

Learning to Reach Goals from Suboptimal Demonstrations via World Models

by

Qasim Ali

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2025

© Qasim Ali 2025

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis builds upon work from the following paper: *MR-CRL: Leveraging Predictive Representations for Contrastive Goal-Conditioned Reinforcement Learning* [1], by Muhammad Qasim Ali, Winnie Trandinh, Howard Nguyen-Huu, and Alexander Wong, published in the *RLBrew: Ingredients for Developing Generalist Agents workshop (RLC 2025)*.

I was the first author and made the primary contributions to the formulation, method design, experiment design, implementation, execution of experiments, analysis of results, and writing of the paper (introduction, methodology, and results). The other authors contributed through editing, figure preparation, and discussion and interpretation of results.

This thesis differs from the paper in several ways, including a stronger motivation, a modified method (utilizing a different loss function for training the model), an enhanced experimental setup, additional ablation studies, and a more in-depth and insightful analysis of the results.

Abstract

A central challenge for training autonomous agents is the scarcity of high-quality and long-horizon demonstrations. Unlike fields such as natural language or computer vision—where abundant internet data exists—many robotics and decision-making domains lack large, diverse, and high-quality datasets. One underutilized resource is leveraging suboptimal demonstrations, which are easier to collect and potentially more abundant.

This limitation is particularly pronounced in goal-conditioned reinforcement learning (GCRL), where agents must learn to reach diverse goal states from limited demonstrations. While methods such as contrastive reinforcement learning (CRL) show promising scaling behavior when given access to abundant and high-quality training demonstrations, they struggle when demonstrations are suboptimal. In particular, when training demonstrations are short or exploratory, CRL struggles to generalize beyond the training demonstrations, and the resulting policy exhibits lower success rates.

To overcome this, we explore the use of self-supervised representation learning to extract general-purpose representations from demonstrations. The intuition is that if an agent can first learn robust representations of environment dynamics—without relying on demonstration optimality—it can then use these representations to guide reinforcement learning more effectively. Such representations can serve as a bridge between noisy demonstrations and goal-directed control, allowing policies to learn faster.

In this thesis, we propose World Model Contrastive Reinforcement Learning (WM-CRL), which augments CRL with representations from a world model (WM). The world model is trained to anticipate future state embeddings from past state-action pairs, thereby encoding the dynamics of the environment. As the world model aims to only learn environment dynamics, it can leverage both high and low quality demonstrations. By integrating these world model embeddings into CRL’s framework, it can help CRL more easily comprehend the environment dynamics and select actions that more effectively achieve its goals.

We evaluate WM-CRL on tasks from the OGBench benchmark. We explore performance on multiple locomotion and manipulation environments and multiple datasets varying in quality.

Our results show that WM-CRL can substantially improve performance over CRL in suboptimal-data settings, such as stitching short trajectories or learning from exploratory behavior. However, we find our method offers limited benefit when abundant expert demonstrations are available. Ablation studies further reveal that success depends critically on the stability of world model training and on how its embeddings are integrated into the agent’s architecture.

Acknowledgements

I want to thank my supervisor, Professor Alex, for his unwavering support and guidance. I would also like to thank my co-supervisor Professor Javad Shafiee for his valuable advice on the video generation project, which significantly contributed to my work.

To Professor Yuhao Chen, thank you for your constant presence, insightful advice about paper writing, helping me recruit URAs, and for always being available to offer your expertise and support.

To Winnie Trandinh, for providing critical discussions throughout many of my successful and unsuccessful endeavours over the course of my master's.

I am grateful to Frank Mokadem for providing much-needed motivation, humor, and levity during the challenging times in my second year.

I am grateful to Griffin Floto and Shahbuland Matiana for providing blunt research advice, particularly on prioritizing time and ideas as a researcher. Your advice and direction are the sole reason I changed my course in my second year.

My sincerest gratitude goes to Saejith Nair. Our daily conversations about AI, technology, and agency were pivotal in helping me develop as an engineer, researcher, and person.

Dedication

This thesis is dedicated to my family and friends, whose unwavering support and encouragement were my foundation. Your belief in me, especially during the most challenging moments, made this work possible.

Table of Contents

Author’s Declaration	ii
Statement of Contributions	iii
Abstract	iv
Acknowledgements	v
Dedication	vi
List of Figures	ix
List of Tables	x
1 Introduction	1
1.1 Contributions	3
2 Background	5
2.1 Supervised Learning	6
2.2 Self-supervised Representation Learning	8
2.2.1 Contrastive Learning	9
2.2.2 Self-Distillation Methods	15
2.3 Reinforcement Learning	19

2.3.1	Formulating Goal-conditioned Reinforcement Learning	20
2.3.2	Model-based vs Model-free Reinforcement Learning	21
2.3.3	Actor-critic Algorithms	23
2.3.4	Contrastive Reinforcement Learning	24
2.4	Related works in Representation Learning for Reinforcement Learning . . .	27
3	Methodology	29
3.1	Motivation	29
3.2	Training the World Model	31
3.3	Integrating Model-Based State Representations into CRL	34
3.4	Training Procedure	36
4	Experiments	38
4.1	Experimental Setup	38
4.2	Comparison with CRL	42
4.2.1	Analysis	43
4.3	Ablations on Integrating Representations with CRL	55
4.4	Ablations on World Model	58
5	Conclusion	65
5.1	Summary of Contributions	65
5.2	Future Research Directions	66
5.2.1	Limitations within the Current Setting	66
5.2.2	Extensions beyond this work	67
	References	69

List of Figures

2.1	Diagram of BYOL’s training loss	16
2.2	Diagram of DINO’s training loss	18
2.3	Actor network during inference (left) and training (right).	23
2.4	Contrastive Reinforcement Learning’s critic network training and positive-negative pair curation.	25
3.1	World model networks and representation	31
3.2	World model loss	33
3.3	Integration of world model embeddings into CRL	34
4.1	Sample environments from OGBench	39
4.2	Point Maze Environment	46
4.3	Ant Maze Environment	48
4.4	Humanoid Maze Environment	50
4.5	Cube Environment	52
4.6	Puzzle Environment	54
4.7	World Model Loss Curves for Ablations	60

List of Tables

4.1	Environment specifications	42
4.2	Comparison of WM-CRL and CRL	44
4.3	Ablation of different integration strategies into CRL’s Actor-Critic	56
4.4	Ablation of different integration strategies into CRL’s Critic components	57
4.5	Ablation on World Model loss (L2 vs CE)	60
4.6	Ablation World Model design choices, when training World Model with L2 loss	61
4.7	Ablation World Model design choices, when training World Model with CE loss	62

Chapter 1

Introduction

A longstanding goal in artificial intelligence (AI) is the development of *autonomous agents* that can act effectively in complex and dynamic environments. Such agents hold promise across a wide range of applications: robotic systems that automate physical tasks, coding assistants that support software development, scientific agents that help analyze experiments, and video game bots [51].

Recent advances in AI have demonstrated remarkable success in domains such as natural language processing [9, 4], image generation [52], visual understanding [19], and game playing [41, 60, 45]. These achievements have largely been driven by unsupervised and self-supervised learning strategies trained with massive computational resources on vast datasets.

Critical to the success of AI in these domains was the availability of large and diverse datasets [50, 57, 41]. Curating these large pre-training datasets has been relatively straightforward in many of these domains because human behavior was already being documented. For example, human activity on the internet has already generated massive repositories of books, articles, and code. Consequently, foundation models can effectively learn long-horizon and diverse patterns of behavior from these pre-training datasets.

In contrast, many agentic domains lack such abundant data. In robotics, for example, high-quality datasets containing diverse examples of successful trajectories are scarce and expensive to collect [66]. Many critical robotics tasks require are long-horizon, requiring multiple stages of planning, acting, and reflecting [37]. Traditional supervised and unsupervised learning approaches would demand large numbers of such demonstrations, which are costly to collect.

Representation learning has historically been employed as a means to mitigate the challenge of data scarcity [2]. In domains such as image classification [33], the limited availability of high-quality labeled data has motivated approaches that go beyond purely supervised learning [7, 30]. Instead, these methods leverage alternative data sources besides their supervised training set, such as data from other tasks or unlabeled samples.

This raises a natural question: can an analogous strategy be applied to autonomous agents, enabling them to acquire useful representations from abundant but suboptimal data, and subsequently transfer these representations to data-scarce decision-making tasks?

Our work builds on a growing body of research that integrates representation learning with reinforcement learning [62], enabling agents to acquire more insights from the same training demonstrations. We focus here on integrating representation learning into *goal-reaching agents*, where agents select actions in order to reach a designated goal state.

One of the many approaches to building goal-reaching agents, *Contrastive Reinforcement Learning* (CRL) [12] has emerged as especially promising. At its core, CRL is an actor-critic framework [32]. The critic learns to evaluate whether a given state-action pair is likely to lead to a goal, while the actor uses the critic’s feedback to choose actions that will take the agent closer to a given goal. What sets CRL apart is its use of contrastive representation, a scalable representation learning approach [38]. This usage of contrastive learning helps CRL to effectively scale with high-quality demonstrations [49].

In this work, we seek to enhance the performance of CRL when training demonstrations are suboptimal—such as when demonstrations are short relative to the behaviours required at test-time, when demonstrations are noisy (inefficiently reaching goal state), or when demonstrations are collected from highly exploratory (non-goal oriented) behaviour.

Our method, termed **World Model Contrastive Reinforcement Learning** (WM-CRL), incorporates representations from a world model into the CRL framework. In particular, we propose training a World Model (WM) [23] to predict future state embeddings given the past state and action. In doing so, it learns the underlying physics of the environment and how actions influence future states. We then integrate the representations from the world model into the CRL framework (either its critic or actor networks). The WM representations can help CRL’s critic evaluate whether a given state-action pair bring the agent closer to a goal state and can help CRL’s actor network learn what actions bring the agent closer to the designated goal state.

Crucially, because the world model’s training objective encourages it to learn only the environment dynamics, it can be trained on a wide variety of data sources. This includes not only successful demonstrations of a task, but any demonstration showcasing interaction

with the environment. This flexibility makes world models a powerful solution to the data scarcity problem in reinforcement learning.

We evaluate WM-CRL against a standard CRL baseline on a diverse benchmark of virtual environments. The benchmark spans both locomotion tasks, such as maze navigation, and manipulation tasks, such as robotic pick-and-place with cube objects. Importantly, we explore datasets varying in quality: some contain complete expert demonstrations, while others consist of noisy, exploratory, or fragmented trajectories. This variation provides a natural test of an agent’s ability to generalize beyond the training distribution.

Our results demonstrate that WM-CRL yields improved performance over CRL when training data is suboptimal. For example, we demonstrate improvements when the agent is limited to short-horizon training trajectories or exploratory demonstrations generated by random, goal-agnostic actions. In particular, WM-CRL improved performance on 5 out of the 6 suboptimal training tasks we tested on (Section 4.2). The results highlight that world model representations can indeed help CRL agents better generalize beyond their training data. However, we find that the additional representations provided by the world model offer limited benefit when abundant high-quality trajectory data is readily available and simply cloning training behaviour is sufficient for solving the task.

To better understand the design choices underlying WM-CRL, we conduct an extensive ablation study. We start by investigating different mechanisms for incorporating world model embeddings into the CRL framework, varying which components of the CRL receive world model embeddings. Then we examine alternative loss functions and training strategies for the world model, and evaluate how that affects the overall CRL performance.

These ablations reveal that the utility of world model representations depends on two key factors. First, the stability of the world model’s training loss effects how informative its representations are for CRL. We find that when the world model’s loss is smooth and monotonic over training, the more likely its representations are to help CRL. Second, the benefit of these representations depends on how and where they are integrated into the CRL framework. Different components of the agent exploit the embeddings in complementary ways, and the choice of whether to provide them to the critic, the actor, or the goal encoder can significantly influence overall performance. We did not find that any particular integration strategy performed the best across all tasks/environments.

1.1 Contributions

This thesis makes the following contributions:

1. **World Model representations for Contrastive Reinforcement Learning (WM-CRL).**

We introduce WM-CRL, a new framework that augments Contrastive Reinforcement Learning with representations from a predictive world model. This is the first work to explore using world model representations in the contrastive reinforcement learning framework.

We conduct extensive ablations 4.3 that test different mechanisms for incorporating world model embeddings into CRL (e.g., critic, actor, and goal encoder). These studies highlight how the placement and stability of predictive embeddings affect downstream reinforcement learning, providing practical guidance for future model-based representation learning methods.

2. **Recasting World Model Representation Learning as Self-Distillation**

We reinterpret world model training through the lens of self-distillation representation learning methods such as BYOL [21] and DINO [6]. In particular, we interpret the process of training world models to be self-consistent as a form of self-distillation. This framing was not used in prior world model representation learning for reinforcement learning work [16, 23]. Using this perspective, we introduce novel design choices for world model training, including a cross-entropy loss over latent dynamics and exponential moving average (EMA) target networks. We find that these changes can improve performance on some tasks (see Ablations 4.4).

3. **Generalization under Data Scarcity.**

We perform one of the first systematic evaluations of world-model embeddings across datasets of varying quality (expert, noisy, exploratory, and fragmented). Our results demonstrate that WM-CRL consistently improves performance when generalization beyond the training distribution is required, while clarifying its limitations in data-rich settings.

Chapter 2

Background

This thesis explores how world models can provide generalizable representations for contrastive reinforcement learning. The central challenge motivating this method is the scarcity of high-quality data for training goal-reaching agents. In this thesis, we aim to find methods to learn from demonstrations that are short in length, collected from noisy policies that inefficiently reach the goal state, or collected from exploratory policies without any goal. The hypothesis is that representations from world models will allow agents to generalize from suboptimal training demonstrations. Our experiments in particular focus on simulated locomotion and manipulation benchmarks, where the task is to reach various goal states from an initial state.

WM-CRL builds on self-supervised representation learning and integrates it into reinforcement learning. While methods like contrastive learning and bootstrapping can extract rich representations without labels, offline reinforcement learning often struggles with limited, high-quality demonstrations. WM-CRL bridges these ideas by using world models to capture environment dynamics and feeding their representations into contrastive reinforcement learning, improving policy learning and generalization under suboptimal training demonstrations.

We start by describing traditional supervised learning and how transfer learning can make the supervised task easier [2.1](#). In the Representation Learning Section [2.2](#), we discuss methods for learning general-purpose representations from unlabelled data. We discuss two methods from the literature that are used in WM-CRL: learning representations by contrasting positive-negative pairs and learning representations by bootstrapping your own labels. Finally, in the Reinforcement Learning Section [2.3](#), we formulate the goal-conditioned reinforcement learning problem, discuss the actor-critic framework, and discuss the con-

trastive reinforcement learning framework.

2.1 Supervised Learning

Supervised learning is a machine learning approach in which predictive models are trained on labeled data to learn a mapping from an input domain to an output range [64]. In this setting, we are given a collection of training examples $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^n$, where each input $x^{(i)}$ has a known label $y^{(i)}$. The goal is to learn a function f_θ that maps inputs to outputs and generalizes well to unseen data. To measure this, the dataset is split into two parts: a *training set*, which is used to fit the model, and a *test set*, which is used only to evaluate how well the model generalizes. The effectiveness of a supervised learning method is assessed by its ability to generalize, as measured by its error on held-out test data.

Empirical risk minimization. The optimal supervised model minimizes the expected error over the true data distribution. Since the true distribution is unknown, we instead minimize the *empirical risk*, the average loss ($\ell(\cdot)$) computed only on the training set:

$$\hat{\mathcal{R}}(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(x^{(i)}), y^{(i)}).$$

This procedure is called *empirical risk minimization* (ERM) [63]. The test set is not used in training, but instead helps provide an unbiased measure of generalization.

Loss functions. The loss function ℓ specifies the performance error between a prediction made by the model and the true label.

For *regression*, where the output is a real-valued quantity, a common choice includes the mean squared error (MSE),

$$\ell_{\text{MSE}}(f_\theta(x), y) = (f_\theta(x) - y)^2,$$

which penalizes squared deviations from the target.

For *classification*, where the output is one of K discrete categories, the most widely used loss is the *cross-entropy* loss. Let the model output a probability distribution over classes,

$$p_\theta(c \mid x) = \frac{\exp(z_c(x))}{\sum_{j=1}^K \exp(z_j(x))}, \quad c \in \{1, \dots, K\},$$

where $z(x) \in \mathbb{R}^K$ are the unnormalized scores (logits). If the true label is $y \in \{1, \dots, K\}$, then the cross-entropy loss is

$$\ell_{\text{CE}}(f_{\theta}(x), y) = - \sum_{c=1}^K \mathbf{1}_{\{c=y\}} \log p_{\theta}(c | x) = -\log p_{\theta}(y | x).$$

This loss is minimized when the predicted probability.

Image Classification as an example task. A representative supervised task in the literature is *image classification* [33]. In this setting, a model is trained to assign each image to one of K possible classes (e.g., dog, cat, tree). Each training example is a pair (x, y) where $x \in \mathbb{R}^{h \times w \times 3}$ is an RGB image of height h and width w , and $y \in \{1, 2, \dots, K\}$ is the class label. The model processes the image and outputs a score or probability distribution over the K classes. Training with empirical risk minimization and the cross-entropy loss encourages the model to assign high probability to the correct class y . Once trained, the model is evaluated on new test images to measure how well it generalizes beyond the training set.

Neural Networks and Optimization. In many modern applications, the function f_{θ} is implemented as a deep neural network (DNN): a layered composition of linear transformations and nonlinear activations [34]. Neural networks can have millions of parameters and multiple layers, allowing them to capture complex input–output relationships.

To minimize the empirical risk, DNNs are typically trained with iterative optimization methods such as *stochastic gradient descent* (SGD) [53]. At each step, the model parameters θ are updated using the gradient of the loss with respect to θ :

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \ell(f_{\theta}(x), y),$$

where η is the learning rate. Variants such as momentum or adaptive methods like Adam [31] build on SGD but follow the same basic principles.

In modern DNNs, gradients are computed efficiently using the *backpropagation* algorithm [54], which applies the chain rule of calculus to propagate errors from the output layer back through the model.

Transfer learning and representation learning. A key idea in modern machine learning is that models benefit from learning general-purpose *representations* of data. Instead of

mapping inputs x directly to outputs with f_θ , one can first learn an *encoder* $g_\phi : \mathcal{X} \rightarrow \mathbb{R}^d$ that extracts a representation $z = g_\phi(x)$ capturing the essential structure of the input. The encoder is typically pretrained on a larger source of data, which is often different but related to the current task. The predictor f_θ then operates on this representation, forming a composite model $f_\theta(g_\phi(x))$. This separation allows the encoder to transfer across tasks: representations learned from a large source dataset (e.g., ImageNet [33]) can be reused in a new target task with limited labeled data, a practice known as *transfer learning*. By leveraging shared structure, transfer learning often yields significantly lower test error on the target task than training f_θ from scratch [46, 7, 30].

