Generating Interpretable Environment Families

by

Gregory Pylypovych
S.B. in Artificial Intelligence and Decision Making and in Mathematics, Massachusetts Institute of Technology (2024)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2025

Authored by:	Gregory Pylypovych EECS May 23, 2025
Certified by:	Pulkit Agrawal Associate Professor EECS Thesis Supervisor
Accepted by:	Katrina LaCurts Chair Master of Engineering Thesis Committee

Generating Interpretable Environment Families

by

Gregory Pylypovych

Submitted to the Department of Electrical Engineering and Computer Science
on May 23rd 2025, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The world holds a vast array of interactive behaviors that can be achieved by formalizing them in the context of a POMDP environment that we train a policy to solve. It is extensively documented that agent learning is environment-specific: the specifics of the environment, such as its action space, reward functions, and other properties, play a dominant role in whether an agent can successfully learn in this loop. Most major RL successes get around this by training on narrow task distributions, owing their success to the engineering of a fast accurate environment. Going through the world's tasks like this is a slow process, and leaves most tasks unsolved due to the difficulty of designing an environment compatible with RL. Exactly because of this difficulty, scientific progress on how environment properties shape learning has been limited. As learning is environment-specific, progress is additionally limited because studying general capabilities agents learn likely requires a large diversity of qualitatively different environments, and we lack approaches to generate these environment families. In this thesis, we outline the challenges and potential of generating these environment families, as well as formalize generating flexible interpretable environment families as a promising path towards this goal. As a step in this direction, we design and assess a toy learned flexible interpretable environment family.

Thesis Supervisor: Pulkit Agrawal
Title: Associate Professor

Acknowledgments

I would like to foremost thank my parents for supporting and being there for me in my life so far and beyond.

I want to mention my gratitude for Prof. Tamara Broderick and Prof. Leslie Kaelbling for supporting this work by appointing me a graduate TA-ship over Fall 2024 for 6.790 Graduate Machine Learning, which I had a pleasure teaching. I also want to mention my gratitude to Prof. Pulkit Agrawal for running the seminar that introduced me to deep reinforcement learning in Fall 2022, and since then providing a friendly lab atmosphere to learn about and work on advancing robotics, providing me compute for my experiments, and for providing support for this project as well as the many UROP projects that let me get to this point.

I additionally want to thank Seungwook Han, Akarsh Kumar, and Srinath Mahankali for sharing our common interests and the many useful conversations resulting from those over the past year at MIT pertaining to the motivating problems and specific details of this thesis.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

We want environments as complex as the real world.

The physics and structures present on earth today are extremely special. Despite strong consensus in the theory of evolution [13, 31], that across all scales our universe today is the product of a long process of selection for structures and interactions in our world that allow for it to become what it is now [26], it is hollow in terms of letting us understand how sensitive this process was to the events that built it up, understand the full distribution of what could be and letting us control our universe to what we want it to be. This is mostly because environment states and dynamics of the evolving world are what determine which structures will appear commonly or which interactions allow for some to be self-replicating, explicitly governing the evolutionary dynamics at play. So if we want to master the universe, the question becomes how do we get better at manipulating the structures that arise from processes of a similar class to ours?

Potentially, simply due to the number of unique interactions that make up the world, "the devil is in the details" and we can't expect models to simulate behavior as complex as we see make up interactions in the real world.

It is still intractable to simulate even 20nm wide structures like ribosomes due to our exact physics models calculating quadratic or even exponential amounts of interactions [1, 25] as well as being very sensitive to initial conditions and system parameters [30, 43] as well as to potentially missing key components. These will likely improve with consistent improvements in scientific instruments for measurement at these scales as well as data coverage of situations covering how these models should behave [22, 41], and lines of work inspired by the success of AlphaFold [23, 40] will be able to take advantage of this to slowly predict more and more dynamics. However, our world dynamics nevertheless have lots of chaotic dimensions to them [30], giving a big reason to remain skeptical that our not fully precise systems will be able to simulate enough of the interactions that govern our world well enough to understand and interventionaly control it based on simulation.

However there remains a strong hope for being able to predict and intervene the world to a serious extent: we can do it because humans can. Both innately and through human scientific understanding humans can feasibly shape and predict the world around them, and leave examples in the form of text, images, and video, but also in our buildings, institutions,

history, ecosystem, genes, and biological structure. The deep learning community has made consistent progress over the past two decades leveraging our vast repositories of internet text, image, and video data for training varying forms of world models such as GPT [48], Stable Diffusion [38], Sora [5], and Genie [7]. These research directions make the bet that the easily accessible plentiful modalities combined with expensive samples for specific behaviors we want to fine-tune will be enough to unlock serious amounts of these general-purpose human abilities. It is likely however that the other modalities are more likely to be responsible for human success, and as of now the relative scarcity of data capturing the specific ways they allow for human awesomeness limits our modern methods.

At both ends of this spectrum, it is possible that we will need to collect vast amounts of samples from our universe past what is economically affordable at the moment. However underlying that, we have two ends of a spectrum of methods that stretches between needing excessive amount of computation and data to simulate the world, and needing an excessive amount of computation and data to figure out which worlds we want to simulate.

In this thesis, I want to make some small concrete steps to inspire the latter direction of work.

1.2 Challenges

The environment design problem ties into critical challenges in various open research areas:

1. Exploration within sparse reward environments
2. Designing open-ended learning systems
3. Building world models
4. Transfer Learning and Meta Reinforcement Learning

1.3 Overview

A more comprehensive motivational section can be found in (chapter 2). A summary of related work and the current state of environment design can be found in (chapter 3) Definitions and formalization of the main perspectives and problems this work attempts to navigate can be found in (chapter 4). Details on the environment training process can be found in (chapter 5)

Chapter 2

Full Motivation and Aspirations

In this chapter we discuss the motivational problems that necessitate further developing the field of environment design. We the reason through the role environment design may play in scientific and engineering progress towards understanding the world. Additionally, we propose hypotheses as to what aspects of environment design will be key for this future.

2.1 Use of designed environment families for transfer

2.1.1 Designing uncontaminated environment families to use as transfer tasks

We are missing simple environment families to test for transfer. We have real world rollouts which are slow, but for fast transfer evaluation in simulation we simply have companies have their own narrow evaluations: robotics companies use a range of simulator configurations [47] and foundation model companies have performance on datasets of text-based questions. However there is a limited amount of environments that come with clear curricula on how to get better, as well as a limited amount to use for transfer learning research in non text-pretrained models. ARC-AGI was an example of such a family [10], however on it's own it's simply not big and diverse enough to sustain that much environment hill-climbing from leading AI labs before it gets saturated [28]. A process for generating diverse interpretable environments would allow study of transfer to controllable classes on new environments [11, 33].

2.1.2 Use of generated environment families for Meta-RL training

Another application of generating environment families would be to use the environment family as a controllable input to a transfer learning pipeline in order to scientifically study which aspects of the environment family lead to what kinds of transfer [32]. In contrast to the rigid or extremely narrow environments of today, if we had an a large diverse environment family that we had significant qualitative control over, we could explicitly study the effects of environment properties in aggregate on agent training [24, 20].

2.2 Use for foundation models in environment design

We haven't yet seen foundation models design creative environment families in interaction space. This is likely due to the lack of environment design data on the internet past a set of concrete families [45, 6]. Nevertheless, we hypothesize that the two types they will likely be useful for are:

1. **Predefined interaction patterns.** Within a programming language or simulator set-up illuminate the space of different configurations that the human baseline for is effectively blind search [18, 9]. Types of objects loaded into a scene, or what kind of entities, or whatever interaction types are easiest to code up in an environment [12, 49]. It's an interesting research direction to automate these and it will likely not be a challenge to illuminate since it is well represented in internet text in game/simulator design documentation [6, 21].

