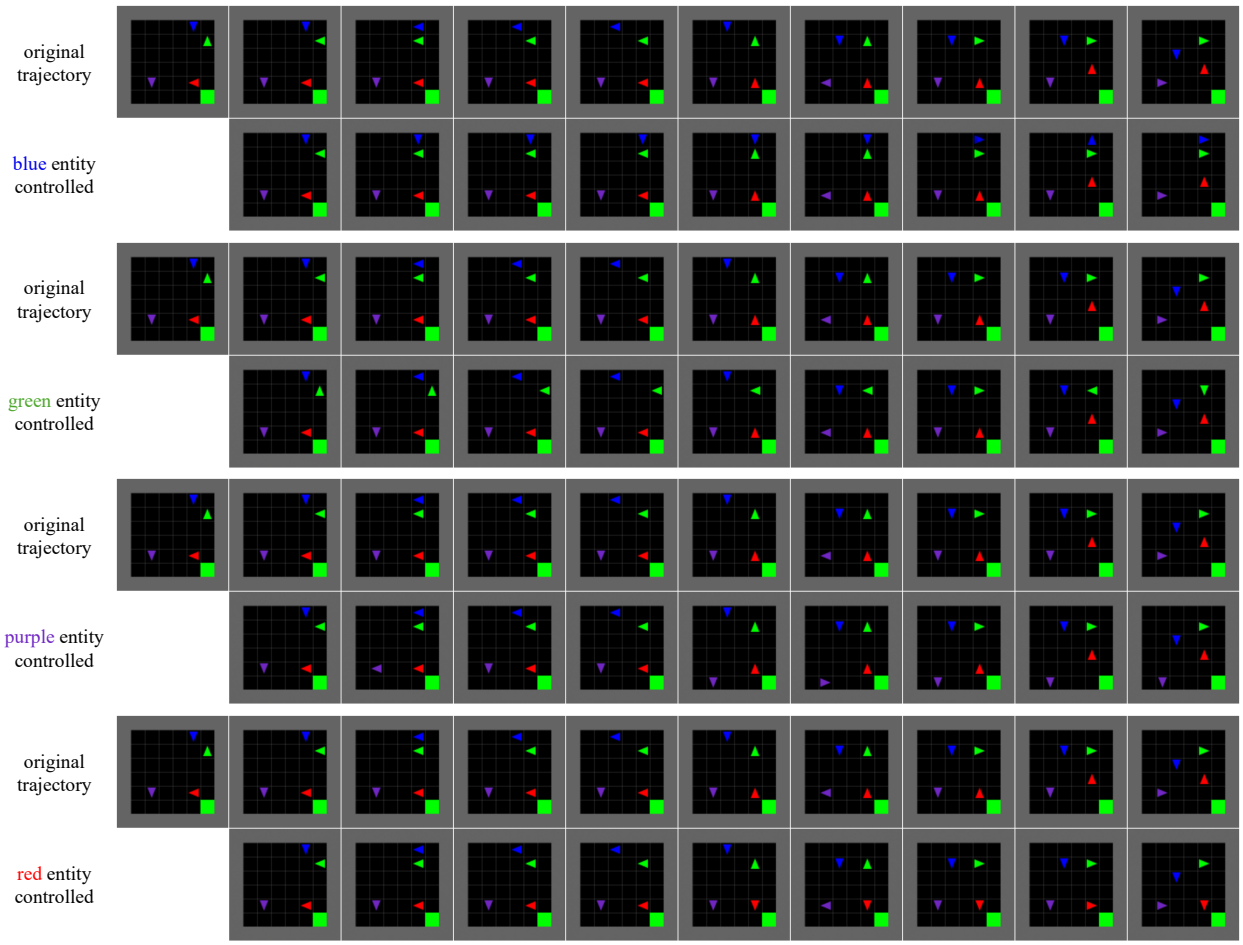


Figure 8.9: FLAM allows for changing / editing the motion of one entity without affecting the others. Here we show controllable video generation on MultiGrid through specifying latent actions for one of the movable agents, while other agents follow their original inferred latent actions.

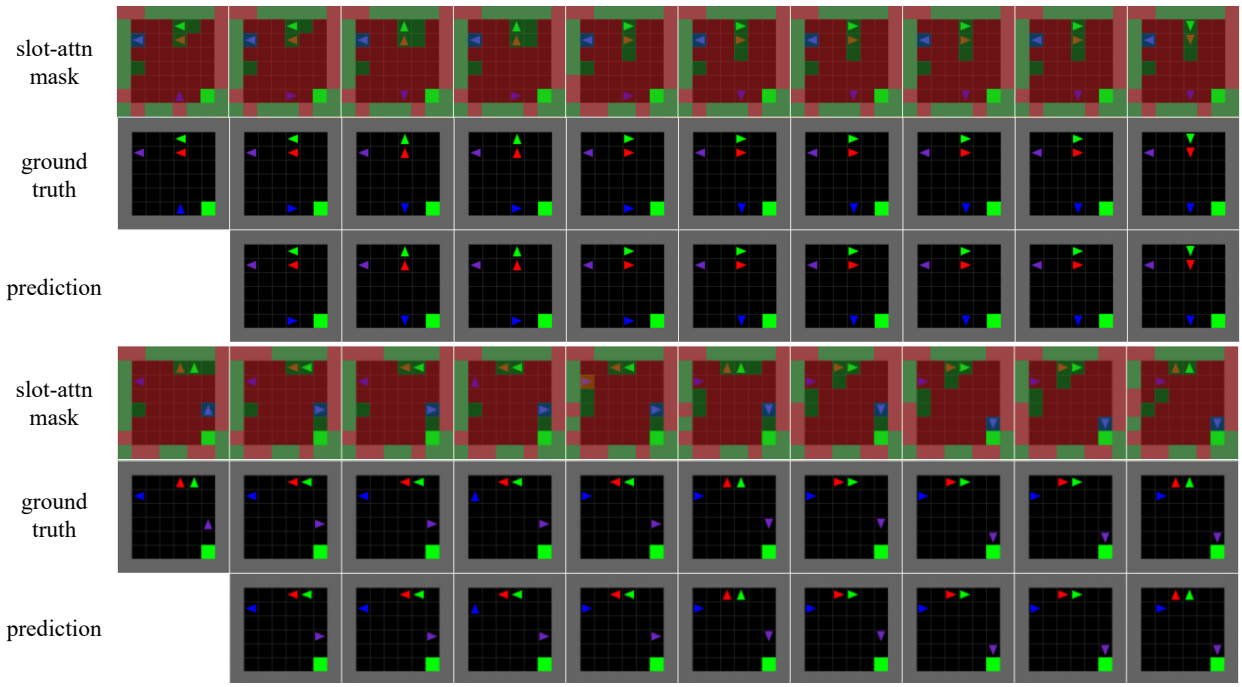
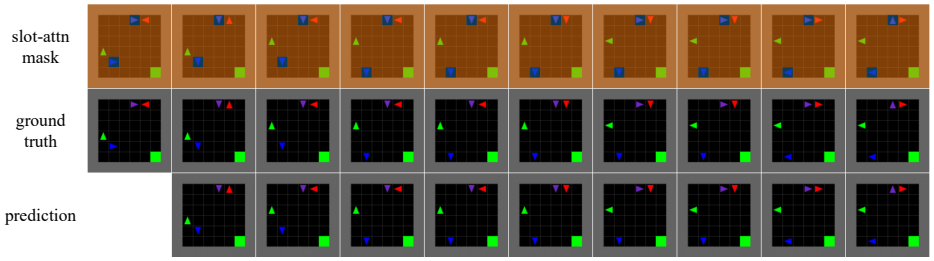
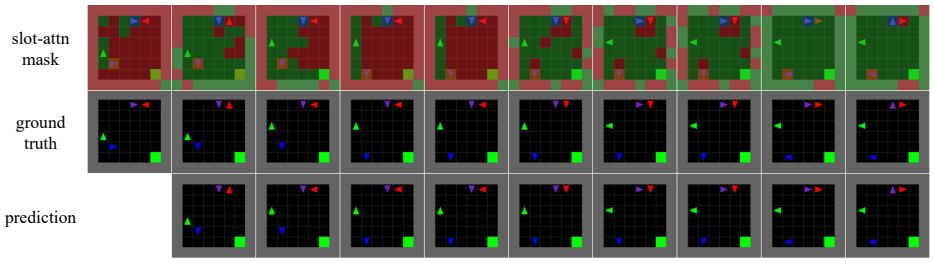


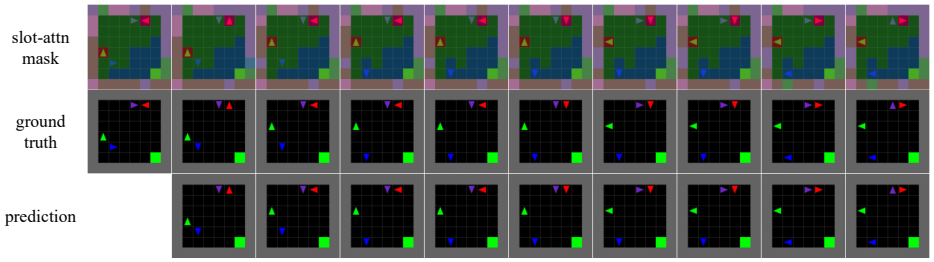
Figure 8.10: Two example rollouts where two entities (green and red ones) share the same actions. FLAM consistently maps the two correlated agents to the same factor.



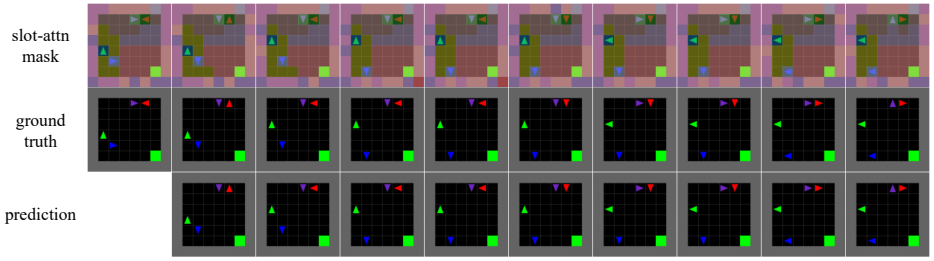
(a) $K = 2$



(b) $K = 4$

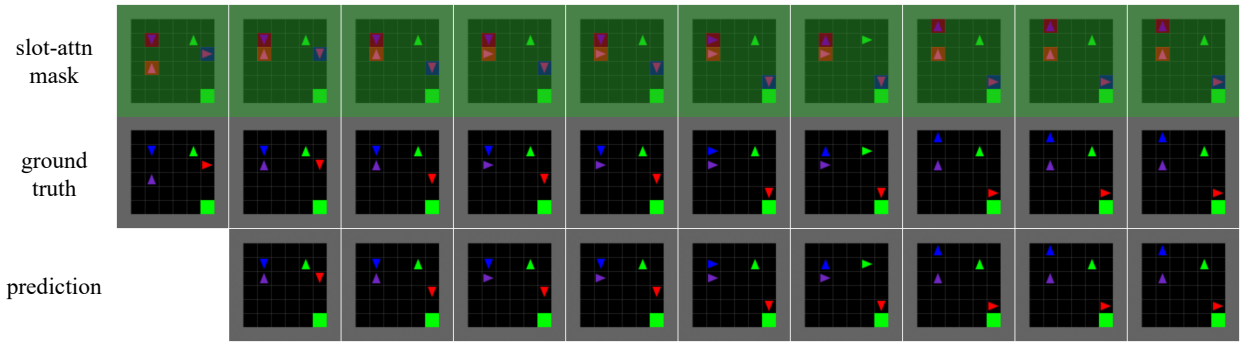


(c) $K = 8$

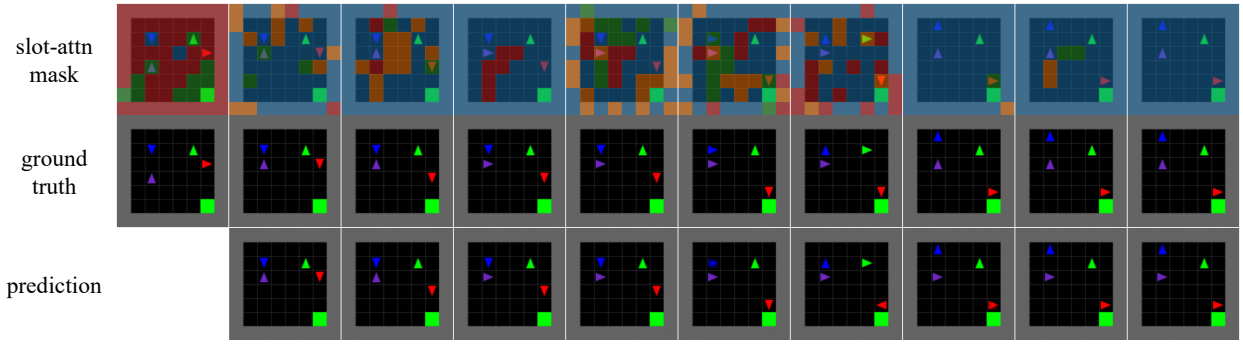


(d) $K = 16$

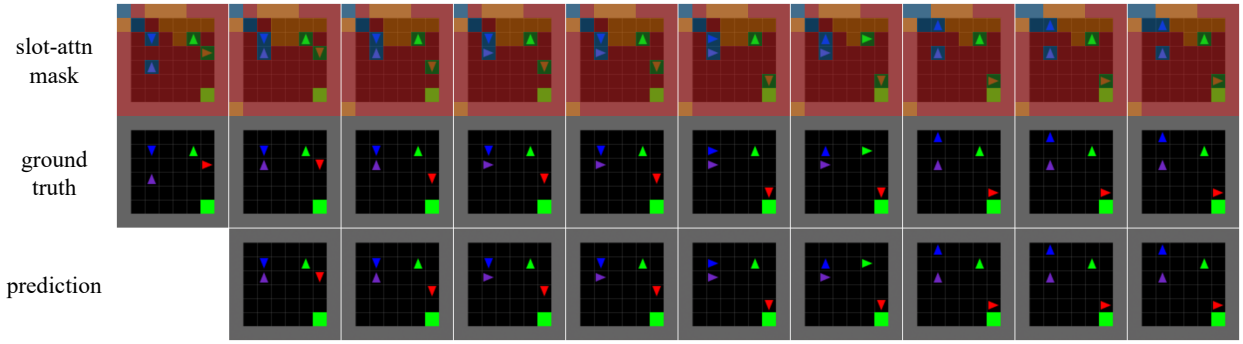
Figure 8.11: Ablation of FLAM on 4-agent MultiGrid dataset, with $K \in \{2, 4, 8, 16\}$. When K is less than the number agents, FLAM assigns two agents to each factor to maximize latent action utilization. When K is equal to or greater than the number of agents, FLAM consistently assigns each entity to a distinct factor, even with excessive K, demonstrating FLAM's robustness to over-specifying K.



(a) Original FLAM



(b) FLAM without factorizer temporal attention



(c) Global-Coupled FLAM

Figure 8.12: Architecture ablations of FLAM on the 4-agent MultiGrid dataset. Removing temporal attention from the factorizer leads to temporally inconsistent factor assignments and degraded long-horizon predictions (notably in the last few timestamps). In contrast, the Global-Coupled variant can produce accurate predictions but tends to assign multiple entities into the same factor, resulting in poorer disentanglement than the original FLAM implementation.

Chapter 9: Related Work

This chapter provides a literature review and explains the relationship between prior work and the contributions of this dissertation. Sections 9.1 and 9.2 review two central themes of this thesis: reinforcement learning, with an emphasis on its limitations in sample efficiency and generalization, and causality. Section 9.3 reviews the literature on model-based RL, which is relevant to our work in Chapters 3-6. Section 9.4 reviews the literature on state abstractions, which is relevant to our work in Chapters 3 and 4. Section 9.5 reviews the literature on intrinsic rewards, which is relevant to our work in Chapter 5. Section 9.6 reviews the literature on skill learning, which is relevant to our work in Chapter 6. Section 9.7 reviews the literature on object-centric representations, which is relevant to our work in Chapter 7. Section 9.8 reviews the literature on latent action models, which is relevant to our work in Chapter 8. Section 9.9 summarizes the contributions of this thesis in comparison to prior work.

9.1 Reinforcement Learning (RL)

Reinforcement learning (RL) studies how an agent can learn to act optimally through trial and error, with formal foundations rooted in Markov decision processes and dynamic programming in the 1950s, including the Bellman equation and early solution methods such as policy iteration (Bellman, 1957a,b; Howard, 1960; Lovejoy, 1991; Whittle, 1982; Bryson, 1996). Building on this foundation, RL matured into a distinct field with core model-free learning algorithms such as temporal-difference learning and Q-learning, and with successful combinations with function approximation (Sutton, 1988; Watkins, 1989; Bertsekas and Tsitsiklis, 1996; Sutton, 1984, 1988). In the past decade, deep neural networks enabled RL to scale to high-dimensional state spaces, producing breakthroughs such as DQN (Mnih et al., 2015), AlphaGo (Silver et al., 2016), PPO (Schulman et al., 2017), GT Sophy (Wurman et al., 2022), and GRPO (Guo et al., 2025).

As RL is a broad field, this section does not cover the entire field. Instead, we focus on two limitations that this thesis targets: low sample efficiency and poor generalization. The following

Sections 9.3-9.6 then review additional RL topics that are directly relevant to this thesis.

9.1.1 Sample Efficiency

A longstanding challenge in reinforcement learning is sample efficiency. Many RL methods are notoriously data-hungry. Agents often require millions of interactions with the environment to reach acceptable performance, which can make RL expensive or infeasible in real-world settings such as physical robotics. Recognizing this issue, a large body of prior work focuses on identifying the causes of poor sample efficiency and proposing methods to address them. Below we outline major directions and representative contributions.

Experience Replay and Off-Policy RL Even within standard model-free RL, algorithms that reuse past experience can be more sample-efficient than on-policy algorithms that discard data after a single update. Experience replay was introduced by Lin (1992), which stores past transitions and replays them to the learning algorithm multiple times, so each costly interaction can be reused. DQN (Mnih et al., 2015) makes experience replay a default component by combining a replay buffer with deep function approximation, showing that replayed transitions can make learning from high-dimensional observations (e.g., pixels) practical and stable. Following DQN, most off-policy algorithms (Silver et al., 2014; Haarnoja et al., 2018; Fujimoto et al., 2018) learn from previously collected data and often achieve higher sample efficiency than on-policy algorithms (Schulman et al., 2015, 2017). Later work developed more sophisticated replay schemes. Prioritized experience replay samples high-error transitions more often to speed learning (Schaul et al., 2016; Brittain et al., 2019; Novati and Koumoutsakos, 2019; Zha et al., 2019; Sun et al., 2020), and hindsight experience replay relabels past episodes with achieved goals to learn effectively from sparse rewards (Andrychowicz et al., 2017; Zhao and Tresp, 2018; Ren et al., 2019). More recent work goes further by learning how to replay (e.g., learning a replay policy), systematically analyzing replay design choices (e.g., buffer size and replay ratio) and their impacts, and studying replay in continual learning settings (Zha et al., 2019; Fedus et al., 2020; Fujimoto et al., 2020; Sinha et al., 2022; Schmitt et al., 2020; Rolnick et al., 2019).