2.2 Self-supervised Representation Learning

Representation learning aims to extract meaningful and efficient representations from data. It accomplishes this goal by learning abstractions that contain the critical essence of data samples while filtering out noise and irrelevant features. The abstractions can then be transferred to other downstream tasks that can leverage the insights of the representation without having seen or processed the noisy raw data.

Self-supervised learning (SSL) is a paradigm within representation learning in which models are trained using unlabeled data, relying on supervision signals outside of training labels. This approach is particularly attractive as it eliminates the dependency on labeled datasets, thereby improving scalability. SSL has gained significant traction in domains where large volumes of unlabeled data are readily available, such as computer vision [46, 36] and natural language processing [40, 9].

SSL methods are commonly categorized into two broad families: *generative* and *discriminative* approaches. Generative methods receive incomplete data inputs and are trained to reconstruct the missing parts. Examples include masked autoencoding techniques, which learn to restore masked regions of the input [9, 29], and sequence modeling methods that predict future elements based on past observations [4]. Discriminative methods, in contrast, aim to produce embeddings that either align similar data or diverge from dissimilar inputs. Representative examples include contrastive approaches, which learn to pull together similar examples while pushing apart dissimilar ones [7, 30]; clustering-based methods, which group similar data points [5]; and bootstrapping methods, which generate their own learning targets without requiring explicit negative examples [21].

In this work, we focus on contrastive methods, as employed in contrastive reinforcement learning, and draw inspiration from self-distillation methods to inform the design of our method. We cover both these methods in detail.

2.2.1 Contrastive Learning

Contrastive learning is a self-supervised learning paradigm that aims to acquire meaningful representations by distinguishing between similar and dissimilar data instances. The central objective is to learn a *scoring function* $s(x, y; \theta)$ to identify when two inputs are semantically related (*positive pairs*) and to differentiate them from unrelated inputs (*negative pairs*). This objective encourages the score function to map similar inputs close together and dissimilar inputs far apart in the representation space. As a result, contrastive learning extracts rich and discriminative features from unlabeled data, which can be readily transferred to downstream tasks.

A representative example is SimCLR [7], in which positive pairs are created by applying two distinct augmentations from the set $\mathcal{T}$ to the same image (e.g. random crop, gaussian blur, etc.), while negative pairs consist of augmented views originating from different images. The model is trained with a contrastive loss, which promotes invariance of the learned representations to data augmentations while maintaining their discriminative power across instances.

In this work, we focus on describing contrastive methods from [22, 38, 44] that are directly applicable to contrastive reinforcement learning. Contrastive reinforcement learning uses a binary classification loss for learning the *density ratio* of the form

$$\frac{p_{Y|X}(y \mid x)}{p(y)}, \tag{2.1}$$

where x and y are samples drawn from random variables X and Y , respectively. We discuss the algorithm and some key theoretical results, but refer the reader to the aforementioned works for a more detailed discussion.

Algorithm 1 Learning to approximate density ratio (2.1) with binary classification loss (following [38]).

1: **Inputs:**

- Training examples $\{(x^{(i)}, y^{(i)})\}_{i=1}^n \in \mathcal{D}$
- Sampling distribution $p_N(\cdot)$ for generating negative examples. We sample negative examples from the marginal distribution $p_N(y) = p_Y(y)$.
- Integer $K \in \mathbb{N}$ specifying number of negative examples per training example
- A trainable model $h(x, y; \theta)$ to approximate $h(x, y; \theta) \approx s(x, y; \theta) - \log(p_N(y))$, where $s(x, y; \theta)$ is the scoring function.

2: **Definitions:**

- From Assumption 2.3, we have $\gamma = \log(Z(x, \theta))$ where Z is the partition function.

3: **Algorithm:**

1. For $i = 1, \dots, n$, and $k = 1, \dots, K$, draw $y^{(i,k)}$ i.i.d. from $p_N(y)$. We choose the positive pair to be the first of K elements in the array $y^{(i,0)} = y^{(i)}$.
2. Define the binary classification objective

$$L_n^B(\theta, \gamma) = \frac{1}{n} \sum_{i=1}^n \left[\log g(x^{(i)}, y^{(i,0)}; \theta, \gamma) + \sum_{k=1}^K \log (1 - g(x^{(i)}, y^{(i,k)}; \theta, \gamma)) \right],$$

where

$$g(x, y; \theta, \gamma) = \frac{\exp(s(x, y; \theta) - \log p_N(y) - \gamma)}{\exp(s(x, y; \theta) - \log p_N(y) - \gamma) + K}.$$

Then let the optimal parameters be denoted as

$$(\hat{\theta}_B, \hat{\gamma}_B) = \arg \max_{\theta \in \Theta, \gamma \in \Gamma} L_n^B(\theta, \gamma).$$

3. Return $\hat{\theta}_B$. Given that the negative sampling distribution was $p_N(y) = p_Y(y)$, an unnormalized density ratio estimate can be computed as a function of $h(x, y; \hat{\theta}_B)$:

$$\frac{\hat{p}_{Y|X}(y | x)}{\hat{p}_Y(y)} = \frac{\exp(s(x, y; \hat{\theta}_B) - \log p_N(y) - \hat{\gamma}_B)}{\sum_{y' \in \mathcal{Y}} \exp(s(x, y'; \hat{\theta}_B) - \log p_N(y') - \hat{\gamma}_B)} \propto \exp(h(x, y; \hat{\theta}_B))$$

A binary classification procedure for estimating the density ratio $\frac{\hat{p}_{Y|X}(y|x)}{\hat{p}_Y(y)}$ is outlined in Algorithm 1 and follows [38]. At a high level, the method trains a neural network $h(x, y; \theta)$ that is a function of x and y to learn the scoring function (also called the similarity function). A higher score should indicate that x is related to y . To learn the score function, the network is trained with binary classification, to classify x, y as either *positive* if they co-occur in the dataset and as *negative* otherwise.

Positive pairs are drawn directly from the dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^n$, where each $(x^{(i)}, y^{(i)})$ reflects a true co-occurrence. For each positive pair, K negative pairs are generated by keeping $x^{(i)}$ fixed and replacing $y^{(i)}$ with independently sampled values $y^{(i,k)} \sim p_Y(y)$ for $k \in \{1, \dots, K\}$. This process yields an augmented dataset containing both positive and negative examples.

For example, in an image-caption retrieval setting, $\mathcal{D}$ could consist of image-caption pairs $(x^{(i)}, y^{(i)})$ where $x^{(i)}$ is an image and $y^{(i)}$ is its corresponding human-annotated caption. Positive pairs are taken directly from $\mathcal{D}$. To create negative examples, we keep the image $x^{(i)}$ fixed and replace its caption $y^{(i)}$ with captions from other, unrelated images in the dataset. This procedure approximates sampling from the marginal distribution $p_Y(y)$ and provides mismatched pairs for contrastive training.

The model $h(x, y; \theta)$ is then optimized using a binary classification loss $L_n^B(\theta, \gamma)$ that encourages it to assign higher scores to positive pairs and lower scores to negative pairs. Once trained, the exponentiated network output $\exp(h(x, y; \theta))$ serves as an unnormalized estimate of the density ratio $\frac{\hat{p}_{Y|X}(y|x)}{\hat{p}_Y(y)}$.

Theory of Contrastive Learning via Binary Classification

We provide theoretical intuition for Algorithm 1. The assumptions and lemmas are based on [38], but have been rewritten in our own words to convey the ideas more clearly. We refer the reader to Gutmann and Hyvärinen [38] for the original and more detailed treatment. The remaining passages in this section can be **skipped** by a reader not interested in the theoretical intuition.

The first assumption 2.2 (following [38]) posits that the true conditional distribution $p_{Y|X}(y | x)$ can be factored with an exponential-family representation normalized by a partition function $Z(x; \theta^*)$ that depends on x . Moreover, it assumes the *existence* of a parameter value $\theta^* \in \Theta$ for which the scoring function $s(x, y; \theta^*)$ exactly characterizes $p_{Y|X}(y | x)$.

Assumption (parametric exponential-family form). There exists $\theta^* \in \Theta$ such that, for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$,

$$p_{Y|X}(y | x) = \frac{\exp(s(x, y; \theta^*))}{Z(x; \theta^*)}, \quad Z(x; \theta^*) := \sum_{y' \in \mathcal{Y}} \exp(s(x, y'; \theta^*)). \quad (2.2)$$

The second assumption 2.3 (following [38]) reformulates the normalization term as $\gamma = \log Z(x; \theta^*)$, treating it as a trainable scalar parameter.

Assumption of constant normalization factor For parameters $\theta^* \in \Theta$, the normalization factor is a constant $\gamma^* \in \mathbb{R}$ and $\gamma^*(x) := \log Z(x; \theta^*)$. Thus the true conditional can be written:

$$p_{Y|X}(y | x) = \exp(s(x, y; \theta^*) - \gamma^*). \quad (2.3)$$

The model parameters are optimized using the *binary classification objective* over K negative examples and one positive example ([38]):

$$L_n^B(\theta, \gamma) = \frac{1}{n} \sum_{i=1}^n \left[\log g(x^{(i)}, y^{(i,0)}; \theta, \gamma) + \sum_{k=1}^K \log (1 - g(x^{(i)}, y^{(i,k)}; \theta, \gamma)) \right],$$

where the ratio $g(x, y; \theta, \gamma)$ is

$$g(x, y; \theta, \gamma) = \frac{\exp(s(x, y; \theta) - \log p_N(y) - \gamma)}{\exp(s(x, y; \theta) - \log p_N(y) - \gamma) + K} = \frac{\exp(h(x, y; \theta))}{\exp(h(x, y; \theta)) + K}.$$

Before analyzing the optimization, we note that $g(x, y; \theta, \gamma)$ does **not** directly yield the target density ratio $\frac{p_{Y|X}(y|x)}{p_Y(y)}$. Under the true parameter θ^* , which correctly specifies the unnormalized conditional distribution $p_{Y|X}(y | x)$, the quantity $g(x, y; \theta^*, \gamma^*)$ instead gives the *Bayes-optimal probability* that (x, y) is a positive example. This probability is defined relative to the binary classification task of distinguishing positive from negative pairs:

Lemma 1 (Density ratio of $g(x, y; \theta^*, \gamma^*)$). *Under the assumption 2.2 that states $\theta^* \in \Theta$ corresponds to unnormalized $p_{Y|X}(y|x)$ and assumption 2.3, then $g(x, y; \theta^*, \gamma^*)$ corresponds to the ratio*

$$g(x, y; \theta^*, \gamma^*) = \frac{p_{Y|X}(y|x)}{p_{Y|X}(y|x) + K p_N(y)}$$

Proof. Under Assumption 2.2,

$$p_{Y|X}(y|x) = \exp(s(x, y; \theta^*) - \gamma^*).$$

$$p_{Y|X}(y|x) = p_N(y) \exp(s(x, y; \theta^*) - \log p_N(y) - \gamma^*).$$

Hence

$$\begin{aligned} \frac{p_{Y|X}(y|x)}{p_{Y|X}(y|x) + K p_N(y)} &= \frac{p_N(y) \exp(s(x, y; \theta^*) - \log p_N(y) - \gamma^*)}{p_N(y) \exp(s(x, y; \theta^*) - \log p_N(y) - \gamma^*) + K p_N(y)} \\ &= \frac{\exp(s(x, y; \theta^*) - \log p_N(y) - \gamma^*)}{\exp(s(x, y; \theta^*) - \log p_N(y) - \gamma^*) + K} \\ &= g(x, y; \theta^*, \gamma^*) \end{aligned}$$

□

Next, we can show that optimizing the binary classification objective 2.2.1 corresponds to finding θ^* and for $g(x, y; \theta, \gamma)$ to model the ratio in Lemma 1. To do that, we first describe the maximization of expected log likelihood:

Lemma 2 (Maximizer of Expected Log-Likelihood). *Let $q = (q_i)_{i=1}^n$ be a probability distribution on a finite set, and let $p = (p_i)_{i=1}^n$ be any distribution on the same set. The problem*

$$\arg \max_p \sum_{i=1}^n q_i \log p_i$$

has the unique solution $p = q$.

Proof. By the identity

$$\sum_{i=1}^n q_i \log p_i = \sum_{i=1}^n q_i \log q_i - D_{\text{KL}}(q \| p),$$

where $D_{\text{KL}}(q \| p) \geq 0$ with equality iff $p = q$. Thus the expression is maximized at $p = q$. □

Let $L_\infty^B(\theta) = \mathbb{E}[L_n^B(\theta)]$ be the expected value of the binary classification loss $L_n^B(\theta)$, where the expectation is taken over i.i.d. samples drawn from the underlying data distribution. The notation L_∞^B emphasizes the population-level objective obtained in the limit as the number of samples $n \rightarrow \infty$. then $L_\infty^B(\theta)$ is minimized when $\theta = \theta^*$ and 1 holds:

Lemma 3 (Minima of expected loss $L_\infty^B(\theta)$). *Under assumption 2.2 and assumption 2.3,*

$$(\theta^*, \gamma^*) \in \arg \max L_B^\infty(\theta, \gamma)$$

Proof. We can expand $L_\infty^B(\theta)$:

$$\begin{aligned} L_\infty^B(\theta) &= \mathbb{E} [L_n^B(\theta)] \\ &= p_{X,Y}(x, y) \log (g(x, y; \theta, \gamma)) + K \cdot p_X(x) p_N(y) \log (1 - g(x, y; \theta, \gamma)) \\ &= (p_{X,Y}(x, y) + K \cdot p_X(x) p_N(y)) \left\{ \frac{p_{Y|X}(y|x)}{p_{Y|X}(y|x) + K \cdot p_N(y)} \log (g(x, y; \theta, \gamma)) \right. \\ &\quad \left. + \frac{K \cdot p_N(y)}{p_{Y|X}(y|x) + K \cdot p_N(y)} \log (1 - g(x, y; \theta, \gamma)) \right\}. \end{aligned}$$

Only $g(x, y; \theta, \gamma)$ is free in the above equations. Using 2, we can maximize the both terms term:

$$\arg \max_{g(x, y; \theta, \gamma)} \left\{ \frac{p_{Y|X}(y|x)}{p_{Y|X}(y|x) + K p_N(y)} \log (g(x, y; \theta, \gamma)) \right\} = \theta^*$$

and

$$\arg \max_{g(x, y; \theta, \gamma)} \left\{ \frac{K p_N(y)}{p_{Y|X}(y|x) + K p_N(y)} \log (1 - g(x, y; \theta, \gamma)) \right\} = \theta^*$$

In both scenarios 1 holds.

□

The binary contrastive learning objective enables estimating the density ratio $p_{Y|X}(y|x)/p(x)$, and in the process learns general-purpose representations.

A critical challenge with contrastive learning is the need to carefully curate training batches that encourage the model to learn disc features. In response to this challenge, many modern representation learning methods have shifted to self-distillation, as they avoid the need for carefully curating negative pairs in training batches.

2.2.2 Self-Distillation Methods

Self-Distillation methods address a central limitation of contrastive learning—the dependence on large numbers of carefully mined negative pairs. If negatives are too easy, the model trained by contrastive loss may learn only superficial features. This requirement makes contrastive methods sensitive to batch size and sampling strategy, and can demand significant computational resources to maintain diverse and informative negatives.

In contrast to contrastive approaches, self-distillation methods remove the need for negative pair mining altogether. They learn representations by aligning different augmented views of the same sample without explicitly contrasting against negatives. By relying on self-consistency rather than inter-sample discrimination, self-distillation methods encourage representations to evolve toward stable and informative features.

Formally, let θ denote the parameters of a neural network f . It employs two networks with different weights: an *online* network f_o parameterized by θ_o and a *target* (or teacher) network f_t parameterized by θ_t . The online network is trained to produce representations that match those generated by the target network.

The target network is constructed by copying a subset of the layers from the online network. Its parameters are updated more slowly, typically as an exponential moving average of the online network’s parameters. This slow update mechanism ensures that the target network provides a relatively stable learning signal for the online network.

During training, different augmented views of the same input sample are passed through both networks. A loss function $\mathcal{L}_{o,t}$ is then used to align the representation produced by the online network with the embedding generated by the target network. In this way, the model *bootstraps* its own target representations, since the teacher model is derived from the online model itself.

BYOL We start by covering Bootstrap your own latent (BYOL) [21], the first large-scale implementation of bootstrapping that matched or beat contrastive methods in Image classification benchmarks (e.g. ImageNet [33]). Its core novelty was creating a target network with an exponential moving average of the online network’s weights, and training the online network with MSE loss to mimic the target network.

Figure 2.1 depicts how the input sample is processed by the online and target networks to compute the loss. The *online* network consists of three sequential components:

- an *encoder* $f_{\theta_o} : \mathbb{R}^N \rightarrow \mathbb{R}^M$,

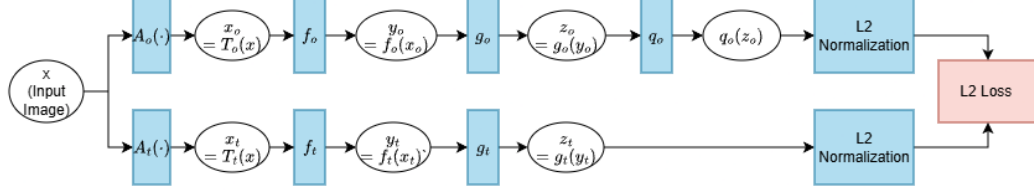


Figure 2.1: Illustration of BYOL’s training setup: the input sample x is processed through the online and target networks, and the training objective is computed by aligning the online network’s prediction with the target network’s representation using an ℓ_2 loss.

- a *projector* $g_{\theta_o} : \mathbb{R}^M \rightarrow \mathbb{R}^D$, and
- a *predictor* $q_{\theta_o} : \mathbb{R}^D \rightarrow \mathbb{R}^D$.

The *target* network is comprised of only the first two components of the online network:

- an *encoder* $f_{\theta_t} : \mathbb{R}^N \rightarrow \mathbb{R}^M$, and
- a *projector* $g_{\theta_t} : \mathbb{R}^M \rightarrow \mathbb{R}^D$.

Let $x \sim \mathcal{D}$ be samples from the dataset, and let A denote a stochastic data augmentation pipeline. Two augmented views are generated through distinct augmentation pipelines:

$$x_o \sim a_o(x), \quad a_o \sim A_o, \quad x_t \sim a_t(x), \quad a_t \sim A_t,$$

producing the *online* and *target* views, respectively.