2. **Fast environment design loop.** With an a) expressive enough environment and b) specific fast transfer evaluation of how well/what skills environments are training in an agent [16, 29, 27, 46], you have enough online interaction samples that you can tractably do some form of zero-th order learning (evolution, reinforce) to get creative environments [44, 39, 14]. It remains to be seen if they are interpretable to humans or not [35, 8].

This being said, the challenge that empirically foundation models cannot achieve end-to-end at the moment are:

1. **Designing new environment languages**. Despite serious assistance as a tool [9, 2], it remains to be seen if foundation models have the ability to design lightweight, efficient programming/abstraction languages that allow for the design of qualitatively new environment families, distinct from those supported by existing languages [18, 42]. This can come in the form of a new simulator with efficiency improvements or a programming language for designing proteins to deploy to influence cell interactions [23, 3]. Combined with human taste and ingenuity, it will likely become significantly easier for domain experts to create novel types of environment families with foundation models [34, 49], but there is no strong evidence that we can automate this yet for use in an end-to-end open-ended system yet [46, 4].

2. **Effective exploration in novel environments**. Any open-ended learning loop will likely require an agent to start solving proposed sparse reward tasks in some part of it's training [17, 36]. The effectiveness of foundation models as high-level action spaces that reduce the effective horizon length in applications such as math competition problem solving [19, 12] shouldn't be taken for granted in other problem settings. Instead, we should progress pretraining high-level action priors through exposure to more designed environments [37, 4].

2.3 Developing Complex Interpretable Environment Families can be enough to transfer to the real world

When searching for transfer to the real world, the choices of environments are key. While much more well studied in setting of transferring to narrow tasks from training in near-accurate simulators, for the majority of transfer tasks we do not have such simulators and will need other methods of transfer.

2.3.1 Complexity is not enough

It is very possible that there will be no meaningful transfer you get from training on arbitrarily complex synthetic environments towards real world tasks we care about, simply because there is no good reason there should be a transfer. We have seen significant progress training on narrow task distributions such as math, coding, and games that only give the somewhat expected transfer improvements, which may be evidence for the hypothesis that we need to train in simulated analogues that contain more of the important interactions in environments we care to transfer to. It would be very interesting to see more research in this direction.

2.3.2 Domain Randomization and how we may just need environments that are complex and interpretable

It is not necessarily the case that we need a digital twin of the real world in order to be able to transfer to the real world. An important direction of advancement in deep reinforcement learning in the late 2010s was domain randomization, the practice of modifying system parameters of simulated environments agents would train in in order to hopefully learn to perform well on the distribution of tasks that captured the real world as well. This approach achieved impressive performance in locomotion and manipulation, as well as other application areas such as self-driving and complex multi-player reasoning games.

If we were to at a higher level apply this methodology to environment design, then perhaps we don't need a digital twin of the world, or anything close. Instead, if we train on worlds of similar complexity and structure to that of our world that may be enough. This according to our definitions in Section 4 comes down to training on complex interpretable sources.

2.4 Understanding What Capabilities our Environments Induce

Despite more of the world investing in the economic sector of training foundation models with RL for specific vertical tasks, due to the serious challenges of test-time reliability, out-of-distribution behavior generalization, and the sim-to-real gap, likely further work will go into pre-training base models in RL environments to have general test-time adaptation abilities. In working to train behavior priors that mesh well with human expectations, it will become

important to understand the connection between environment systems and the capabilities they train in order to control our systems on the level of the environments they are trained in.

2.5 Environments we have simulators for but very sparse reward

A ton of our fast simulator environments are defined more by human demonstrations of how to properly do tasks. Research math is the prominent example, in addition to scientific progress in the approximate simulators we do make up. If we could make flexible versions of these environments then we could make curricula denser with natural feedback loops.

Chapter 3

Related Work

In this chapter we discuss work related to environment design

3.1 Related Work

3.1.1 Unsupervised Environment Design

UED focuses on developing methods for training agents that can leverage a provided parameterized Environment Family that the agent can explicitly choose their environment from at all times.

Most state of the art work on unsupervised environment design takes lessons from self-supervised learning and focuses on it as an end-to-end problem, asking us to design environments and agents hand in hand optimizing towards a specific goal.

The main issues come across either in problems with exploration collapse or training over environment families not supporting high enough of a diversity of different interactions. Most work gets around these issues by focusing on parameterized environments hand-designed to have a small amount of interaction depth, but it remains unclear what sources will be able to scale up the generation of these environment families.

3.1.2 Generative World Models

Generative world models such as Veo3, Sora, the World Labs model, and Genie are trained on significant amounts of video data, and have been achieving impressive realistic video generation results from prompts and explicit action conditioning. With respect to environment design, these models can potentially serve as a source of environments that capture real-world dynamics and interactions. However, finding a way to covert these to an action-conditioned dynamics model with rich action spaces has not been straightforward, so the direction of progress has been building up environments with bigger and bigger action sets that compose interactions very well represented in video data.

A challenge of diffusion transformer world models is their dynamics being harder to interpret. It will require further work to discover what grounding we can rely on in environments derived from these world models, such as whether they can provide past in-distribution interactions or in what situations agents optimizing in these environments will be able to stay in a meaningful state distribution inside the neural world model.

3.2 Examples of Dynamics Models

1. Game of Life
2. ARC-AGI
3. Foundation World Models
4. Video Games
5. Robotics Simulators
6. Coding Environments

3.3 Observations about environment design

3.3.1 In most expressive parameterizations randomly selected environments are trivial

Taking chess as an example, if pieces were sampled from a distribution of possible piece moves, the chance that either a side will a a quick forced with or the game is a stalemate grows. More generally, most of our hand designed game/simulated environments support self-play and interesting behaviors solely because of the strong filtering over time of humans selecting for the rare variety of environment with "interesting" interactions.

3.3.2 The goal is increasing the number of qualitatively distinct interactions

Interactions are the real currency, and not behaviors. We can explain away Moravec's paradox by reconsidering human behaviors to instead be the product of training in an environment composed of a lot of interactions for a long time. Hence, it would be expected that an agent without a simulator that contains all of these interactions with their natural feedback loops would not be able to display the higher level behavior.

3.3.3 We can speed up our interaction

Defining human interpretable structure is linear in human-time. Despite this, it may take less time to define interactions if they arise from satisfying constraints we set. Not having to hard-code all the interactions we want agent to see and learn from in an environment/question of if that's what we're ultimately doing

3.3.4 The low-dimensional manifold of interactive tasks

The manifold hypothesis explains that what allows deep neural networks to learn and generalize in incredibly high dimensional spaces is that the high-dimensional data lies on a

low-dimensional manifold that the model learns. It is an open research question to figure out what the manifold of interactions found in the real world looks like, and any insight would give us better ideas on how to parametrize our environment design with the goal to simulate the important interactions that shaped human evolution in the real world.

3.4 Current bottlenecks of designing environments from human intuition

If we have a list of vague explanations of types of interactions we want what force stops us from directly feeding them to the agent? A big reason could be that most of natural human thoughts reference the real world environment that most foundation models do not have a good grasp of. Ideally the initial environments we train our agents in make it easy for our agent to act in a way that bestows it with the useful lessons that environment can teach. We then transition to environments that the agent can collect useful data in given its past experience and keep on putting the agents in more and more environments with interesting structure that the agent succeeds in collecting.

What stops this normally?

1. One big issue is that the curriculum is hard to define and expensive to do with human labor especially for arbitrary complicated tasks.