Model-Based Reinforcement Learning (MBRL) Compared to model-free RL, incorporating explicit models of the environment can improve sample efficiency by enabling the agent to simulate experience. We discuss relevant MBRL work in Section 9.3.

State Abstractions, Temporal Abstractions, and Intrinsic Rewards Another cause of poor sample efficiency is ineffective exploration, where agents waste many samples on uninformative or repetitive experience. Research on state abstractions, temporal abstractions, and intrinsic motivation addresses this issue by reducing the exploration space and providing learning signals that guide exploration. We discuss relevant work on these topics in Sections 9.4-9.6.

Transfer Learning and Meta-Learning A different approach to reducing sample complexity is to leverage prior knowledge from other tasks or from the agent's own past experience. Transfer learning reuses knowledge (e.g., policies, representations, dynamics models, or value estimates) learned in a source task to accelerate learning in a related target task, thereby reducing redundant relearning. Early transfer methods include probabilistic policy reuse, which mixes previously learned policies to guide behavior in a new task (Fernández and Veloso, 2006), and inter-task mappings, which align state and action features across tasks so that value updates in the target can be bootstrapped from source experience (Taylor et al., 2007). One representative line of work transfers value-relevant structure across tasks, for example via successor features that decouple dynamics from rewards, allowing an agent to adapt to new reward functions with little additional data (Barreto et al., 2017). Another line of work transfers deep parameters and representations, ranging from policy distillation that compresses and merges expert policies into a single network (Rusu et al., 2016a) to progressive networks that reuse frozen features through lateral connections to speed learning on new tasks while mitigating interference (Rusu et al., 2016b). Other directions include meta-learning for rapid adaptation by learning initializations that fine-tune in a few gradient steps (Finn et al., 2017; Duan et al., 2016).

RL from Demonstrations and Offline RL Compared to relying only on online interactions, reinforcement learning from demonstrations (RLfD) and offline RL both aim to improve sample

efficiency by leveraging previously collected data (expert demonstrations or logged behavior trajectories) to provide an initialization or learning signals (Ramírez et al., 2022; Prudencio et al., 2023). Early RLfD methods learn from expert behavior either by inferring a reward or objective that explains demonstrations or by using demonstrations to guide exploration and value learning, thereby reducing the search space (Abbeel and Ng, 2004; Ramírez et al., 2022). A representative line of RLfD work injects demonstrations directly into deep RL updates, for example by initializing the replay buffer with expert transitions or adding auxiliary imitation-style losses to warm up off-policy learning (Hester et al., 2018; Vecerík et al., 2017). A representative line of offline RL addresses distribution shift and extrapolation error by constraining policy improvement to stay close to the dataset support or by learning conservative (pessimistic) value estimates, which reduces overestimation on out-of-distribution actions (Fujimoto et al., 2019; Kumar et al., 2019, 2020; Kostrikov et al., 2022). Additional directions include model-based offline RL that penalizes model uncertainty to limit error when generalizing beyond the dataset, and offline-to-online fine-tuning frameworks that use offline data to bootstrap fast online learning (Yu et al., 2020; Nair et al., 2020; Chen et al., 2021; Prudencio et al., 2023).

In summary, sample efficiency is critical for applying RL to real-world problems, and prior work approaches it from many angles. These developments enable contemporary RL algorithms to learn effective policies with far fewer interactions than early or naive methods, although closing the gap to human-level efficiency remains an active research frontier.

9.1.2 Generalization

Another issue in reinforcement learning is poor generalization, i.e., an agent's inability to apply learned knowledge to new, unseen situations. Early RL successes often involved training and testing in the same environment, but policies learned by deep RL can overfit to their training environments and fail when even slight changes are introduced. Cobbe et al. (2020b) illustrate this generalization gap: agents trained on certain procedurally generated video game levels perform well on the training levels but poorly on held-out test levels, suggesting they effectively memorize level layouts rather than learn general skills. Prior work also shows that deep RL policies can be

brittle to adversarial or out-of-distribution inputs. For example, small perturbations to observations can significantly degrade an agent's performance, indicating a lack of robustness to even minor changes in the input data (Huang et al., 2017; Kos and Song, 2017; Behzadan and Munir, 2017). These findings highlight that reliable generalization in RL remains a critical challenge, especially for deploying agents in unpredictable real-world settings. In response, a growing body of work focuses on diagnosing the causes of poor generalization in RL and developing techniques to address them. We summarize key approaches below.

Diverse Environments Training on diverse environments aims to learn policies that perform well across a distribution of scenarios, increasing the likelihood of generalization to unseen environments compared to policies trained in a single environment. A representative benchmark-driven line of work uses procedural generation to create large train/test splits of levels (e.g., CoinRun and its extension Procgen), showing that scaling the number of distinct training environments reduces overfitting and improves performance on unseen levels (Cobbe et al., 2019, 2020a). A second line of work targets sim-to-real transfer via domain and dynamics randomization, where policies are trained under randomized textures, lighting, camera parameters, and physical dynamics so that the real world becomes another draw from the training distribution, enabling zero-shot or low-shot deployment (Tobin et al., 2017; Sadeghi and Levine, 2017; Peng et al., 2018; OpenAI et al., 2019). Related directions include robustness objectives that optimize worst-case performance over subsets of an environment collection (Rajeswaran et al., 2017).

Data Augmentation and Function Regularization Data augmentation and function regularization facilitate generalization and sample efficiency by injecting inductive bias (e.g., invariances or smoothness), encouraging policies to perform well beyond the exact training observations (Ma et al., 2025; Korkmaz, 2024). Early work showed that simple image augmentations could be integrated into off-policy pipelines, reducing overfitting to spurious visual details and improving data efficiency (Laskin et al., 2020; Kostrikov et al., 2021; Igl et al., 2019). Subsequent methods tune, select, or restructure augmentations to better match the task structure, for example by automatically searching augmentation policies or mixing observations (Raileanu et al., 2021; Wang et al., 2020;

Hansen and Wang, 2021). Function regularization instead constrains the learned value functions and policies through objective-level or architecture-level inductive biases, such as information bottlenecks and noise injection, treating smaller discount factors as an explicit regularizer, modifying temporal difference targets with action likelihood as reward augmentations, or stabilizing large networks via dropout-based ensembles, so that policies learned from limited data become smoother and more robust (Igl et al., 2019; Amit et al., 2020; Vieillard et al., 2020; Hiraoka et al., 2022).

Adversarial Methods for RL Generalization Adversarial methods study how worst-case perturbations (to observations, dynamics, or opponents) can break learned policies, and use such perturbations during training to facilitate robustness and generalization. One representative direction constructs adversarial perturbations on agent observations to expose brittle decision boundaries in deep RL and motivates defenses against small but harmful input shifts (Huang et al., 2017; Pan et al., 2022). A second direction adopts minimax formulations, where an adversary applies disturbances or selects hard environment instances while the agent learns a policy that performs well under worst-case conditions (Pinto et al., 2017; Rajeswaran et al., 2017; Pattanaik et al., 2018). A third line of work considers multi-agent adversaries, where an attacker learns an adversarial policy that induces natural but challenging state distributions, uncovering failures without requiring direct access to the agent's observations and encouraging robustness against strategic opponents (Gleave et al., 2020).

In summary, generalization in RL aims to ensure that a learned policy is not merely a brittle solution for a narrow training scenario, but instead captures patterns that transfer to unseen settings. Prior work shows that RL agents can substantially overfit. Despite that this issue can be mitigated by the techniques described above, achieving reliable generalization in RL remains challenging, especially when there are large differences between training and test environments. In this thesis, as described in Chapters 3 and 4, we study generalization from a causality perspective, using causal discovery to reduce spurious correlations and support robust predictions under distribution shift.

9.2 Causality

This section provides an overview of causality, which is another central theme of this thesis in addition to RL. In particular, we review foundational work and representative developments in causal modeling, causal discovery, actual causality, causal RL, and causal representations.

Causality Early ideas about causality trace back to controlled experimentation and structural explanations of dependence, including path analysis that links observed correlations to directed causal influences (Wright, 1921; Fisher, 1935). Causality was later formalized through modern mathematical frameworks for counterfactual reasoning, including the potential-outcomes view (Rubin, 1974), structural causal models and causal diagrams with do-calculus (Pearl, 1995), and constraint-based causal discovery with graphical models (Spirtes et al., 1993). Subsequent developments expanded both methodology and scope, with work on external validity and transportability across populations (Pearl and Bareinboim, 2014), mature treatments that integrate identification and estimation in practice (Hernán and Robins, 2010; Peters et al., 2017), and efforts to connect causality with modern ML through causal representation learning and invariance-based generalization (Schölkopf et al., 2021).

Actual Causality Actual causality studies whether a particular event in a specific situation causes another event, building on philosophical analyses of causal claims (e.g., but-for counterfactual reasoning) and their pitfalls (Lewis, 1973; Mackie, 1980). A widely used formalization of actual causality models the world with structural causal models and defines an actual cause through counterfactual reasoning (Halpern and Pearl, 2005; Chockler and Halpern, 2004). This definition also supports quantitative notions such as degrees of responsibility. Subsequent work expands these definitions, proposes alternative criteria and variants, and applies actual causality to explanation tasks such as attributing responsibility for query answers, including why-so and why-not explanations (Meliou et al., 2009; Halpern, 2016; Kueffner, 2021).

Causal Discovery Traditional causal discovery infers a causal graph from data by combining assumptions (e.g., causal Markov conditions, causal faithfulness, causal sufficiency, and explicit

handling of hidden confounders) with statistical tests or scoring objectives, and it largely falls into three families:

- **Constraint-based methods.** These approaches recover graph structure by testing conditional independencies implied by a directed acyclic graph and then orienting edges using sound logical rules. Representative algorithms include the Peter-Clark algorithm, which outputs a partially directed acyclic graph that represents the Markov equivalence class under causal sufficiency, and the Fast Causal Inference algorithm, which extends this line of work to allow hidden confounders and outputs a partial ancestral graph (Spirtes et al., 2000). Recent work also leverages heterogeneous environments or distribution shifts as additional constraints to identify directions beyond an independent and identically distributed (i.i.d.) equivalence class, for example under the sparse mechanism shift hypothesis (Perry et al., 2022).

- **Score-based methods.** These methods search over graph structures to maximize likelihood with a complexity penalty (e.g., Bayesian information criterion or minimum description length), with Greedy Equivalence Search (Chickering, 2002) as a representative algorithm. More recent work extends score-based discovery to deep learning, including continuous optimization with an acyclicity constraint (NOTEARS) and neural score models, as well as differentiable Bayesian and amortized inference approaches that learn to predict graphs directly (Zheng et al., 2018; Lachapelle et al., 2020; Lorch et al., 2021, 2022). Another complementary line of work uses reinforcement learning as a search strategy over graphs or variable orderings, optimizing a reward function that combines a score with acyclicity and regularization terms (Zhu et al., 2020a; Wang et al., 2021b).

- **Functional causal model methods.** This family identifies causal dependencies by assuming an asymmetric data generation mechanism, such as additive-noise models where only the correct causal direction matches the independent noise assumption (Peters et al., 2014). Recent extensions improve scalability and realism, for example by using score matching to recover nonlinear additive-noise models efficiently, handling missing data, or extracting causal relationships from identifiable representation learning objectives (e.g., Jacobian-based

nonlinear independent component analysis) (Rolland et al., 2022; Gao et al., 2022; Reizinger et al., 2023).

The causal discovery methods used in Chapters 3 and 4 belong to the family of constraint-based methods. Compared to score-based methods, for which it can be difficult to balance regularization and prediction accuracy (especially for rare dependencies), we find that constraint-based methods better capture crucial dependencies from limited data.

Granger Causality Granger causality defines a directed relation between time series based on predictability: a process X "Granger-causes" Y if past values of X improve the prediction of future Y compared to only using the past of Y (Granger, 1969). Classical Granger causality often assumes linear vector autoregressive models and conducts tests on model coefficients in bivariate settings (Geweke, 1982), and later work extended Granger causality to more complex setups. Some work studies multivariate settings and conditions the test on other observed series to mitigate confounding (Basu et al., 2015), while other work introduces graphical interpretations that represent Granger relations with dynamic path diagrams to make temporal dependencies easier to analyze (Eichler, 2007). Recent advances further relax modeling assumptions by extending Granger analysis to nonlinear models (e.g., neural networks) (Marinazzo et al., 2008; Tank et al., 2022), developing tests that work with irregular or unevenly sampled observations (Heerah et al., 2021), and supporting mixed-frequency settings where variables have different sampling rates (Ghysels et al., 2016).

A major distinction between causality and Granger causality is that Granger causality focuses on time series and identifies relationships via incremental predictability from past values (Granger, 1969; Eichler, 2007), whereas causal discovery focuses on the data generation process and aims to identify interventional or counterfactual relationships under stronger structural assumptions (Pearl, 2009; Peters et al., 2017; Stokes and Purdon, 2017). Regarding the causal discovery methods introduced in Chapters 3 and 4, although their testing form (whether $\mathcal{S}_t^i$ helps predict $\mathcal{S}_{t+1}^j$) resembles Granger causality, under Assumptions 1-6 we show that they identify causal relationships rather than the weaker notion of Granger relationships.

Causal Representation Learning (CRL) Causal representation learning aims to learn latent variables that correspond to the underlying causal factors of a data generation process (rather than arbitrary features), so that the learned representation supports interventions, modular reuse, and robust generalization under distribution shifts (Schölkopf et al., 2021; Deng et al., 2025). Early work in this area focused on learning disentangled representations using generative models with inductive biases (Chen et al., 2016; Higgins et al., 2017; Kumar et al., 2018; Chen et al., 2018; Kim and Mnih, 2018), but purely unsupervised disentanglement is not identifiable without additional assumptions or supervision, motivating explicitly causal formulations (Locatello et al., 2019). One representative line of work exploits signals beyond i.i.d. observations, such as auxiliary variables, weak supervision, multiple environments, or interventional regimes, connecting CRL to nonlinear independent component analysis and invariance principles (Khemakhem et al., 2020; Arjovsky et al., 2019; Brehmer et al., 2022; Ahuja et al., 2023; Lippe et al., 2023). A second major line of work builds causal structure directly into the latent generative model by parameterizing a structural causal model over latent factors (often with a learned causal graph) and training end to end to obtain causally meaningful, manipulable representations (Yang et al., 2021; Shen et al., 2022; Brehmer et al., 2022; Lippe et al., 2023; Deng et al., 2025). More recently, CRL explores alternative generative backbones beyond variational autoencoders, for example using diffusion models (Karimi Mamaghan et al., 2024).

Despite recent progress, CRL remains a challenging research frontier. In Chapter 7, we instead focus on the simpler setting of learning object-centric representations, and we review related work in Section 2.7.

Causality in Reinforcement Learning Incorporating causality into reinforcement learning has received growing interest in recent years due to its potential benefits for generalization, like learning generalizable representations (Zhang et al., 2019b, 2020; Sontakke et al., 2021; Volodin et al., 2020; Tomar et al., 2021; Fu et al., 2021; Zhang et al., 2019a; Huang et al., 2022b). It can also facilitate policy learning by enabling intervention-aware decision making and counterfactual reasoning (Buesing et al., 2019; Nair et al., 2019; Mozifian et al., 2020; Lyle et al., 2021; Seitzer et al., 2021; Lu et al., 2020; Sonar et al., 2021; Huang et al., 2022a; Pitis et al., 2020; Liao et al.,

2024). Finally, causal inductive biases can support dynamics learning by encouraging structured, mechanism-level modeling that is reusable across tasks and environments (Li et al., 2020; Zhu et al., 2025).

9.3 Model-Based Reinforcement Learning (MBRL)

Model-based reinforcement learning has a long history, ranging from early integrated architectures that interleave learning, planning, and acting, such as Dyna (Sutton, 1991, 1990), to optimal-control and planning methods that leverage learned or analytic dynamics models, such as iterative linear quadratic regulators and LQR-Trees (Li and Todorov, 2004; Tedrake, 2009), as well as data-efficient probabilistic model-based policy search, such as PILCO (Deisenroth and Rasmussen, 2011). Modern MBRL methods often follow the same high-level strategy. They learn a dynamics model (often together with a reward predictor) from collected trajectories and then use the learned model for planning and control (Williams et al., 2017; Chua et al., 2018; Nagabandi et al., 2018; Li and Todorov, 2004; Tedrake, 2009; Deisenroth and Rasmussen, 2011; Du et al., 2020; Janner et al., 2022), generating imagined experience or counterfactual rollouts that support policy learning and generalization (Sutton, 1991; Kurutach et al., 2018; Janner et al., 2019; Pitis et al., 2022; Urpí et al., 2024; Venkatesh et al., 2024; Ha and Schmidhuber, 2018; Hafner et al., 2020, 2021, 2025; Micheli et al., 2023, 2024; Cohen et al., 2024; Nagabandi et al., 2018), or providing model-based targets for Q-value estimation (Feinberg et al., 2018; Amos et al., 2021).

Despite strong empirical performance, many methods use dense dynamics models in which each state factor at the next time step depends on all current state factors and the action. Such dense dependencies are vulnerable to spurious correlations and can generalize poorly under distribution shift or out-of-distribution states. Moreover, reward predictors often share similar dense architectures, which can further exacerbate these issues. As described in Chapters 3 and 4, CDL and CBM address this limitation by leveraging causal discovery and retaining only necessary causal dependencies in the model. A closely related line of work designs structured dynamics models that encourage modularity and sparsity in how factors interact (Goyal et al., 2021c,b,a; Liu et al., 2023; Feng and Magliacane, 2023; Zhao et al., 2022; Sehgal et al., 2024; Baek et al., 2025). These

structural biases can support generalization by reducing unnecessary dependencies among state factors. However, without explicitly modeling causal relationships, sparse dependencies can still learn spurious correlations and generalize poorly.

Implicit Models for Dynamics Learning Implicit models (Teh et al., 2003; Welling and Hinton, 2002) appear broadly across machine learning, including image generation (Du and Mordatch, 2019), natural language processing (Bakhtin et al., 2021; He et al., 2021), and density estimation (Saremi et al., 2018; Song et al., 2020). One practical appeal is that implicit formulations can unify probabilistic and deterministic views of prediction and inference (LeCun et al., 2006; Song and Kingma, 2021). In reinforcement learning (and closely related imitation learning), implicit models have been used for policies (Florence et al., 2022), value functions (Haarnoja et al., 2017), and dynamics models (Pfrommer et al., 2020; Wang et al., 2021a). In robotics settings with discontinuous contact, implicit approaches can better capture sharp transitions, whereas explicit models can incorrectly interpolate across discontinuity boundaries when data are finite (Pfrommer et al., 2020; Florence et al., 2022), which motivates CBM to adopt implicit dynamics models in Chapter 4. Beyond learning dynamics accurately, implicit energy-based models also support planning directly in model space (Du et al., 2020), and diffusion-based generative models provide a way to jointly learn dynamics and perform sampling-based planning (Janner et al., 2022).

