Environment Generation for Autonomous Agents for Sequential Decision Making

By

Abdus Salam Azad

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ion Stoica, Chair
Professor Pieter Abbeel
Professor Sanjit Seshia
Professor Joshua Bloom

Summer 2024

Environment Generation for Autonomous Agents for Sequential Decision Making

Abstract

Environment Generation for Autonomous Agents for Sequential Decision Making

By

Abdus Salam Azad

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Ion Stoica, Chair

Autonomous agents have seen tremendous advancements in solving sequential decision-making problems in recent years, primarily driven by Reinforcement Learning (RL), and more recently, by Large Generative Models. The capability of these autonomous agents depends crucially on the quality and diversity of the learning environments they are trained in. This thesis presents research on designing frameworks and algorithms to formulate and systematically generate environments that improve the generalization capabilities of autonomous agents in solving sequential decision-making tasks. First, we explore the benefits of human-guided programmatic environment generation for training, testing, and debugging autonomous agents in complex real-time strategic (RTS) environments. We present a novel framework that, for the first time, demonstrates the benefits of using scenario specification languages (e.g., SCENIC) for systematic modeling and generation of realistic and diverse RTS RL environments (e.g., Soccer). Next, we discuss a class of algorithms called adaptive teacher Unsupervised Environment Design (UED), which automatically generates training tasks with an RL teacher agent. UED shows promising zero-shot generalization by simultaneously learning a task distribution (i.e., curriculum) and agent policies on the generated tasks. This is a non-stationary process where the task distribution evolves along with agent policies, creating instability over time. While prior works demonstrated the potential of such approaches, training the teacher remained a practical challenge. To this end, we introduce Curriculum Learning via Unsupervised Task Representation Learning (CLUTR): a novel unsupervised curriculum learning algorithm that decouples task representation and curriculum learning into a two-stage optimization to solve the training instability by pretraining a latent task manifold. Following that, we present MultiModal Reasoning and Critique for web navigation (MMRC), which introduces augmented environments with multimodal critic agents to enhance the performance of Large Foundational Multimodal Language agents on autonomous web navigation tasks. Together, these approaches portray the importance and usefulness of environment formulation and generation encompassing traditional RL-based and contemporary LLM-based agents.

To My Parents

Contents

List of Figures

List of Tables

Acknowledgments

I feel incredibly fortunate to have had Prof. Ion Stoica as my advisor. He has been an exceptional mentor and supporter, providing invaluable guidance while granting me an unparalleled level of autonomy to explore my research interests. I am deeply grateful for his kindness, support, and patience, especially during the significant challenges I faced throughout my Ph.D. When I decided to completely change my research direction two and a half years into the program, Ion's understanding and encouragement were instrumental in my progress.

I extend my sincerest appreciation to Prof. Pieter Abbeel, whose guidance, support, and boundless patience have been invaluable. I am also deeply thankful to Prof. Sanjit Seshia for his unwavering support, particularly during my transition to a new research field. Switching topics so late in the program was a difficult decision, but I was able to navigate these challenges and complete my Ph.D. thanks to the kindness and encouragement of these amazing mentors. I also wish to thank Prof. Joshua Bloom for his ongoing guidance and support since my qualifying exam. Working with such esteemed scholars and legends has been beyond anything I could have ever imagined.

I would like to express my heartfelt gratitude to Izzeddin Gur from Google DeepMind for his constant support and mentorship. I have learned so much from him and am incredibly grateful for the opportunity to collaborate with him throughout much of my Ph.D. I am also deeply thankful to Aleksandra Faust from Google DeepMind for her guidance. Additionally, I want to express my gratitude to my close collaborators Edward Kim and Kimin Lee. Special thanks go to Jasper Emhoff and Qiancheng Wu, the most dedicated undergraduate and Master's students I have had the pleasure of working with. I am also grateful to my other collaborators, Nathaniel Alexis, Hiroki Furuta, Jai Sharma, Kevin Zakka, Xingyu Lin, Abhik Bhattacharjee, and Tahmid Hasan, each of whom has significantly impacted my research.

During my Ph.D. at Berkeley, I had the privilege of forming friendships with exceptional individuals, without whom this journey would have been incomplete, difficult, and far less enjoyable. Tanveer Ahmed Siddique, thank you for all the late-night discussions about physics, the universe, and so much more. Vinamra Benara, your support during the toughest times was invaluable. Zunaid Omair, I feel privileged to have spent so much time with you: one of the smartest people I have ever met and an incredibly honest friend. Ahmad Us Saleheen, I'm deeply grateful for your incredible support during my dissertation talk—I will never forget your help. To the incredibly fun Berkeley and Bay Area folks: Abrar Amin Khan, Anika Sohaana, Kashfia Nehrin, Mahfuza Islam (special thanks for all the delicious meals), Md Ishfak Tahmid Mahin, Md Nazibul Islam, Monir Uz Zaman, Munif Ishad Mujib, Nazmul Ahasan (special thanks for the deep conversations that broadened my horizons), Tarannum Sarwat Sahar, Mohd. Elius, Urmita Sikder (for her intellectual speeches), Maruf Ahmed & Sifat Tanzim (for being a home away from home), Sheikh Waheed Baksh, Arefa Hossain (for the yummiest Bangladeshi patties), Suman Hossain (for his insightful conversations that expanded my perspective), Imran Khan, Rumi Karim, Yasser Khan, Sifat Sharmeen Muin, Arunoday Saha (for the late-night discussions about almost anything), Swarna Saha, Kamrul Hasan, Tania Islam, Konok Chapa Jui, Saikat Chakraborty, Anurag Roy (the only Bengali 'Dada' I know in the 2018 cohort), and Jemma Malia, thank you all. Lastly, I want to extend my gratitude to Rohan Bavishi, Rohan Padhye, Edward Kim, and Caroline Lemieux for their incredible guidance during my early years at Berkeley.