2. No guarantees that agent will never get stuck at a step in the curriculum rendering that curriculum useless

What if we instead trained our *environments* to contain capacity for interesting behavior. UED seeks to do this on fixed handmade environment distributions, but these distributions may be too narrow. Intuitively, decision making problems can take any form, whether it be chess or meta learning or bandits or deciding on which environments to train in. One advantage of thinking in terms of environments is that it is a term that allows us to abstract away the potentially vast number of forms that our decision making problem can take. We're now free to focus on how these environments differ, what makes some of them special, and how properties of them affect agent learning,

3.4.1 Real-world human capabilities are counterintuitively hard to define

A lot of the behaviors we think are simple are actually composed of way more interactions than we give credit for. Humans do not appreciating how hard certain tasks we find simple are. This is an example of Moravec's paradox, where despite humans being able to solve task families such as manipulation in the real world we struggle to fully articulate them computationally.

Chapter 4

Environment Design Framework

In this chapter we define and discuss properties of environments that compose to environment generation. We define a system of useful abstractions when designing more general types of environments, and discuss the goals and challenges compose to the usefulness of the environment for use cases such as human interaction or generating complex worlds.

4.1 Definitions

4.1.1 Environment Systems

The results of recent foundation model training efforts suggest that models continuously improve with larger diversities of interesting learning signals related to things in the real world. There is an abundance of environment instances which relate to understanding and creating tools for the real world. So finding ways to design *environment systems* training any real-world related ability (tool capable of achieving certain outcomes) past those we already demonstrated is useful.

We define an *environment system* to be a collection of:

1. Dynamics/parametrized process
2. Success objective
3. Human guidance overseeing/helping the process

All three of these work together to produce a certain ability. Simple examples are behavior cloning text with humans shaping model training based on downstream evaluation metrics or training robotics policies with RL in sim with humans shaping rewards, domain randomization, and system parameters trying to get visually convincing behaviors before deploying in the real world.

When defining standard RL environments, component 3) is often skipped, which doesn't make much sense when you consider that most successful deep learning decision making applications succeed exactly because of the expensive step of human guidance (e.g. hand-picked task demonstrations, specifically chosen pre-trained encoders, reward shaping, curriculum construction, etc.)

Viewing environment design through the lense of environment system design is more natural as we will discover below, because it formalizes that an environment is worth only as much as humans can interpret what goals the environment rewards for in agents.

4.1.2 Structures and Interactions

When designing standard well-studied environment systems such as researchers improving score on an episodic multi-agent video game or accuracy on a RLVR reasoning task, clarity about the exact details of what agents are being trained with what credit assignment towards what goals is already trickier than it appears on a surface level. When going past these to environment systems that become increasingly multi-agent, where persistence becomes poorly defined, and agents and action spaces are ambiguous, it is worth it to adopt language to discuss the grounding elements of these environments. Two simple examples useful for describing contents of an environment are structures and interactions.

A *structure* is a pattern that can be identified in an environment state. In the case where we treat environment state as translation-invariant, then structures also have position. We say a structure is *interpretable* if a human can notice when/where it is present with high accuracy. Examples could include agents represented as dots, a ring representing some barrier, some cell patterns known to be a molecule, or a robot arm identified with a monochrome patch of cells.

We refer to a progression of environment states with interpretable structures as an *interaction instance* or an *interaction*. Examples are the state trajectory that a specific humanoid goes through to grab an object from a specific orientation, the set of states an agent goes through to win a particular poker hand, or the sequence of structures a cell undergoing meiosis passes through.

4.1.3 Complex Interpretable Sources

There is an important difference between the classic self supervised learning and reinforcement learning problems when compared to the environment design problem. It is that at core, the environment design problem is not posed in the context of a *environment source* such as a large language dataset or a video game.

Define an *environment source* be any artificial or real-world process that the training environment of a policy is based on. This can be Conway's Game of Life, for which the corresponding environment can just be the simulation of game of life, or it can be internet videos of our world, in which case the environment is the video prediction task most amenable to deep learning.

Call it an *interpretable source* when it comes with an argument for why certain structures or interactions in the environment correspond to real world systems, or abstractions of those systems that we care about studying.

Call it a *complex source* when it contains a rich source of interactions that have potential for being as complex as those that make up the real world.

4.1.4 Interpretability interface

We define the *interpretability interface* to be the dense space where a human can ultimately reason about, understand and control the interactions the environment is composed of. This can be through language, logic, or any complex analogues for real-world reasoning that humans use.

4.1.5 Interpretable Environment Family

We define an environment system that can fit a variety of complex interpretable sources and also contains a functional interpretability interface an *interpretable environment family.* An interpretable environment family should be a straightforward way to generate a number of diverse environments that scale in complex interpretable sources and humans adept at using the interpretability interface.

4.2 Challenges

4.2.1 Parameterizing Environment Families is Hard

Deciding on some finite amount of parameters to change in a family empirically often leads to both:

1. Not being enough parameters to be able to get at the core intuition for why an environment is useful for training, such as there being a non-trivial equilibria we observe in a real version of the system, or there is some reward signal that is obvious in the real-world analog of the agents we train but is hard to specify in the simulator.

2. The parameters generating a set of interactions that grow repetitive

This has been seen in robotics, computational cognitive science, and probabilistic programming languages.

The most straightforward way to parameterize an environment family is to explicitly choose the axis of variation we want to vary the environment family over. This often is chosen to be attributes easily modified in the environment code, such as object placements, agent attributes, or general environment variables. The set of interactions an agent can experience in its environment is really sensitive to the environment's specific dynamics, and variation on these levels often leaves it unchanged, or radically changed to a set no longer useful to learn.

4.2.2 Transfer Learning

Scientific wisdom says that the right way to measure transfer is by having a set of pretraining tasks with held out transfer tasks that the model is evaluated on. We've recently seen lots of frontier work training large neural networks, however we're left with tasks that don't fall into the reasoning paradigm. Empirically it has been shown that data diversity and not data quantity is what drives capability improvements. However we have yet to see an example of transfer learning to an environment with interactions not present in training data.

4.2.3 Being able to effectively track diversity is unsolved

The exploration and open-endedness communities care about the problem of ascertaining the novelty or interestingness of a behavior to get signal for an AI system to explore in behavior space or update its curriculum. An equivalent problem is being able to numerically tell the diversity of a set of behaviors. In spite of continuing research advancement on metrics for diversity that work well in certain scenarios, such as gzip or Omni-EPIC, empirical evidence suggests developing automated measures for interestingness suffers from Moravec's paradox. All current mathematical definitions of diversity are either 1) intractable to compute (AIXI) 2) rely on representations which are poor for the aspects of tasks that keep them unsolved 3) aligned with human interestingness only on a toy environment distribution.

Developing automated interestingess measures seems approachable due to humans coming naturally pre-equipped with extremely effective. However this ability may be much rarer than this describes it to be, and instead be a product of human evolutionary ancestry having been selected to survive in Earth-like environments for billions of years.

4.2.4 Is designing interpretable environments too hard?

Humanity needed lots of scientific tools to be able to interpret the world the way we do now. Human ability to interpret with no education is best suited for in-the-wild survival, so its necessary to accept that our education is what allows for scientific understanding of any part of the world. Designing interpretable environments will have to be something driven by humans or AI systems that have a good enough understanding of parts of the world to design environments that use that complexity.

4.2.5 Requiring use of a zeroth order training method

It is much more sample-efficient efficient to get behavior from neural networks through first order gradient methods, which we can do whenever our objective is a differentiable function of the neural network outputs. If there are concrete outputs we have differentiable classifiers for, we can utilize a first order method.