The online network processes x_o to produce:

$$y_o = f_o(x_o), \quad z_o = g_o(y_o), \quad q_o(z_o) = q_o(z_o).$$

The target network processes x_t to produce:

$$y_t = f_t(x_t), \quad z_t = g_t(y_t).$$

Both representations are ℓ_2 -normalized:

$$\bar{q}_o(z_o) = \frac{q_o(z_o)}{\|q_o(z_o)\|_2}, \quad \bar{z}_t = \frac{z_t}{\|z_t\|_2}.$$

The training objective encourages the online network’s predictions to match the target network’s normalized embeddings, while preventing gradient flow through the target network using the stop-gradient operator $\text{sg}(\cdot)$. The general loss is:

$$\mathcal{L}_{o,t}(\bar{q}_o(z_o), \text{sg}(\bar{z}_t)),$$

and, in the case of BYOL, is instantiated as a mean squared error (MSE):

$$\mathcal{L}_{o,t}(\bar{q}_o(z_o), \text{sg}(\bar{z}_t)) = \|\bar{q}_o(z_o) - \bar{z}_t\|_2^2.$$

After training, representations produced by the online encoder backbone f_o are used for downstream tasks. Importantly, while the training loss is computed on features obtained from the *projector* network, downstream tasks utilize only the representations from the encoder $f_\theta : \mathbb{R}^N \rightarrow \mathbb{R}^M$. This design choice has been found to be essential for preventing *representational collapse*.

Theoretically, BYOL is prone to representational collapse, where both encoders f_o and f_t produce identical constant outputs for all inputs x , yielding zero loss without learning meaningful features [55]. In practice, however, collapse is not observed. One possible explanation the authors give is that using a slowly updated exponential moving average (EMA) for the target network parameters helps break symmetry and encourage the online network to match an improved version of itself.

DINO Another notable contribution in this domain is *Self-Distillation with No Labels* (DINO) [6], which reinterprets the BYOL framework in terms of knowledge distillation [69]. It introduced several enhancements that produced more semantically meaningful features, improved downstream image classification performance, and produced general-purpose features that could be fine-tuned with diverse strategies.

The two major distinctions from BYOL were (1) the removal of the predictor network and (2) using cross entropy instead of MSE loss between the online and target networks.

As in BYOL, the online and target networks process augmented views of the same image through an encoder f_o and a projector g_o . However, unlike BYOL, DINO uses identical architectures for the online and target networks by omitting the predictor stage.

As depicted in Figure 2.2, the input sample is processed by the online network $z_o = g_o(f_o(x_o))$ and the target network $z_t = g_t(f_t(x_t))$. Both representations are then ℓ_2 -normalized:

$$\bar{z}_o = \frac{z_o}{\|z_o\|_2}, \quad \bar{z}_t = \frac{z_t}{\|z_t\|_2}.$$

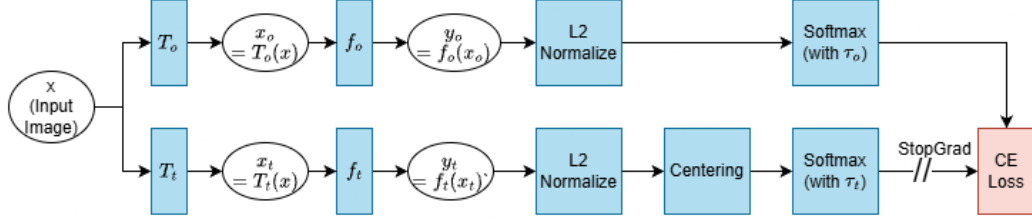


Figure 2.2: Illustration of DINO’s training setup: the input sample x is processed through the online (student) and target (teacher) networks, and the training objective is computed by aligning the online network’s prediction with the target network’s representation using cross-entropy loss.

DINO introduces a *centering* operation on the teacher’s output:

$$z_t \leftarrow z_t + c,$$

where the bias c is updated as an exponential moving average of the teacher’s outputs:

$$c \leftarrow mc + (1 - m) \frac{1}{B} \sum_{i=1}^B g_{\theta_t}(x_i),$$

with batch size B and momentum parameter $m \in [0, 1]$.

Both teacher and student outputs are then passed through a softmax with distinct *temperature* parameters τ_t and τ_o :

$$p^i = \frac{\exp(z^i/\tau)}{\sum_{d=1}^D \exp(z^d/\tau)},$$

where τ controls the sharpness of the distribution and D is the dimensionality of z .

Finally, the training objective is the cross-entropy between the student and teacher probability distributions:

$$\mathcal{L}_{o,t}(p_o, p_t) = - \sum_{i=1}^D p_t^{(i)} \log p_o^{(i)},$$

where $p_o = \text{softmax}(\bar{z}_o/\tau_o)$ and $p_t = \text{softmax}(\bar{z}_t/\tau_t)$.

DINO reinterpreted the BYOL framework through the lens of *knowledge distillation* (KD) [69] between a teacher and a student classifier, but without hard labels. The online and target networks can be interpreted as the *teacher* and *student* networks, respectively.

In standard KD, a smaller student network is trained to match the softened output distribution of a larger teacher network, typically by minimizing the cross-entropy between the student’s outputs and the teacher’s sharpened outputs. Crucially, the teacher’s outputs are softened by temperature-scaling, which helps them capture the teacher’s decision boundaries more effectively than one-hot labels. It has been widely applied across domains, from image classification to large language models [20].

Unlike conventional KD, where the teacher is pre-trained and fixed, DINO’s teacher is updated dynamically during training as an exponential moving average (EMA) of the student’s parameters. In this self-supervised interpretation, the EMA acts as a *mean teacher*, effectively forming an ensemble of the student’s parameters over time and providing a progressively improving learning signal.

Within the DINO framework, representational collapse cannot be theoretically ruled out; however, it is not observed empirically. The authors attribute this stability to the combined effect of *centering* and *sharpening*. Centering normalizes feature distributions to prevent any single dimension from dominating, but in isolation it may lead to collapse toward a uniform distribution. Sharpening amplifies differences between features, mitigating uniform collapse, yet on its own can cause a different failure mode where one dimension becomes overly dominant. When applied together, centering and sharpening counterbalance each other’s weaknesses, thereby helping to prevent collapse in practice.

2.3 Reinforcement Learning

The world is populated by numerous agents, both artificial and biological, that interact with their environments to accomplish objectives. Reinforcement Learning (RL) provides a framework for training such agents to act within an environment and accomplish a predefined task. Among the many formulations of RL, this work focuses on *goal-conditioned reinforcement learning* (GCRL) in the *offline* setting. In particular, the agent is trained to reach goal states by learning from a fixed dataset of prior experience.

Within the spectrum of GCRL approaches, we concentrate on *Contrastive Reinforcement Learning* (CRL), a model-free, actor–critic method. CRL leverages contrastive representation learning objectives for value estimation, offering improved simplicity, stability, and scalability over other RL methods.

2.3.1 Formulating Goal-conditioned Reinforcement Learning

Reinforcement Learning (RL) is a framework in which an agent interacts with an environment $\mathcal{M}$ with the objective of maximizing a measure of long-term performance, typically the *cumulative reward* R .

We start by formally describing the classical Markov Decision Process (MDP) by defining the following items:

- a set of states $\mathcal{S}$, with $s \in \mathcal{S}$;
- a set of actions $\mathcal{A}$ available to the agent, with $a \in \mathcal{A}$;
- the initial state distribution $p_0(s)$ describing the possible starting states of the agent;
- a reward model $r(a, s) \in \mathbb{R}$ that specifies the reward received for taking a particular action at a particular state. The agent does not have access to the reward model;
- a transition dynamics model $p(s_{t+1} \mid s_t, a_t)$.
- a policy $\pi(a|s)$ learned by the agent, encoding the distribution of actions the agent picks at a given state;

At each discrete time step t , the agent observes a state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$ according to its policy $\pi(a \mid s)$. In response to taking the action, the agent receives a scalar reward $r(s_t, a_t)$, and transitions to a new state determined by the dynamics model $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$. The objective of the agent is to learn a policy π that maximizes the *expected discounted return*:

$$R_t = \sum_{k=0}^H \gamma^k r(s_{t+k}, a_{t+k}), \quad (2.4)$$

where H is the finite horizon length and $\gamma \in [0, 1)$ is the discount factor.

Reinforcement learning (RL) is the problem of learning to act optimally within such a Markov Decision Process by learning from experience. In RL, the agent does not have access to the transition dynamics model, the reward model, or the optimal policy. Instead, it must learn a policy $\pi(a \mid s)$ that maximizes the expected discounted return (reward) by interacting with the environment and observing outcomes.

In this work, we focus on the *Goal-Conditioned Reinforcement Learning* (GCRL) problem. In GCRL, the agent is tasked with reaching a specific *goal state* s_g rather than directly

maximizing an externally defined reward. This objective can be expressed in the standard RL framework by defining a goal-conditioned reward function:

$$r_g(s_t, a_t) = (1 - \gamma) p(s_{t+1} = s_g \mid s_t, a_t), \quad (2.5)$$

which represents the *discounted probability* of reaching the goal state s_g in the next step.

As such, we can use the **traditional** reinforcement learning formulation but with some modifications. The policy is now expressed as a *goal-conditioned policy* $\pi(a \mid s, s_g)$, where the agent selects actions based not only on the current state s but also on the desired goal state s_g . The new cumulative reward objective becomes:

$$\max_{\pi} \mathbb{E}_{p_g(s_g), \pi(\tau \mid s_g)} \left[\sum_{t=0}^{\infty} \gamma^t r_g(s_t, a_t) \right], \quad (2.6)$$

where $p_g(s_g)$ denotes the distribution over possible goals and $\pi(\tau \mid s_g)$ is the probability of trajectory τ under policy π .

2.3.2 Model-based vs Model-free Reinforcement Learning

Reinforcement learning algorithms can be classified in multiple ways, a critical one being *model-based* vs *model-free* methods. Our approach combines elements of both paradigms: we learn a world model and leverage its representations as a model-free reinforcement learning algorithm (as done in [27, 28]).

Model-based methods explicitly learn a *World Model* $\hat{p}(s_{t+1} \mid s_t, a_t)$, otherwise known as a learned dynamics model. This model is trained to learn the true dynamics model of the environment $p(s_{t+1} \mid s_t, a_t)$. In essence, these models learn the outcomes of taking various actions at a given state. As world models primarily learn the physics of an environment, they can potentially be trained on diverse data sources (such as data from an ineffective policy or data from an unrelated task).

Model-based reinforcement learning (MBRL) has long been studied for its improved data efficiency and generalization over model-free methods. The seminal work of [23] demonstrated that RL tasks can leverage world models for continuous control tasks. While they primarily used world models for representations in their controller, they also showed that the world model could serve as the sole training environment/simulator for the controller (a process they called “dreaming”). More recent works have advanced this idea by using world models to generate synthetic trajectories that augment limited datasets, most

notably in the Dreamer series [26, 25, 27] and in UniSim [67]. A parallel line of research employs world models as planning tools, where the model is queried to simulate possible futures and select actions that maximize long-term rewards [56, 28, 10]. We discuss using a world model for representation learning in Section 2.4.

In contrast, *model-free* methods do not attempt to model the environment dynamics explicitly. Instead, they directly learn a policy or a value function from experience. Model-free methods avoid the complexity of training a world model, the computational cost of running a world model, and the compounding errors of world model rollouts.

A large body of influential work has focused on advancing model-free reinforcement learning. Early breakthroughs included value-based methods, like Deep Q-Networks (DQN) [41], which successfully played Atari games. Subsequent research refined methods known as policy gradients, most notably with Proximal Policy Optimization (PPO) [58], which has since become a standard baseline in many RL tasks. To address instability in continuous action spaces, algorithms such as Deep Deterministic Policy Gradient (DDPG) [35], Twin Delayed DDPG (TD3) [17], and Soft Actor–Critic (SAC) [24] introduced techniques to mitigate overestimation bias and improve exploration. We discuss a notable class of actor-critic methods in Section 2.3.3.

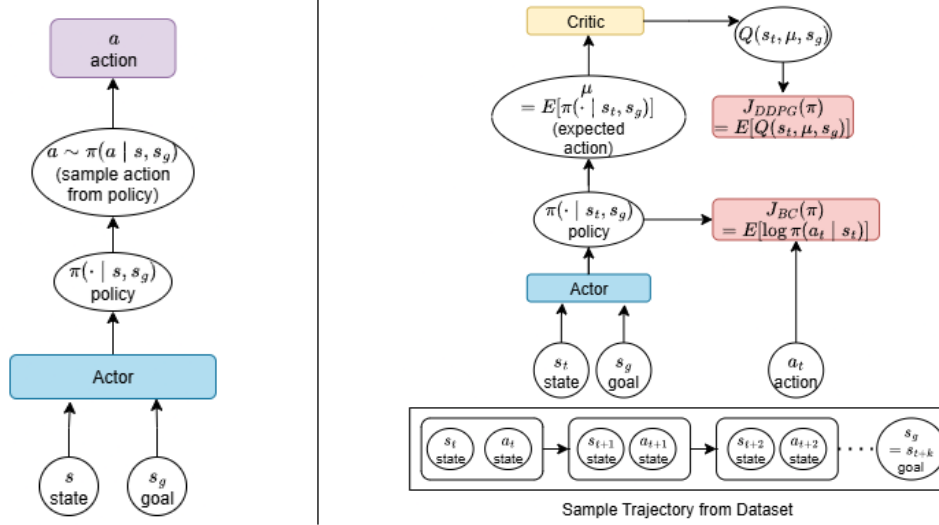


Figure 2.3: Actor network during inference (left) and training (right). The actor outputs a policy, a distribution over actions conditioned on the current state and goal. At inference time, actions are sampled directly from this policy (left). During training, the actor is optimized with a composite loss: a DDPG objective that encourages actions maximizing the critic’s value estimate, and a behavior cloning objective that encourages consistency with actions observed in the dataset (right).

2.3.3 Actor-critic Algorithms

We now focus on a class of model-free methods known as *actor-critic* algorithms [32], in the context of goal-conditioned reinforcement learning. At a high level, these methods train a critic network to estimate the value of taking different actions at different states, and then train an actor/policy to find actions that the critic suggests are optimal. Figure 2.3 shows an overview of the actor, depicting how actions are sampled at inference time and how training loss is computed.

In these methods, a *critic* network estimates the value of a state–action pair via the Q -function:

$$Q(s, a) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a \right], \quad (2.7)$$

where $\gamma \in [0, 1)$ is the discount factor, and the expectation is taken over trajectories induced by the policy π and the environment dynamics. Intuitively, $Q(s, a)$ represents cumulative reward by taking action a in state s , and subsequently following the policy π .

An alternative formulation of the Q -function can be expressed in terms of the *discounted state occupancy measure*. For a goal-conditioned policy $\pi(\cdot \mid \cdot, s_g)$, the discounted state occupancy measure is defined as:

$$p^{\pi(\cdot \mid \cdot, s_g)}(s_{t+} = s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t p_t^{\pi(\cdot \mid \cdot, s_g)}(s_t = s), \quad (2.8)$$

where $p_t^{\pi(\cdot \mid \cdot, s_g)}(s)$ denotes the probability distribution of states after following policy π for t steps and s_{t+} denotes the state samples from the discounted state occupancy measure. Sampling states from the s_{t+} discounted state occupancy measure is straightforward: (1) sample a time offset t from the geometric distribution $t \sim \text{Geom}(1 - \gamma)$; (2) follow the policy for exactly t steps; (3) record the state.

Given this definition, the Q -function for goal-conditioned policy π can be equivalently written as:

$$Q^{\pi, s_g}(s, a) = p^{\pi(\cdot \mid \cdot, s_g)}(s_{t+} = s_g \mid s, a). \quad (2.9)$$

as per the proof in [12]. This identity shows that the Q -function is equivalent to the *probability of reaching the goal state s_g* under the discounted state occupancy measure. This alternative definition of the Q -function will be critical to showing why contrastive learning objectives can effectively learn a Q -function.

Given a critic, the *actor* is trained simultaneously to extract a policy from it. Since the critic estimates the value of each state–action pair, the optimal policy should select actions that maximize these value estimates. In our work, we adopt the *Deterministic Deep Policy Gradient* (DDPG) algorithm [35]. As we adopt the offline setting, we follow prior work [47, 12] and regularize the actor with a behavior cloning (BC) term. This term encourages the policy to take actions a observed in the dataset $\mathcal{D}$, thereby reducing the risk of reaching out-of-distribution states at inference. The resulting objective is:

$$\max_{\pi} J_{\text{DDPG+BC}}(\pi) = \mathbb{E}_{s, a \sim \mathcal{D}} [Q(s, \mu) + \alpha \log \pi(a \mid s)], \quad (2.10)$$

where α controls the strength of the BC regularization and $\mu = E_a[\pi(a \mid s)]$ is the average action chosen by the policy at state a .

2.3.4 Contrastive Reinforcement Learning

Contrastive Reinforcement Learning (CRL) [12] is a model-free actor–critic method for GCRL. Its key novelty lies in using *contrastive learning* objective to train the critic and

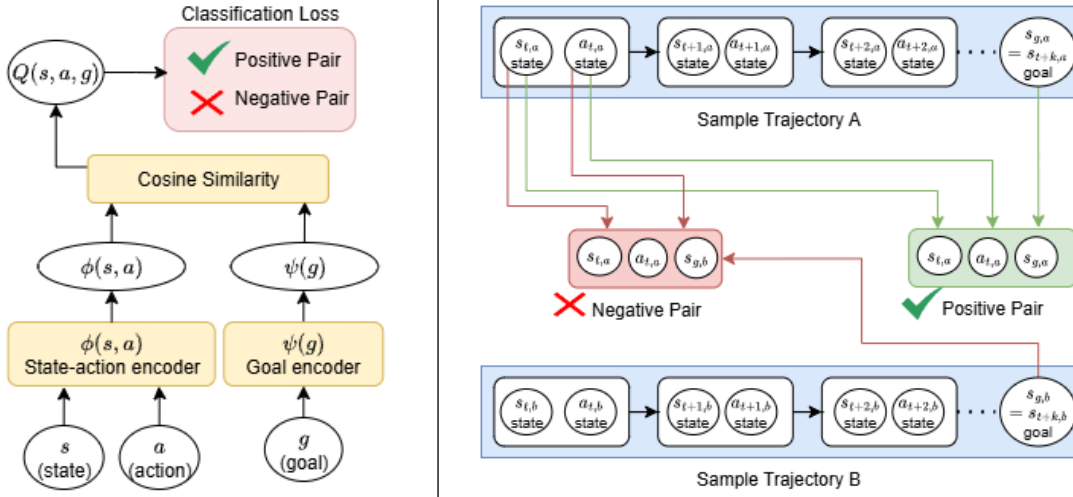


Figure 2.4: CRL’s critic network training and positive-negative pair curation. In CRL, the critic network evaluates the physical plausibility of state–action–goal triplets by embedding both the state–action and goal into a shared representation space and computing their dot product. It is trained with a similarity-based classification loss to distinguish real triplets from false ones (left). Positive and negative training pairs are curated by sampling goal states from the same trajectory (positive) or from different trajectories (negative) (right).

estimate the Q -function. Compared to traditional TD-Learning based methods for learning Q -functions, this approach offers several advantages: it simplifies training (avoiding the instability often associated with TD learning, which typically requires techniques such as exponential moving average bootstrapping or ensemble methods), and it scales more effectively when high-quality training data is available [12, 49].