9.4 State Abstraction

State abstraction maps a large state space to a smaller abstract space that preserves decision-relevant information. By removing irrelevant information, state abstractions can facilitate RL sample efficiency and generalization by reducing the learning space. When an abstraction (approximately) preserves the reward and dynamics information that matters for control, planning or learning in the abstract space can yield near-optimal behavior in the original environment.

Classical state abstraction methods decide which states to aggregate based on equivalence or similarity over the MDP model, value functions, or optimal actions. Bisimulation uses a strict model equivalence criterion, aggregating states only when their reward and transition functions

match under every action, yielding an exact abstract MDP (Givan et al., 2003). Homomorphisms generalize model equivalence by allowing a structure-preserving mapping with flexible action matching, which can capture symmetries and other relational structure while remaining exact when the homomorphism conditions hold (Ravindran and Barto, 2003). Approximate bisimulation and bisimulation metrics relax exact model equivalence by permitting bounded deviations, producing a MDP with bounded errors that is suitable for approximate planning (Dean et al., 1997a; Ferns et al., 2004). Stochastic dynamic programming in factored MDPs builds aggregation trees and groups states that have equivalent transition and reward functions under a fixed policy, and can be viewed as a special case of model equivalence abstractions (Boutilier et al., 2000a; Givan et al., 2003). The G algorithm performs online, statistically tested aggregation by grouping states with equivalent rewards and Q-values for each action, under assumptions that allow testing each feature's relevance independently (Chapman and Kaelbling, 1991). Utile distinction methods also use statistical tests online, merging states that share the same optimal action and have similar Q-values for those actions (McCallum, 1996). Policy irrelevance abstractions group states solely by having the same optimal action, which can be coarse and may not preserve the optimal policy for the ground MDP (Jong and Stone, 2005).

Modern work extends these principles to learning approximate abstractions and task-relevant representations from high-dimensional observations. Abel et al. (2016) analyzed multiple approximate abstraction families and derived near-optimality bounds for policies computed under these abstractions, and Abel (2019) further developed a broader theory of abstraction in RL. In representation learning, DeepMDP learns a continuous latent model by matching rewards and latent transition distributions, connecting latent model learning to abstraction-style guarantees (Gelada et al., 2019). Zhang et al. (2021a) developed invariant representations that discard task-irrelevant visual distractors without reconstruction, using a bisimulation-inspired latent geometry, and MICo provides a scalable sampling-based state similarity objective for shaping deep RL representations (Castro et al., 2021). Task-informed abstraction separates task-relevant factors from nuisance variation using reward signals (Fu et al., 2021), while Denoised MDPs further splits features based on controllability and reward relevance while filtering distractors (Wang et al., 2022a). Action-Sufficient State Representations aim to learn minimal representations that remain sufficient for optimal control under

structural constraints, emphasizing sufficiency rather than reconstruction fidelity (Huang et al., 2022b). Recent extensions adapt these ideas to broader control settings, including goal-conditioned bisimulation (Hansen-Estruch et al., 2022), model-invariant abstractions for model-based RL (Tomar et al., 2021), and learning Markov abstractions (Allen et al., 2021).

The state abstractions introduced by CBM in Chapter 4 are closely related to bisimulation. However, in contrast to classical bisimulation that derives abstractions by merging states, CBM identifies task-irrelevant state factors and derives a state abstraction by ignoring those factors.

9.5 Intrinsic Reward Functions

The idea of using intrinsic rewards to facilitate exploration in reinforcement learning has long been studied and traces back to early work on curiosity and "model-building" agents (Schmidhuber, 1991). In this section, we discuss intrinsic reward methods that are related to ELDEN introduced in Chapter 5.

Curiosity-Based Intrinsic Reward Functions Curiosity-based methods reward an agent for visiting "novel" or "surprising" states, where different methods define novelty in different ways. The first family uses (pseudo-)counts to measure state novelty (Brafman and Tennenholtz, 2002; Kearns and Singh, 2002). In particular, Bellemare et al. (2016) derived intrinsic reward from pseudo-counts estimated by a density model, and Ostrovski et al. (2017) extended this direction with neural density models for high-dimensional observations. Other approaches discretize or generalize counting through representations, such as locality-sensitive hashing for approximate visitation counts (Tang et al., 2017), feature space pseudo-counts that reuse value function features (Martin et al., 2017), and successor-representation-based counting that propagates novelty through expected future occupancies (Machado et al., 2020). Relatedly, Lee et al. (2019) reframed exploration as matching a target state distribution and shows that many intrinsic reward heuristics fit this objective.

The second family measures novelty through prediction errors or uncertainties in learned predictive models. Early approaches trained forward models and use prediction errors as intrinsic rewards (Stadie et al., 2015), while later methods designed representations to handle uncontrollable

noise, for example by measuring prediction error in an inverse dynamics feature space (Pathak et al., 2017) or by using fixed random features and rewarding the agent when a predictor fails to match them (Burda et al., 2019b). Uncertainty-aware variants replace raw prediction error with disagreement or information gain, such as rewarding information gain about dynamics parameters (Houthooft et al., 2016), surprise-based bonuses that encourage experiences that change an agent's beliefs (Achiam and Sastry, 2017), and optimizing the policy toward transitions that maximize ensemble prediction disagreement (Hester and Stone, 2013; Shyam et al., 2019; Pathak et al., 2019). ELDEN is closely related to these curiosity-based methods, but differs in what it considers "novel". Rather than rewarding novelty of raw states or next state prediction errors, it rewards novelty in the local dependencies between state factors, aiming to guide exploration toward new interactions rather than toward inaccurate prediction.

Empowerment-Based Intrinsic Reward Functions Empowerment-based methods adopt a different notion of what makes a state "interesting" from curiosity-based methods. They formalize interest as control, typically via mutual information between an agent's actions and the resulting future states, meaning that states where the agent has distinct, reliably achievable outcomes are valuable for exploration (Klyubin et al., 2005b; Salge et al., 2014). Later work adopts variational lower bounds that make mutual information estimation tractable in high-dimensional settings (Mohamed and Rezende, 2015; Gregor et al., 2017), and also develops formulations from a goal-conditioned RL perspective (Levy et al., 2023; Levy, 2025). Recent work extends this perspective to influence and controllability in multi-entity environments, for example by maximizing mutual information between an agent-centric state and surrounding states (Zhao et al., 2021) or by rewarding states where the agent's action has high state-dependent causal influence as measured via conditional mutual information (Seitzer et al., 2021). A limitation of empowerment-style objectives is that, when approximated myopically (e.g., with short-horizon or one-step estimates), they can bias exploration toward the easiest-to-control state factors and miss indirect, complex behaviors such as tool use. ELDEN is related to causal-influence-based intrinsic rewards (Seitzer et al., 2021) but differs in two key ways. It further models dependencies between state factors so that it can

encourage indirect interactions, and it prioritizes novel interaction patterns rather than maximizing controllability, reducing the tendency to repeatedly manipulate a single easy-to-control object.

9.6 Unsupervised Skill Discovery (USD)

SkiLD introduced in Chapter 6 falls within unsupervised skill discovery, where an agent learns a set of reusable skills during a reward-free pretraining phase. A common formulation conditions the policy on a latent skill variable z and learns diverse, distinguishable behaviors under different values of z. When solving downstream tasks, the learned skills serve as temporally extended actions in an options-style hierarchy (Sutton et al., 1999), where a high-level policy learns to select skills based on the task reward (and some methods also fine-tune the skills), which improves sample efficiency compared to using primitive actions.

We organize prior USD work into four major lines.

- **Skill discovery based on mutual information.** These methods maximize mutual information between the skill variable and visited states, transitions, or termination states so that different skills induce distinguishable behaviors (Florensa et al., 2017; Achiam et al., 2018; Lee et al., 2019; Liu and Abbeel, 2021a; Zhang et al., 2021b; Campos et al., 2020; Kim et al., 2021; Laskin et al., 2022). Variational Intrinsic Control maximizes mutual information between options and their termination states, linking skill discovery to empowerment (Gregor et al., 2017). DIAYN similarly maximizes a mutual information objective so that skills are identifiable from visited states, thereby producing diverse behaviors (Eysenbach et al., 2019). Dynamics-aware variants, such as DADS, encourage skills whose outcomes are predictable under a skill-conditioned dynamics model and support planning in the learned latent space (Sharma et al., 2020).

- **Skill discovery based on state coverage.** This line of work focuses on covering the reachable space (or a learned representation space), aiming to avoid a skill set that is diverse but narrow (Liu and Abbeel, 2021b). EDL studies forward mutual information and reverse mutual information skill objectives from a coverage perspective and proposes an optimization

procedure that directly promotes state-covering skills (Campos et al., 2020). To avoid learning skills that are limited to simple behaviors, recent work replaces mutual information objectives with rewards that encourage large travel distances in the state space, such as LSD (Park et al., 2022) and CSD (Park et al., 2023). METRA extends those methods to image observations by learning a compact latent space that is metrically connected to the state space via temporal distances and then learning to cover this latent space (Park et al., 2024b).

- **Goal-conditioned RL.** A closely related line of work views "skills" as goals and learns goal-conditioned policies that can be reused and composed, which aligns naturally with hierarchical control and downstream task learning (Liu et al., 2025; Eysenbach et al., 2024; Myers et al., 2024, 2025; Durugkar et al., 2021; Durugkar, 2023; Ma et al., 2022; Agarwal et al., 2023). In unsupervised goal-conditioned RL, the agent generates its own goals and or learns its own goal metric, such as DISCERN (Warde-Farley et al., 2019), RIG (Nair et al., 2018), and Skew-Fit (Pong et al., 2020). A further extension connects goal-conditioned RL to representation learning via contrastive objectives, starting from classifier-based views of goal reaching (Eysenbach et al., 2021) to formalizing how an InfoNCE objective yields a goal-conditioned value function (Eysenbach et al., 2022). Building on this contrastive perspective, subsequent work develops automatic waypoint curricula for long-horizon goals (Zhang et al., 2022b), derives temporal difference objectives that support goal-conditioned control (Zheng et al., 2024b), and demonstrates stable offline-to-real robotic goal reaching with contrastive self-supervision (Zheng et al., 2024a).

- **RL basis functions.** This line of work aims to learn task-agnostic basis functions that separate environment dynamics from downstream rewards, so that many new tasks can be solved by fitting a small set of weights at test time with a small number of interactions. Successor feature approaches are a representative family of work in this line, but their effectiveness depends strongly on the choice of the underlying "elementary" features, motivating geometry-aware bases such as Laplacian eigenfunctions (Wu et al., 2019). Forward-backward (FB) representations take a closely related occupancy factorization view and address this sensitivity by jointly learning both elementary and successor components, enabling near-optimal policies

for arbitrary rewards specified at test time (Touati and Ollivier, 2021; Touati et al., 2023). Recent work further expands basis function learning to long-horizon, high-dimensional settings (Park et al., 2024a; Agarwal et al., 2025; Sikchi et al., 2025).

Unlike prior work that primarily focuses on the state space or the goal space, SkiLD introduced in Chapter 6 focuses on evaluating whether state factors interact by identifying local causal dependencies and then learning how to induce such interactions. Compared to prior work that must learn a huge (combinatorial) number of skills to cover a large factored space, SkiLD targets skill learning in the smaller space of state factor interactions, which also often aligns better with downstream tasks in many environments.

9.7 Object-Centric Representations

Object-centric representation learning decomposes a scene into a set of object-level features (often called slots), rather than encoding an observation as a single monolithic vector. By imposing a permutation-invariant, compositional structure, object-centric representations facilitate generalization, enable object-level reasoning and manipulation, and provide a natural interface for modeling object interactions.

Early unsupervised generative approaches such as MONet (Burgess et al., 2019) and IODINE (Greff et al., 2019) achieve multi-object decomposition via attention and iterative amortized inference, and GENESIS (Engelcke et al., 2020) extends this direction with structured scene generation. Slot Attention (Locatello et al., 2020) then introduces a simple bottleneck that serves as the foundation for many subsequent methods that learn object-centric features in an unsupervised manner from images (Seitzer et al., 2023; Wu et al., 2023c; Jiang et al., 2023; Wu et al., 2023a) or videos (Kipf et al., 2022; Elsayed et al., 2022; Aydemir et al., 2023). To learn high-quality object-centric representations in real-world environments, subsequent work leverages features from pretrained encoders (Seitzer et al., 2023; Aydemir et al., 2023; Zadaianchuk et al., 2023; Manasyan et al., 2025), strengthens object binding (Fan et al., 2024; Singh et al., 2024), enhances identifiability of learned slots (Kori et al., 2024), and adopts high-fidelity decoders for object-centric

generation (Wu et al., 2023c; Singh et al., 2025; Akan and Yemez, 2025). Finally, object-centric representations increasingly serve as building blocks for world models, latent action models, and RL policies, where object-level representations support long-horizon prediction and generalizable decision making (Baek et al., 2025; Mosbach et al., 2025; Jiang et al., 2024; Yoon et al., 2023).

Dyn-O described in Chapter 7 builds on the SOLV framework (Aydemir et al., 2023), introducing improvements to more accurately extract object-centric representations with the guidance of segmentation masks. Another closely related approach is the Slot State Space Model (SlotSSM) (Jiang et al., 2024), which uses state-space models to capture long-range temporal dependencies in a structured way. However, SlotSSM has only been validated on domains with limited complexity, such as basic shapes, and it tightly couples slot representation learning and dynamics modeling by training them jointly. In contrast, Dyn-O first learns a strong slot feature encoder and then builds object-centric world models on top of it.

9.8 Latent Action Models (LAMs)

Given the abundance of videos and the scarcity of action labels, several methods learn dynamics and control from pure observations. ILPO (Edwards et al., 2019) learns a latent policy together with a forward dynamics model, and then maps the latent policy output to real actions through an action remapping network. LAPO (Schmidt and Jiang, 2024) and Genie (Bruce et al., 2024) jointly learn an inverse dynamics model with a forward dynamics model. Nikulin et al. (2025) then introduce a small portion of ground-truth actions as supervision for dynamics modeling. Ye et al. (2025) expands learning from observations from vision only to vision and language modalities. Prior work (Menapace et al., 2021; Zhang et al., 2022a; Ye et al., 2023; Baker et al., 2022; Ye et al., 2025; Chen et al., 2024) demonstrates the value of dynamics modeling without action labels in applications such as learning to drive, play games, and manipulate with robot arms from videos.

However, many domains include multiple entities with independent actions, and prior work struggles to model such complex action combinations, motivating FLAM introduced in Chapter 8 that addresses this challenge through factorization. Two methods that are closely related to FLAM are PlaySlot (Villar-Corrales and Behnke, 2025) and LAPO-slot (Klepach et al., 2025), which follow

a two-stage procedure: object-centric representations are first learned in isolation, and a latent action model is then trained on top of these fixed representations. As a result, the learned slots largely mirror conventional object-centric representations and are primarily organized according to visual appearance. In contrast, FLAM jointly optimizes slot extraction and the latent action model with the prediction loss, with each slot endowed with an independent latent action in both the inverse dynamics model and the forward dynamics model. This end-to-end factorized dynamics learning encourages the model to separate entities based on the independence of their dynamics rather than visual similarity, yielding action-driven representations.

9.9 Summary

Within this growing body of research on RL sample efficiency and generalization, this thesis fills several gaps:

- Regarding state abstractions, while prior work mainly derives abstractions by merging states, CDL and CBM instead identify task-irrelevant state factors and derive state abstractions by ignoring those factors.
- Regarding intrinsic rewards and unsupervised skill discovery, while prior work mainly focuses on visiting novel states or learning skills to reach novel states, ELDEN and SkiLD instead target the space of state factor interactions and learn to induce novel interactions.
- Regarding representation learning, unlike prior work, Dyn-O disentangles static and dynamic features to enable generating trajectories with novel appearances. Meanwhile, no prior work factorizes representations based on the independence of entity actions in the way FLAM does.

Nonetheless, even with these contributions, there remains substantial room for new RL research to narrow the gap between RL agents and humans. The next chapter outlines several potential future directions and summarizes the contributions of this thesis.

Chapter 10: Conclusion and Future Work

This chapter summarizes the contributions in this thesis and outlines several directions for future research.

10.1 Summary of Contributions

This thesis presents six methods that address the question in Section 1.1: How can a reinforcement learning agent reason about and leverage causal relationships in the environment's dynamics and reward function to learn generalizable, sample-efficient policies? The four main contributions are summarized as follows.

1. **State Abstractions from General Causal Dependencies.** RL agents often learn from an unnecessarily large state space in which many state factors are irrelevant to the task. Relying on irrelevant factors causes dynamics models and policies to learn spurious correlations and to generalize poorly to unseen scenarios. To address this issue, CDL introduced in Chapter 3 and CBM introduced in Chapter 4 learn to identify causal relationships in transition dynamics and reward functions. Based on these causal dependencies, they derive state abstractions that ignore irrelevant state factors and reduce the effective learning space, thereby facilitating sample efficiency and generalization in RL.

2. **Intrinsic Reward Functions from Actual Causal Dependencies.** For many real-world tasks, only a sparse reward signal based on task success or failure is available, which hinders efficient exploration. To address this issue, ELDEN introduced in Chapter 5 identifies actual causal dependencies between state factors and uses uncertainty about these dependencies as an intrinsic reward to facilitate exploration. This intrinsic reward function encourages the agent to induce novel interactions between state factors, enabling efficient exploration in the space of factor interactions rather than in the vast state space in which most states are uninformative for solving the task.

3. **Skill Discovery from Actual Causal Dependencies.** Another RL challenge arises from the temporal dimension. For tasks with long horizons, the number of possible action sequences grows exponentially, making it increasingly difficult to discover action sequences that accomplish a particular task. To address this issue, SkiLD introduced in Chapter 6 uses the same intrinsic motivation as ELDEN but leverages it to learn diverse skills, where each skill is trained to induce a specific actual dependency. During downstream task learning, using these learned skills in place of primitive actions shortens the effective exploration horizon and facilitates sample efficiency from an action space perspective.

4. **State Factor Extraction.** The three contributions above assume that state factors are given, but in real-world settings we typically only observe low-level inputs such as images. To address this limitation, we present algorithms that extract state factors from observations in two forms. Dyn-O introduced in Chapter 7 adopts an object-centric representation approach. Compared to prior work that typically learns representations in an unsupervised manner, it improves representation quality by distilling from segmentation masks. FLAM introduced in Chapter 8 adopts a different inductive bias by factorizing representations based on independent actions across entities. Specifically, FLAM introduces a factorized latent action model in which the scene is decomposed into independent factors, each inferring its own latent action and predicting its own next-step factor value. Compared to monolithic models that treat the scene as a whole, this architecture encourages the model to separate entities with independent actions into different factors and to better utilize latent action capacity, thereby enabling the model to learn factorized representations and improving prediction accuracy in multi-entity scenarios.