However, in general, environment design already operates under the assumption that we can't exactly describe the full set of interactions we want to see in the environment, and if we can, then defining an environment that way isn't very scalable.

If we consider the mental model of layers of progressively higher order functions of our dynamics model in more abstract spaces, the literal outputs would lie in the inner core, followed by any concretely definable patterns in terms of those outputs, eventually building up to a hierarchy of larger and larger effects that we can't explain easily in terms of the dynamics model but can observe and select for.

First-order methods work well up until we hit these higher order effects that we can most easily directly identify and are hard to develop classifiers for without having already achieved the behaviour. At that point we likely need to rely fully on zeroth order methods.

4.3 Desiderata for Environments based on Complex Interpretable Sources

4.3.1 Same level of multi-modality as real world

The real world is composed of many interactions with high capacity for actions to control the future of the environment state. Deep learning systems and world models empirically struggle to be able to fully model the multimodal world, often collapsing to more likely outcomes, something that would be extremely detrimental to an environment where we want its intricate structure to guide an agent's understanding of the multimodality of the world.

4.3.2 Long-term dependence similar to real world

The real world is also characterized by long-term dependencies that most living organisms and inorganic matter obey. Persistence across time, long-term consequences of actions, as well explicit conscious choices as are all different aspects that contribute to the complexity of the real world.

4.3.3 A different type of environment flexibility

The types of environments most lacking currently are environments flexible enough to be able to reward for intuitive measures such as survival meaningfully. Most environments are designed for very narrow task families with fixed action spaces that are hard to use to define natural curricula like those that shaped evolution in the real world.

4.3.4 Speed

An environment that is parallelizable and fast is likely necessary to be able to iterate the design of this environment with respect to any outer loop.

Chapter 5

Training Environments

This chapter details describes the training process ofthe first steps toward an interpretable environment family. We cover specific design choices, challenges, and limitations of our method.

5.1 Guiding design principles

In trying to design an flexible interpretable environment family, we seek to make the the generation process for the family scale super-linearly in human time. Hypothesizing that we can find to break past needing to hard-code interactions linearly in human effort, we instead pursue generating interactions by spending most human effort 1) defining structures in the environment and 2) incentivizing interaction types that make sense within the interpretability interface.

5.2 Environment Parameterization

With the goal of getting environments with mechanisms that we believe we can transfer learn from, we train a neural network f_θ to be a state to state transition function $s_{t+1} = f_\theta(s_t)$. To make the environment more interpretable, we choose f to both:

1. Provide clear signal/structure as to which parts of the previous state individual parts of the future state depend on

2. minimize sensitivity to changes in the state that aren't semantically meaningful

We additionally want the dynamics function to be expressive enough to capture multi-modal stochastic distributions of futures based on the past. This combined with a render function (that we take to be the identity) $o = r(s)$ gives us a process $o_1, o_2, ..., o_T$ parametrized by θ. The render function should make defining human rewards/objectives on what values o_i should take where based on past o_j as easy as possible.

A simple starting point is to choose our state space to be an $n \times n$ grid of a discrete set of $k = 10$ colors. To train f_θ we pick a human-defined solid objective that makes sense in the context of this system (multi-agent competition, self-replication across scale, diversity across space, diversity across the temporal dimension, generation of specific structures, etc.) and

train f_θ on this objective with some zeroth order method. The main immediate bottleneck to this will be exploration, but this is where the flexibility of f_θ allows human interpretation of the process to reward shape however we like and through whatever intermediate behaviors it would make sense to do so.

The specific choice of f_θ is relevant both for the speed of the environment as well as the concrete interpretability interface of understanding our environment. We choose f to be a lightweight CNN with 3 layers, kernel size 3, and hidden dimension 128 employing skip connections. This choice not only speeds up our environment design process, but also allows us our interpretability interface an easier time identifying time-evolution of behavior to be a consequence of spatial propagation of structures in the environment in a causal way, as opposed to being unsure what is driving environment evolution behavior.

We additionally use JAX to JIT both our dynamics steps and our reward computation to drastically speed up our environment.

5.3 Initial State Distribution is Important

It would be cool to aim for abiogenesis by rolling out long context from from some initial states and train to get dynamics capable of generating something from nothing. However, there haven't been a dynamics models found yet that can be tractably rolled out and expected to generate an interpretable structure that can function as an agent in its environment, potentially because:

1. We haven't ran on big enough of an environment
2. Haven't rolled out the environments enough times to get this low-probability behavior
3. Our current environments aren't designed to allow for such emergent behavior to arise in the dynamics with any positive density

An initial state with meaningful structure can make it a much easier exploration task to train dynamics that allow for interactions that would involve the type of state the initial state is. Nevertheless, deciding what initial states to start the environment in adds extra complexity and so we stick to random initialization of the state by sampling all pixels uniformly among the colors.

5.4 Training Environments

5.4.1 Method

Aiming for a flexible environment design method that could scale the number of qualitatively different interactions in the number of defined structures in the system, we choose the expressive environment parameterization above that, importantly, is differentiable, meaning we can learn it.

We choose to formulate the structures, interactions, and constraints from our interpretability interface as rewards as functions of the state of the environment, and use RL to train this environment to maximize the rewards we specify.

As part of our interpretability interface, and to help with credit assignment, we define both global and local rewards on the state, with each generated pixel getting separate rewards from its neighbors.

We settle on a value function free method, PPO without value learning, as credit assignment is ambiguous in this setting and using value function is unneeded complexity.

5.4.2 Illuminating the interaction space of the model

In order to test whether our method can generate environments with dynamics that are non-trivial, we train our dynamics model on the local reward of cell-replication. A cell (i, j) gets rewarded for it's color being transmitted to a neighboring cell the next step. Some randomly chosen states within a single RL environment evolution run can be found in 5.1.

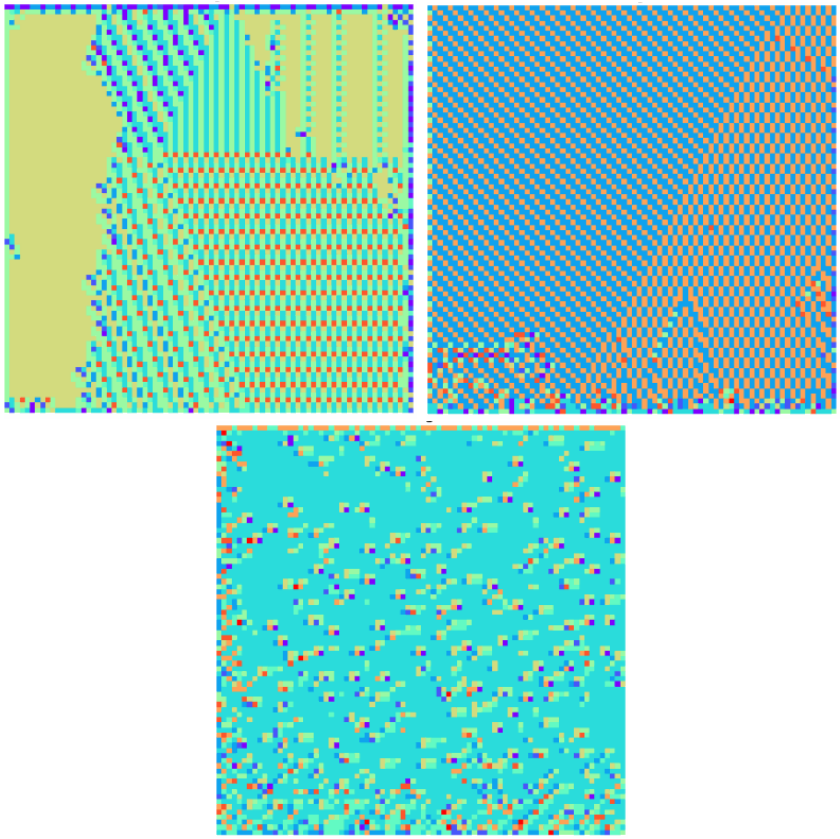

Figure 5.1: States explored by training environment with RL on single-cell self-replication objective

5.4.3 Training Setup

The above can a diverse number of different interactions, but at the price of not having control over the type. For environment design it is crucial to be honest towards what structures you are intentionally capable of bringing about, because without intentional control there is no guarantee that scaling the method will generate more than just the same.