In CRL [12], value estimation is reframed as a discriminative task: determining whether a given state–action pair (s, a) will lead to a particular goal state s_g in the future. This formulation draws on principles from self-supervised representation learning, allowing the critic to be trained via contrastive objectives rather than bootstrapped targets. Figure 2.4 depicts how the critic network computes state-action-goal similarity and how positive/negative pairs are curated.

To understand how contrastive learning can be applied to train the critic, we begin by normalizing the Q -function by the marginal probability of the goal state, $p(s_g)$. Using the earlier expansion of Q , we have:

$$\frac{Q^{\pi, s_g}(s, a)}{p(s_g)} = \frac{p^{\pi(\cdot)}(s_{t+} = s_g \mid s, a)}{p(s_g)}. \quad (2.11)$$

This ratio is equivalent to the Q -function up to a constant normalization factor that does not depend on the policy. Importantly, this form is particularly convenient because it can be learned directly using the binary classification objective for contrastive learning (discussed in 1).

To implement contrastive loss, positive and negative pairs $\{(s_t, a_t), (s_f)\}$ need to be curated first. First, state-action pairs (s_t, a_t) are sampled from the dataset of prior experience. Then we sample s_f^+ . For *positive* examples, we sample s_f^- from the discounted state occupancy measure:

$$s_f \sim p^{\pi(\cdot|\cdot)}(s_{t+} \mid s_t, a_t). \quad (2.12)$$

For *negative* examples, we sample future states s_f from trajectories different from the one used to sample (s_t, a_t) .

With these definitions, the critic is composed of two networks $\phi(s, a)$ and $\psi(s_g)$ whose dot product encodes the Q estimate:

$$\begin{aligned} f(s, a, s_g) &= \phi^T(s, a)\psi(s_g) \\ \frac{p^{\pi(\cdot|\cdot)}(s_{t+} = s_g \mid s, a)}{p(s_g)} &= \exp(f(s, a, s_g)) \end{aligned}$$

The contrastive training objective can be written as:

$$\max_f \mathbb{E}_{(s,a) \sim p(s,a), s_f^+ \sim p^\pi(s_{t+}|s,a), s_f^- \sim p(s_{t+})} \mathcal{L}(s, a, s_f^+, s_f^-), \quad (2.13)$$

where

$$\mathcal{L}(s, a, s_f^+, s_f^-) = \log \sigma(\phi(s, a)^\top \psi(s_f^+)) + \log [1 - \sigma(\phi(s, a)^\top \psi(s_f^-))]. \quad (2.14)$$

where σ is the logistic sigmoid. This loss corresponds to binary classification of positive and negative pairs 2.2.1, and is provably minimized when the (exponentiated) similarity function $\exp(f(s, a, s_g))$ estimates the density ratio $\frac{p^{\pi(\cdot|\cdot)}(s_{t+}=s_g|s,a)}{p(s_g)}$ (3).

Intuitively, the critic learns embeddings for both the current state-action pair and potential future states, and measures how well these embeddings align. This alignment reflects how likely it is that taking the given action in the current state will lead to the goal, thereby serving the same role as a Q -function.

Once the critic is trained, the actor (goal-conditioned policy) is optimized using the DDPG+BC objective described earlier 2.10. The critic guides the actor toward actions that increase the likelihood of reaching the specified goal state s_g .

2.4 Related works in Representation Learning for Reinforcement Learning

A major limitation of modern reinforcement learning methods is their sample inefficiency: many algorithms require hundreds of thousands to millions of interactions to achieve reliable performance [58, 17, 16], especially in complex high-dimensional environments [60, 56]. In addition, RL agents often exhibit poor generalization beyond the training distribution, as they can overfit to spurious correlations in demonstrations or exploit biases in the reward function [39, 14].

Representation learning has emerged as a promising direction to mitigate these limitations. By introducing an explicit representation stage, raw inputs are transformed into a structured latent space that abstracts away irrelevant details while preserving task-relevant dynamics. Such learned representations provide richer learning signals, improve robustness to environmental variations, and facilitate transfer across tasks with differing input modalities (e.g., images, proprioceptive states, or action vectors) [3].

Representation learning has been incorporated in many prior RL methods. One line of representation learning to RL research focuses on compressing high-dimensional observations into compact latent spaces that retain only task-relevant information, thereby reducing the complexity of policy learning [13]. Beyond simply compressing observations, subsequent works demonstrated that stronger representations can be obtained by learning to predict future observations/states [61, 42, 43, 59, 18].

More recent work has found that coupling representation learning with explicit world models can be substantially helpful. In particular, these are model-based methods that learn world/dynamics models that can be rolled out across a time horizon like a simulator. The seminal *World Models* work [23] demonstrated this approach by training a generative model to reconstruct high-dimensional observations and using its latent features within a simple linear controller trained via evolutionary strategies. Despite being evaluated on relatively simple environments, it showcased that learning world models was now computationally feasible and that world model representations could enhance downstream policies.

Building on this foundation, TD7 [15] integrated representations from a world model into a downstream model-free policy, specifically TD3 [17]. On OpenAI Gym benchmarks, this combination yielded substantial performance gains—up to 276.7% and 50.7% over TD3 at 300k and 5M steps, respectively. MR.Q [16] further refined this framework by introducing auxiliary losses and implementation improvements, while preserving the TD7 training pipeline. These refinements enabled robust performance across a single set of hyperparameters and achieved state-of-the-art sample efficiency.

WM-CRL lies within the broader line of work on model-based representation learning, but it departs from prior approaches in several important ways:

1. **Focus on Generalization.** A central motivation of this thesis is to address the scarcity of high-quality demonstrations by finding methods that leverage suboptimal demonstrations. In contrast, much of the prior work has been driven primarily by concerns of *sample efficiency*, i.e., achieving high performance with fewer high-quality interactions.

To this end, we are the first to systematically evaluate world-model representations on datasets of varying demonstration quality. By analyzing performance across different dataset types for each environment, this thesis demonstrates that embeddings derived from world models can enable agents to generalize beyond their training distribution. Moreover, we provide ablation studies that identify the conditions under which such embeddings are most beneficial for generalization.

2. **Goal-conditioned setting.** We focus on goal-conditioned problems, where the objective is to reach specific states rather than maximize external rewards. This setting makes it natural to learn model-based representations, rather than prior works that attempt to learn predictive embeddings of both future state and reward. Many prior works explore learning to maximize cumulative reward, which typically requires intricate engineering to design a suitable reward function [16].
3. **Integration with CRL.** We incorporate model-based representations within contrastive reinforcement learning, rather than other model-free frameworks. For example, many prior works used the alternative TD3 framework [17]). CRL is especially well-suited to benefit from model-based representation learning, as it already leverages contrastive methods for state-action and goal representation learning. CRL also has benefits over other model-free methods, such as not requiring a reward function, avoidance of unstable TD-learning, and scalability to more complex long-horizon tasks [49, 12].
4. **Connection to Self-Distillation.** This work draws a novel connection between representation learning in world models and self-distillation methods. In particular, we adopt techniques such as cross-entropy-based objectives and exponential moving average (EMA) target network updates, inspired by recent advances in self-supervised learning [21, 6]. To our knowledge, this framing—together with the accompanying ablations—has not been explored in prior work.

Chapter 3

Methodology

We now describe the method proposed in this thesis **WM-CRL**— which trains a world model, and then integrates the *representations from the world model* into the contrastive reinforcement learning framework. The world model is trained to predict future state embeddings from past state–action sequences.

3.1 Motivation

A central challenge in training autonomous agents is the lack of large, high-quality datasets. Unlike fields such as natural language processing [50] and computer vision [33, 57], where massive pretraining corpora exist, many agentic domains—especially robotics—suffer from data scarcity [66]. Collecting diverse demonstrations of long-horizon behaviors is expensive and time-consuming, making it difficult to apply traditional sample-inefficient reinforcement algorithms at scale [60, 39, 14].

One avenue to overcome the lack of high-quality data is to find methods that can learn from suboptimal data. *Suboptimal data*, whether incomplete, shorter, or highly exploratory, are far easier to collect than long-horizon expert trajectories. Methods that can effectively leverage suboptimal data can potentially provide richer insights than relying solely on the relatively few expert examples. In domains like natural language processing, Large language models can achieve human-level conversation by pre-training on vast amounts of noisy web data and post-training with smaller quantities of curated chat dialogue [4, 19].

While high-quality demonstrations are expensive, suboptimal demonstrations are potentially more plentiful, such as short trajectories that do not demonstrate the longer

behaviors required at inference, or noisy trajectories that do not efficiently direct the agent to the goal.

Representation learning methods have historically been used to learn from raw data, such as unlabelled training examples or data unrelated to the task. By mapping raw sensory inputs into abstract latent spaces, representation learning enables the extraction of transferable features that are robust to noise and irrelevant variation. This approach has proven successful in many computer vision problems: representations learned from large collections of unlabeled images can be transferred to downstream tasks like classification [7, 36].

A viable avenue to overcome the bottleneck of high-quality demonstrations is to develop representation learning methods that can extract useful representations from suboptimal demonstrations. Such representations could help reinforcement learning agents succeed in tasks where high-quality demonstrations are scarce but suboptimal training data are plentiful. Ideally, representation learning should help agents generalize beyond the suboptimal training demonstrations.

CRL is a notable model-free actor-critic method that provides near state-of-the-art results with excellent scaling properties (Section 2.3.4). However, like many other offline reinforcement learning algorithms, its ability to generalize is constrained by the limitations of contrastive representation learning itself, which depends on large amounts of carefully curated negative pairs to produce robust features.

One promising direction to enhance CRL is the integration of world model representations. These representations can potentially offer two key advantages. First, unlike static encoders, world models capture the changes in the environment, and thus retain information about future time-step observations. Second, world models can be trained on suboptimal data, since meaningful dynamics can be learned from virtually any interaction with the environment.

The thesis investigates this question through *World Model Contrastive Reinforcement Learning* (WM-CRL), a framework designed to address generalization from plentiful but suboptimal training demonstrations. By leveraging predictive representations from a world model, WM-CRL aims to enable superior performance on suboptimal datasets. If successful in its aims, WM-CRL showcases the potential of this research direction in enabling agents to generalize beyond their training distribution and ultimately learn from both suboptimal data and high-quality demonstrations.

3.2 Training the World Model

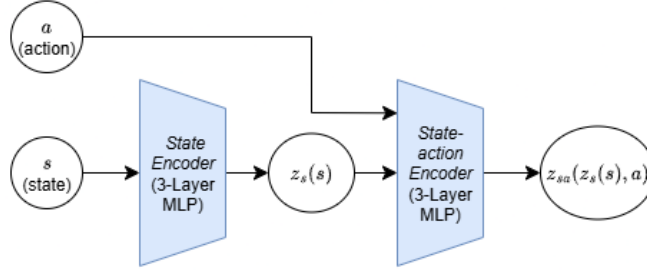


Figure 3.1: The world model is composed of two networks, whose representations are composed. The representation z_s is a function of only the state s . The representation z_{sa} is computed using z_s (not the raw state s) and raw action a .

We consider a standard reinforcement learning setting in which an agent interacts with an environment over discrete time steps. At time t , the agent observes a state $s_t \in \mathcal{S}$, representing the current configuration of the environment (e.g., high-dimensional sensory observations), and selects an action $a_t \in \mathcal{A}$, corresponding to a control input. In many practical domains, the state space $\mathcal{S}$ is high-dimensional and contains both controllable and uncontrollable factors, while the action space $\mathcal{A}$ is comparatively low-dimensional. This motivates learning compact latent representations that capture the controllable aspects of the environment relevant for predicting future dynamics.

We design our world model’s structure to learn to predict controllable and relevant aspects about the future. The world model will be trained to predict future latent embeddings instead of reconstructing raw future observations. Therefore, we decompose the world model into multiple networks (Figure 3.1):

- A *state encoder* $z_s : \mathcal{S} \rightarrow \mathbb{R}^d$, which maps each state to a compact d -dimensional latent vector. It represents controllable aspects of the environment.
- A *state-action encoder* $z_{sa} : \mathbb{R}^d \times \mathcal{A} \rightarrow \mathbb{R}^d$, which maps a state-action pair to a latent vector. Its features are trained to represent the subsequent state embedding.

A central goal behind world models is to build an internal model of how the environment evolves in response to a given action. Our loss functions are therefore designed with this intuition in mind: we want the state-action representation z_{sa} to carry predictive information about the future. Formally, given the current state embedding $z_s(s_t)$ and

action a_t , the predictor z_{sa} is trained to approximate the embedding of the subsequent state:

$$z_{sa}(z_s(s_t), a_t) \approx z_s(s_{t+1}).$$

This formulation trains z_{sa} to capture how actions drive state transitions, effectively encoding the environment’s dynamics into the latent space.

If we had instead trained the world model to directly reconstruct future observations s_{t+1} , it would encourage the model to also learn uncontrollable aspects of the environment, including stochastic or irrelevant variations [28, 8]. In addition, such an approach typically requires a computationally powerful generative model capable of capturing the distribution of high-dimensional observations, which can be inefficient and unstable [28, 27].

Our design encourages the world model to focus on controllable dynamics by predicting the future state embedding $z_s(s_{t+1})$ rather than the raw observation s_{t+1} . This formulation biases the encoder z_s to learn features that are predictable and controllable, while ignoring irrelevant variations. Moreover, learning to predict embeddings $z_s(s_{t+1})$ can be achieved with relatively simple neural networks and can be further stabilized with modern representation learning techniques (Section 2.2).

The world model’s objective function ($\mathcal{L}_{\text{world model}}$) is composed of one-step *transition losses* $\mathcal{L}_{\text{transition loss}}(s_{t+1}, s_t, a_t)$ summed over a multiple timesteps.

One-step Transition Loss Our thesis uniquely frames the world model as both learning a predictive world model [16, 15, 27] and self-supervised representation learning via self-distillation/bootstrapping. Inspired by prior self-distillation works like BYOL [21], which uses mean squared error (MSE), and DINO [6], which uses cross-entropy (CE), we explore both losses for the transition loss ($\mathcal{L}_{\text{transition loss}}(s_{t+1}, s_t, a_t)$).

The L2 Loss is defined as:

$$\mathcal{L}_{\text{transition}}(s_{t+1}, s_t, a_t) = \left\| \text{sg}(z'_s(s_{t+1})) - z_{sa}(z_s(s_t), a_t) \right\|^2, \quad (3.1)$$

where sg denotes the stop-gradient operation, and z'_s is a slowly updated copy of z_s . The target network z'_s is refreshed periodically every p steps to serve as targets. This loss function is also used in other model-based reinforcement learning works [16, 15, 27]. By using an L2 loss, the model is trained to regress directly in a continuous Euclidean embedding space, rather than predict a discrete distribution over latent states. The L2 loss frames the world modelling task to be a regression problem.

The CE loss is defined as:

$$\mathcal{L}_{\text{transition}} = \text{CE} \left(\text{sg} \left(\text{softmax} \left(\frac{z'_s(s_{t+1})}{\tau_t} \right) \right), \text{softmax} \left(\frac{z_{sa}(z_s(s_t), a_t)}{\tau_s} \right) \right), \quad (3.2)$$

where τ_s denotes and τ_t denoting the respective student and teacher temperatures. This loss function has not been used in prior model-based reinforcement learning works [16, 15, 27]. By virtue of using the softmax activation function, the embeddings z_s, z_{sa} will be discrete the world model’s task can be interpreted as a form of classification.

Features from z_s and z_{sa} are used for the downstream reinforcement learning agent. Many prior representation learning works [21, 6, 30] suggest directly comparing embeddings z_s, z_{sa} instead of the representations of the intermediate embeddings leads to less effective representations. However, in our ablations 4.4 we find that directly using features from z_s, z_{sa} gives better performance than intermediate embeddings.

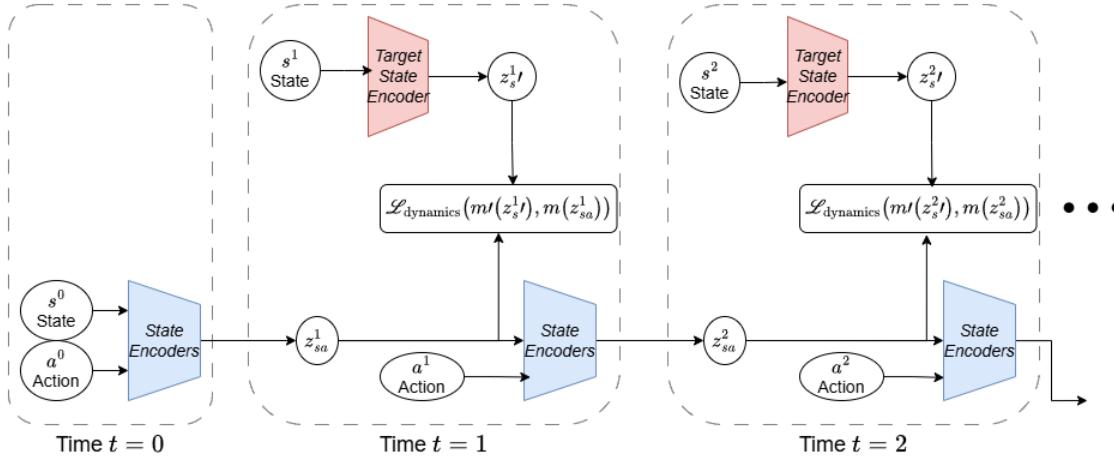


Figure 3.2: The world model’s loss is composed of multiple one-step transition losses summed over a time horizon H . To save space, we denote the time step as a superscript instead of a subscript. The representations z_{sa} are rolled out over time (the previous z_{sa}^{t-1} is used to compute z_{sa}^t). At a given time step, the transition loss compares the model’s predicted state representation $z_{sa}(z_s^{t-1}, a^{t-1})$ with the state representation of the target network $z_s'(s^t)$.