Taken together, these contributions substantially address the dissertation question. In the next section, we outline several promising directions for future research.

10.2 Future Directions

Building on the contributions in this dissertation, several promising future research directions arise. Here, we group them into short-term and long-term avenues. Short-term directions involve research that can directly build on the findings of this dissertation, while long-term directions target broader challenges that could benefit from these contributions.

10.2.1 Short-Term Directions

1. **Efficient Data Collection for Dynamics Causal Discovery.** As described in Section 2.2, causal discovery accuracy depends strongly on data quality. For example, if a causal dependency is never exposed, the algorithm has no basis to identify it. The same limitation applies to CDL and CBM, even though CDL proposes a data collection policy based on the prediction error gap between a dense model and a causal model. However, this criterion aggregates prediction errors across state factors, which can obscure which dependency is uncertain, and it can be vulnerable to inherently unpredictable transitions. A promising alternative is to identify causal dependencies that the model remains uncertain about and then actively induce and collect data for those dependencies, akin to active causal discovery (Tigas et al., 2022), while additionally learning policies that conduct the required interventions.

2. **Causal Discovery in POMDPs.** As described in Chapters 3 and 4, we focus on MDP settings without hidden confounders. However, in realistic settings, RL agents typically observe only part of the world, which leads to POMDPs in which hidden confounders may exist. This shift makes causal discovery harder because some apparent dependencies may be driven by unobserved confounders rather than direct causal mechanisms. Even though causal discovery in temporal domains with hidden confounders remains an open research question, RL agents have a unique advantage: they can actively choose actions to conduct interventions, test if hidden confounders exist, and evaluate their influence if applicable, potentially by leveraging ideas from related fields such as active perception.

3. **Mixed Intrinsic Reward Functions for Tradeoffs Between Interactions and State Coverage.** While ELDEN introduces an intrinsic reward function that encourages the agent to

induce novel local dependencies, it can fall short in environments that require precise control of the agent or of a particular object. Although SkiLD further combines local dependency rewards with DIAYN-style diversity bonuses, more research is needed to understand how to balance depth and breadth. In particular, the agent may need to dive deep to discover multiple ways to induce a given dependency, while also getting broad by shifting to other dependencies so that exploration and learned skills transfer to a wide range of downstream tasks.

4. **Sparsification Regularization without Hurting Prediction.** In ELDEN, we introduce a regularizer on model partial derivatives to encourage predictions to depend on a sparse subset of inputs. However, this term is difficult to tune. Overly strong regularization can suppress true positive dependencies and degrade prediction quality, whereas overly weak regularization fails to remove spurious correlations. Future work can investigate regularizers or mechanisms that better balance sparsity and predictive performance.

5. **Quantification of Long-Horizon Causal Dependencies and Empowerment.** The intrinsic reward function in ELDEN can be viewed as a coarse proxy for long-horizon empowerment if we assume that the agent induces all local dependencies. For example, the stove cooks the dish because the robot places the dish on the stove earlier. From this perspective, the intrinsic reward function quantifies the agent's controllability over local dependencies over time. However, this quantification can be inaccurate. For instance, if other agents exist, they may induce some local dependencies, and the agent may receive credit for effects it does not induce. To better measure the agent's controllability over the environment, we need improved quantification of long-horizon empowerment, not limited to single-agent settings but also applicable to multi-agent environments. Such quantification can enable the agent to learn more complex behaviors that require prerequisites or collaboration with other agents.

6. **Guiding Skill Discovery with Human Priors.** SkiLD introduces an unsupervised skill discovery objective that enables agents to learn skills that induce diverse local dependencies, which often aligns better with downstream tasks than approaches that maximize state space coverage. However, not all local dependencies are useful, and some may even be dangerous, such as inducing an interaction between a hammer and a vase. To guide skill discovery, we

need priors about which interactions humans value in downstream tasks, so that the agent can focus on useful dependencies and avoid harmful ones. Such priors may be provided by modern foundation models such as LLMs or VLMs.

7. **Efficient Experience Synthesis.** A key promise of the model-based RL methods in this thesis (CDL, CBM, and Dyn-O) is that they can generate synthetic rollouts for efficient policy learning. However, these rollouts typically start from random states sampled from the replay buffer, which can waste expensive synthesis if the agent already learns well in those regions. To generate rollouts that are most helpful for policy learning, future work can investigate criteria for choosing rollout start states, such as states with large regret.

8. **A Universal Factored Latent Action Model.** FLAM introduces a new paradigm for learning latent action models. However, due to data and compute constraints, we learn a separate model for each dataset and only demonstrate the potential of learning a universal model across datasets at a small scale. Specifically, we show that using a large number of factors can accommodate different entity densities across datasets. Future work can explore learning a foundation FLAM across diverse datasets and evaluate its ability to group entities based on action similarity, which may enable stronger transfer across environments.

9. **Camera as a Factor in FLAM.** FLAM identifies factors based on action independence. In many scenarios such as autonomous driving or robotics, the camera is controlled by a separate entity. Future work can explore how to distinguish camera-induced motion from object motion, for example, by modeling the camera as a separate factor.

10. **Representations with Finer Granularity.** In this thesis, we focus on object-centric representations and action-grouped representations. However, object-centric representations remain coarse because they entangle many object properties into a single vector. When identifying causal dependencies, it is often desirable to access causal variables at a finer granularity. For example, when learning a dynamics model, we may want to operate on position and orientation while ignoring other properties. Possible directions include disentangled or causal

representations, as well as methods such as Neural Systematic Binder (Singh et al., 2023; Stammer et al., 2024).

10.2.2 Long-Term Directions

In the long term, we aim to learn a foundation causal world model that learns from diverse environments and datasets, facilitates efficient policy learning, and enables sample-efficient adaptation to new environments.

Specifically, the foundation causal world model unifies the contributions of all previous technical chapters. It consists of a perception model that extracts factorized representations, a dynamics model that predicts the next step value for each factor, and a policy module that learns skills to induce desirable interactions in an unsupervised way while adapting quickly to downstream tasks when needed. Beyond this unification, the foundation causal model should address the following open problems to generalize across diverse training environments and adapt efficiently to unseen environments or tasks.

1. **Representation learning and appropriate abstractions.** Dyn-O in Chapter 7 extracts object-centric representations, while FLAM in Chapter 8 extracts action-grouped representations, but both are learned purely from visual appearance or dynamics signals and may not yield ideal representations. On the one hand, to generalize well, we want the world model and policy to focus on task-relevant entities when performing a task, which suggests that representations should be task-dependent. In that case, object-centric representations may include many irrelevant factors, while action-driven representations may be too coarse because task relevant objects can remain stationary and thus are ignored. On the other hand, we also want the agent to be versatile, so focusing on entity A and ignoring entity B for one task should not cause the agent to lose the ability to recognize entity B when another task requires it. This requirement raises the question of what representation granularity the model should operate on. One option is to extract entity-level representations for the world model and then use a separate model to filter task-irrelevant entities when learning policies. Another option is to have both the world model and the policy operate at the same level, with a representation module that is

itself task-dependent. In summary, an important long-term effort lies in deciding and learning representations at an appropriate level to balance task coverage and task-specific conciseness.

2. **Dynamics learning and generalizable causal models.** As described in Chapter 3, CDL focuses on identifying causal relationships within a single MDP. However, for a new MDP, it learns everything from scratch even if the new MDP contains objects that also appear in previous MDPs, which is inefficient. Instead, we want to learn a causal dynamics model that generalizes across environments, so that knowledge about recurring entities and interactions transfers to new settings. To achieve this goal, a key missing component in CDL and CBM is to understand the semantic meaning of each state factor (i.e., whether it is a block, or a cup, etc) rather than simply treating it as the i-th state factor. In this way, when the model is in a new environment, it can recognize each state factor and reuse the corresponding learned causal dependencies. Such semantic meaning could be provided by humans, or preferably, learned from observations. For example, when the agent is put into a new environment, it can infer from appearance that a state factor looks like a cup, so it should reuse its learned knowledge about how a cup interacts with other entities in the environment. Meanwhile, we need to account for special cases where appearance is misleading (e.g., lighters are crafted into various looks), so the agent may need a few interactions to identify what the entity really is. How to collect such data efficiently for fast entity identification is another open question for future research.

3. **Transfer / continual learning.** One promise of a causal model is that, when facing a new environment, the modular design allows the model to adapt only the modules that need to change while keeping others unchanged, or to learn new modules when necessary. In contrast, a monolithic model typically learns from scratch or finetunes the whole model for the new environment. However, to realize this promise, several open problems remain.

    - **Generalizable perception.** As mentioned, in addition to learning factorized representations, the perception model also needs to identify what kind of entity each factor corresponds to, so the dynamics model can reuse the corresponding causal dependencies. When encountering a new domain, the perception model needs to generalize well or

adapt in a sample-efficient way. The generalization has at least two requirements: 1) recognizing a known entity under a new appearance (e.g., a billboard with new ads), and 2) determining that an entity is unseen so its causal dependencies need to be learned. Both aspects benefit from uncertainty estimates that detect novelty, so the agent can decide when it should interact with an entity to determine whether it is known or unseen.

- **Dynamics adaptation.** In a new environment, the dynamics of some entities may change, but which entities change can be unknown. The agent therefore needs to detect these changes and adapt only the relevant components rather than updating the entire model. This setting raises challenges in both diagnosis and data collection, including how to select informative interactions that reveal which dynamics components mismatch and how to collect such data efficiently.

4. **Integration with existing foundation models.** Current popular foundation models such as LLMs, VLMs, and VLAs typically use transformer architectures operating on language, vision, and action tokens. Meanwhile, the causal world models proposed in this thesis operate on state factors and use separate modules for each factor. It is an important direction to investigate how foundation causal models could unify with existing foundation models, both architecturally and in tokenization, such as how to construct tokens that convey causal dependencies.

10.3 Conclusion

This thesis presents initial attempts to answer the question: How can we train generalizable and sample-efficient RL agents? We approach this question through the lens of causality, more specifically by reasoning about and leveraging causal relationships in transition dynamics and reward functions. By identifying general causal relationships, two methods derive state abstractions that enable RL agents to focus on task-relevant state factors, thereby facilitating the generalization and sample efficiency of learned policies. By identifying local causal relationships, two methods enable RL agents to understand the effects of their actions and to purposely learn behaviors or skills that induce novel dependencies, thereby facilitating exploration. Finally, when state factors are not

given, we introduce two methods that extract state factors from observations, based on either visual appearance or entities' independent actions.

Collectively, these methods represent meaningful steps toward facilitating RL sample efficiency and generalization from multiple perspectives, including the state space, reward space, action space, and representation space. While they do not provide a comprehensive solution, I hope they inspire and provide foundations for future research toward building general causal RL agents that understand causal relationships in diverse environments, learn generalizable policies by focusing on task-relevant state factors, and adapt to new environments in a sample-efficient way by identifying and updating only the components that change or are previously unseen. I hope these contributions encourage continued progress toward AI agents that can robustly learn, reason, and act across diverse settings.

Appendix A: Notation Summary

The notation used in each chapter is listed below:

Chapter 2.1

- $\mathcal{M}$: a Markov decision process (MDP).
- $\mathcal{S}$: the state space of an MDP.
- $d_{\mathcal{S}}$: the number of state factors.
- $\mathcal{S}^i$: the space of the i-th state factor.
- $\mathcal{O}$: the observation space of an MDP.
- $\mathcal{A}$: the action space of an MDP.
- $d_{\mathcal{A}}$: the action dimension.
- $\mathcal{P}$: the transition probability of an MDP.
- $\mathcal{R}$: the reward function of an MDP.
- γ: the discount factor of an MDP.
- π: a policy that maximizes the expected discounted return.
- t: the time step.
- s_t^i: the i-th state factor at time t.
- s_t: the state at time t, i.e., $\{s_t^1, \cdots, s_t^{d_{\mathcal{S}}}\}$.
- o_t: the observation at time t.
- a_t: the action taken at time t.

- x_t: the union of the state factors and the action at time t, i.e., $\{s_t^1, \cdots, s_t^{ds}, a_t\}$.
- $x_t^{\neg i}$: all variables in x_t except s_t^i, i.e., $x_t \setminus \{s_t^i\}$.
- s_{t+1}: the state at the next time step.
- $y_{1:t}$: shorthand for the set $\{y_1, \ldots, y_t\}$ for a generic variable y.
- $y^{1:N}$: shorthand for the set $\{y^1, \ldots, y^N\}$ for a generic N.

Chapter 2.2

- d: the number of variables.
- $\mathbf{G}$: the space of causal graphs.
- $\mathcal{G}$: the general causal graph of an MDP.
- V: the nodes in a causal graph.
- E: the edges in a causal graph.
- X: a set of random variables.
- $p(X)$: the probability distribution of the random variable X.
- $\mathbf{PA}(X)$: the parents of variable X in the causal graph.
- $\bar{X}$: a subset of X.
- $\mathbf{Desc}(X)$: the descendants of variable X in the causal graph.
- $\mathbf{Sep}(X^i, X^j)$: the separating set of X^i and X^j.
- $\mathcal{G}_t(x_t)$: the actual causal graph activated at time t, based on x_t.
- $X^i \to X^j$: X^i is a parent of X^j in the causal graph.
- $A \perp\!\!\!\perp_{\mathcal{G}} B \mid C$: two nodes A and B are d-separated by a third node C in a DAG $\mathcal{G}$.

Chapter 2.4

- ϕ: a state abstraction.
- $\bar{\mathcal{S}}$: the abstract state space.
- $\bar{s}$: an abstract state.
- ϕ^{-1}: the inverse function of state abstraction, i.e., finding the set of states that are mapped the same abstract state.

Chapter 2.5

- r_{task}: the task reward.
- $r_{\text{intrinsic}}$: the intrinsic reward.
- β: the coefficient of the intrinsic reward.
- $C(s)$: the number of times visiting to s.
- f: a learned dynamics model.
- $\{f^i\}_{i=1}^N$: an ensemble of N learned dynamics model.
- Var: the variance.

Chapter 2.6

- z: the skill parameter.
- q: the discriminator.
- g: the goal in goal-conditioned reinforcement learning.

Chapter 2.7

- e: the patch-level features extracted by the encoder.
- n_e: the number of patches in an image.
- d_e: the feature dimension.
- ξ: the slots.
- ξ^k: the k-th slot.
- K: the number of slots.
- d_ξ: the slot dimension.
- ξ_0^k: the learnable initialization for the k-th slot.
- T_{SA}: the number of slot attention iterations (refinements).
- d_{SA}: slot attention model dimension.
- W_q: learned query projection.
- W_k: learned key projection.
- W_v: learned value projection.
- $query_t^i$: the query computed from the i-th slot at iteration t.
- key^j: the key computed from the j-th patch-level feature.
- $value^j$: the value computed from the j-th patch-level feature.
- ℓ_t^{ij}: the unnormalized attention score between the i-th slot at iteration t and j-th patch-level feature.
- $\langle a, b \rangle$: the dot product between two generic vectors a and b.
- $\mathbf{W}_t^{ij}$: the attention score between the i-th slot at iteration t and j-th patch-level feature, normalized across slots for each feature.

- $\bar{\mathbf{W}}_t^{ij}$: the attention score between the i-th slot at iteration t and j-th patch-level feature, computed by further normalizing $\mathbf{W}_t^{ij}$ across features.
- $\xi_{T_{\text{SA}}}^i$: the i-th final slot.
- $\hat{o}^i$: the reconstruction computed from the i-th slot.
- κ^i: the unnormalized logits of the i-th slot's contribution to each pixel.
- ρ^i: the normalized score of the i-th slot's contribution to each pixel.
- $\hat{o}$: the final reconstruction.

Chapter 2.8

- α: the latent action.

Chapter 3

- $\mathcal{S}^{\mathcal{C}}$: the controllable state factors.
- $\mathcal{S}^{\mathcal{AR}}$: the action-relevant state factors.
- $\mathcal{S}^{\mathcal{AI}}$: the action-irrelevant state factors.
- f: the learned dynamics model.
- f^{ϕ}: the dynamics model in the abstract state space.
- $\hat{\mathcal{R}}$: the learned reward prediction model.
- $\mathcal{S}_{<t}$: the historical states, i.e., $\mathcal{S}_1, \ldots, \mathcal{S}_{t-1}$.
- $\mathcal{A}_{<t}$: the historical actions.
- Q: a cofounder.
- CMI^{ij}: the condition mutual information between $\mathcal{S}_t^i$ and $\mathcal{S}_{t+1}^j$.

- $x^i \perp\!\!\!\perp x^j | x^k$: x^i and x^j are independent conditioned on x^k.
- $x^i \not\perp\!\!\!\perp x^j | x^k$: x^i and x^j are not independent conditioned on x^k.
- $x^i \dashrightarrow x^j$: x^i is an ancestor of x^j in the causal graph.
- $\epsilon_{\mathcal{G}}$: the threshold on the conditional mutual information used to determine whether a general causal relationship exists between two variables.
- $\varphi^a, \varphi^1, \ldots, \varphi^{d_{\mathcal{S}}}$: the features extracted from the action and all state factors.
- M: the binary mask to mask certain features to $-\infty$.
- φ: the feature aggregated from $\varphi^a, \varphi^1, \ldots, \varphi^{d_{\mathcal{S}}}$.
- $\hat{p}(x)$: the learned probability distribution of the random variable X.
- L_f: the loss used to optimize the dynamics model.
- η: when computing the reward for data collection policy, the scaling factor of prediction difference between dense and causal models.
- L: the planning length.
- $\mathcal{N}$: a multi-variant normal distribution.
- μ: the mean of a normal distribution.
- σ: the standard deviation of a normal distribution.
- ω: the distribution for a categorical variable.
- $\mathbf{AN}(X)$: the ancestors of variable X in the causal graph.

Chapter 4

- f^{expl}: a learned explicit dynamics model.

- f^{impl}: a learned critic function for the implicit dynamics model.
- $\text{CMI}^{i,\mathcal{R}}$: the condition mutual information between $\mathcal{S}_t^i$ and $\mathcal{R}_t$.
- $f^{\text{impl,i}}$: a learned critic function for the implicit dynamics model for state factor bS_{t+1}^j.
- L_{NCE}: the contrastive InfoNCE loss (van den Oord et al., 2018).
- $\{\tilde{s}_{t+1}^{i,n}\}_{n=1}^N$: the negative examples for contrastive learning.
- $L_{\text{NCE}}(F(x;y))$: the InfoNCE loss that encourages a generic model F to extract information about x from y.
- $\phi(s_{t+1}^j; s_t^i | x_t^{\neg i})$: a conditioned model trained to capture the additional information about s_{t+1}^j in s_t^i that is not present in $x_t^{\neg i}$.
- $\psi(s_{t+1}^j; x_t^{\neg i})$: a critic model trained to extract information about s_{t+1}^j from $x_t^{\neg i}$.
- ψ^*: the optimal ψ after training.
- w_n: importance weights computed using ψ for the n-th negative sample.