To explore how to train the environment to have intentional structures, we shape the environment to have red 2by2 squares with no surrounding red or blue to be "agents" that we will shape to interact and persist in the environment.

We go about this in a three step process:

1. Define and shape structures into an environment
2. Shape some primary interactions involving structures
3. Train agents in this environment that identify with structures from step 1 that interact with the environment partially guided by step 2.

The first two steps are done by training our f_θ to maximize a chosen reward $r(s, s')$. We can additionally guide exploration by hard-coding simple actions in certain states when necessary.

5.4.4 Generating Intentional Interpretable Structures

We show the results of training red 2by2 structures to exist in the environment, training red 2by2 structures that move up/down/left/right by one square each step to exist in the environment, as well as training both red 2by2 and blue 2by2 structures that move up/down/left/right by one square each step to exist in the environment.

We use $\gamma = 0$ with an entropy term of 0.03, and $\epsilon = 0.2$, training with a minibatches of size 16 grids, and collecting 120 length 4 episodes for each update.

To give intuition for how the exploration challenge of environment design with our method scales, for creating red 2by2 structures, we find that we need no extra exploration tricks, for getting red 2by2 structures that can move around we need to hard code movements into actions of collected data to help exploration, and for getting both red and blue moving 2by2s we find that this task requires significant reward shaping in addition to exploration guidance. Resulting states can be seen in 5.2.

5.4.5 Multi-modality of the environment

The real world is multi-modal, with agents having empowered actions that qualitatively influence the outcome of the world around them. While there are still no formal measures for what this means precisely ([15] is a good first step for 3d world models), a good intermediate metric is how sensitive the environment evolution is to initial state. As we can see in 5.3, the environment consistency is rather fixed, with just positions of structures being the varying factor between different worlds. If we wanted this dynamics to be able to represent an environment with more than this one setting, our method may need further modification, ideally getting around the hypothesized multimodal collapse seen in RL training.

This may also be an issue with our initial states being too random, leading to all being effectively learned to correspond to the same state.

Alternately, this may be an issue with us not actively exploring the actions that may take us more out of this distribution.

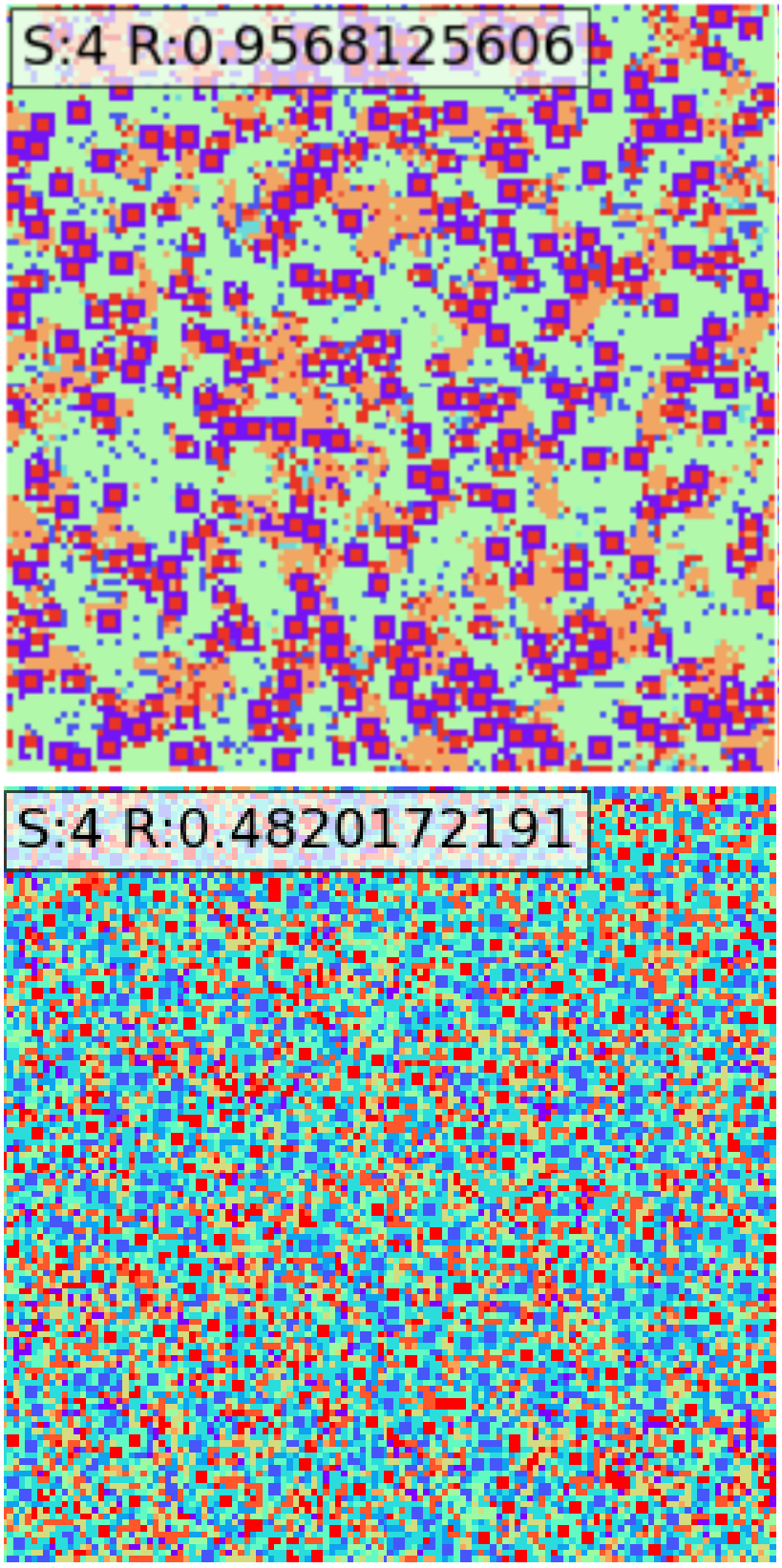

Figure 5.2: States in environment shaped to contain red 2x2 agents (above) and red 2x2 and blue 2x2 agents (below)

This also serves as an example of the environment being overly robust to agents optimizing in it, simply because it is so heavily regularized to staying within a certain state distribution and the agents have little empowerment.