World Model Loss To encourage the world model to learn long-range behaviour, the overall loss is computed by rolling out predictions over a horizon H and summing the one-step transition losses (Figure 3.2):

$$\mathcal{L}_{\text{model}} = \sum_{h=1}^H \mathcal{L}_{\text{transition}}(s_{t+h}, s_{t+h-1}, a_{t+h-1}). \quad (3.3)$$

This also promotes representations that remain coherent over multiple timesteps.

3.3 Integrating Model-Based State Representations into CRL

We extend the CRL framework by incorporating embeddings from the world model networks, namely the state embedding z_s and the state-action embedding z_{sa} . These representations are integrated into **both** the actor and critic networks, as illustrated in Figure 3.3.

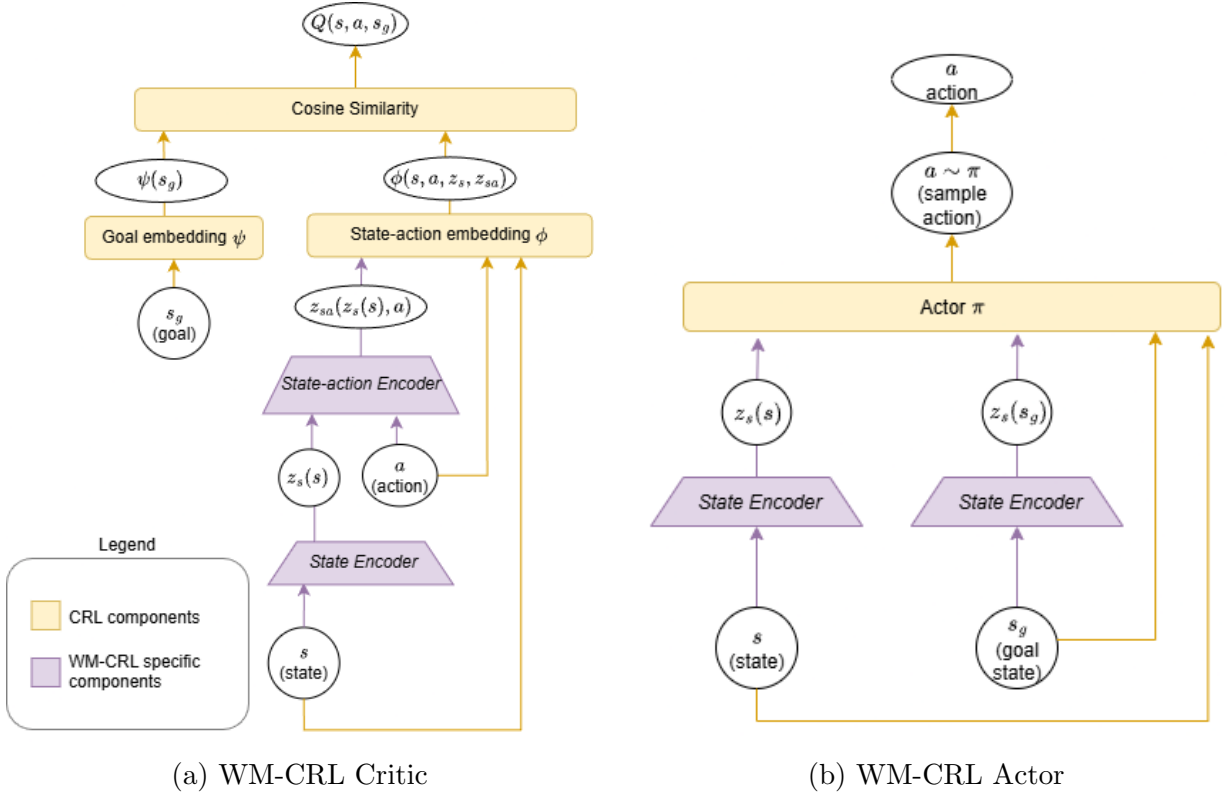


Figure 3.3: Integration of world model embeddings into Contrastive RL (CRL), forming WM-CRL. The critic’s state-action encoder ϕ incorporates z_s and z_{sa} alongside (s, a) and $\psi(g)$, while the actor π conditions on s , s_g , and the world model representations $z_s(s)$ and $z_s(s_g)$.

Critic integration. In the original CRL, the state-action critic encoder ϕ produces a d -dimensional embedding $\phi(s, a)$ from the raw state s and action a , while the goal encoder ψ computes $\psi(s_g)$ from the goal state s_g . In our WM-CRL variant, we augment ϕ by

appending the world model representations z_s and z_{sa} . The resulting similarity function becomes

$$f(s, a, s_g) = \phi(s, a, z'_s(s), z'_{sa}(z'_s(s), a))^\top \psi(s_g), \quad (3.4)$$

where z'_s and z'_{sa} are obtained from frozen target encoders. We choose to use the target networks instead of the online networks, as they are updated more slowly and thus provide more stable features for training. The new critic is visualized in Figure 3.3a.

Empirically, integrating WM embeddings into the critic substantially improves performance on certain locomotion tasks (Section 4.3). A plausible explanation is that CRL already requires the critic to capture environment dynamics, and the WM encoders provide structured, dynamics-aware representations that facilitate this process. Specifically, $z_s(s)$ summarizes the controllable aspects of the current state, while $z_{sa}(z_s(s), a)$ encodes predictive information about likely future states. Together, they enrich the critic with temporally meaningful features that improve its ability to evaluate whether (s, a) leads toward s_g .

Actor integration. The original CRL policy $\pi(s, s_g)$ conditions on the state s and goal s_g to output a distribution over actions, from which actions are sampled as $a \sim \pi(s, s_g)$. In WM-CRL, we augment the actor by providing WM embeddings of both the current and goal states:

$$a \sim \pi(s, s_g, z'_s(s), z'_s(s_g)), \quad (3.5)$$

as illustrated in Figure 3.3b.

We observe that adding WM embeddings solely to the actor (without modifying the critic) yields improvements in several manipulation tasks (Section 4.3). This is intuitive, since the actor must implicitly model the environment dynamics to select actions that guide the agent toward s_g . By operating in a latent space informed by the world model, the actor benefits from compact and semantically rich representations of the current and goal states, thereby simplifying the action selection problem.

Alternative integration strategies. In Section 4.3, we systematically compare several integration strategies: supplying WM embeddings to the goal encoder ψ , to the critic network ϕ only, or to the actor π only. While each approach improves performance in some environments, it also leads to degradation in others. The performance gains were consistently observed when training data was suboptimal as opposed to high-quality dataset settings. Interestingly, combining strategies often produced an averaging effect, where gains in some cases were offset by losses in others. No single strategy universally dominated.

We choose to integrate WM embeddings into *both* the actor and critic, as this design yielded the most **consistent** performance across environments. Other integration strategies could achieve superior results on a **specific** dataset or environment, but this integration strategy enhanced CRL more substantially across all tasks.

3.4 Training Procedure

Algorithm 2 Networks and training loop for WM-CRL. Here, let $s_t \in \mathcal{S}$ denote the state at time t , $a_t \in \mathcal{A}$ the action, s_g the goal state, H the trajectory horizon, T the number of training steps, w the warm-up period, and p the target network update interval.

Networks		Training loop
World Model		for t in 1:T
State encoder	$z_s(s)$	Sample trajectory $\{(s_t, a_t)\}_{t=1}^H$
State-action encoder	$z_{sa}(s, a)$	and goal s_g from training set.
Contrastive RL		Update z_s, z_{sa}, m with world model
Critic goal	$\psi(g)$	loss (Eq. 3.3)
Critic state-action	$\phi(s, a, z_s(s), z_{sa}(s, a))$	if t > w then
Policy	$\pi(s, s_g, z_s(s), z_s(s_g))$	Update ϕ, ψ with contrastive
		loss (Eq. 2.2.1)
		Update π with actor
		loss (Eq. 2.10).
		if t % p == 0 then
		Update target networks:
		$z'_s, z'_{sa}, m' \leftarrow z_s, z_{sa}, m$

Algorithm 2 summarizes the WM-CRL training loop. The design is intentionally simple: at each iteration, the state encoders, actor, and critic are trained jointly. To ensure stable world model representations, however, CRL training is delayed by a warmup of w timesteps.

We start by sampling a trajectory of length H from the training dataset $\{(s_t, a_t)\}_{t=1}^H$ and a goal state s_g . The trajectory is used to train the world model, and the goal state is used by the CRL framework. The state s_g is sampled as done in standard CRL (i.e. s_g is sampled t_g steps ahead of s_H where t_g is drawn from a geometric distribution) 2.3.4.

At every training step, the world model z_s, z_{sa} is trained with using the sampled training trajectory $\{(s_t, a_t)\}_{t=1}^H$. The world model loss is summed over the full horizon H as

described in Eq. 3.3. One can use either of the transition losses (Eq. 3.1 or Eq. 3.2), as we found both to give similar performance 4.4.

After w warmup timesteps, the actor and critic networks begin being trained on the state-action-goal triplets (s_H, a_H, s_g) using the standard CRL setup: the actor π is optimized with the DDPG objective 2.10 and the critic ϕ, ψ with the contrastive binary classification loss (Eq. 2.2.1).

We update the target networks z'_s, z'_{sa} every p steps. Every p steps, we hard-copy weights from the online network, following the approach in [15]. As such, the target networks z'_s, z'_{sa} are slowly updating versions of their online counterparts z_s, z_{sa} and provide stable training targets for the world model. We explore alternative ways to update the target network, namely with an exponentially moving average, in Section 4.4.

Chapter 4

Experiments

4.1 Experimental Setup

We evaluate WM-CRL on various goal-conditioned reinforcement learning tasks and compare it to standard CRL. We find that world model representations from WM-CRL do benefit CRL under low-quality or suboptimal demonstrations (5/6 tasks). Here, suboptimal demonstrations refer to training data generated by policies that are noisy, exploratory, or do not reliably achieve the task goal, resulting in trajectories that deviate from expert or near-optimal behavior.

Benchmark. We utilize tasks from the OGBench benchmark [48], which contains a diverse suite of tasks specifically designed for offline goal-conditioned reinforcement learning. OGBench spans a wide range of locomotion and manipulation environments that differ in state-space complexity, degree of multi-step reasoning required, and dataset quality.

To ensure the learning problem remains tractable for the agents, we provide observations in the form of low-dimensional state features rather than high-dimensional images. These state features include quantities such as (x, y) positions, joint angles, and velocities, which directly describe the agent and environment dynamics. Exploring image-based observations, which introduce additional challenges in representation learning and are too computationally demanding for our compute infrastructure, is left for future work.

We evaluated our method on several manipulation tasks, where the agent has control of a 6-DoF UR5e robot arm [68] with a table in front of it:

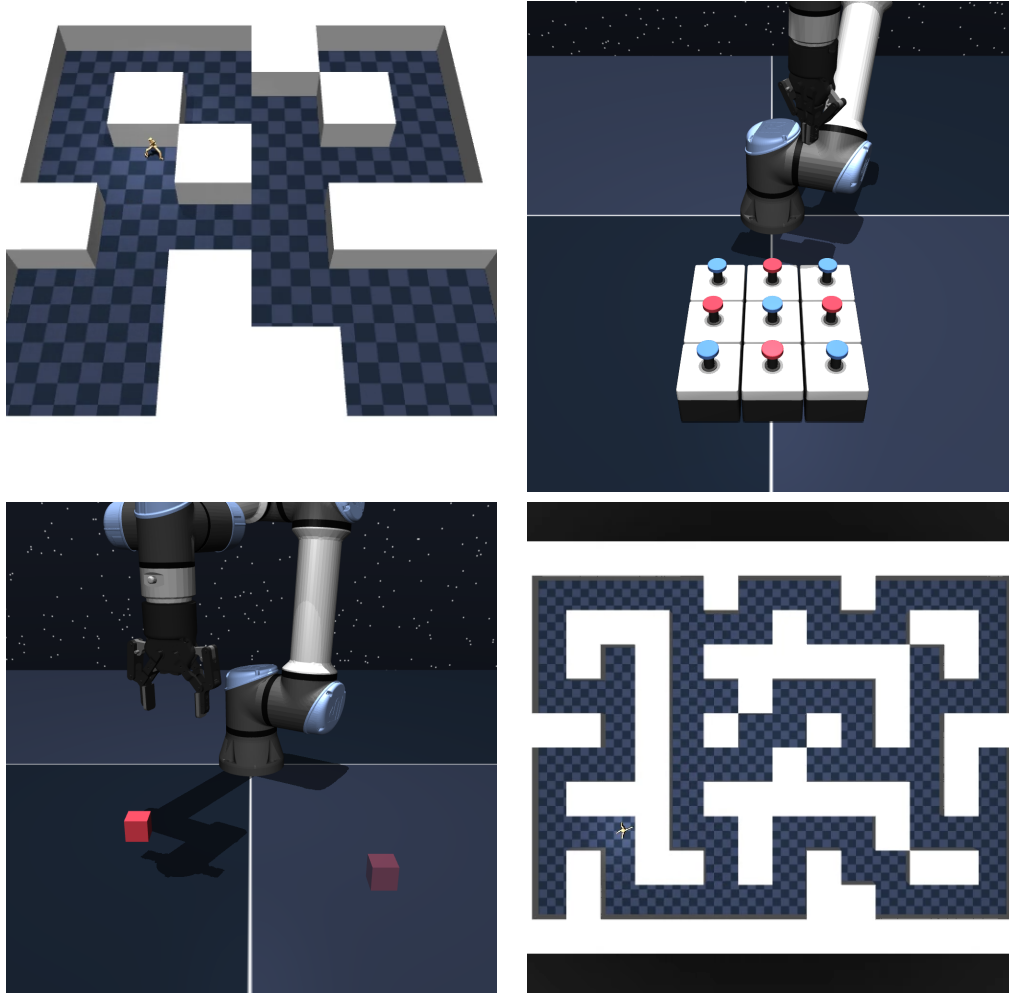


Figure 4.1: Example environments from the OGBench benchmark [48]. Top row: *HumanoidMaze* (left) and *Puzzle-3x3* (right). Bottom row: *Cube (single)* (left) and *AntMaze* (right).

- **Cube:** Placed on the table is a set of colored cubes, and the agent is given a target configuration. The task is to rearrange and stack the cubes to match the target configuration. Solving the task may involve stacking, moving, permuting, or swapping cubes. We adopt the *single* variant, where only one cube is manipulated. The agent is successful if the cube is within a specified distance from the target location, regardless of orientation.

- **Puzzle:** Inspired by "Lights Out" puzzles, the agent is placed in front of a square grid of buttons. Each button is either red or blue, and pressing it toggles its color as well as that of its neighbors. The goal is to make all buttons blue. Compared to Cube, this task poses a greater challenge due to its combinatorial nature: the agent must generalize across an enormous number of grid configurations while still mastering fine-grained low-level control. We use the 3×3 grid variant.

For both tasks, datasets are provided in two quality settings:

- *Play*: Consists of trajectories collected by an expert policy already capable of solving the problem.
- *Noisy*: Consists of trajectories collected by an expert policy corrupted with high levels of uncorrelated gaussian noise. Thus, the trajectories are highly random/exploratory and do not showcase efficient goal-directed behaviour.

In addition to the manipulation tasks, we also evaluate our method on several locomotion tasks. The agent starts in a 2D maze and succeeds if it reaches a designated goal location within a specified distance:

- **PointMaze:** The agent is a point-mass with 2 degrees of freedom (DoF).
- **AntMaze:** The agent is a quadruped with 8 degrees of freedom (DoF). As the agent has more DoF than the point-mass, this task is harder than PointMaze.
- **HumanoidMaze:** The agent is a humanoid with 21 degrees of freedom (DoF). This is the hardest locomotion task, as it involves combining high-level navigation with complex low-level locomotion.

Each locomotion task is provided with multiple dataset variants, including the navigate variant and one or both of stitch or explore:

- *Navigate*: Trajectories collected from an expert policy already capable of solving the task. Almost all trajectories demonstrate successful maze-solving behavior.
- *Explore*: Random exploratory trajectories with no specified goal but broad state-space coverage.

- *Stitch*: Short, disjoint trajectory segments where training goals are very close to the start state (at most four cell units away). Nearly all training trajectories successfully reach the goal state from the start state. At test time, the agent is tasked with reaching goals much further away. Thus, the agent is required to compose up to eight fragmented behaviors into coherent long-horizon strategies.

We list the input state dimensions, action dimensions, and maximum horizon length for each environment in Table 4.1.

Baseline. We compare against Contrastive Reinforcement Learning (CRL) [12], as our method directly extends this approach by introducing representation learning.

Training. Following the standard OGBench protocol [48], each agent is trained for 1 million steps on the offline datasets. Agents are trained and evaluated on multiple random seeds: 3 seeds for ablation results and 6 seeds for the final results.

Evaluation metrics. We evaluate each trained policy by measuring its ability to reach a specified goal state within a fixed time horizon. Episodes terminate immediately upon successful goal attainment. As our primary metric, we report the *average success rate* (in %) computed over 100 episodes (5 distinct goal states and 20 distinct start states).

Hyperparameters for CRL components All CRL networks- ϕ , ψ , and π — are implemented as three-layer multilayer perceptrons (MLPs) with 512 hidden units per layer and GELU activations. We use the AdamW optimizer with a batch size of 1024 and a learning rate of 10^{-4} across all networks. The critic networks ϕ , ψ embeddings are 512 dimensions. We note that alternative dimensions could have been used, but we stay consistent with OGBench [48] when possible.

The actor’s input/output dimensions are particular to the task and displayed in Table 4.1. We do not use any ensembling in the critic function as done in [17, 12], as we did not observe any performance improvement with ensembling turned on and did not find ensembling parameters stated in the paper [48]. We set the behavior cloning weight to $\alpha = 0.1$ for all tasks, except for `pointmaze-medium-stitch` and `cube-single-play` where we follow [12] and use $\alpha = 0.03$ and $\alpha = 3.0$, respectively.

Environment	State Dimension	Action Dimension	Maximum Episode Length
pointmaze-medium-stitch	2	2	200
pointmaze-medium-navigate	2	2	1000
antmaze-large-navigate	29	8	200
antmaze-medium-explore	29	8	500
antmaze-medium-stitch	29	8	1000
humanoidmaze-medium-navigate	69	21	2000
humanoidmaze-medium-stitch	69	21	400
cube-single-noisy	28	5	1000
cube-single-play	28	5	1000
puzzle-3x3-play	55	5	1000
puzzle-3x3-noisy	55	5	1000

Table 4.1: Environments used in our experiments, along with their state dimensions, action dimensions, and maximum episode lengths.

Hyperparameters for World Model components For all our WM-CRL experiments, we update the target networks z'_s, z'_{sa} every $p = 500$ steps by hard-copying the online network weights. We compute the dynamics losses over rollouts with horizon length $H = 15$, consistent with prior work [15, 16]. The world model embeddings were $d = 256$ dimensions, and both networks z_s, z_{sa} were three-layer MLPs with 512 hidden units per layer and GELU activation functions. Training includes a warm-up period of $w = 50,000$ steps, during which the world model embeddings z_s, z_{sa} are trained but the CRL actor-critic networks ϕ, ψ, π are not.