Chapter 5

- f: a learned dynamics model.
- $\{f^i\}_{i=1}^N$: an ensemble of N learned dynamics model.
- f^i: the i-th model in the ensemble of learned dynamics models.
- λ_∂: the regularization coefficient of partial derivatives.
- ϵ_∂: the threshold on the partial derivatives used to determine whether an actual causal relationship exists between two variables.
- L_f: the loss used to optimize each dynamics model in the ensemble.
- r_{task}: the task reward.

- $r_{\text{intrinsic}}$: the intrinsic reward.
- β: the coefficient of the intrinsic reward.
- $\varphi^a, \varphi^1, \ldots, \varphi^{d_S}$: features extracted from the action and all state factors.
- $\vartheta^a, \vartheta^1, \ldots, \vartheta^{d_S}$: features extracted by self-attention for prediction.

Chapter 6

- $\mathcal{Z}$: the skill representation space.
- z: the skill parameter.
- $\pi_{\mathcal{G}_t}$: the high-level graph selection policy that chooses target local dependencies to induce.
- π_{skill}: the low-level skill policy that learns to induce the specified local dependencies using primitive actions.
- $\mathcal{R}_{\mathcal{G}_t}$: the reward for the high-level policy.
- $\mathcal{R}_{\text{skill}}$: the reward for the low-level policy.
- $\mathcal{G}_t$: the desired actual causal graph.
- $\mathcal{B}$: the space of the the diversity indicator.
- b: the diversity indicator.
- $\mathcal{G}_{t,\text{induced}}$: the induced actual causal graph.
- $C(\mathcal{G}_{t,\text{induced}})$: the number of times that we have seen the induced graph in the collected transition.
- q: the discriminator.
- λ_b: the coefficient of diversity reward.

- $\epsilon_{\mathcal{G}_t}$: the threshold on the point-wise conditional mutual information used to determine whether a local causal relationship exists between two variables.
- I: the mutual information.
- π_{task}: the policy that utilizes learned skills to solve downstream tasks.
- K_{pt}: the number of pre-training steps for skill learning.
- K_{ft}: the number of finetuning steps for downstream task learning.

Chapter 7

- H: the image height.
- W: the image width.
- e: the patch-level features extracted by the encoder.
- n_e: the number of patches in an image.
- d_e: the feature dimension.
- m: the segmentation mask computed by SAM2.
- ξ: the slots.
- ξ^k: the k-th slot.
- ξ_{init}: the learnable initialization for slots.
- K: the number of slots.
- d_ξ: the slot dimension.
- u: the information about slot interaction.
- h: the hidden state of a state-space model.

- τ_t: whether the episode terminates at time step t.
- r_{cls}: the learnable query token used to generate reward prediction.
- τ_{cls}: the learnable query token used to generate termination prediction.
- $\hat{r}$: the predicted reward.
- $\hat{\tau}$: the predicted termination.
- $\hat{\xi}$: the predicted slots.
- CE: the cross-entropy loss.
- c: the static features.
- v: the dynamic features.
- $\mathtt{M}_c$: the network to extract static features.
- $\mathtt{M}_v$: the network to extract dynamic features.
- $\cos$: the cosine similarity.
- $\tilde{C}$: the negative samples for static feature learning.
- $\mathtt{M}_\xi$: the network to reconstruct the slot.
- $\mathtt{sg}$: the stop-gradient operator.
- $\mathtt{Disc}$: the discriminator used to minimize static information in dynamic features.
- LeCam: LeCam regularization.

Chapter 8

- T: the video length.
- H: the image height.

- W: the image width.
- e: the patch-level features extracted by the encoder.
- n_e: the number of patches in an image.
- d_e: the feature dimension.
- e_q: the quantized patch-level features.
- z_q: quantized features.
- ξ: the slots.
- ξ^k: the k-th slot.
- K: the number of slots.
- α: the latent actions.
- α^k: the latent action for the k-th slot.
- $\tilde{\xi}$: the feature after self-attention in IDM and FDM, encoding slot interaction information.
- $\tilde{\alpha}$: the information about the latent actions.
- w: the observation history length.
- $q(\alpha_t^k)$: the posterior distribution of the latent action α_t^k.
- $p(\alpha_t^k)$: the prior distribution of the latent action α_t^k.
- $\texttt{M}_\mu$: the network that maps $\tilde{\alpha}_t^i$ to the mean of the posterior distribution.
- $\texttt{M}_\sigma$: the network that maps $\tilde{\alpha}_t^i$ to the standard deviation of the posterior distribution.
- $\hat{\xi}$: the predicted slots.
- $\hat{e}$: predicted features.

- β_{KL}: the KL regularization coefficient.
- D: dataset of expert demonstration.
- π: the imitation policy.
- $\mathtt{M}_a$: the network that maps the latent action to the ground-truth action.
- $\hat{o}$: the predicted video.

Appendix B: Acronym Summary

The acronyms used in each Chapters are listed as follows:

Chapter 1.

- RL: Reinforcement Learning.

Chapter 2.

- MDP: Markov Decision Processes.
- CGM: Causal Graphic Model.
- DAG: Directed Acyclic Graph.
- PC algorithm: Peter-Clark algorithm.
- CIT: Conditional Independence Test.
- CMI: Conditional Mutual Information.
- pCMI: pointwise Conditional Mutual Information.
- MBRL: Model-Based Reinforcement Learning.
- USD: Unsupervised Skill Discovery.
- DIAYN: Diversity Is All You Need (Eysenbach et al., 2019).
- CSD: Controllability-aware Skill Discovery (Park et al., 2023).
- METRA: Metric-Aware Abstraction (Park et al., 2024b).
- RIG: RL with Imagined Goals (Nair et al., 2018).
- Slot-Attn: Slot Attention (Locatello et al., 2020).

- GRU: Gated Recurrent Unit.
- MLP: Multi-Layer Perceptron.
- LAM: Latent Action Model.
- IDM: Inverse Dynamics Model.
- FDM: Forward Dynamics Model.
- VAE: Variational AutoEncoder (Kingma and Welling, 2014).

Chapter 3.

- CDL: Causal Dynamics Learning (Wang et al., 2022b).
- CEM: Cross Entropy Method.
- EEF: End-EFfector
- TIA: Task Informed Abstraction (Fu et al., 2021).
- DMC: DeepMind Control Suite.
- ID: in-distribution.
- OOD: out-of-distribution.
- PPO: Proximal Policy Optimization (Schulman et al., 2017).

Chapter 4.

- CBM: Causal Bisimulation Modeling (Wang et al., 2024b).
- InfoNCE: Information Noise-Contrastive Estimation (van den Oord et al., 2018).
- SAC: Soft Actor Critic (Haarnoja et al., 2018).

Chapter 5.

- ELDEN: Exploration via Local Dependencies (Wang et al., 2023).

Chapter 6.

- SkiLD: Skill Discovery from Local Dependencies (Wang et al., 2024a).
- COInS: Chain of Interaction Skills (Chuck et al., 2023).

Chapter 7.

- Dyn-O: Building Structured World Models with Object-Centric Representations.
- WM: World Model.
- SAM2: Segment Anything Model 2 (Ravi et al., 2025).
- SSM: State-Space Model.
- SOLV: Self-supervised Object-centric Learning for Video (Aydemir et al., 2023).
- Slot-Attn: Slot Attention.
- Self-Attn: Self-Attention.
- Cross-Attn: Cross-Attention.
- CE: Cross-Entropy.
- InfoNCE: Info Noise Contrastive Estimation (van den Oord et al., 2018).
- Procgen: Procedurally-Generated Game-Like Gym-Environments (Cobbe et al., 2020b).
- PPG: Phasic Policy Gradient (Cobbe et al., 2021).
- FR-ARI: ForeGround Adjusted Rand Index.

- LPIPS: Learned Perceptual Image Patch Similarity (Zhang et al., 2018b).
- FVD: Fréchet Video Distance (Unterthiner et al., 2018).
- SSIM: Structural Similarity Index Measure.
- PSNR: Peak Signal-to-Noise Ratio.
- MSE: Mean Squared Error.

Chapter 8.

- FLAM: Factored Latent Action Model.
- LAM: Latent Action Model.
- IDM: Inverse Dynamics Model.
- FDM: Forward Dynamics Model.
- VQ-VAE: Vector Quantized Variational AutoEncoder (van den Oord et al., 2017).
- FSQ: Finite Scalar Quantization (Mentzer et al., 2024).
- KL divergence: KullbackLeibler divergence.
- MultiGrid: Theory of mind as intrinsic motivation for multi-agent reinforcement learning (Oguntola et al., 2023).
- nuPlan: Towards learning-based planning: The nuPlan benchmark for real-world autonomous driving (Karnchanachari et al., 2024).
- LAPO: Latent Action Policies from Observation (Schmidt and Jiang, 2024).
- Genie: Generative Interactive Environments (Bruce et al., 2024).
- AdaWorld: Learning Adaptable World Models with Latent Actions (Gao et al., 2025).

- PlaySlot: Learning Inverse Latent Dynamics for Controllable Object-Centric Video Prediction and Planning (Villar-Corrales and Behnke, 2025).
- SlotFormer: Unsupervised Visual Dynamics Simulation with Object-Centric Models (Wu et al., 2023b).
- SAVi: Slot Attention for Video (Kipf et al., 2022).

Chapter 9.

- DQN: Deep Q-Network (Mnih et al., 2015).
- GRPO: Group Relative Policy Optimization (Guo et al., 2025).
- RLfD: Reinforcement Learning from Demonstrations.
- i.i.d.: independent and identically distributed.
- CRL: Causal Representation Learning.
- LQR: Linear Quadratic Regulator.
- PILCO: Probabilistic Inference for Learning COntrol (Deisenroth and Rasmussen, 2011).
- MICo: Matching under Independent Couplings (Castro et al., 2021).
- DADS: Dynamics-Aware Unsupervised Discovery of Skills (Sharma et al., 2020).
- EDL: Explore, Discover and Learn (Campos et al., 2020).
- LSD: Lipschitz-constrained Skill Discovery (Park et al., 2022).
- DISCERN: Discriminative Embedding Reward Networks (Warde-Farley et al., 2019).
- MONet: Multi-Object Network (Burgess et al., 2019).
- IODINE: Iterative Object Decomposition Inference NEtwork (Greff et al., 2019).

- GENESIS: GENErative Scene Inference and Sampling (Engelcke et al., 2020).
- SlotSSM: Slot State Space Model (Jiang et al., 2024).
- ILPO: Imitating Latent Policies from Observation (Edwards et al., 2019).

Works Cited

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*, page 1. ACM, 2004. doi: 10.1145/1015330.1015430. URL https://doi.org/10.1145/1015330.1015430.

David Abel. A theory of state abstraction for reinforcement learning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, volume 33, pages 9876–9877, 2019. doi: 10.1609/aaai.v33i01.33019876. URL https://doi.org/10.1609/aaai.v33i01.33019876.

David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48, pages 2915–2923. PMLR, 2016. URL https://proceedings.mlr.press/v48/abel16.html.

Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017. URL https://arxiv.org/abs/1703.01732.

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018. URL https://arxiv.org/abs/1807.10299.

Siddhant Agarwal, Ishan Durugkar, Peter Stone, and Amy Zhang. f-policy gradients: A general framework for goal-conditioned RL using f-divergences. In *Advances in Neural Information Processing Systems*, volume 36 of *NeurIPS*, pages 12100–12123, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/27f4d95417bb722015 97bf4d67cbacc-Abstract-Conference.html.

Siddhant Agarwal, Harshit Sikchi, Peter Stone, and Amy Zhang. Proto successor measure: Representing the behavior space of an RL agent. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=mUDnPzopZF.

Kartik Ahuja, Divyat Mahajan, Yixin Wang, and Yoshua Bengio. Interventional causal representation learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pages 372–407. PMLR, 2023. URL https://proceedings.mlr.press/v202/ahuja23a.html.

Adil Kaan Akan and Yucel Yemez. Slot-guided adaptation of pre-trained diffusion models for object-centric learning and compositional generation. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=kZvor5aaz7.

Cameron Allen, Neev Parikh, Omer Gottesman, and George Konidaris. Learning markov state abstractions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34 of *NeurIPS*, pages 8229–8241, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/454cecc4829279e64d624cd8a8c9ddf1-Abstract.html.

Ron Amit, Ron Meir, and Kamil Ciosek. Discount factor as a regularizer in reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 269–278. PMLR, 2020. URL https://proceedings.mlr.press/v119/amit20a.html.

Brandon Amos, Samuel Stanton, Denis Yarats, and Andrew Gordon Wilson. On the model-based stochastic value gradient for continuous reinforcement learning. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control (L4DC)*, volume 144, pages 6–20. PMLR, 2021. URL https://proceedings.mlr.press/v144/amos21a.html.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular Multitask Reinforcement Learning with Policy Sketches. In *International Conference on Machine Learning (ICML)*, pages 166–175. PMLR, 2017. URL https://proceedings.mlr.press/v70/andreas17a.html.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 5048–5058, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html.

Martín Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. URL https://arxiv.org/abs/1907.02893.

Görkay Aydemir, Weidi Xie, and Fatma Güney. Self-supervised object-centric learning for videos. In *Advances in Neural Information Processing Systems*, volume 36 of *NeurIPS*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/67b0e7c7c2a5780aeefe3b79caac106e-Abstract-Conference.html.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=Sye57xStvB.

Junyeob Baek, Yi-Fu Wu, Gautam Singh, and Sungjin Ahn. Dreamweaver: Learning compositional world models from pixels. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=e5mTvjXG9u.

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (VPT): Learning to act by watching unlabeled online videos. In *Advances in Neural Information Processing Systems*, volume 35 of *NeurIPS*, pages 24639–24654, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9c7008aff45b5d8f0973b23e1a22ada0-Abstract-Conference.html.

Anton Bakhtin, Yuntian Deng, Sam Gross, Myle Ott, Marc'Aurelio Ranzato, and Arthur Szlam. Residual energy-based models for text. *Journal of Machine Learning Research*, 22 (40):1–41, 2021. URL http://jmlr.org/papers/v22/20-326.html.

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado P. van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30 of *NeurIPS*, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/350db081a661525235354dd3e19b8c05-Abstract.html.

Andrew G. Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning (ICDL)*, pages 112–119. IEEE, 2004.

Sumanta Basu, Ali Shojaie, and George Michailidis. Network Granger causality with inherent grouping structure. *Journal of Machine Learning Research*, 16:417–453, 2015. URL https://jmlr.org/papers/v16/basu15a.html.

Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International conference on machine learning and data mining in pattern recognition*, pages 262–275. Springer, 2017.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. doi: 10.1613/jair.3912. URL https://doi.org/10.1613/jair.3912.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 1471–1479, 2016. URL https://papers.nips.cc/paper/6383-unifying-count-based-exploration-and-intrinsic-motivation.

Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957a.

Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6 (5):679–684, 1957b. URL https://www.jstor.org/stable/24900506.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*, volume 3 of *Optimization and Neural Computation Series*. Athena Scientific, 1996. ISBN 978-1886529106.

Gianluca Bontempi and Maxime Flauder. From dependency to causality: A machine learning approach. *Journal of Machine Learning Research*, 16(74):2437–2457, 2015. URL https://jmlr.org/papers/v16/bontempi15a.html.

Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107, 2000a. doi: 10.1016/S0004-3702(00)00033-3. URL https://doi.org/10.1016/S0004-3702(00)00033-3.

Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1–2):49–107, 2000b. doi: 10.1016/S0004-3702(00)00033-3. URL https://doi.org/10.1016/S0004-3702(00)00033-3.

Ronen I. Brafman and Moshe Tennenholtz. R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct): 213–231, 2002. URL https://jmlr.org/papers/v3/brafman02a.html.

Johann Brehmer, Pim de Haan, Phillip Lippe, and Taco Cohen. Weakly supervised causal representation learning. In *Advances in Neural Information Processing Systems*, volume 35 of *NeurIPS*, pages 38319–38331, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/fa567e2b2c870f8f09a87b6e73370869-Abstract-Conference.html.

Marc Brittain, Josh Bertram, Xuxi Yang, and Peng Wei. Prioritized sequence experience replay. *arXiv preprint arXiv:1905.12726*, 2019. URL https://arxiv.org/abs/1905.12726.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235. PMLR, 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Arthur E. Bryson. Optimal control—1950 to 1985. *IEEE Control Systems Magazine*, 16(3): 26–33, 1996. doi: 10.1109/37.506395. URL https://doi.org/10.1109/37.506395.

Lars Buesing, Theophane Weber, Yori Zwols, Sebastien Racaniere, Arthur Guez, Jean-Baptiste Lespiau, and Nicolas Heess. Woulda, Coulda, Shoulda: Counterfactually-Guided Policy Search. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=BJG0voC9YQ.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-Scale Study of Curiosity-Driven Learning. In *International Conference on Learning Representations (ICLR)*, 2019a. URL https://openreview.net/forum?id=rJNwDjAqYX.

Yuri Burda, Harri Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation. In *International Conference on Learning Representations (ICLR)*, 2019b. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. MONet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019. URL https://arxiv.org/abs/1901.11390.

Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giro-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 1317–1327. PMLR, 2020. URL https://proceedings.mlr.press/v119/campos20a.html.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. MICo: Improved representations via sampling-based state similarity for markov decision processes. In *Advances in Neural Information Processing Systems*, volume 34 of *NeurIPS*, pages 12580–12591, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/fd06b8ea02fe5b1c2496fe1700e9d16c-Abstract.html.

David Chapman and Leslie Pack Kaelbling. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 726–731, 1991. URL https://www.ijcai.org/Proceedings/91-2/Papers/018.pdf.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34 of *NeurIPS*, pages 15084–15097, 2021. URL https://proceedings.neurips.cc

/paper_files/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html.

Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, volume 31 of *NeurIPS*, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/1ee3dfcd8a0645a25a35977997223d22-Abstract.html.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 29 of *NeurIPS*, 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/hash/7c9d0b1f96aebd7b5eca8c3edaa19ebb-Abstract.html.

Xiaoyu Chen, Junliang Guo, Tianyu He, Chuheng Zhang, Pushi Zhang, Derek Cathera Yang, Li Zhao, and Jiang Bian. IGOR: Image-goal representations are the atomic control units for foundation models in embodied AI. *arXiv preprint arXiv:2411.00785*, 2024. URL https://arxiv.org/abs/2411.00785.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=rJeXCo0cYX.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002. URL https://jmlr.org/papers/v3/chickering02b.html.

Hana Chockler and Joseph Y. Halpern. Responsibility and blame: A structural-model approach. *Journal of Artificial Intelligence Research*, 22:93–115, 2004. doi: 10.1613/jair.1391. URL https://doi.org/10.1613/jair.1391.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31 of *NeurIPS*, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3de568f8597b94bda53149c7d7f5958c-Abstract.html.