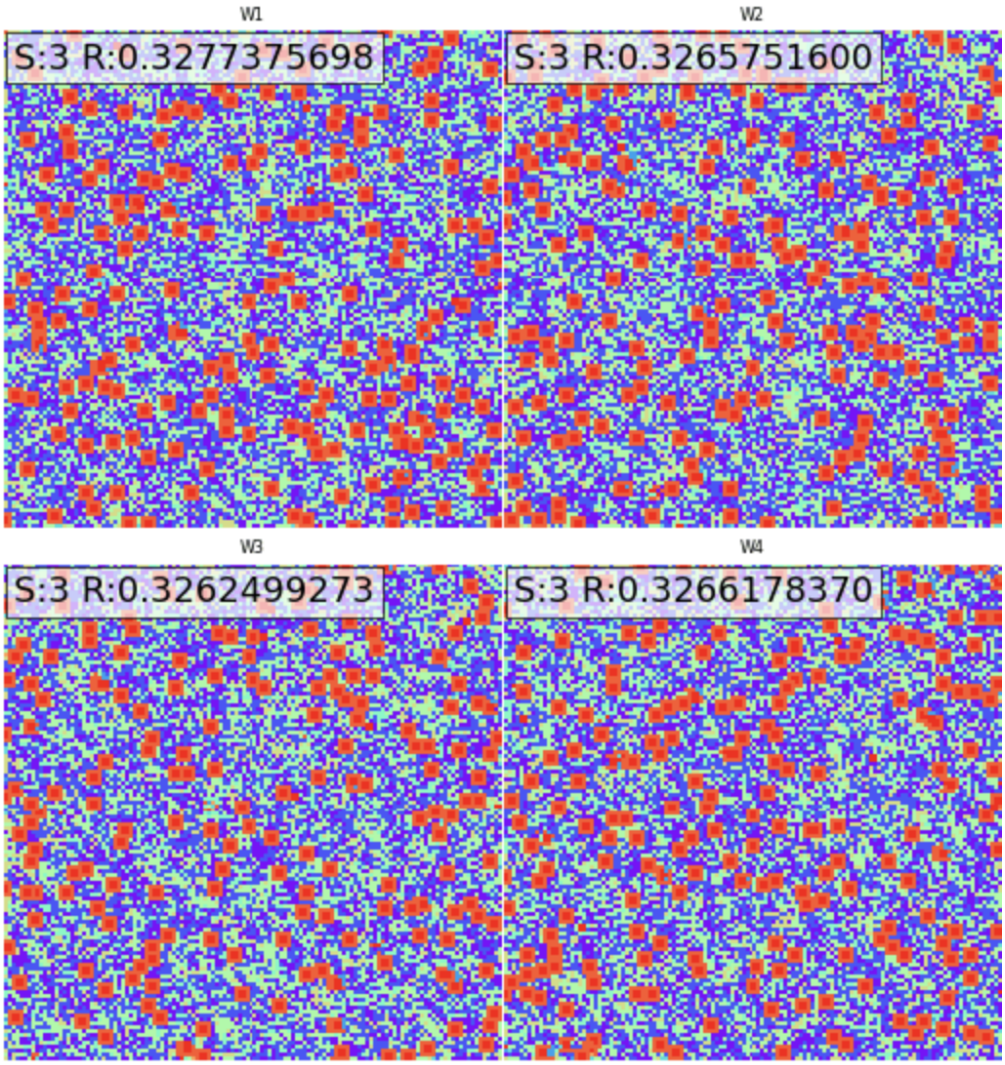

Figure 5.3: A snapshot of a single dynamics model rolled out over four different seeds

5.4.6 Action Space

An advantage of designing environments as a state transitioning dynamics model is that the action space is not fixed to the set of interactions that are pre-coded into the environment. This can give more flexibility to define environments that have more amorphous agents with more multi-agent dynamics and interaction. We show in 5.4 and 5.5 how actions sampled from two different action spaces empower the modification of an agent's surroundings and its own persistence in the environment.

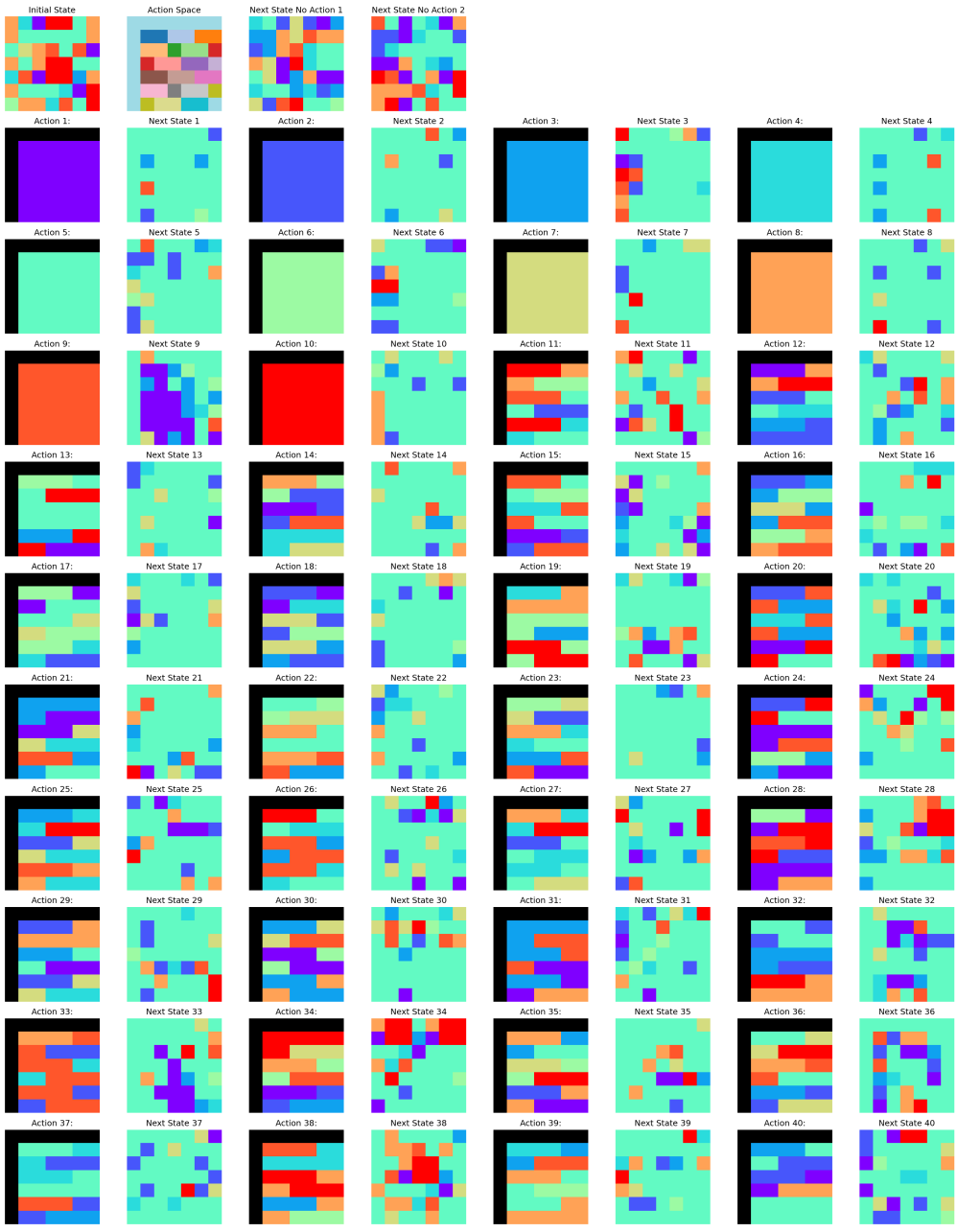

Figure 5.4: A snapshot of how a single agent with a larger but less refined action-space of the 6x6 neighborhood surrounding it can affect its environment over one step