4.2 Comparison with CRL

In this section, we compare the results of WM-CRL with standard CRL. Table 4.2 shows the average success rates after training the agents on each dataset with six random seeds. Overall, we found that world model representations are particularly valuable when training data is suboptimal, and successful performance requires the agent to generalize beyond the demonstrated trajectories.

Experimental Setup. We evaluated WM-CRL on a dozen datasets from OGBench, with six being suboptimal datasets. Among the suboptimal datasets were three stitch tasks (pointmaze-medium-stitch, antmaze-medium-stitch, humanoidmaze-medium-stitch), two noisy tasks (cube-single-noisy, puzzle-3x3-noisy), and one exploratory task (antmaze-medium-explore).

WM-CRL Setup We chose to train the final world model with L2 Loss, but expect similar results and observations had we trained with CE Loss (see 4.4). In addition, the final world model integrates the world model embeddings into two CRL components: the state-action critic $\phi(s, a, z_s(s), z_{sa}(s, a))$ and policy $\pi(s, s_g, z_s(s), z_s(s_g))$. The average success rates are computed after training the agents on each dataset with six random seeds.

4.2.1 Analysis

Our main conclusion is that world model representations benefit Contrastive Reinforcement Learning (CRL) trained agents when the training data is suboptimal, and success at inference time requires the agent to generalize beyond the demonstrated trajectories. As shown in Table 4.2, WM-CRL improved performance on **7 out of 12** benchmark datasets. The majority of these gains occurred in the *suboptimal data* settings: WM-CRL improved performance on 5 of 6 such tasks (all three stitch datasets, the exploratory dataset, and one noisy manipulation dataset). In contrast, improvements were observed on only 2 of 6 tasks drawn from *high-quality datasets* (navigate and play). In high-quality datasets, demonstrations already provide strong supervision, and introducing world model representations may inject unnecessary inductive bias or modeling error, which can interfere with CRL’s learned representations and lead to marginal performance degradation.

Our analysis in this section and the ablation studies (Sections 4.3, 4.4) provide further insight into these trends. They reveal that the benefit of WM-CRL generally depends on two factors. Firstly, the quality of the learned world model, as reflected in predictive loss and training stability, plays an important role in predicting whether the world model embeddings benefit CRL. Secondly, performance is very sensitive to which specific components of CRL make use of the learned representations. We find that no single integration strategy dominates across all environments.

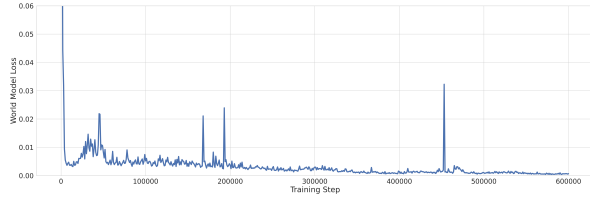
Environment	Dataset	CRL	WM-CRL
PointMaze	pointmaze-medium-navigate	39.0	23.7
	pointmaze-medium-stitch	13.5	19.2
AntMaze	antmaze-large-navigate	78.0	85.7
	antmaze-giant-navigate	22.0	17.0
	antmaze-medium-explore	2.3	3.0
	antmaze-medium-stitch	25.8	28.0
HumanoidMaze	humanoidmaze-medium-navigate	45.0	44.7
	humanoidmaze-medium-stitch	32.2	33.2
Cube	cube-single-play	23.7	17.0
	cube-single-noisy	45.5	43.3
Puzzle	puzzle-3x3-play	11.7	14.5
	puzzle-3x3-noisy	19.5	36.7

Table 4.2: Average success-rate on various OGBench environments across six random seeds. All metrics are average success-rate (%) across six random seeds. When both methods are available, the higher score is **bolded**.

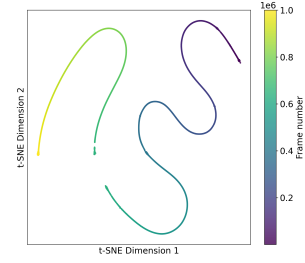
Point Maze Within the Point Maze domain, we evaluate WM-CRL on two datasets: `pointmaze-medium-stitch` and `pointmaze-medium-navigate`.

WM-CRL achieves a substantial improvement on the `stitch` task (+5.7%) but performs notably worse on the `navigate` variant (-15.3%) compared to the CRL baseline. The world model exhibits stable and monotonic loss curves throughout training (Figures 4.2a, 4.2c). The training loss was very low towards the end (0.01). In addition, its learned embeddings evolve smoothly over time, as illustrated by the t-SNE visualization in Figure 4.2b, where successive timesteps form a coherent and almost continuous trajectory in the latent space. Across ablations, the `stitch` task consistently yielded large gains, typically around +15.0% (Table 4.6).

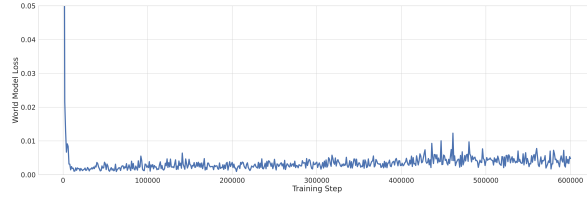
The Point Maze environment illustrates a clear example of where world model representations can meaningfully support CRL. The environment requires moving an agent with two degrees of freedom (DoF), which makes the transition dynamics relatively simple to approximate. This simplicity is reflected in the smooth loss behavior of the world model (Figure 4.2). In this setting, the `stitch` tasks provide clear evidence that world model embeddings help the agent generalize beyond the limited set of observed trajectories. By contrast, performance on the `navigate` variant decreases, suggesting that world model embeddings provide little benefit when high-quality demonstrations are already present.



(a) pointmaze-medium-stitch, WM Loss



(b) pointmaze-medium-stitch, t-SNE of z_{sa} across 1M timesteps



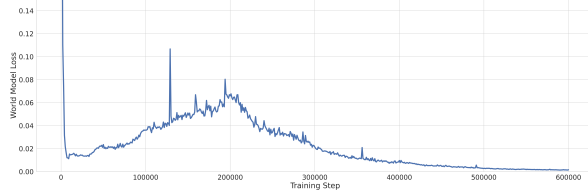
(c) pointmaze-medium-navigate, WM Loss

Figure 4.2: Plots from the **Point Maze** environment. Subfigures (a,c) showcase the world model loss 3.3 curves up to 600k timesteps. Subfigure b presents a t-SNE visualization of world model embeddings z_{sa} , compressed from 1M frames of a held-out validation sample. Colors indicate temporal progression through a validation trajectory, showing that the learned embeddings z_{sa} form smooth, continuous paths in the latent space as the agent moves through different states.

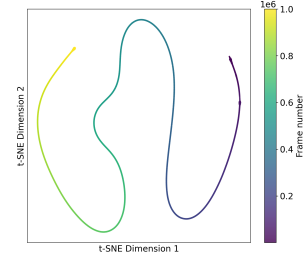
Ant Maze Within the Ant Maze domain, we evaluate WM-CRL on two suboptimal datasets: `antmaze-medium-stitch`, `antmaze-medium-explore`. We also evaluate on two high-quality datasets: `antmaze-large-navigate` and `antmaze-giant-navigate`.

WM-CRL provides modest gains on the `stitch` (+2.2%) and `explore` (+0.7%) tasks. Interestingly, WM-CRL showed gains on the `medium-navigate` variant (+7.7%), but caused a pronounced drop in performance on the `giant-navigate` (-5.0%) task. During training, we observed that the world model loss was not consistently monotonic, with loss sometimes increasing for several thousand iterations before decreasing again (Figure 4.3a, 4.3c). However, the learned world model embeddings evolve smoothly over time, as illustrated by the t-SNE visualization in Figure 4.3b, where successive timesteps form a coherent trajectory in the latent space.

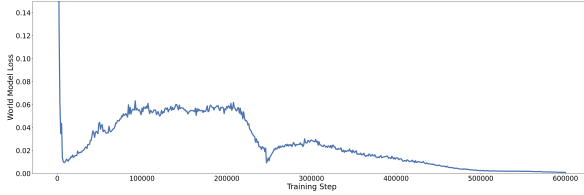
The results on Ant Maze suggest that even when the model struggles to learn the dynamics, its representations can still provide a useful signal for CRL. The Ant Maze environment requires controlling an agent with eight degrees of freedom, making it substantially more challenging than the two-degree-of-freedom Point Maze. This increased complexity is reflected in the non-monotonic training losses observed across tasks (Figures 4.3a, 4.3c, 4.3e, 4.3f). Despite these difficulties, the learned representations were sufficient to yield moderate improvements in the `stitch`, `explore`, and `medium-navigate` settings. In contrast, performance dropped on the `giant-navigate` variant, suggesting once again that world model representations provide limited benefit when the task can be solved effectively by direct imitation of the demonstrations.



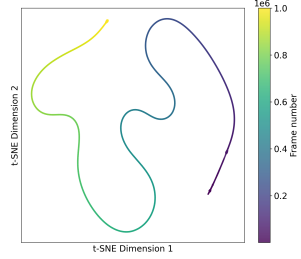
(a) antmaze-medium-stitch, WM Loss



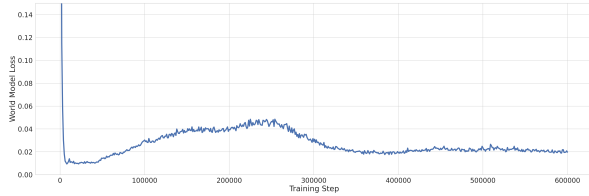
(b) antmaze-medium-stitch, t-SNE of z_{sa} across 1M timesteps



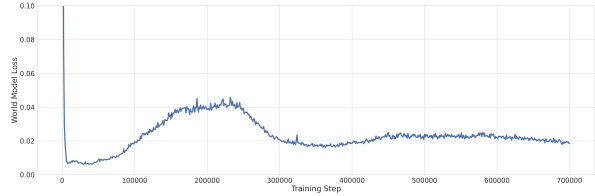
(c) antmaze-medium-stitch, WM Loss



(d) antmaze-medium-explore, t-SNE of z_{sa} across 1M timesteps



(e) antmaze-large-navigate, WM Loss



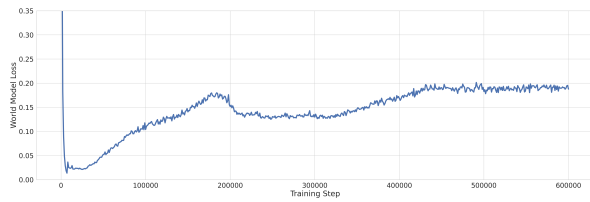
(f) antmaze-giant-navigate, WM Loss

Figure 4.3: Plots from the **Ant Maze** environment. Subfigures (a,c, e, f) showcase the world model loss 3.3 curves up to 600k timesteps. Subfigures (b,d) present a t-SNE visualization of world model embeddings z_{sa} , compressed from 1M frames of a held-out validation sample. Colors indicate temporal progression through a validation trajectory, showing that the learned embeddings z_{sa} form smooth, continuous paths in the latent space as the agent moves through different states.

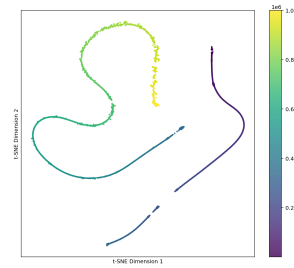
HumanoidMaze Within the Humanoid Maze domain, we evaluate WM-CRL on two datasets: `humanoidmaze-medium-stitch` and `humanoidmaze-medium-navigate`.

WM-CRL yields only a modest improvement on the `stitch` task (+1.0%) and a nearly negligible decrease on the `navigate` variant (-0.3%) relative to the CRL baseline. The world model losses were neither stable nor monotonic during training (Figure 4.4a). Also, the world model losses remained relatively high after 200k iterations (approximately 0.15). The learned embeddings evolved somewhat smoothly at first, but exhibited a pronounced discontinuity around 300k training steps and became increasingly fragmented in the final 200k iterations (Figure 4.4b).

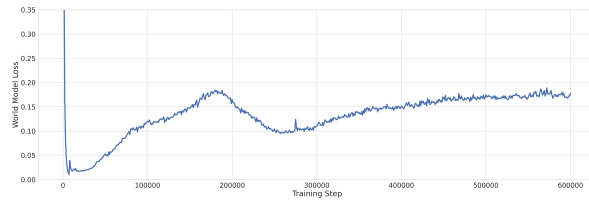
Humanoid Maze showcases how WM-CRL performs when the world model representations are poor. This is a substantially more difficult setting than the previous mazes, as the agent must control a 21-degree-of-freedom humanoid with complex, highly coupled dynamics. Due to the complex dynamics, WM-CRL struggled to model such dynamics, as reflected in the unstable loss curves. Nevertheless, the resulting representations were sufficient to provide a small benefit on the `stitch` task and did not degrade performance on the `navigate` task. The comparable performance between WM-CRL and CRL on `navigate` suggests that the CRL agent may have learned to downweight or effectively ignore the less reliable world model embeddings in this setting.



(a) humanoidmaze-medium-stitch, WM Loss



(b) humanoidmaze-medium-stitch, t-SNE of z_{sa} across 1M timesteps



(c) humanoidmaze-medium-navigate, WM Loss

Figure 4.4: Plots from the **Humanoid Maze** environment. Subfigures (a,c) showcase the world model loss 3.3 curves up to 600k timesteps. Subfigure b presents a t-SNE visualization of world model embeddings z_{sa} , compressed from 1M frames of a held-out validation sample. Colors indicate frame number, illustrating the trajectory and evolution of the embeddings over the course of training.

Cube We evaluate WM-CRL on two datasets: `cube-single-noisy` and `cube-single-play`.

WM-CRL exhibits performance reductions in both Cube tasks: the `noisy` variant (-2.2%) and the `navigate` variant (-6.7%) relative to the CRL baseline. In both cases, the world model training loss eventually converged to a low value (approximately 0.05). For the `noisy` variant, however, the loss was not consistently monotonic, often increasing for several thousand iterations before decreasing again (Figure 4.5a). By contrast, the loss curve for the `play` variant (Figure 4.5c) was relatively more monotonic. The corresponding embeddings evolved smoothly over time, as shown in Figure 4.5b.

Among the ablations, only the use of a cross-entropy (CE) loss for the world model (Table 4.7) produced more monotonic training behavior and led to improved performance on `cube-single-noisy` compared to CRL.

The Cube environment demonstrates a setting where WM-CRL embeddings do not generally benefit critic or policy learning. Manipulation tasks such as Cube involve high-dimensional state spaces and additional complexity from object interactions, which make environment dynamics difficult to approximate. Although the embeddings contained meaningful information (Figure 4.5)—as further supported by ablations using only embeddings z_s, z_{sa} (Section 4.3)—the downstream CRL components were unable to leverage them effectively. The improvement observed in the CE-loss ablation supports our claim that world model embeddings (z_s, z_{sa}) can aid CRL, but only when the features are stable and the training loss converges in a monotonic manner. Meanwhile, the absence of improvements in other ablations further confirms that, for Cube, WM-CRL representations offer limited or no advantage over standard CRL.

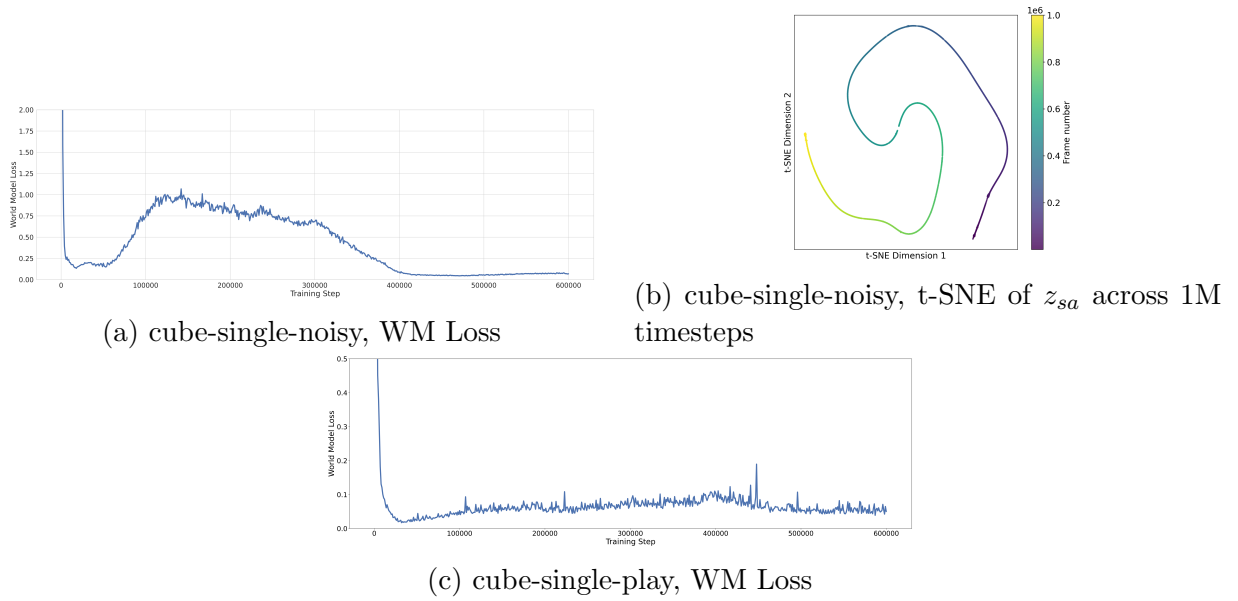
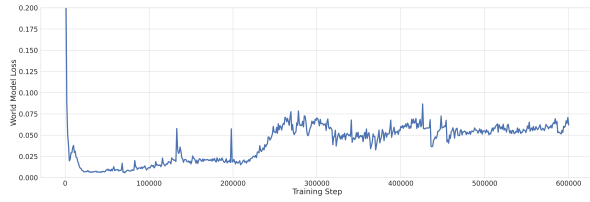


Figure 4.5: Plots from the **Cube** environment. Subfigures (a,c) showcase the world model loss 3.3 curves up to 600k timesteps. Subfigure b presents a t-SNE visualization of world model embeddings z_{sa} , compressed from 1M frames of a held-out validation sample. Colors indicate temporal progression through a validation trajectory, showing that the learned embeddings z_{sa} form smooth, continuous paths in the latent space as the agent moves through different states.