Caleb Chuck, Kevin Black, Aditya Arjun, Yuke Zhu, and Scott Niekum. Granger causal interaction skill chains. *arXiv preprint arXiv:2306.09509*, 2023.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 1282–1289. PMLR, 2019. URL https://proceedings.mlr.press/v97/cobbe19a.html.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 2048–2056. PMLR, 2020a. URL https://proceedings.mlr.press/v119/cobbe20a.html.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 2048–2056. PMLR, 2020b. URL https://proceedings.mlr.press/v119/cobbe20a.html.

Karl W. Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pages 2020–2027. PMLR, 2021. URL https://proceedings.mlr.press/v139/cobbe21a.html.

Lior Cohen, Kaixin Wang, Bingyi Kang, and Shie Mannor. Improving token-based world models with parallel observation prediction. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235. PMLR, 2024. URL https://arxiv.org/abs/2402.15506.

Thomas Dean and Robert Givan. Model minimization in Markov decision processes. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI)*, pages 106–111. AAAI Press, 1997.

Thomas Dean, Robert Givan, and Sonia Leach. Model reduction techniques for computing approximately optimal solutions for markov decision processes. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 124–131. Morgan Kaufmann, 1997a.

Thomas Dean, Robert Givan, and Sonia Leach. Model reduction techniques for computing approximately optimal solutions for markov decision processes. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, UAI'97, page 124131, San Francisco, CA, USA, 1997b. Morgan Kaufmann Publishers Inc. ISBN 1558604855.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 465–472, 2011.

Zizhen Deng, Hu Tian, Xiaolong Zheng, and Daniel Dajun Zeng. Deep causal learning: Representation, discovery and inference. *ACM Comput. Surv.*, 58(2), September 2025. ISSN 0360-0300. doi: 10.1145/3762179. URL https://doi.org/10.1145/3762179.

Wenhao Ding, Haohong Lin, Bo Li, and Ding Zhao. Generalizing goal-conditioned reinforcement learning with variational causal reasoning. In *Advances in Neural Information Processing Systems*, volume 35 of *NeurIPS*, pages 31466–31479, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/a96368eb38bce0956a1132154d70d72d-Abstract-Conference.html.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, volume 32 of *NeurIPS*, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/378a063b8fdb1db941e34f4bde584c7d-Abstract.html.

Yilun Du, Toru Lin, and Igor Mordatch. Model-based planning with energy-based models. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 374–383. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/du20a.html.

Yuqing Du, Eliza Kosoy, Alyssa Dayan, Maria Rufova, Pieter Abbeel, and Alison Gopnik. What can AI learn from human exploration? intrinsically-motivated humans and agents in open-world exploration. In *Proceedings of the NeurIPS 2023 Workshop on Intrinsically Motivated Open-ended Learning (IMOL)*, 2023. URL https://neurips.cc/virtual/2023/77602.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint*, 2016. URL https://arxiv.org/abs/1611.02779.

Ishan Durugkar. *Estimation and Control of Visitation Distributions for Reinforcement Learning*. PhD thesis, The University of Texas at Austin, 2023. URL https://repositories.lib.utexas.edu/items/649294a7-56b7-4975-ad82-f324fa212aec.

Ishan Durugkar, Mauricio Tec, Scott Niekum, and Peter Stone. Adversarial intrinsic motivation for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34 of *NeurIPS*, pages 8622–8636, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/486c0401c56bf7ec2daa9eba58907da9-Abstract.html.

Cian Eastwood and Christopher K. I. Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://openreview.net/forum?id=By-7dz-AZ.

Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 1755–1763. PMLR, 2019. URL https://proceedings.mlr.press/v97/edwards19a.html.

Michael Eichler. Granger causality and path diagrams for multivariate time series. *Journal of Econometrics*, 137(2):334–353, 2007. doi: 10.1016/j.jeconom.2005.06.032. URL https://doi.org/10.1016/j.jeconom.2005.06.032.

Gamaleldin F. Elsayed, Aravindh Mahendran, Sjoerd Van Steenkiste, Klaus Greff, Michael C. Mozer, and Thomas Kipf. SAVi++: Towards end-to-end object-centric learning from real-world videos. In *Advances in Neural Information Processing Systems*, volume 35 of *NeurIPS*, pages 28940–28954, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/ba1a6ba05319e410f0673f8477a871e3-Abstract-Conference.html.

Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=BkxfaTVFwH.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is All You Need: Learning Skills Without a Reward Function. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=tc5qisoB-C.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35 of *NeurIPS*, pages 35603–35620, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/e7663e974c4ee7a2b475a4775201ce1f-Abstract-Conference.html.

Benjamin Eysenbach, Vivek Myers, Ruslan Salakhutdinov, and Sergey Levine. Inference via interpolation: Contrastive representations provably enable planning and inference. In *Advances in Neural Information Processing Systems*, volume 37 of *NeurIPS*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/6c49d2ad55e50c5ebc1002fdc50e48e5-Abstract-Conference.html.

Ke Fan, Zechen Bai, Tianjun Xiao, Tong He, Max Horn, Yanwei Fu, Francesco Locatello, and Zheng Zhang. Adaptive slot attention: Object discovery with dynamic slot number. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23201–23211, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Fan_Adaptive_Slot_Attention_Object_Discovery_with_Dynamic_Slot_Number_CVPR_2024_paper.html.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 3061–3071. PMLR, 2020. URL https://proceedings.mlr.press/v119/fedus20a.html.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018. URL https://arxiv.org/abs/1803.00101.

Fan Feng and Sara Magliacane. Learning dynamic attribute-factored world models for efficient multi-object reinforcement learning. In *Advances in Neural Information Processing*

Systems, volume 36 of *NeurIPS*, 2023. URL https://openreview.net/forum?id=bsNslV3Ahe.

Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 720–727. ACM, 2006. doi: 10.1145/1160633.1160762. URL https://doi.org/10.1145/1160633.1160762.

Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous Markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011. doi: 10.1137/10080484X. URL https://doi.org/10.1137/10080484X.

Norman Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 162–169, 2004. URL https://arxiv.org/abs/1207.4114.

Stefano Ferraro, Pietro Mazzaglia, Tim Verbelen, and Bart Dhoedt. Focus: object-centric world models for robotic manipulation. *Frontiers in Neurorobotics*, Volume 19 - 2025, 2025. ISSN 1662-5218. doi: 10.3389/fnbot.2025.1585386. URL https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2025.1585386.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 1126–1135. PMLR, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

R. A. Fisher. *The Design of Experiments*. Oliver and Boyd, Edinburgh, 1935.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit Behavioral Cloning. In *Conference on Robot Learning (CoRL)*, volume 164, pages 158–168. PMLR, 2022. URL https://proceedings.mlr.press/v164/florence22a.html.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=B1oK8aoxe.

Xiang Fu, Ge Yang, Pulkit Agrawal, and Tommi Jaakkola. Learning Task Informed Abstractions. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pages 3480–3491. PMLR, 2021. URL https://proceedings.mlr.press/v139/fu21b.html.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 2052–2062. PMLR, 2019. URL https://proceedings.mlr.press/v97/fujimoto19a.html.

Scott Fujimoto, David Meger, and Doina Precup. An equivalence between loss functions and non-uniform sampling in experience replay. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS*, pages 14219–14230, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/a3bf6e4db673b6449c2f7d13ee6ec9c0-Abstract.html.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *ICML*, pages 1180–1189. PMLR, 2015. URL https://proceedings.mlr.press/v37/ganin15.html.

Erdun Gao, Ignavier Ng, Mingming Gong, Li Shen, Wei Huang, Tongliang Liu, Kun Zhang, and Howard Bondell. MissDAG: Causal discovery in the presence of missing data with continuous additive noise models. In *Advances in Neural Information Processing Systems*,

volume 35 of *NeurIPS*, pages 5024–5037, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/206361867abf7eb01746c3943078da3c-Abstract-Conference.html.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. AdaWorld: Learning adaptable world models with latent actions. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. URL https://openreview.net/forum?id=QQegZj99sk.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 2170–2179. PMLR, 2019. URL https://proceedings.mlr.press/v97/gelada19a.html.

John Geweke. Measurement of linear dependence and feedback between multiple time series. *Journal of the American Statistical Association*, 77(378):304–313, 1982. doi: 10.1080/01621459.1982.10477803. URL https://doi.org/10.1080/01621459.1982.10477803.

Eric Ghysels, Jonathan B. Hill, and Kaiji Motegi. Testing for Granger causality with mixed frequency data. *Journal of Econometrics*, 192(1):207–230, 2016. doi: 10.1016/j.jeconom.2015.07.007. URL https://doi.org/10.1016/j.jeconom.2015.07.007.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1):163–223, 2003. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(02)00376-4. URL https://www.sciencedirect.com/science/article/pii/S0004370202003764. Planning with Uncertainty and Incomplete Information.

Adam Gleave, Michael Dennis, Neel Kant, Cody Wild, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on*

Learning Representations (ICLR), 2020. URL https://openreview.net/forum?id=HJgEMpVFwB.

Anirudh Goyal, Aniket Didolkar, Nan Rosemary Ke, Charles Blundell, Philippe Beaudoin, Nicolas Heess, Michael C. Mozer, and Yoshua Bengio. Neural Production Systems. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pages 25673–25683, 2021a. URL https://proceedings.neurips.cc/paper/2021/hash/d785bf9067f8af9e078b93cf26de2b54-Abstract.html.

Anirudh Goyal, Alex Lamb, Phanideep Gampa, Philippe Beaudoin, Sergey Levine, Charles Blundell, Yoshua Bengio, and Michael C. Mozer. Object files and schemata: Factorizing declarative and procedural knowledge in dynamical systems. In *International Conference on Learning Representations (ICLR)*, 2021b. URL https://openreview.net/forum?id=VVdmjgu7pKM.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent Independent Mechanisms. In *International Conference on Learning Representations (ICLR)*, 2021c. URL https://openreview.net/forum?id=mLcmdlEUxy-.

C. W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969. doi: 10.2307/1912791. URL https://doi.org/10.2307/1912791.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 2424–2433. PMLR, 2019. URL https://proceedings.mlr.press/v97/greff19a.html.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=Skc-Fo4Yg.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31 of *NeurIPS*, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 1352–1361. PMLR, 2017. URL https://proceedings.mlr.press/v70/haarnoja17a.html.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 2555–2565. PMLR, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Joseph Y. Halpern. *Actual Causality*. MIT Press, 2016. ISBN 9780262035026. URL https://mitpress.mit.edu/9780262035026/actual-causality/.

Joseph Y. Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part I: Causes. *The British Journal for the Philosophy of Science*, 56(4):843–887, 2005. doi: 10.1093/bjps/axi147. URL https://doi.org/10.1093/bjps/axi147.

Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 13611–13617. IEEE, 2021. doi: 10.1109/ICRA48506.2021.9561103. URL https://doi.org/10.1109/ICRA48506.2021.9561103.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Philippe Hansen-Estruch, Amy Zhang, Ashvin Nair, Patrick Yin, and Sergey Levine. Bisimulation makes analogies in goal-conditioned reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162, pages 8407–8426. PMLR, 2022. URL https://proceedings.mlr.press/v162/hansen-estruch22a.html.

Tianxing He, Bryan McCann, Caiming Xiong, and Ehsan Hosseini-Asl. Joint energy-based model training for better calibrated natural language understanding models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1076–1089. Association for Computational Linguistics,

2021. doi: 10.18653/v1/2021.eacl-main.92. URL https://aclanthology.org/2021.eacl-main.151/.

Sachin Heerah, Roberto Molinari, Stéphane Guerrier, and Amy Marshall-Colon. Granger-causal testing for irregularly sampled time series with application to nitrogen signalling in arabidopsis. *Bioinformatics*, 37(16):2450–2460, 03 2021. ISSN 1367-4803. doi: 10.1093/bioinformatics/btab126. URL https://doi.org/10.1093/bioinformatics/btab126.

Miguel A. Hernán and James M. Robins. *Causal Inference: What If*. Chapman & Hall/CRC, Boca Raton, FL, 2010.

Todd Hester and Peter Stone. TEXPLORE: Real-time sample-efficient reinforcement learning for robots. *Machine Learning*, 90(3):385–429, 2013. doi: 10.1007/s10994-012-5322-7. URL https://doi.org/10.1007/s10994-012-5322-7.

Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John P. Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep Q-learning from demonstrations. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, volume 32, pages 3223–3230, 2018. doi: 10.1609/aaai.v32i1.11757. URL https://doi.org/10.1609/aaai.v32i1.11757.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout Q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. VIME: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, volume 29 of *NeurIPS*, pages 1109–1117, 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/hash/abd815286ba1007abfbb8415b83ae2cf-Abstract.html.

Ronald A. Howard. *Dynamic Programming and Markov Processes*. The MIT Press and John Wiley & Sons, Cambridge, MA, USA, 1960.

Biwei Huang, Fan Feng, Chaochao Lu, Sara Magliacane, and Kun Zhang. AdaRL: What, where, and how to adapt in transfer reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022a. URL https://openreview.net/forum?id=8H5bpVwvt5.

Biwei Huang, Chaochao Lu, Liu Leqi, José Miguel Hernández-Lobato, Clark Glymour, Bernhard Schölkopf, and Kun Zhang. Action-sufficient state representation learning for control with structural constraints. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162, pages 9260–9279. PMLR, 2022b. URL https://proceedings.mlr.press/v162/huang22f.html.

Sandy H. Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017. URL https://arxiv.org/abs/1702.02284.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *Advances in Neural Information Processing Systems*, volume 32 of *NeurIPS*, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/e2ccf95a7f2e1878fcafc8376649b6e8-Abstract.html.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 32 of *NeurIPS*, 2019.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162, pages 9902–9915. PMLR, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Jindong Jiang, Fei Deng, Gautam Singh, and Sungjin Ahn. Object-centric slot diffusion. In *Advances in Neural Information Processing Systems*, volume 36 of *NeurIPS*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b3ceb8a495a63ced4a48f8429ccdcd8-Abstract-Conference.html.

Jindong Jiang, Fei Deng, Gautam Singh, Minseung Lee, and Sungjin Ahn. Slot state space models. In *Advances in Neural Information Processing Systems*, volume 37 of *NeurIPS*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/158ac5698e36a01ee5ca9e6732685b34-Abstract-Conference.html.

Emily Jin, Jiaheng Hu, Zhuoyi Huang, Ruohan Zhang, Jiajun Wu, Li Fei-Fei, and Roberto Martín-Martín. Mini-BEHAVIOR: A procedurally generated benchmark for long-horizon decision-making in embodied AI. In *Advances in Neural Information Processing Systems*, volume 36 of *NeurIPS*, 2023. URL https://neurips.cc/virtual/2023/79902.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2901–2910, 2017. URL https://openaccess.thecvf.

com/content_cvpr_2017/html/Johnson_CLEVR_A_Diagnostic_CVPR_2017_paper.html.

Nicholas K. Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 752–757, 2005. URL https://www.ijcai.org/Proceedings/05/Papers/1655.pdf.

Amir Mohammad Karimi Mamaghan, Andrea Dittadi, Stefan Bauer, Karl Henrik Johansson, and Francesco Quinzan. Diffusion-based causal representation learning. *Entropy*, 26(7), 2024. ISSN 1099-4300. doi: 10.3390/e26070556. URL https://www.mdpi.com/1099-4300/26/7/556.

Napat Karnchanachari, Dimitris Geromichalos, Kok Seang Tan, Nanxiang Li, Christopher Eriksen, Shakiba Yaghoubi, Noushin Mehdipour, Gianmarco Bernasconi, Whye Kit Fong, Yiluan Guo, et al. Towards learning-based planning: The nuplan benchmark for real-world autonomous driving. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 629–636. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10610077. URL https://doi.org/10.1109/ICRA57147.2024.10610077.

Nan Rosemary Ke, Aniket Rajiv Didolkar, Sarthak Mittal, Anirudh Goyal, Guillaume Lajoie, Stefan Bauer, Danilo Jimenez Rezende, Michael Curtis Mozer, Yoshua Bengio, and Christopher Pal. Systematic Evaluation of Causal Discovery in Visual Model-Based Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=gp5Uzbl-9C-.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232, 2002. doi: 10.1023/A:1017984413808. URL https://doi.org/10.1023/A:1017984413808.

Ilyes Khemakhem, Diederik P. Kingma, Ricardo Pio Monti, and Aapo Hyvärinen. Variational autoencoders and nonlinear ICA: A unifying framework. In *Proceedings of the 23rd*

International Conference on Artificial Intelligence and Statistics (AISTATS), volume 108, pages 2207–2217. PMLR, 2020. URL https://proceedings.mlr.press/v108/khemakhem20a.html.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 2649–2658. PMLR, 2018. URL https://proceedings.mlr.press/v80/kim18b.html.

Jaekyeom Kim, Seohong Park, and Gunhee Kim. Unsupervised skill discovery with bottleneck option learning. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pages 5572–5582. PMLR, 2021. URL https://proceedings.mlr.press/v139/kim21j.html.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014. URL https://openreview.net/forum?id=33X9fd2-9FyZd.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive Learning of Structured World Models. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=H1gax6VtDB.

Thomas Kipf, Gamaleldin F. Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=aD7uesX1GF_.

Albina Klepach, Alexander Nikulin, Ilya Zisman, Denis Tarasov, Alexander Derevyagin, Andrei Polubarov, Nikita Lyubaykin, and Vladislav Kurenkov. Object-centric latent action learning. *arXiv preprint arXiv:2502.09680*, 2025. URL https://arxiv.org/abs/2502.09680.

Alexander S. Klyubin, Daniel Polani, and Chrystopher L. Nehaniv. All Else Being Equal Be Empowered. In *Advances in Artificial Life: 8th European Conference (ECAL)*, pages

744–753. Springer, 2005a. doi: 10.1007/11553090_75. URL https://doi.org/10.1007/11553090_75.

Alexander S. Klyubin, Daniel Polani, and Chrystopher L. Nehaniv. Empowerment: A universal agent-centric measure of control. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135, 2005b. doi: 10.1109/CEC.2005.1554676. URL https://doi.org/10.1109/CEC.2005.1554676.

Avinash Kori, Francesco Locatello, Ainkaran Santhirasekaram, Francesca Toni, Ben Glocker, and Fabio De Sousa Ribeiro. Identifiable object-centric representation learning via probabilistic slot attention. In *Advances in Neural Information Processing Systems*, NeurIPS, 2024. URL https://openreview.net/forum?id=qmoVQbwmCY.

Ezgi Korkmaz. A survey analyzing generalization in deep reinforcement learning. *arXiv preprint arXiv:2401.02349*, 2024. URL https://arxiv.org/abs/2401.02349.

Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017. URL https://arxiv.org/abs/1705.06452.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-Learning. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Konstantin Raphael Kueffner. A comprehensive survey of the actual causality literature. Diploma thesis, Technische Universität Wien, 2021. URL https://doi.org/10.34726/hss.2021.90003.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://openreview.net/forum?id=H1kG7GZAW. Also known as DIP-VAE.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-Learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, volume 32 of *NeurIPS*, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/c2073ffa77b5357a498057413bb09d3a-Abstract.html.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS*, pages 1179–1191, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations (ICLR)*, 2018.

Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient Based Neural DAG Learning. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=rklbKA4YDS.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS*, pages 19884–19895, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/e615c82aba461681ade82da2da38004a-Abstract.html.

Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. URLB: Unsupervised reinforcement learning benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/091d584fced301b442654dd8c23b3fc9-Abstract-round2.html.

Michael Laskin, Hao Liu, Adam Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Contrastive intrinsic control for unsupervised reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35 of *NeurIPS*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/debf482a7dbdc401f9052dbe15702837-Abstract-Conference.html.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*, pages 191–246. MIT Press, 2006.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019. URL https://arxiv.org/abs/1906.05274.

Andrew Levy. *Unsupervised Skill Discovery with Empowerment*. PhD thesis, Brown University, 2025.

Andrew Levy, Sreehari Rammohan, Alessandro Allievi, Scott Niekum, and George Konidaris. Hierarchical empowerment: Towards tractable empowerment-based skill-learning. *arXiv preprint arXiv:2307.02728*, 2023. URL https://arxiv.org/abs/2307.02728.

David Lewis. Causation. *The Journal of Philosophy*, 70(17):556–567, 1973. doi: 10.2307/2025310. URL https://doi.org/10.2307/2025310.

Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov,

Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. iGibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, volume 164, pages 455–465. PMLR, 2022. URL https://proceedings.mlr.press/v164/li22b.html.

Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pages 222–229, 2004. doi: 10.5220/0001143902220229. URL https://doi.org/10.5220/0001143902220229.

Yunzhu Li, Antonio Torralba, Anima Anandkumar, Dieter Fox, and Animesh Garg. Causal Discovery in Physical Systems from Videos. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 9180–9192, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6822951732be44edf818dc5a97d32ca6-Abstract.html.

Luofeng Liao, Zuyue Fu, Zhuoran Yang, Yixin Wang, Dingli Ma, Mladen Kolar, and Zhaoran Wang. Instrumental variable value iteration for causal offline reinforcement learning. *Journal of Machine Learning Research*, 25(303):1–56, 2024. URL https://jmlr.org/papers/v25/22-0965.html.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3):293–321, 1992. doi: 10.1007/BF00992699. URL https://doi.org/10.1007/BF00992699.

Phillip Lippe, Sara Magliacane, Sindy Löwe, Yuki M. Asano, Taco Cohen, and Efstratios Gavves. BISCUIT: Causal representation learning from binary interactions. In *Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 216, pages 1264–1274. PMLR, 2023. URL https://proceedings.mlr.press/v216/lippe23a.html.

Grace Liu, Michael Tang, and Benjamin Eysenbach. A single goal is all you need: Skills and exploration emerge from contrastive RL without rewards, demonstrations, or subgoals. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=xCkgX4Xfu0.

Hao Liu and Pieter Abbeel. APS: Active pretraining with successor features. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pages 6736–6747. PMLR, 2021a. URL https://proceedings.mlr.press/v139/liu21b.html.

Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. In *Advances in Neural Information Processing Systems*, volume 34 of *NeurIPS*, pages 18459–18473, 2021b. URL https://proceedings.neurips.cc/paper/2021/hash/99bf3d153d4bf67d640051a1af322505-Abstract.html.

Yu-Ren Liu, Biwei Huang, Zhengmao Zhu, Honglong Tian, Mingming Gong, Yang Yu, and Kun Zhang. Learning world models with identifiable factorization. In *Advances in Neural Information Processing Systems*, volume 36 of *NeurIPS*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/65496a4902252d301cdf219339bfbf9e-Abstract-Conference.html.

Joseph T. Lizier and Mikhail Prokopenko. Differentiating information transfer and causal effect. *The European Physical Journal B*, 73(4):605–615, 2010. doi: 10.1140/epjb/e2010-00034-5. URL https://doi.org/10.1140/epjb/e2010-00034-5.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 4114–4124. PMLR, 2019. URL https://proceedings.mlr.press/v97/locatello19a.html.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS*, pages 11525–11538, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/8511df98c02ab60aea1b2356c013bc0f-Abstract.html.

Lars Lorch, Jonas Rothfuss, Bernhard Schölkopf, and Andreas Krause. DiBS: Differentiable Bayesian structure learning. In *Advances in Neural Information Processing Systems*, volume 34 of *NeurIPS*, pages 24111–24123, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/ca6ab34959489659f8c3776aaf1f8efd-Abstract.html.

Lars Lorch, Scott Sussex, Jonas Rothfuss, Andreas Krause, and Bernhard Schölkopf. Amortized inference for causal structure learning. In *Advances in Neural Information Processing Systems*, volume 35 of *NeurIPS*, pages 12739–12753, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/54f7125dee9b8b3dc798bb9a082b09e2-Abstract-Conference.html.

William S. Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28(1):47–65, 1991. doi: 10.1007/BF02055574. URL https://doi.org/10.1007/BF02055574.

Chaochao Lu, Biwei Huang, Wang Ke, José Miguel Hernández-Lobato, Kun Zhang, and Bernhard Schölkopf. Sample-Efficient Reinforcement Learning via Counterfactual-Based Data Augmentation. In *NeurIPS 2020 Workshop on Offline Reinforcement Learning*, 2020. URL https://arxiv.org/abs/2012.09092.

Clare Lyle, Amy Zhang, Minqi Jiang, Joelle Pineau, and Yarin Gal. Resolving causal confusion in reinforcement learning via robust exploration. In *ICLR 2021 Workshop on Self-Supervision for Reinforcement Learning*, 2021. URL https://arxiv.org/abs/2104.04351.

Guozheng Ma, Zhen Wang, Zhecheng Yuan, Xueqian Wang, Bo Yuan, and Dacheng Tao. A comprehensive survey of data augmentation in visual reinforcement learning. *International Journal of Computer Vision*, 133:7368–7405, 2025. doi: 10.1007/s11263-025-02472-w. URL https://doi.org/10.1007/s11263-025-02472-w.

Yecheng Jason Ma, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. How far i'll go: Offline goal-conditioned reinforcement learning via f-advantage regression. In *Advances in Neural Information Processing Systems*, volume 35 of *NeurIPS*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/022a39052abf9ca467e268923057dfc0-Abstract.html.

Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, volume 34, pages 5125–5133, 2020. doi: 10.1609/aaai.v34i04.5955. URL https://doi.org/10.1609/aaai.v34i04.5955.

J. L. Mackie. *The Cement of the Universe: A Study of Causation*. Clarendon Press, Oxford, 1980.

Anna Manasyan, Maximilian Seitzer, Filip Radovic, Georg Martius, and Andrii Zadaianchuk. Temporally consistent object-centric learning by contrasting slots. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Daniele Marinazzo, Mario Pellicoro, and Sebastiano Stramaglia. Kernel method for nonlinear Granger causality. *Physical Review Letters*, 100(14):144103, 2008. doi: 10.1103/PhysRevLett.100.144103. URL https://doi.org/10.1103/PhysRevLett.100.144103.

Jarryd Martin, S. Suraj Narayanan, Tom Everitt, and Marcus Hutter. Count-based exploration in feature space for reinforcement learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2471–2478, 2017. doi: 10.24963/ijcai.2017/344. URL https://doi.org/10.24963/ijcai.2017/344.

Atalanti A. Mastakouri, Bernhard Schölkopf, and Dominik Janzing. Necessary and Sufficient Conditions for Causal Feature Selection in Time Series with Latent Common Causes. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 7502–7511. PMLR, 2021. URL https://proceedings.mlr.press/v139/mastakouri21a.html.

Andrew Kachites McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1996. URL https://people.cs.umass.edu/~mccallum/papers/mccallum-thesis.ps.gz.

Christopher Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 403–410, 1995. URL https://arxiv.org/abs/1302.4972.

Alexandra Meliou, Wolfgang Gatterbauer, Katherine F Moore, and Dan Suciu. Why so? or why no? functional causality for explaining query answers. *arXiv preprint arXiv:0912.5340*, 2009.

Willi Menapace, Stéphane Lathuilière, Sergey Tulyakov, Aliaksandr Siarohin, and Elisa Ricci. Playable video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10061–10070, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Menapace_Playable_Video_Generation_CVPR_2021_paper.html.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=8ishA3LxN8.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

Vincent Micheli, Eloi Alonso, and François Fleuret. Efficient world models with context-aware tokenization. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235. PMLR, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236.

Shakir Mohamed and Danilo Jimenez Rezende. Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 2125–2133, 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/hash/f30c6913192078696d5b035f8505e83c-Abstract.html.

Malte Mosbach, Jan Niklas Ewertz, Angel Villar-Corrales, and Sven Behnke. SOLD: Slot object-centric latent dynamics models for relational manipulation learning from pixels. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2025. URL https://openreview.net/forum?id=XOUpHJPYRX&noteId=uBVPRVGhN1.

Melissa Mozifian, Amy Zhang, Joelle Pineau, and David Meger. Intervention Design for Effective Sim2Real Transfer. In *arXiv preprint arXiv:2012.02055*, 2020. URL https://arxiv.org/abs/2012.02055.

Vivek Myers, Chongyi Zheng, Anca Dragan, Sergey Levine, and Benjamin Eysenbach. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, pages 37076–37096. PMLR, 2024. URL https://proceedings.mlr.press/v235/myers24a.html.

Vivek Myers, Catherine Ji, and Benjamin Eysenbach. Horizon generalization in reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=BH8Nrt2dPf.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566, 2018. doi: 10.1109/ICRA.2018.8463189. URL https://doi.org/10.1109/ICRA.2018.8463189.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020. URL https://arxiv.org/abs/2006.09359.

Ashvin V. Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, volume 32 of *NeurIPS*, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/7ec69dd44416c46745f6edd947b470cd-Abstract.html.

Suraj Nair, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Causal Induction from Visual Observations for Goal Directed Tasks. In *arXiv preprint arXiv:1910.01751*, 2019. URL https://arxiv.org/abs/1910.01751.

Akihiro Nakano, Masahiro Suzuki, and Yutaka Matsuo. Interaction-based disentanglement of entities for object-centric world models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=JQc2VowqCzz.

Soroush Nasiriany, Huihan Liu, and Yuke Zhu. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks. In *Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7477–7484, 2022. doi:

10.1109/ICRA46639.2022.9812140. URL https://doi.org/10.1109/ICRA46639.2022.9812140.

Alexander Nikulin, Ilya Zisman, Denis Tarasov, Nikita Lyubaykin, Andrei Polubarov, Igor Kiselev, and Vladislav Kurenkov. Latent action learning requires supervision in the presence of distractors. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. URL https://openreview.net/forum?id=2gcEQCT7QW&noteId=URj8pOdrae.

Guido Novati and Petros Koumoutsakos. Remember and forget for experience replay. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 4851–4860. PMLR, 2019. URL https://proceedings.mlr.press/v97/novati19a.html.

NVIDIA Cosmos Team. Cosmos world foundation model platform for physical AI. *arXiv preprint arXiv:2501.03575*, 2025. URL https://arxiv.org/abs/2501.03575.

Ini Oguntola, Joseph Campbell, Simon Stepputtis, and Katia Sycara. Theory of mind as intrinsic motivation for multi-agent reinforcement learning. *arXiv preprint arXiv:2307.01158*, 2023. URL https://arxiv.org/abs/2307.01158.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. URL https://arxiv.org/abs/1910.07113.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust

visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024. URL https://openreview.net/forum?id=a68SUt6zFt.

Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 2721–2730. PMLR, 2017. URL https://proceedings.mlr.press/v70/ostrovski17a.html.

Pierre-Yves Oudeyer, Frédéric Kaplan, and Verena V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11 (2):265–286, 2007. doi: 10.1109/TEVC.2006.890271. URL https://doi.org/10.1109/TEVC.2006.890271.

Xinlei Pan, Chaowei Xiao, Warren He, Shuang Yang, Jian Peng, Mingjie Sun, Mingyan Liu, Bo Li, and Dawn Song. Characterizing attacks on deep reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, page 10101018, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136.

Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=BGvt0ghNgA.

Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. Controllability-aware unsupervised skill discovery. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pages 27225–27245. PMLR, 2023. URL https://proceedings.mlr.press/v202/park23h.html.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with Hilbert representations. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, pages 39459–39478. PMLR, 2024a. URL https://proceedings.mlr.press/v235/park24g.html.

Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable unsupervised RL with metric-aware abstraction. In *International Conference on Learning Representations (ICLR)*, 2024b. URL https://openreview.net/forum?id=c5pwL0Soay.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-Driven Exploration by Self-Supervised Prediction. In *International Conference on Machine Learning (ICML)*, volume 70, pages 2778–2787. PMLR, 2017. URL https://proceedings.mlr.press/v70/pathak17a.html.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-Supervised Exploration via Disagreement. In *International Conference on Machine Learning (ICML)*, volume 97, pages 5062–5071. PMLR, 2019. URL https://proceedings.mlr.press/v97/pathak19a.html.

Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 2040–2042. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2018. URL https://dl.acm.org/doi/10.5555/3237383.3238064.

Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995. doi: 10.1093/biomet/82.4.669. URL https://doi.org/10.1093/biomet/82.4.669.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009. doi: 10.1017/CBO9780511803161. URL https://doi.org/10.1017/CBO9780511803161.

Judea Pearl and Elias Bareinboim. External validity: From Do-Calculus to transportability across populations. *Statistical Science*, 29(4):579–595, 2014. doi: 10.1214/14-STS486. URL https://doi.org/10.1214/14-STS486.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *Proceedings of the IEEE*

International Conference on Robotics and Automation (ICRA), pages 1–8. IEEE, 2018. doi: 10.1109/ICRA.2018.8460528. URL https://doi.org/10.1109/ICRA.2018.8460528.

Ronan Perry, Julius von Kügelgen, and Bernhard Schölkopf. Causal discovery in heterogeneous environments under the sparse mechanism shift hypothesis. In *Advances in Neural Information Processing Systems*, volume 35 of *NeurIPS*, pages 10904–10917, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/46a126492ea6fb87410e55a58df2e189-Abstract-Conference.html.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*, volume 155, pages 188–204. PMLR, 2021. URL https://proceedings.mlr.press/v155/pertsch21a.html.

Jonas Peters, Joris M. Mooij, Dominik Janzing, and Bernhard Schölkopf. Causal discovery with continuous additive noise models. *Journal of Machine Learning Research*, 15: 2009–2053, 2014. URL https://jmlr.org/papers/v15/peters14a.html.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, 2017. ISBN 9780262037310.

Samuel Pfrommer, Mathew Halm, and Michael Posa. ContactNets: Learning of discontinuous contact dynamics with smooth, implicit representations. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*, volume 155, pages 2279–2291. PMLR, 2020. URL https://proceedings.mlr.press/v155/pfrommer21a.html.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 2817–2826. PMLR, 2017. URL https://proceedings.mlr.press/v70/pinto17a.html.

Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS*, pages 3976–3990, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/294e09f267683c7ddc6cc5134a7e68a8-Abstract.html.

Silviu Pitis, Elliot Creager, Ajay Mandlekar, and Animesh Garg. MoCoDA: Model-based counterfactual data augmentation. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/7314e20a73542bbfff25030d1185ce88-Abstract.html.

Vitchyr H. Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 7783–7792. PMLR, 2020. URL https://proceedings.mlr.press/v119/pong20a.html.

Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. doi: 10.1109/TNNLS.2023.3250269. URL https://doi.org/10.1109/TNNLS.2023.3250269.

Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34 of *NeurIPS*, pages 5402–5415, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/2b38c2df6a49b97f706ec9148ce48d86-Abstract.html.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. EPOpt: Learning robust neural network policies using model ensembles. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=SyWvgP5el.

Jorge Ramírez, Wen Yu, and Adolfo Perrusquía. Model-free reinforcement learning from expert demonstrations: A survey. *Artificial Intelligence Review*, 55(4):3213–3241, 2022. doi: 10.1007/s10462-021-10085-1. URL https://doi.org/10.1007/s10462-021-10085-1.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ha6RTeWMd0.

Balaraman Ravindran and Andrew G. Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi-markov decision processes. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1011–1016. Morgan Kaufmann, 2003.

Patrik Reizinger, Yash Sharma, Matthias Bethge, Bernhard Schölkopf, Ferenc Huszár, and Wieland Brendel. Jacobian-based causal discovery with nonlinear ICA. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=2Yo9xqR6Ab.

Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. In *Advances in Neural Information Processing Systems*, volume 32 of *NeurIPS*, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/57db7d68d5335b52d5153a4e01adaa6b-Abstract.html.

Paul Rolland, Volkan Cevher, Matthäus Kleindessner, Chris Russell, Dominik Janzing, Bernhard Schölkopf, and Francesco Locatello. Score matching enables causal discovery of nonlinear additive noise models. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162, pages 18741–18753. PMLR, 2022. URL https://proceedings.mlr.press/v162/rolland22a.html.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32 of *NeurIPS*, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Abstract.html.

Donald B. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688–701, 1974. doi: 10.1037/h0037350. URL https://doi.org/10.1037/h0037350.

Reuven Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997. doi: 10.1016/S0377-2217(96)00385-2. URL https://doi.org/10.1016/S0377-2217(96)00385-2.

Andrei A. Rusu, Sergio Gómez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *International Conference on Learning Representations (ICLR)*, 2016a. URL https://arxiv.org/abs/1511.06295.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016b. URL https://arxiv.org/abs/1606.04671.

Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real single-image flight without a single real image. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017. doi: 10.15607/RSS.2017.XIII.034. URL http://www.roboticsproceedings.org/rss13/p34.html.

Christoph Salge, Cornelius Glackin, and Daniel Polani. Changing the environment based on empowerment as intrinsic motivation. *Entropy*, 16(5):2789–2819, 2014. doi: 10.3390/e16052789. URL https://doi.org/10.3390/e16052789.

Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. Deep energy estimator networks. *arXiv preprint arXiv:1805.08306*, 2018. URL https://arxiv.org/abs/1805.08306.

Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=SkeK3s0qKQ.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=rvUq3cxpDF.

Simon Schmitt, Matteo Hessel, and Karen Simonyan. Off-policy actor-critic with shared experience replay. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 8545–8554. PMLR, 2020. URL https://proceedings.mlr.press/v119/schmitt20a.html.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021. doi: 10.1109/JPROC.2021.3058954. URL https://doi.org/10.1109/JPROC.2021.3058954.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine*

Learning (ICML), volume 37, pages 1889–1897. PMLR, 2015. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Atharva Sehgal, Arya Grayeli, Jennifer J. Sun, and Swarat Chaudhuri. Neurosymbolic grounding for compositional world models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=4KZpDGD4Nh.

Maximilian Seitzer, Bernhard Schölkopf, and Georg Martius. Causal influence detection for improving efficiency in reinforcement learning. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/c1722a7941d61aad6e651a35b65a9c3e-Abstract.html.

Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, and Francesco Locatello. Bridging the gap to real-world object-centric learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=b9tUk-f_aG.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=HJgLZR4KvH.

Xinwei Shen, Furui Liu, Hanze Dong, Qing Lian, Zhitang Chen, and Tong Zhang. Weakly supervised disentangled generative causal representation learning. *Journal of Machine Learning Research*, 23(241):1–55, 2022. URL https://jmlr.org/papers/v23/21-0080.html.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 5779–5788. PMLR, 2019. URL https://proceedings.mlr.press/v97/shyam19a.html.

Harshit Sikchi, Siddhant Agarwal, Pranaya Jajoo, Samyak Parajuli, Caleb Chuck, Max Rudolph, Peter Stone, Amy Zhang, and Scott Niekum. RLZero: Direct policy inference from language without in-domain supervision. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=eyH8QLn2Qx.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, volume 32, pages 387–395. PMLR, 2014. URL https://proceedings.mlr.press/v32/silver14.html.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529 (7587):484–489, 2016. doi: 10.1038/nature16961. URL https://doi.org/10.1038/nature16961.

Gautam Singh, Yeongbin Kim, and Sungjin Ahn. Neural systematic binder. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=ZPHE4fht19t.

Gautam Singh, Yue Wang, Jiawei Yang, Boris Ivanovic, Sungjin Ahn, Marco Pavone, and Tong Che. Parallelized spatiotemporal slot binding for videos. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235. PMLR, 2024. URL https://proceedings.mlr.press/v235/singh24g.html.

Krishnakant Singh, Simone Schaub-Meyer, and Stefan Roth. GLASS: Guided latent slot diffusion for object-centric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Singh_GLASS_Guided_Latent_Slot_Diffusion_for_Object-Centric_Learning_CVPR_2025_paper.pdf.

Samarth Sinha, Jiaming Song, Animesh Garg, and Stefano Ermon. Experience replay with likelihood-free importance weights. In *Proceedings of the 4th Annual Learning for Dynamics and Control Conference (L4DC)*, volume 168, pages 110–123. PMLR, 2022. URL https://proceedings.mlr.press/v168/sinha22a.html.

Anoop Sonar, Vincent Pacelli, and Anirudha Majumdar. Invariant policy optimization: Towards stronger generalization in reinforcement learning. In *Proceedings of the 3rd Annual Learning for Dynamics and Control Conference (L4DC)*, volume 144, pages 21–33. PMLR, 2021. URL https://proceedings.mlr.press/v144/sonar21a.html.

Wonil Song, Sangryul Jeon, Hyesong Choi, Kwanghoon Sohn, and Dongbo Min. Learning disentangled skills for hierarchical reinforcement learning through trajectory autoencoder with weak labels. *Expert Systems with Applications*, 230:120625, 2023. doi: 10.1016/j.eswa.2023.120625. URL https://doi.org/10.1016/j.eswa.2023.120625.

Yang Song and Diederik P. Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021. URL https://arxiv.org/abs/2101.03288.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 115, pages 574–584. PMLR, 2020. URL https://proceedings.mlr.press/v115/song20a.html.

Sumedh A. Sontakke, Arash Mehrjou, Laurent Itti, and Bernhard Schölkopf. Causal curiosity: RL agents discovering self-supervised experiments for causal representation learning. In

Proceedings of the 38th International Conference on Machine Learning (ICML), volume 139, pages 9848–9858. PMLR, 2021. URL https://proceedings.mlr.press/v139/sontakke21a.html.

Alessandro Sordoni, Nouha Dziri, Hannes Schulz, Geoff Gordon, Philip Bachman, and Remi Tachet Des Combes. Decomposed mutual information estimation for contrastive representation learning. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pages 9859–9869. PMLR, 2021. URL https://proceedings.mlr.press/v139/sordoni21a.html.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Springer, New York, NY, 1993.

Peter Spirtes, Clark N. Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT Press, 2nd edition, 2000. doi: 10.7551/mitpress/1754.001.0001. URL https://mitpress.mit.edu/9780262194402/causation-prediction-and-search/.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 4732–4741. PMLR, 2018. URL https://proceedings.mlr.press/v80/srinivas18b.html.

Bradly C. Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models. In *arXiv preprint arXiv:1507.00814*, 2015. URL https://arxiv.org/abs/1507.00814.

Wolfgang Stammer, Antonia Wüst, David Steinmann, and Kristian Kersting. Neural concept binder. In *Advances in Neural Information Processing Systems*, volume 37 of *NeurIPS*, pages 71792–71830, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/8402cf3031d649066ada24514739f0dd-Abstract-Conference.html.

Patrick A. Stokes and Patrick L. Purdon. A study of problems encountered in Granger causality analysis from a neuroscience perspective. *Proceedings of the National Academy of Sciences*, 114(34):E7063–E7072, 2017. doi: 10.1073/pnas.1704663114. URL https://doi.org/10.1073/pnas.1704663114.

Peiquan Sun, Wengang Zhou, and Houqiang Li. Attentive experience replay. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, volume 34, pages 5900–5907, 2020. doi: 10.1609/aaai.v34i04.6049. URL https://doi.org/10.1609/aaai.v34i04.6049.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988. doi: 10.1007/BF00115009. URL https://doi.org/10.1007/BF00115009.

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning (ICML)*, pages 216–224. Morgan Kaufmann, 1990.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin*, 2(4):160–163, 1991. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

Richard Stuart Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst, 1984.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. #Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing*

Systems 30 (NIPS), pages 2753–2762, 2017. URL https://papers.nips.cc/paper/6868-exploration-a-study-of-count-based-exploration-for-deep-reinforcement-learning.

Alex Tank, Ian Covert, Nick Foti, Ali Shojaie, and Emily B. Fox. Neural Granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4267–4279, 2022. doi: 10.1109/TPAMI.2021.3065601. URL https://doi.org/10.1109/TPAMI.2021.3065601.

Matthew E. Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(73):2125–2167, 2007. URL https://jmlr.org/papers/v8/taylor07a.html.

R. Tedrake. LQR-trees: Feedback motion planning on sparse randomized trees. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009. doi: 10.15607/RSS.2009.V.003.

Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4: 1235–1260, 2003. URL https://jmlr.org/papers/v4/teh03a.html.

Panagiotis Tigas, Yashas Annadani, Andrew Jesson, Bernhard Schölkopf, Yarin Gal, and Stefan Bauer. Interventions, where and how? experimental design for causal models at scale. In *Advances in Neural Information Processing Systems*, volume 35 of *NeurIPS*, pages 24130–24143, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/98a5c0470e57d518ade4e56c6ee0b363-Abstract-Conference.html.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017. doi: 10.1109/IROS.2017.8202133. URL https://doi.org/10.1109/IROS.2017.8202133.

Manan Tomar, Amy Zhang, Roberto Calandra, Matthew E. Taylor, and Joelle Pineau. Model-invariant state abstractions for model-based reinforcement learning. In *arXiv preprint arXiv:2102.09850*, 2021. URL https://arxiv.org/abs/2102.09850.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In *Advances in Neural Information Processing Systems*, volume 34 of *NeurIPS*, pages 13–23, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/003dd617c12d444ff9c80f717c3fa982-Abstract.html.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=MYEap_OcQI.

Hung-Yu Tseng, Lu Jiang, Ce Liu, Ming-Hsuan Yang, and Weilong Yang. Regularizing generative adversarial networks under limited data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7921–7931, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Tseng_Regularizing_Generative_Adversarial_Networks_Under_Limited_Data_CVPR_2021_paper.html.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. doi: 10.1016/j.simpa.2020.100022. URL https://doi.org/10.1016/j.simpa.2020.100022.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018. URL https://arxiv.org/abs/1812.01717.

Núria Armengol Urpí, Marco Bagatella, Marin Vlastelica, and Georg Martius. Causal action influence aware counterfactual data augmentation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30 of *NeurIPS*, 2017. URL https://papers.nips.cc/paper/7210-neural-discrete-representation-learning.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. URL https://arxiv.org/abs/1807.03748.

Matej Vecerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017. URL https://arxiv.org/abs/1707.08817.

Rahul Venkatesh, Honglin Chen, Kevin T. Feigelis, Daniel M. Bear, Khaled Jedoui, Klemen Kotar, Felix J. Binder, Wanhee Lee, Sherry Liu, Kevin A. Smith, Judith E. Fan, and Daniel L. K. Yamins. Counterfactual world modeling for physical dynamics understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 368–387, 2024. URL https://arxiv.org/abs/2312.06721.

Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS*, pages 4235–4246, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/2c6a0bae0f071cbbf0bb3d5b11d90a82-Abstract.html.

Angel Villar-Corrales and Sven Behnke. PlaySlot: Learning inverse latent dynamics for controllable object-centric video prediction and planning. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. URL https://openreview.net/forum?id=P9gwpJDo8i.

Sergei Volodin, Nevan Wichers, and Jeremy Nixon. Resolving spurious correlations in causal models of environments via interventions. *arXiv preprint arXiv:2002.05217*, 2020. URL https://arxiv.org/abs/2002.05217.

Jianren Wang, Yujie Lu, and Hang Zhao. CLOUD: Contrastive learning of unsupervised dynamics. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*, volume 155, pages 1004–1018. PMLR, 2021a. URL https://proceedings.mlr.press/v155/wang21c.html.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS*, pages 7968–7978, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/5a751d6a0b6ef05cfe51b86e5d1458e6-Abstract.html.

Tongzhou Wang, Simon S. Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised MDPs: Learning world models better than the world itself. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162, pages 22591–22612. PMLR, 2022a. URL https://proceedings.mlr.press/v162/wang22c.html.

Xiaoqiang Wang, Yali Du, Shengyu Zhu, Liangjun Ke, Zhitang Chen, Jianye Hao, and Jun Wang. Ordering-based causal discovery with reinforcement learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3566–3573, 2021b. doi: 10.24963/ijcai.2021/491. URL https://doi.org/10.24963/ijcai.2021/491.

Zizhao Wang, Xuesu Xiao, Yuke Zhu, and Peter Stone. Task-independent causal state abstraction. In *Proceedings of the 35th International Conference on Neural Information Processing Systems, Robot Learning workshop*, 2021c.

Zizhao Wang, Xuesu Xiao, Zifan Xu, Yuke Zhu, and Peter Stone. Causal dynamics learning for task-independent state abstraction. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23151–23180. PMLR, 17–23 Jul 2022b. URL https://proceedings.mlr.press/v162/wang22ae.html.

Zizhao Wang, Jiaheng Hu, Peter Stone, and Roberto Martín-Martín. ELDEN: Exploration via local dependencies. In *Advances in Neural Information Processing Systems*, volume 36 of *NeurIPS*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/31ed129feae64a7e44a15b148c15558d-Abstract-Conference.html.

Zizhao Wang, Jiaheng Hu, Caleb Chuck, Stephen Chen, Roberto Martín-Martín, Amy Zhang, Scott Niekum, and Peter Stone. SkiLD: Unsupervised skill discovery guided by factor interactions. In *Advances in Neural Information Processing Systems*, volume 37 of *NeurIPS*, 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/8e3b2fb66db1930ae7fcee712770e48a-Abstract-Conference.html.

Zizhao Wang, Caroline Wang, Xuesu Xiao, Yuke Zhu, and Peter Stone. Building minimal and reusable causal state abstractions for reinforcement learning. In *Proceedings of the 38th Annual AAAI Conference on Artificial Intelligence*, volume 38, pages 15778–15786, 2024b. doi: 10.1609/aaai.v38i14.29507. URL https://doi.org/10.1609/aaai.v38i14.29507.

Zizhao Wang, Kaixin Wang, Li Zhao, Peter Stone, and Jiang Bian. Dyn-o: Building structured world models with object-centric representations. In *Advances in Neural Information Processing Systems*, volume 38 of *NeurIPS*, 2025. URL https://openreview.net/forum?id=b2u1yrTwFK.

David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=r1eVMnA9K7.

Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, United Kingdom, 1989.

Martin Weiss, Nasim Rahaman, Francesco Locatello, Chris Pal, Yoshua Bengio, Bernhard Schölkopf, Erran Li Li, and Nicolas Ballas. Neural Attentive Circuits. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 7741–7754, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/32f227c41a0b4e36f65bebb4aeda94a2-Abstract-Conference.html.

Max Welling and Geoffrey E. Hinton. A new learning algorithm for mean field Boltzmann machines. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 251–256. Springer, 2002. doi: 10.1007/3-540-46084-5_57. URL https://doi.org/10.1007/3-540-46084-5_57.

Peter Whittle. *Optimization Over Time*. John Wiley & Sons, Inc., 1982.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic model predictive control for model-based reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017. doi: 10.1109/ICRA.2017.7989202. URL https://doi.org/10.1109/ICRA.2017.7989202.

Sewall Wright. Correlation and causation. *Journal of Agricultural Research*, 20(7):557–585, 1921.

Yi-Fu Wu, Minseung Lee, and Sungjin Ahn. Object-centric semantic vector quantization. In *UniReps: The First Workshop on Unifying Representations in Neural Models*, 2023a.

Yifan Wu, George Tucker, and Ofir Nachum. The Laplacian in RL: Learning representations with efficient approximations. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=HJlNpoA5YQ.

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. SlotFormer: Unsupervised visual dynamics simulation with object-centric models. In *International Conference on Learning Representations (ICLR)*, 2023b. URL https://openreview.net/forum?id=TFbwV6I0VLg.

Ziyi Wu, Jingyu Hu, Wuyue Lu, Igor Gilitschenski, and Animesh Garg. SlotDiffusion: Object-centric generative modeling with diffusion models. In *Advances in Neural Information Processing Systems*, volume 36 of *NeurIPS*, 2023c. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/9fa03b16dbd6cabc7601fe98c6ec291e-Abstract-Conference.html.

Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022. doi: 10.1038/s41586-021-04357-7. URL https://doi.org/10.1038/s41586-021-04357-7.

Mengjiao Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. CausalVAE: Disentangled representation learning via neural structural causal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9593–9602, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Yang_CausalVAE_Disentangled_Representation_Learning_via_Neural_Structural_Causal_Models_CVPR_2021_paper.html.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from

videos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=VYOe2eBQeh.

Weirui Ye, Yunsheng Zhang, Pieter Abbeel, and Yang Gao. Become a proficient player with limited data through watching pure videos. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=Sy-o2N0hF4f.

Jaesik Yoon, Yi-Fu Wu, Heechul Bae, and Sungjin Ahn. An investigation into pre-training object-centric representations for reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. URL https://proceedings.mlr.press/v202/yoon23c.html.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS*, pages 14129–14142, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/a322852ce0df73e204b7e67cbbef0d0a-Abstract.html.

Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Object-centric learning for real-world videos by predicting temporal feature similarities. In *Advances in Neural Information Processing Systems*, volume 36 of *NeurIPS*, pages 57286–57303, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/c1fdec0d7ea1affa15bd09dd0fd3af05-Abstract-Conference.html.

Daochen Zha, Kwei-Herng Lai, Kaixiong Zhou, and Xia Hu. Experience replay optimization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4243–4249, 2019. doi: 10.24963/ijcai.2019/589. URL https://doi.org/10.24963/ijcai.2019/589.

Amy Zhang, Zachary C. Lipton, Luis Pineda, Kamyar Azizzadenesheli, Anima Anandkumar, Laurent Itti, Joelle Pineau, and Tommaso Furlanello. Learning causal state representations

of partially observable environments. *arXiv preprint arXiv:1906.10437*, 2019a. URL https://arxiv.org/abs/1906.10437.

Amy Zhang, Zachary C. Lipton, Luis Pineda, Kamyar Azizzadenesheli, Anima Anandkumar, Laurent Itti, Joelle Pineau, and Tommaso Furlanello. Learning Causal State Representations of Partially Observable Environments. In *arXiv preprint arXiv:1906.10437*, 2019b. URL https://arxiv.org/abs/1906.10437.

Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and Doina Precup. Invariant Causal Prediction for Block MDPs. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 11214–11224. PMLR, 2020. URL https://proceedings.mlr.press/v119/zhang20t.html.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations (ICLR)*, 2021a. URL https://openreview.net/forum?id=-2FCwDKRREu.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations (ICLR)*, 2018a. URL https://openreview.net/forum?id=r1Ddp1-Rb.

Jesse Zhang, Haonan Yu, and Wei Xu. Hierarchical reinforcement learning by discovering intrinsic options. In *International Conference on Learning Representations (ICLR)*, 2021b. URL https://openreview.net/forum?id=r-gPPHEjpmw.

Qihang Zhang, Zhenghao Peng, and Bolei Zhou. Learning to drive by watching youtube videos: Action-conditioned contrastive policy pretraining. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 111–128, Cham, 2022a. Springer Nature Switzerland. ISBN 978-3-031-19809-0.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018b. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html.

Tianjun Zhang, Benjamin Eysenbach, Ruslan Salakhutdinov, Sergey Levine, and Joseph E. Gonzalez. C-planning: An automatic curriculum for learning goal-reaching tasks. In *International Conference on Learning Representations (ICLR)*, 2022b. URL https://openreview.net/forum?id=K2JfSnLBD9.

Linfeng Zhao, Lingzhi Kong, Robin Walters, and Lawson L.S. Wong. Toward compositional generalization in object-oriented world modeling. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162, pages 26841–26864. PMLR, 2022. URL https://proceedings.mlr.press/v162/zhao22b.html.

Rui Zhao and Volker Tresp. Energy-based hindsight experience prioritization. In *Proceedings of the 2nd Conference on Robot Learning (CoRL)*, volume 87, pages 113–122. PMLR, 2018. URL https://proceedings.mlr.press/v87/zhao18a.html.

Rui Zhao, Yang Gao, Pieter Abbeel, Volker Tresp, and Wei Xu. Mutual information state intrinsic control. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=OthEq8I5v1.

Chongyi Zheng, Benjamin Eysenbach, Homer Rich Walke, Patrick Yin, Kuan Fang, Ruslan Salakhutdinov, and Sergey Levine. Stabilizing contrastive RL: Techniques for robotic goal reaching from offline data. In *International Conference on Learning Representations (ICLR)*, 2024a. URL https://openreview.net/forum?id=Xkf2EBj4w3.

Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive difference predictive coding. In *International Conference on Learning Representations (ICLR)*, 2024b. URL https://openreview.net/forum?id=0akLDTFR9x.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 9472–9483, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e347c51419ffb23ca3fd5050202f9c3d-Abstract.html.

Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. RoboDreamer: Learning compositional world models for robot imagination. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235. PMLR, 2024. URL https://arxiv.org/abs/2404.12377.

Shengyu Zhu, Ignavier Ng, and Zhitang Chen. Causal Discovery with Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2020a. URL https://openreview.net/forum?id=S1g2skStPB.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020b.

Zhengmao Zhu, Honglong Tian, Xionghui Chen, Kun Zhang, and Yang Yu. Offline model-based reinforcement learning with causal structured world models. *Frontiers of Computer Science*, 19(4):194347, 2025.