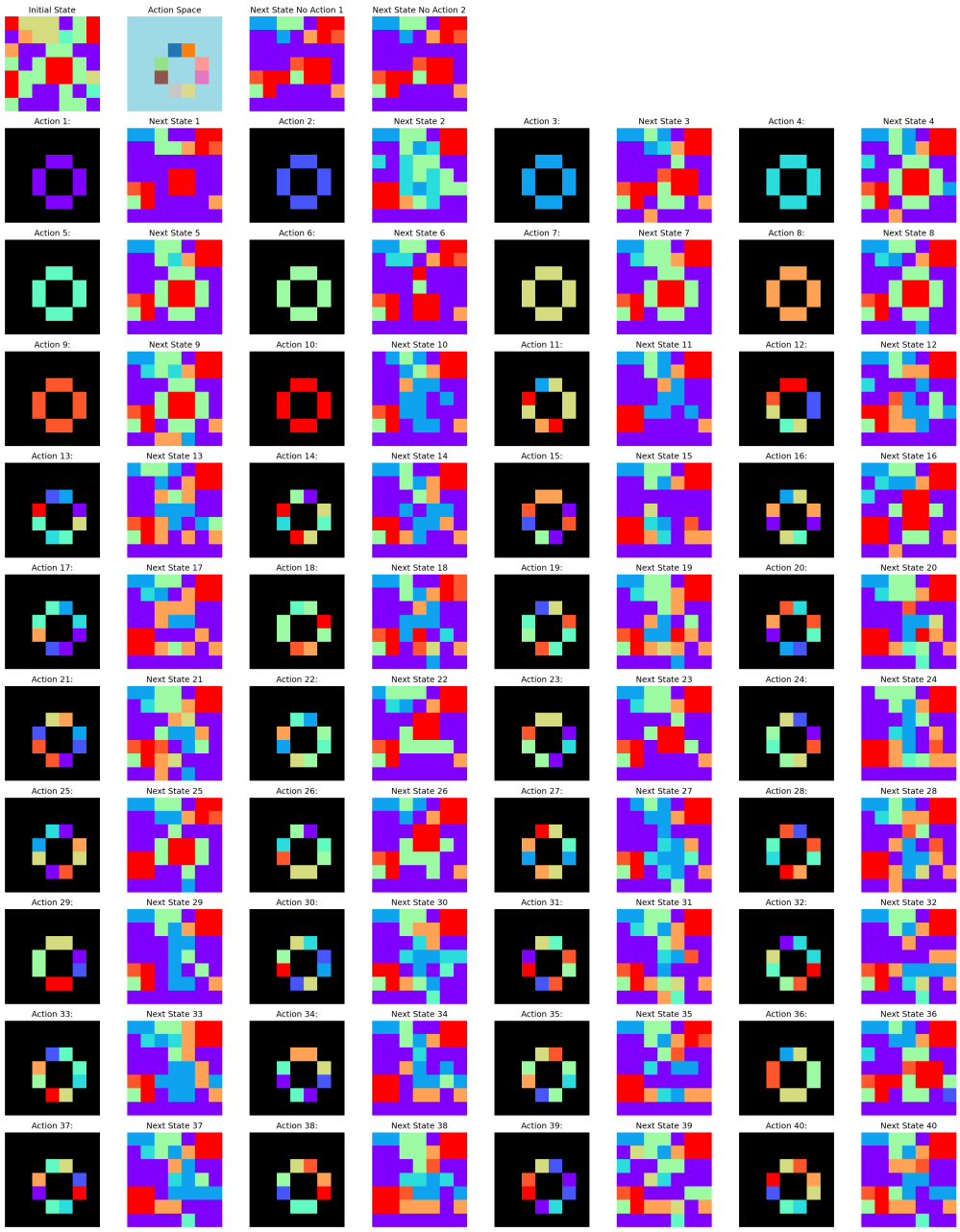

Figure 5.5: A snapshot of how a single agent with an action-space of its neighboring cells can affect its environment over one step

5.4.7 Training persistence

The final step towards being able to train environment-generated agents with environment-generated interactions on agentic tasks with some horizon is to give them persistence in the environment. Without persistence, credit assignment is impossible and the role of the agent is unclear.

It is the role of the interpretability interface of an environment to decide what persisting means. We run a test where we define persistence to be a 2by2 square staying within 1 of its position on the previous step, and train agents in this multi-agent environment on the persistence reward of 1 for every step they persist.

Blacking out the rest of the environment to emphasize the agents, 5.6 shows that we can get non-surviving agents to stay alive long enough in the environment in order to define other rewards on them to make use of this interpretable environment family.

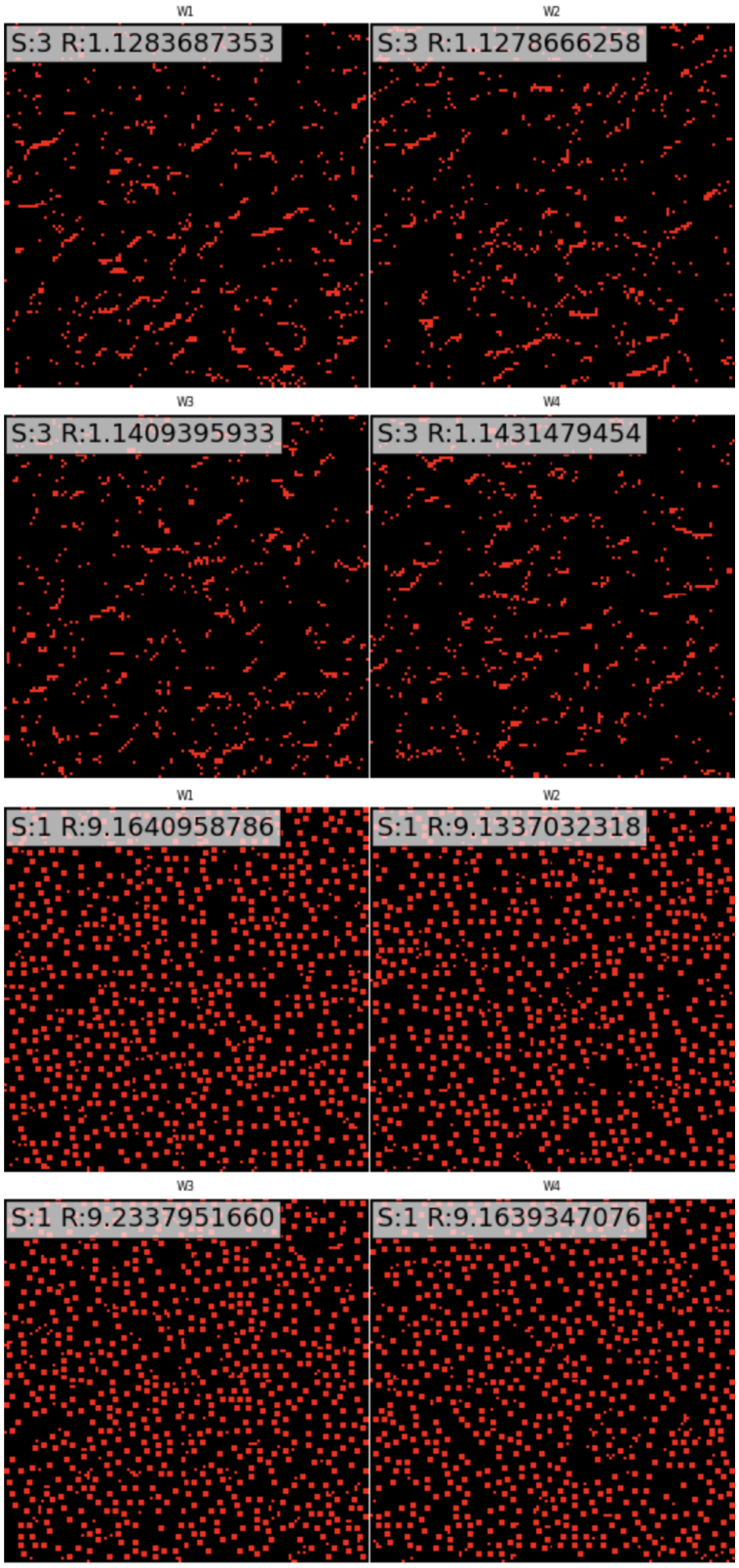

Figure 5.6: Two state snapshots from trajectories before (above) and after (below) training red 2x2 agents for persistence in an environment

Chapter 6

Conclusion

6.1 Limitations

Qualitative Experimental Results Most RL work enjoys grounding in a fixed environment family, with relatively agreed upon metrics such as score to measure the performance of the proposed algorithm. Due to the unstructured and very underdeveloped nature of the field of environment design, it's much less clear what kind of metrics are scientifically useful to report. Some concerns to using quantitative results are:

1. Whether the results are specific to an narrow environment family or can be interpreted more generally

2. Identifying the axis of the algorithm that can be used to create more general environments

With the development of environment design, these limitations may end up resolved because we will have created non-narrow environment families with appropriate practices to evaluate design on. This may additionally solve reproducbility issues deep RL has faced as well.