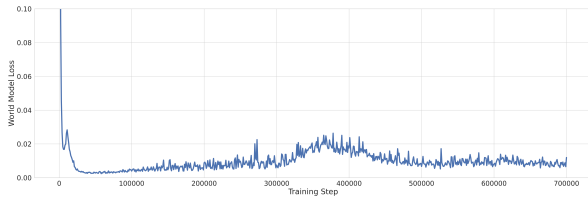
Puzzle We evaluate WM-CRL on two datasets: `puzzle-3x3-noisy` and `puzzle-3x3-play`.

WM-CRL exhibits clear performance gains in both tasks: the `noisy` variant (+17.2%) and the `play` variant (+2.8%) relative to the CRL baseline. In both settings, the world model training loss converged to very low values. For the `noisy` variant, however, the loss was not strictly monotonic and exhibited a sharp increase after 200k iterations (Figure 4.6a), though it remained near (0.05). By contrast, the loss curve for the `play` variant (Figure 4.6c) was more monotonic, stabilizing around 0.01. The corresponding embeddings evolved smoothly over time with minimal discontinuities, as shown in Figure 4.6b.

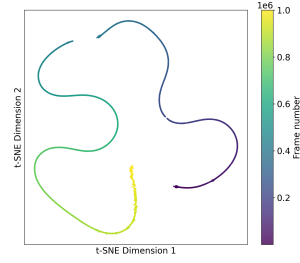
The Puzzle environment illustrates a setting where world model representations can substantially benefit CRL. Like Cube, Puzzle is a manipulation task involving a 6-DoF robotic arm, and both tasks feature high-dimensional state spaces. However, Puzzle differs in that it requires solving a combinatorial button-pressing challenge inspired by the “Lights Out” puzzle: pressing a button toggles its own color as well as that of its neighbors, and the goal is to make the entire grid uniform in color. Unlike Cube, Puzzle does not involve complex object interaction (such as pick and place), which makes the underlying transitions easier to model. As a result, the world model was able to learn useful representations of the environment (Figure 4.6). Crucially, the downstream CRL components—particularly the policy π (Section 4.3)—were able to effectively leverage these representations, leading to significant improvements in the noisy variant and modest gains in the play variant. These findings suggest that world model embeddings support *combinatorial generalization and reasoning*, enabling the agent to generalize beyond observed trajectories and solve novel grid configurations.



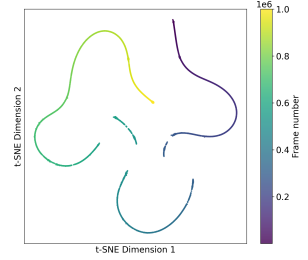
(a) puzzle-3x3-noisy, WM Loss



(c) puzzle-3x3-play, WM Loss



(b) puzzle-3x3-noisy, t-SNE of z_{sa} across 1M timesteps



(d) puzzle-3x3-play, t-SNE of z_{sa} across 1M timesteps

Figure 4.6: Plots from the **Puzzle** environment. Subfigures (a,c) showcase the world model loss 3.3 curves up to 600k timesteps. Subfigures(b, d) present a t-SNE visualization of world model embeddings z_{sa} , compressed from 1M frames of a held-out validation sample. Colors indicate temporal progression through a validation trajectory, showing that the learned embeddings z_{sa} form smooth, continuous paths in the latent space as the agent moves through different states.

4.3 Ablations on Integrating Representations with CRL

In the standard WM-CRL framework, the world model’s representations z_s, z_{sa} are integrated into both the critic’s state-action network ϕ and the policy π , while the remaining component—critic network ψ —are trained as in standard CRL. To better understand the role of these choices, we explored several alternative integration strategies. Although some variants achieved higher performance on specific environments, no single approach consistently outperformed the baseline across all tasks.

Over the course of many ablation on the integration strategy, we consistently observed the central theme of this work: world model representations are particularly valuable when training on suboptimal datasets that require generalization beyond the training distribution. We also run an ablation to verify that the embeddings z_s and z_{sa} encode semantically meaningful information, verifying their use in isolation of other features is sufficient to yield non-trivial downstream performance.

We make several experimental setup choices specific to this section. For efficiency, we restrict this study to six suboptimal datasets and three standard datasets, each evaluated over three random seeds. Furthermore, all world models are trained with L2 Loss. We omit CE-based variants, both due to computational constraints and because prior results (Section 4.4) demonstrated that L2 and CE losses exhibited similar qualitative trends.

Integration with Critic state-action network ϕ and Actor In this ablation, we evaluate how world model representations (z_s, z_{sa}) influence the downstream agent when integrated into the critic state-action network ϕ and the actor π . In standard CRL, the critic evaluates state-action pairs $\phi(s, a, z_s(s))$ and the actor outputs actions conditioned on state and goal $\pi(s, s_g)$. We extend this setup by incorporating world model embeddings in three ways, corresponding to the three columns of Table 4.3: (i) critic-only integration $\phi(s, a, z_s(s), z_{sa}(s, a))$ (first column), (ii) actor-only integration $\pi(s, s_g, z_s(s), z_s(s_g))$ (second column), and (iii) integration into both networks simultaneously (third column), which defines the standard WM-CRL agent.

Several trends emerge. The critic-only ablation (first column) outperforms the actor-only ablation (second column) **pointmaze-medium-stitch** by +21.7%. Meanwhile, the actor-only ablation (second column) outperforms the critic-only ablation (first column) on **antmaze-large-navigate** (+43.6%), **puzzle-3x3-noisy** (+30.0%), and **humanoidmaze-medium-stitch** (+10.7%).

Interestingly, the critic+actor ablation (third column) often yields performance that lies between only critic or only actor intergration, as seen on **pointmaze-medium-stitch**,

antmaze-large-navigate, puzzle-3x3-noisy, and humanoidmaze-medium-stitch. However, notable exceptions to this trend include antmaze-medium-stitch, where it performs substantially worse than both other ablations, and cube-single-noisy, where WM-CRL achieves better than either ablation.

These findings indicate that no single integration strategy dominates across tasks. Instead, different environments benefit from different modes of integration, and combining strategies tends to average their effects. Consistent with our main results, improvements from world model representations primarily appear in suboptimal dataset settings (stitching and noisy tasks), reaffirming the conditional utility of world model embeddings for CRL.

Dataset	Critic ϕ with WM	Actor π with WM	Critic ϕ + Actor π with WM
pointmaze-medium-stitch	37.7	16.0	25.7
antmaze-large-navigate	45.7	89.3	85.0
antmaze-medium-stitch	32.3	30.1	25.3
antmaze-medium-explore	4.5	5.3	3.3
humanoidmaze-medium-navigate	46.0	45.3	40.1
humanoidmaze-medium-stitch	21.0	31.7	30.1
cube-single-noisy	39.3	35.0	45.0
puzzle-3x3-play	13.7	14.0	12.7
puzzle-3x3-noisy	17.7	47.7	39.0

Table 4.3: Comparison of performance when using WM embeddings in the critic state-action network (ϕ), actor (π), or both (as used in standard WM-CRL 4.2). All metrics are average success-rate (%) across three random seeds. The best score for each dataset is **bolded**. Across many tasks, integrating WM embeddings into both networks yields more moderate, averaged performance, whereas integrating them into only the actor or only the critic leads to more extreme, task-dependent behavior.

Critic $\phi + \psi$ with WM In this ablation, we evaluate the effect of incorporating world model representations into the goal encoder ψ , such that $\psi(s_g, z_s(s_g))$. Recall that standard CRL uses two critic networks, where the value estimate is given by the cosine similarity between the embeddings of the state-action critic $\phi(s, a)$ and the goal encoder $\psi(s_g)$

(Eq. 2.3.4, 2.3.4). We extend this setup by incorporating world model embeddings into both networks: $\phi(s, a, z_s(s), z_{sa}(s, a))$ and $\psi(s_g, z_s(s_g))$. We then compare performance against the simpler strategy of integrating embeddings only into $\phi(s, a, z_s(s), z_{sa}(s, a))$.

Results are reported in Table 4.4 (first and second columns). Adding world model embeddings to ψ yields modest gains in some environments (e.g., **puzzle-3x3-noisy** (+3.6%), **cube-single-noisy** (+5.4%)) but leads to a notable performance drop in **antmaze-medium-stitch** (-10.3%).

Overall, this ablation shows that extending world model integration to both critic networks does not yield consistent benefits. While certain tasks show small improvements, these gains are offset by substantial drops in others. One possible explanation is that augmenting ψ introduces redundancy: both ϕ and ψ are then conditioned on z_s , which may bias the critic toward overemphasizing the world model features. As a result, the added complexity provides little net advantage over the simpler critic-only integration.

Dataset	Baseline (Critic ϕ with WM)	Critic $\phi + \psi$ with WM	Critic ϕ with only WM
pointmaze-medium-stitch	37.7	39.0	31.0
antmaze-large-navigate	45.7	46.7	30.0
antmaze-medium-stitch	32.3	22.0	33.7
antmaze-medium-explore	4.5	7.7	3.0
humanoidmaze-medium-navigate	46.0	38.3	34.3
humanoidmaze-medium-stitch	21.0	16.7	23.3
cube-single-noisy	39.3	44.7	39.3
puzzle-3x3-play	13.7	14.0	20.0
puzzle-3x3-noisy	17.7	21.3	18.3

Table 4.4: Ablation study on integrating world model (WM) embeddings into the CRL critic networks. All metrics are average success-rate (%) across three random seeds. The best score for each dataset is **bolded**. None of the ablations consistently improve performance across all tasks relative to the baseline.

Critic ϕ with only WM A natural question is whether the world model representations (z_s, z_{sa}) carry meaningful semantic information on their own. To evaluate this, we train

the critic state-action network using only these embeddings, i.e., $\phi(z_s(s), z_{sa}(s, a))$, without access to the raw state-action inputs (s, a) . Recall that in standard CRL, the critic operates directly on $\phi(s, a)$.

Results are reported in the third column of Table 4.4. Removing raw inputs leads to reduced performance on 5 of 9 tasks compared to the critic-only integration baseline (first column of Table 4.4). For instance, in `pointmaze-medium-stitch`, `antmaze-large-navigate`, and `humanoidmaze-medium-navigate`, success rates decrease by roughly 15%. Nevertheless, the critic still achieves around 30% success on these tasks, indicating that the embeddings alone preserve non-trivial semantic content.

Overall, this ablation demonstrates two points. First, world model embeddings indeed capture meaningful information that can support policy learning, even in the absence of raw observations and actions. Second, their effectiveness is maximized when combined with (s, a) inputs, suggesting that the embeddings complement rather than replace raw state-action features.

4.4 Ablations on World Model

We conduct a series of ablation studies on the world model (WM) in order to understand which design choices most influence downstream Contrastive Reinforcement Learning (CRL) performance. These ablations help to verify the important design decisions and understand how the world model features become useful for the CRL agent.

We make several experimental setup choices specific to this section. Due to computational constraints, we evaluate these ablations across the six suboptimal datasets, but with only three random seeds and three normal datasets.

We compare all ablations to the *Baseline WM-CRL*. In this baseline, we integrate world model embeddings z_s, z_{sa} only into the state-action critic $\phi(s, a, z_s(s), z_{sa}(s, a))$. The integration strategy is a control variable, and because we found it to be particularly influential downstream performance (Section 4.3), we choose just one strategy. However, we expect similar trends to hold with other integration strategies.

We begin by analyzing the effect of training the world model on L2 loss versus CE Loss (Figure 4.7, Table 4.5). Then we explore four types of ablation on the world model: (i) the loss function used to train the WM, (ii) the WM layer from which activations are passed to CRL, (iii) the prediction horizon H during WM training, and (iv) the update strategy for the target network (Table 4.6, Table 4.7). For cross-entropy loss, we set the

student temperature to $\tau_s = 1.0$, while the teacher temperature τ_t is annealed using a cosine schedule, starting at 0.04 and increasing to 0.07, following the setup in DINOv2 [46].

World Model loss function: L2 vs CE In this ablation, we compare performance when training the world model with an L2 reconstruction loss (Eq. 3.1) versus a Cross-Entropy (CE) classification loss (Eq. 3.2).

As discussed in prior representation learning works (Section 2.2.2), these two objectives induce qualitatively different representations. With L2 loss, the transition embedding $z_{sa}(s_t, a_t)$ is optimized to *reconstruct* the embedding of the next state $z_s(s_{t+1})$, yielding what we term *continuous-L2 embeddings*. In contrast, CE loss frames prediction as a classification task: embeddings (z_s, z_{sa}) are discretized through a softmax function and optimized to identify the most likely future state among a candidate set. We refer to these as *discrete-CE embeddings*.

Table 4.5 summarizes the results. Across the nine environments, both objectives yield broadly comparable downstream performance: continuous-L2 embeddings outperform discrete-CE embeddings on 5 environments, while CE embeddings perform better on the other 4. Although most differences are negligible, several tasks exhibit pronounced divergence.

- On **antmaze-medium-stitch**, continuous-L2 embeddings outperform CE embeddings by +5.0%. Here, CE loss led to representational collapse, with the world model loss vanishing midway through training (Figure 4.7a). Such collapse is a well-documented limitation of the CE (classification-style) objectives, which, unlike reconstruction objectives do not guarantee preservation of representation variance [6, 55]. This representational collapse makes the embeddings uninformative for the downstream critic ϕ , whereas L2-based training, though less monotonic, converged to useful representations (Figure 4.3a).
- By contrast, on **cube-single-noisy**, discrete-CE embeddings outperform continuous-L2 embeddings by +11.0%. In this case, CE training produced stable and monotonic convergence (Figure 4.7b), while L2 training exhibited non-monotonic fluctuations (Figure 4.5a), resulting in less reliable representations. Here, the stability of CE embeddings translated directly into higher downstream success rates.

These ablations highlight how different losses create different semantic embeddings, different learning curves, and ultimately yield different downstream performance. L2 training

provides robustness against collapse but often yields less monotonic convergence, while CE training produces smoother loss curves but is vulnerable to representational collapse.

Dataset	L2 Loss	CE Loss
pointmaze-medium-stitch	37.7	38.0
antmaze-large-navigate	45.7	51.3
antmaze-medium-stitch	32.3	27.3
antmaze-medium-explore	4.5	2.0
humanoidmaze-medium-navigate	46.0	47.0
humanoidmaze-medium-stitch	21.0	13.7
cube-single-noisy	39.3	50.3
puzzle-3x3-play	13.7	11.7
puzzle-3x3-noisy	17.7	16.7

Table 4.5: Ablation comparing different loss functions for WM-CRL. Results are shown for L2 and cross-entropy (CE) losses. All metrics are average success-rate (%) across three random seeds. The best score for each dataset is **bolded**. None of the ablations consistently improve performance across all tasks relative to the baseline, but reveal interesting information about WM-CRL.

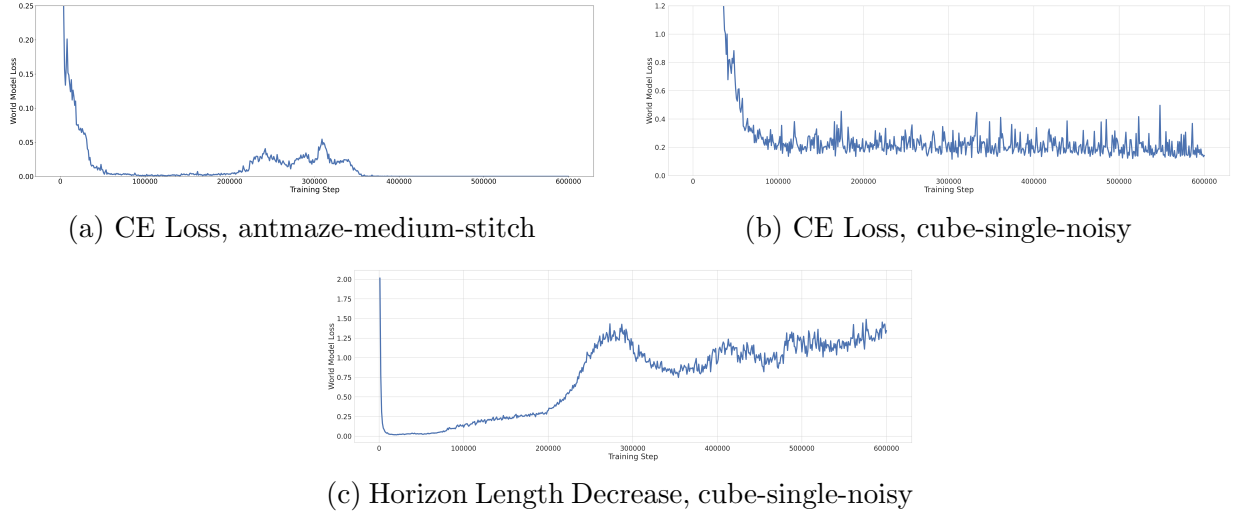


Figure 4.7: World model loss curves corresponding to ablations specified in the subcaption. To save space, we only show curves for specific datasets that illustrate particular concepts.

Dataset	Baseline WM-CRL (L2 Loss)	Intermediate Features	Horizon Length Decrease	EMA Target Network
pointmaze-medium-stitch	37.7	40.0	37.0	37.3
antmaze-large-navigate	45.7	44.3	40.7	45.3
antmaze-medium-stitch	32.3	21.0	27.7	28.0
antmaze-medium-explore	4.5	4.0	5.3	5.7
humanoidmaze-medium-navigate	46.0	45.0	40.0	37.7
humanoidmaze-medium-stitch	21.0	19.3	14.0	15.3
cube-single-noisy	39.3	39.5	39.0	47.3
puzzle-3x3-play	13.7	7.7	15.0	13.0
puzzle-3x3-noisy	17.7	9.3	14.0	12.0

Table 4.6: Ablations on training and usage of world models trained with **L2 Loss**. All metrics are average success rate (%) across three random seeds. Each column removes or modifies one component of WM-CRL. The best score for each dataset is **bolded**.

Intermediate Features In this ablation, we compare performance when using representations from the penultimate world model layer in the downstream CRL. More specifically, we modify the world model by introducing a linear projection layer m upon z_s, z_{sa} whose features are used to compute the world model loss (Eq. 3.3). We still integrate the representations into CRL’s ϕ network using the intermediate features z_s, z_{sa} (before projection m).

This ablation is inspired by prior representation learning works [7, 6], where it is common to leverage intermediate activations of the encoder for downstream tasks. A common explanation for this phenomenon is that penultimate features retain richer information, while the final layer collapses those features for the final prediction.

Our results indicate that incorporating intermediate features consistently degrades downstream performance: success rates decrease in 7 out of 9 environments under both L2 Loss (Table 4.6) and CE loss (Table 4.7).

This ablation suggests that the CRL agent interprets world model representations primarily as proxies for future states. Performance decreases when intermediate representations are used, indicating that the downstream CRL components struggle to exploit these unprojected—but potentially richer—features. We hypothesize that this reduction

Dataset	Baseline WM-CRL (CE Loss)	Intermediate Features	Horizon Length Decrease	EMA Target Network
pointmaze-medium-stitch	38.0	40.0	35.0	34.3
antmaze-large-navigate	51.3	44.3	51.0	42.7
antmaze-medium-stitch	27.3	21.0	22.3	20.0
antmaze-medium-explore	2.0	4.0	6.3	8.7
humanoidmaze-medium-navigate	47.0	45.0	42.3	44.7
humanoidmaze-medium-stitch	13.7	19.3	9.3	15.0
cube-single-noisy	50.3	39.5	40.7	44.7
puzzle-3x3-play	11.7	7.7	8.3	10.7
puzzle-3x3-noisy	16.7	9.3	19.7	18.3

Table 4.7: Ablations on training and usage of world models trained with **Cross-Entropy Loss**. All metrics are average success rate (%) across three random seeds. Each column removes or alters one component of the full WM-CRL model. The best score for each dataset is **bolded**.

arises because the critic is less able to interpret intermediate embeddings, which may be semantically misaligned with the task objectives.

A potential remedy would be to increase the capacity of the critic so that it is able to comprehend and use the world model features. For example, the critic could use additional layers or hidden units. Verifying this, however, remains an open question for future work.

Horizon Length Decrease We next ablate the prediction horizon H used in training the world model, decreasing it from $H = 15$ to $H = 7$. A shorter horizon should, in principle, simplify the dynamics prediction problem, since the model only needs to anticipate a few steps into the future rather than many.

Empirically, we find that success rates decrease in 7 out of 9 environments when using L2 Loss (Table 4.6) and in 8 out of 9 environments when using CE loss (Table 4.7).

This ablation suggests that using a longer horizon actually helps the representations become stronger. Our initial hypothesis was that shorter horizons would benefit challenging dynamics tasks (such as Humanoid Maze, Cube, and Puzzle) by simplifying the prediction problem to fewer timesteps. Closer inspection of world model training losses provides evidence against the initial hypothesis and towards the argument that longer horizons yield more regularized representations.

On tasks such as `cube-single-noisy`, shorter horizons yield unstable and non-monotonic convergence (Figure 4.7c), whereas longer horizons produce smoother and more stable behavior (Figure 4.5a). We hypothesize that predicting multiple steps ahead forces the world model to capture meaningful long-term structure in the environment dynamics, which reduces sensitivity to short-term noise. In this sense, longer prediction horizons act as a form of regularization, leading to more robust representations for downstream tasks.

EMA Target Network In this ablation, we evaluate the effect of updating target networks using an exponential moving average (EMA) procedure rather than the hard periodic update employed in baseline WM-CRL. In the baseline, the target encoders (z'_s, z'_{sa}) are synchronized with the online encoders every p iterations. By contrast, EMA updates adjust the target parameters as a moving average of the online parameters. EMA is often argued to provide smoother and more stable targets for representation learning. We follow [46] and apply a cosine schedule to the decay parameter, increasing it smoothly from 0.992 to 1.0 over the course of training.

Empirically, however, EMA updates did not yield consistent improvements. Under L2 loss, performance decreased in 7 of 9 environments (Table 4.6); under CE loss, performance

decreased in 6 of 9 environments (Table 4.7). The only notable gains were observed on `cube-single-noisy` with L2 loss (+8.0%) and on `antmaze-medium-explore` with CE loss (+6.7%).

These results suggest that, in our setting, the simpler hard update mechanism is sufficient to provide the student network with stable and useful targets. Rather than benefiting from the additional smoothing introduced by EMA, the downstream critic appears to perform better when the target encoders are sharply synchronized with the online encoders.

Chapter 5

Conclusion

5.1 Summary of Contributions

This thesis has explored how representation learning can be leveraged to address the challenges of data scarcity for reinforcement learning agents. While large-scale pre-training has enabled transformative advances in natural language processing, vision, and game playing, many agentic domains—such as robotics—lack access to vast, high-quality demonstration datasets. In these settings, agents need to learn from alternative data sources, such as suboptimal data (like noisy and fragmented demonstrations). Our work builds on the idea that predictive representations of environmental dynamics can help agents learn from suboptimal training distributions.

To this end, we introduced **World Model Contrastive Reinforcement Learning (WM-CRL)**, a framework that augments Contrastive Reinforcement Learning with representations from a learned world model. By incorporating predictive dynamics into the actor-critic of CRL, WM-CRL enables agents to extract richer information from suboptimal trajectories. This design allows agents to benefit from diverse sources of training data, including fragmented or exploratory demonstrations.

Our contributions can be summarized as follows:

- We proposed WM-CRL, the first integration of world model representations into contrastive reinforcement learning, and demonstrated its effectiveness on locomotion and manipulation benchmarks. We showed that predictive representations can help CRL agents learn under suboptimal datasets, showing WM-CRL outperformed CRL on 5/6 suboptimal tasks (Section 4.2).

- We recast world model training as a form of self-distillation, drawing connections to methods such as BYOL and DINO. This perspective motivated new design choices, including EMA target networks and cross-entropy objectives, which improved stability and downstream performance.
- We conducted a systematic evaluation and ablation study. We demonstrated that WM-CRL improved performance on noisy, fragmented, and exploratory demonstrations. However, the benefits of world model representations diminish when abundant expert data is available, and their effectiveness depends critically on the world model learning being stable and smooth.

Taken together, these findings underscore the potential of world model representations as a bridge between suboptimal training data and effective reinforcement learning. While our study has focused on goal-reaching agents, the framework and insights developed here open pathways for applying predictive representation learning more broadly to agentic tasks. By demonstrating both the benefits and limitations of WM-CRL, this work contributes practical guidance for future research at the intersection of representation learning, dynamics modeling, and reinforcement learning.

5.2 Future Research Directions

This work contributes to a growing line of research on helping goal-reaching agents extract more insight and generalize beyond their training data through representation learning. While our method advances this agenda by integrating world models with contrastive reinforcement learning (CRL), several important directions remain open for future exploration.

5.2.1 Limitations within the Current Setting

Our study focused on goal-reaching agents in locomotion and manipulation tasks, yet even within this domain, there are unresolved challenges.

Capturing dynamics in complex and interactive environments. Our state encoders sometimes failed to adequately capture environment dynamics in challenging settings. In particular, all ablation variants struggled with the high degrees of freedom (DoF) dynamics in `humanoidmaze-medium-navigate` (Section 4.2), as well as environments characterized by extensive object-agent interactions such as `cube-single-noisy`. Designing

training methods that yield more stable and expressive world models—capable of reliably capturing such complex dynamics—remains an open research problem.

Using multiple previous states and actions Our world models used architectures that conditioned on a single previous state embedding and action. A more powerful approach would be to train world models that can condition on multiple previous timesteps. Using sequence-to-sequence architectures (e.g., RNNs [11], Transformers [65]) could provide such capabilities.

Predicting multiple timesteps into the future Finally, our world models are trained to predict one timestep into the future. That is, given a stack of state-action pairs of length t , our world model could be rolled out to predict the next state embedding at time $t + 1$. However, a more powerful world model would be able to predict multiple timesteps into the future (e.g. $t + 2, t + 3, \dots$). While a sequence-to-sequence model could help here, it raises another challenge: the longer the horizon, the more strongly the learned dynamics become coupled to the policy that generated the training data. In suboptimal-data settings such as ours, designing multi-step world models that remain robust and policy-agnostic is a key open research problem.

5.2.2 Extensions beyond this work

Beyond our controlled experiments, several broader scenarios that change the experimental setup could present promising opportunities.

Real-world environments Firstly, while we evaluated exclusively in simulated game environments, deploying WM-CRL on real-world environments would provide a more rigorous and practical testbed. A particularly pertinent environment would be robotic systems, possibly simple tasks like pick and place.

Partially observable state Our experiments assumed fully observable states (e.g., direct access to positions, velocities, and accelerations). Extending to partially observable environments, where agents receive high-dimensional sensory inputs such as images or video streams, would constitute a more challenging and realistic setting.

Stochastic environments Our evaluation was restricted to deterministic environments. Many real-world domains exhibit a mixture of deterministic and stochastic dynamics. Incorporating stochastic world models—such as Recurrent State-Space Models (RSSMs) [27]—into the WM-CRL framework could enable agents to capture this richer structure. Investigating how such models balance deterministic and stochastic components of dynamics would represent a valuable case study.

Multi-task world models. In our approach, a separate world model is trained for each task, and our results demonstrated how world models can exploit suboptimal training data tailored to that specific task. However, a central advantage of world models is their ability to leverage data beyond the task of interest. Ideally, world models should learn dynamics for a particular environment from both task-specific and task-agnostic data, thereby providing more general representations that support downstream control. For instance, an agent trained across multiple maze environments could share structural knowledge of navigation. Another example could be an agent trained on several manipulation tasks could transfer insights across related subtasks within the manipulation domain.

Leveraging both high and low quality data with a two-stage training protocol In our experiments, we considered only single-stage training, where the entire dataset consisted either of expert demonstrations or of suboptimal trajectories (stitched, noisy, or exploratory). In contrast, many large-scale foundation models [4] adopt a two-stage process: a broad pre-training phase on diverse, potentially lower-quality data, followed by post-training or fine-tuning on high-quality demonstrations. Exploring how a similar two-stage strategy could benefit goal-reaching agents. We would start by first pre-training WM-CRL on suboptimal data. Then we would refine or post-train the world model, actor, and critic networks with expert demonstrations. This setup is more realistic, as in the real world, we typically have both low-quality and high-quality demonstrations.

In summary, while this work demonstrates that world models can enhance contrastive reinforcement learning under suboptimal data settings, advancing toward more general, robust, and real-world-capable agents will require progress on these open fronts.

References

- [1] Muhammad Qasim Ali, Winnie Trandinh, Howard Nguyen-Huu, and Alexander Wong. MR-CRL: Leveraging predictive representations for contrastive goal-conditioned reinforcement learning. In *Workshop on Reinforcement Learning Beyond Rewards @ Reinforcement Learning Conference 2025*, 2025.
- [2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.
- [3] Nicolò Botteghi, Mannes Poel, and Christoph Brune. Unsupervised representation learning in deep reinforcement learning: A review. *IEEE Control Systems*, 45(2):26–68, 2025.
- [4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.
- [5] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924. Curran Associates, Inc., 2020.

- [6] Mathilde Caron, Matthieu Touvron, Hugo ande, Piotr Joulin, Armand andb, Piotr Bojanowski, and Hervé Jégou. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9630–9640. IEEE, 2021.
- [7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020.
- [8] Anna Dawid and Yann LeCun. Introduction to latent variable energy-based models: a path toward autonomous machine intelligence. *Journal of Statistical Mechanics: Theory and Experiment*, 2024(10):104011, October 2024.
- [9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Tamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [10] Yilun Du, Mengjiao Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B Tenenbaum, et al. Video language planning. *arXiv preprint arXiv:2310.10625*, 2023.
- [11] Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.
- [12] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 35603–35620. Curran Associates, Inc., 2022.
- [13] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2016.
- [14] John Foley, Emma Tosch, Kaleigh Clary, and David Jensen. Toybox: Better atari environments for testing reinforcement learning agents, 2019.

- [15] Scott Fujimoto, Wei-Di Chang, Edward J. Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For sale: State-action representation learning for deep reinforcement learning, 2023.
- [16] Scott Fujimoto, Pierluca D’Oro, Amy Zhang, Yuandong Tian, and Michael Rabbat. Towards general-purpose model-free reinforcement learning, 2025.
- [17] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1582–1591, 2018.
- [18] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2170–2179. PMLR, 09–15 Jun 2019.
- [19] Gemini Team. Gemini: A family of highly capable multimodal models. 2024.
- [20] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, March 2021.
- [21] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020.
- [22] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Machine Learning*, pages 375–382. Omnipress, 2010.
- [23] David Ha and Jürgen Schmidhuber. World models. 2018.
- [24] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In

- Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.
- [25] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Mastering atari with discrete world models. In *International Conference on Learning Representations (ICLR)*, 2021.
 - [26] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Peter Flennerhag, and Marc Deisenroth. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.
 - [27] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.
 - [28] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.
 - [29] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022.
 - [30] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
 - [31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR (Poster)*, 2015.
 - [32] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.
 - [33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 25, 2012.
 - [34] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

- [35] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Gabriel Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- [36] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering*, 35(1):857–876, 2021.
- [37] Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724*, 2023.
- [38] Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. *arXiv preprint arXiv:1809.01812*, 2018.
- [39] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents, 2017.
- [40] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.
- [41] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- [42] Jelle Munk, Jens Kober, and Robert Babuška. Learning state representation for deep actor-critic control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4667–4673, 2016.
- [43] Tianwei Ni, Benjamin Eysenbach, Erfan SeyedSalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive RL. In *The Twelfth International Conference on Learning Representations*, 2024.
- [44] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [45] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael

- Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019.
- [46] 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*, 2024. Featured Certification.
 - [47] Seohong Park, Kevin Frans, Sungyoon Cho, Younggyo Seo, Jonathan Gordon, and Pieter Abbeel. Is value learning really the main bottleneck in offline rl? In *Advances in Neural Information Processing Systems*, 2024.
 - [48] Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *The Thirteenth International Conference on Learning Representations*, 2025.
 - [49] Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable, 2025.
 - [50] Guilherme Penedo, Hynek Kydlíček, Colin Raffel, Louis Fevry, Quentin Lhoest, Bram Van Der Goes, Lewis Tunstall, and Thomas Wolf. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. *arXiv preprint arXiv:2406.17557*, 2024.
 - [51] Francesco Piccialli, Diletta Chiaro, Sundas Sarwar, Donato Cerciello, Pian Qi, and Valeria Mele. Agentai: A comprehensive survey on autonomous agents in distributed ai for industry 4.0. *Expert Systems with Applications*, 291:128404, 2025.
 - [52] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.
 - [53] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
 - [54] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

- [55] Emanuele Sansone, Tim Lebailly, and Tinne Tuytelaars. Collapse-proof non-contrastive self-supervised learning. In *Forty-second International Conference on Machine Learning*, 2025.
- [56] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.
- [57] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. LAION-400m: Open dataset of CLIP-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.
- [58] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Chen. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [59] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021.
- [60] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Koray Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [61] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9870–9879. PMLR, 18–24 Jul 2021.
- [62] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, 2nd edition, 2018.
- [63] Vladimir N. Vapnik. Principles of risk minimization for learning theory. *Advances in Neural Information Processing Systems*, 4:831–838, 1992.
- [64] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

- [65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.
- [66] Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, Chelsea Finn, Max Du, Moo Jin Kim, Alexander Khazatsky, Jonathan Heewon Yang, Tony Z. Zhao, Ken Goldberg, Ryan Hoque, Lawrence Yunliang Chen, Simeon Adebola, Gaurav S. Sukhatme, Gautam Salhotra, Shivin Dass, Lerrel Pinto, Zichen Jeff Cui, Siddhant Halder, Anant Rai, Nur Muhammad Mahi Shafiullah, Yuke Zhu, Yifeng Zhu, Soroush Nasiriany, Shuran Song, Cheng Chi, Chuer Pan, Wolfram Burgard, Oier Mees, Chenguang Huang, Deepak Pathak, Shikhar Bahl, Russell Mendonca, Gaoyue Zhou, Mohan Kumar Srirama, Sudeep Dasari, Cewu Lu, Hao-Shu Fang, Hongjie Fang, Henrik I Christensen, Masayoshi Tomizuka, Wei Zhan, Mingyu Ding, Chenfeng Xu, Xinghao Zhu, Ran Tian, Youngwoon Lee, Dorsa Sadigh, Yuchen Cui, Suneel Belkhale, Priya Sundaresan, Trevor Darrell, Jitendra Malik, Ilija Radosavovic, Jeannette Bohg, Krishnan Srinivasan, Xiaolong Wang, Nicklas Hansen, Yueh-Hua Wu, Ge Yan, Hao Su, Jiayuan Gu, Xuanlin Li, Niko Suenderhauf, Krishan Rana, Ben Burgess-Limerick, Federico Ceola, Kento Kawaharazuka, Naoaki Kanazawa, Tatsuya Matsushima, Yutaka Matsuo, Yusuke Iwasawa, Hiroki Furuta, Jihoon Oh, Tatsuya Harada, Takayuki Osa, Yujin Tang, Oliver Kroemer, Mohit Sharma, Kevin Lee Zhang, Beomjoon Kim, Yoonyoung Cho, Junhyek Han, Jaehyung Kim, Joseph J Lim, Edward Johns, Norman Di Palo, Freek Stulp, Antonin Raffin, Samuel Bustamante, João Silvério, Abhishek Padalkar, Jan Peters, Bernhard Schölkopf, Dieter Buehler, Jan Schneider, Simon Guist, Jiajun Wu, Stephen Tian, Haochen Shi, Yunzhu Li, Yixuan Wang, Mingtong Zhang, Heni Ben Amor, Yifan Zhou, Keyvan Majd, Lionel Ott, Giulio Schiavi, Roberto Martín-Martín, Rutav Shah, Yonatan Bisk, Jeffrey T Bingham, Tianhe Yu, Vidhi Jain, Ted Xiao, Karol Hausman, Christine Chan, Alexander Herzog, Zhuo Xu, Sean Kirmani, Vincent Vanhoucke, Ryan Julian, Lisa Lee, Tianli Ding, Yevgen Chebotar, Jie Tan, Jacky Liang, Igor Mordatch, Kanishka Rao, Yao Lu, Keerthana Gopalakrishnan, Stefan Welker, Nikhil J Joshi, Coline Manon Devin, Alex Irpan, Sherry Moore, Ayzaan Wahid, Jialin Wu, Xi Chen, Paul Wohlhart, Alex Bewley, Wenxuan Zhou, Isabel Leal, Dmitry Kalashnikov, Pannag R Sanketi, Chuyuan Fu, Ying Xu, Sichun Xu, brian ichter, Jasmine Hsu, Peng Xu, Anthony Brohan, Pierre Sermanet, Nicolas Heess, Michael Ahn, Rafael Rafailov, Acorn Pooley, Kendra Byrne, Todor Davchev, Kenneth Oslund, Stefan Schaal, Ajinkya Jain, Keegan Go, Fei Xia, Jonathan Tompson, Travis Armstrong, and Danny Driess. Open x-embodiment: Robotic learning datasets and RT-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill*

Acquisition @ CoRL2023, 2023.

- [67] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator, 2023.
- [68] Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022.
- [69] Tianzhu Zhang, Jiashi Feng, Yew-Soon Ong, and Jianfei Cai. Relational knowledge distillation. In *arXiv preprint arXiv:1904.05068*, 2019.