While this thesis can only contain static images, gifs of the environment rollouts will be made available in future work.

Lack of Clear Environment Generation Metrics. It is hard to be completely confident that we are approaching the environment design problem correctly empirically. Due to not having concrete evaluations for what means to generate a useful environment family, we need to resort to qualitative measures that are convincing to be worth pursuing to get closer to our goals. This can include foresight about the environment properties we should aim for, like multi-modality or flexibility, defining parts of the environment system concretely in order to be able to focus on and improve them, or definitions of intermediate metrics that are acknowledged to be faulty. For now, it feels like the by and far best direction to take is to generate more examples of interpretable environment families that would allow to better iterate on our baselines of how we can shape our environment design space.

6.2 Future Work

6.2.1 Environment Robustness to Agent Goal Optimization Distribution Shift

One important question for neural environments is how environment training changes how easily the state of the environment can be taken out of distribution of the train-time state manifold. One possible approach with this method is if you make an environment state very hard to change with actions, then it might be difficult to go out of distribution. This remains an important question for designing environments both from scratch neurally and from existing deep neural network world models.

6.2.2 Discovering directions that environments can scale in diversity in

With a flexible interpretable interpretable family, it can potentially scale the number of qualitatively different interactions it can hold by scaling any of:

1. Environment size
2. Number of defined structures
3. Number of shaped interactions
4. Number of rewards defined on structures
5. Number of high-variance RL runs

Determining which of these are most important is an interesting direction for future work.

6.2.3 Accelerating Specific Types of Reward Design

When training large multi-agent environments, reward computation can quickly grow to be expensive for more complicated shaping rewards. This work utilizes the speed of JIT-able local reward functions to shape the environment structures and interactions to generate. However, to maintain speed, reward functions need to be written in convoluted JAX-friendly code. Future work that can make it easier to design efficiently implemented versions of rewards intuitive to humans would significantly speed up the environment design cycle.

6.2.4 Training Constrained Models of the World

Another direction is to train generative models that are restricted to simulate physics to match reality. This can be training some cellular automata with a rendering function to project onto 2D video that matches our datasets. This is a hard exploration problem if we don't have an explicit way to to back-propogate directly through the rendering and we're forced to use zeroth order methods, but potentially a local diffusion transformer trained on

video could work. This type of world model is significantly harder to train than the existing generation, but comes with the advantage of having physics that can be re-appropriated to designing a significant amount of interactions in environments.

6.2.5 Training predictive world models

Training on predicting video data is likely suited as an objective for getting better high level action spaces for embodied agents. This would help with exploration in more general environment families, similar to how improved base models allow for exploration on math. Getting more clarity on whether video-trained predictive world models can serve as good environment on the other hand is tough due to the lack of evaluations for whether environments are useful outside of downstream tasks performance which is a very high variance metric. Nevertheless, the direction remains useful.

6.2.6 Determining if task complexity matters

It would be useful to get better clarity on what aspects of environments are important in getting transfer and whether the interactions hypothesis holds. Do we need sources for environments, or can we get away with directly designing environments with tasks in mind?

Bibliography

[1] Michael P. Allen and Dominic J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, Oxford, 2nd edition, 2017.

[2] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[3] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[4] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

[5] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. OpenAI Technical Report, 2024.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Volodymyr Mnih, Sergio Gonzalez, Demis Hassabis, et al. Genie: Generative interactive environments. *arXiv preprint arXiv:2402.15391*, 2024.

[8] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 4(3):e00015, 2019.

[9] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[10] François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

[11] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

[12] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[13] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, London, 1859.

[14] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

[15] Haoyi Duan, Hong-Xing Yu, Sirui Chen, Fei-Fei Li, and Jiajun Wu. Worldscore: A unified evaluation benchmark for world generation. *arXiv preprint arXiv:2504.00983*, 2025.

[16] Sam Earle and Julian Togelius. Autoverse: An evolvable game language for learning robust embodied agents. *arXiv preprint arXiv:2407.04221*, 2024.

[17] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[18] Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. Program synthesis. *Foundations and Trends in Programming Languages*, 4(1-2):1–119, 2017.

[19] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[20] Timm Hess, Martin Mundt, Iuliia Pliushch, and Visvanathan Ramesh. A procedural world generation framework for systematic evaluation of continual learning. *arXiv preprint arXiv:2106.02585*, 2021.

[21] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[22] Scott A. Hollingsworth and Ron O. Dror. Molecular dynamics simulation for all. *Neuron*, 99(6):1129–1143, 2018.

[23] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[24] Niels Justesen, Rubén R Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, 2(8):428–436, 2020.

[25] Martin Karplus and J. Andrew McCammon. Molecular dynamics simulations of biomolecules. *Nature Structural Biology*, 9(9):646–652, 2002.

[26] Stuart A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, New York, 1993.

[27] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

[28] Mike Knoop, François Chollet, Marcus Hutter, et al. Arc prize 2024: Technical report. *arXiv preprint arXiv:2412.04604*, 2024.

[29] Akarsh Kumar, Chris Lu, Louis Kirsch, Yujin Tang, Kenneth O. Stanley, Phillip Isola, and David Ha. Automating the search for artificial life with foundation models. *arXiv preprint arXiv:2412.17799*, 2024.

[30] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130–141, 1963.

[31] Ernst Mayr. Cause and effect in biology. *Science*, 134(3489):1501–1506, 1961.

[32] Thomas Miconi. Procedural generation of meta-reinforcement learning tasks. *arXiv preprint arXiv:2302.05583*, 2023.

[33] Matthias Müller-Brockhausen, Mike Preuss, and Aske Plaat. Procedural content generation: Better benchmarks for transfer reinforcement learning. *arXiv preprint arXiv:2105.14780*, 2021.

[34] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

[35] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

[36] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *International conference on machine learning*, pages 2778–2787, 2017.

[37] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *Transactions on Machine Learning Research*, 2022.

[38] 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.

[39] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[40] Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander W.R. Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[41] David E. Shaw, Paul Maragakis, Kresten Lindorff-Larsen, Stefano Piana, Ron O. Dror, Michael P. Eastwood, Joseph A. Bank, John M. Jumper, John K. Salmon, Yibing Shan, et al. Atomic-level characterization of the structural dynamics of proteins. *Science*, 330(6002):341–346, 2010.

[42] Armando Solar-Lezama. Program synthesis and semantic parsing with learned code idioms. *Advances in neural information processing systems*, 31, 2018.

[43] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Westview Press, Boulder, CO, 2nd edition, 2014.

[44] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

[45] Gemini Team. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[46] Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

[47] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017.

[48] 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.

[49] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023.

This bibliography was partially compiled with assistance from Deep Research.