Special thanks go to Siddhartha Shankar Das and Probal Chandra Dhar, for being the most supportive friends—without your help, I might not have made it through. I am also deeply grateful to my undergraduate supervisor, Prof. Md. Monirul Islam, for laying the foundation of my research skills, convincing me to persevere with my Ph.D., and encouraging me to change research topics when I was on the verge of quitting. My gratitude also extends to Prof. Syed Ishtiaque Ahmed, Prof. Rifat Shahriyar, Rifat Ahsan, and Swarna Saha for their support and encouragement during that challenging time. I am thankful to my childhood friends Olive Hasan, Ariful Islam, and Rahat Islam, who have always been there for me. I would also like to express my appreciation to Shirley Salanio, Jean Nguyen, and Roxana Infante for their unwavering support throughout my Ph.D., especially in responding to every single email regarding official matters.

This Ph.D. thesis represents the culmination of my academic journey, which began in my undergraduate years at BUET. I owe a debt of gratitude to my family, particularly my mother, who has dedicated her life to our family, and my elder brother and sister, especially my nephews Kabbo and Ayaan. I wish my father were here to share in this moment. Finally, I want to thank my loved ones for their continuous support and encouragement throughout this journey. I apologize to those whose names I may have missed unintentionally.

Chapter 1

Introduction

1.1 Background and Motivation

Autonomous agents have made significant strides in addressing sequential decision-making problems in recent years, driven by advancements in Reinforcement Learning (RL) and, more recently, Large Generative Models, enabling them to solve increasingly complex and realistic challenges. The effectiveness of these agents relies crucially on the quality and diversity of the learning environments in which they are trained. This thesis focuses on designing frameworks and algorithms to formulate and generate environments that improve the generalization capabilities of autonomous agents in solving a wide variety of realistic and complex sequential decision-making tasks.

MDP and Environment

First, we discuss Markov Decision Process (MDP), a mathematical framework, which is typically used to formulate sequential decision-making tasks. MDPs are foundational in reinforcement learning (RL), where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. Similarly, MDPs also offer the mathematical framework for agents based on Large Generative Models. Formally an MDP is as the tuple (S, A, T, R, S_0), where each component has the following meaning:

- **State Space (S):** The set of all possible states in the environment.
- **Action Space (A):** The set of all possible actions an agent can execute on the environment.
- **Transition Dynamics (T):** The probability $T(s'|s, a)$ of transitioning from state s to state s' given action a. Here, a is an action executed in the environment.
- **Reward Function (R):** The reward function $R(s, a)$ that gives the reward received after taking action a in state s.
- **Initial State Distribution ($\mathbf{S_0}$):** The distribution over the initial states in the environment.

Often, the agents do not have direct access to the true state but instead receive observations that provide partial information about the state. Such environments are formulated as Partially Observable Markov Decision Processes (POMDP). A POMDP is represented as (S, A, T, R, S_0, O, Z) with the following additional components:

- **Observation Space (O):** The set of all possible observations that the agent can receive.
- **Observation Function (Z):** The probability $Z(o|s, a)$ of receiving observation o given state s and action a.

In this thesis, we define environment as a specific instance of an MDP. By generation of environment, we mean generating the distributions of each component of an MDP/POMDP (S, A, T, R, S_0, O, Z). We show how formulating and generating distributions of environments can improve the performance of autonomous agents.

1.2 Overview of Methods

The following subsections provide a brief overview of the subsequent chapters.

Scenic4RL: Programmatic Modeling and Generation of Real-Time Strategic Soccer Environments for Reinforcement Learning [3]

In Chapter 2, we introduce Scenic4RL, a novel approach that utilizes the formal scenario specification language, SCENIC [26, 27, 28], to programmatically generate diverse and realistic learning environments [77] for reinforcement learning (RL) agents in real-time strategy (RTS) settings [3]. Unlike traditional simulators that rely on randomly generated environments with limited flexibility, Scenic4RL allows researchers to systematically model and create complex environments. To demonstrate its effectiveness, we integrated Scenic4RL with the Google Research Football (GRF) simulator, releasing a benchmark of 32 realistic scenarios to train RL agents and evaluate their generalization capabilities. Additionally, we demonstrate how Scenic4RL enables researchers and practitioners to incorporate domain knowledge to expedite the training process or debug agents by modeling stochastic programmatic policies.

CLUTR: Curriculum Learning via Unsupervised Task Representation Learning [2]

While Chapter 2 explores how Scenic4RL enables human-guided systematic environment generation, in Chapter 3, we introduce CLUTR: a novel unsupervised environment design/curriculum learning (UED) algorithm that automatically generates environments to address the challenges of sample inefficiency and difficult generalization in Reinforcement Learning (RL). CLUTR operates in two stages: first, it trains a recurrent variational autoencoder on randomly generated tasks to learn a latent task manifold, and then a teacher agent creates a curriculum based on a minimax REGRET-based objective. This approach overcomes the instability commonly observed in training teachers in adaptive UED methods and enhances the stability of the training process. Our experiments show that CLUTR significantly outperforms the popular UED method, PAIRED, in both CarRacing and navigation environments, achieving substantial improvements in zero-shot generalization and sample efficiency.

MMRC: Multimodal Reasoning and Critique for Web Navigation

Chapters 2 and 3 explore systematically modeling and generating environments for traditional deep neural network-based agents trained with reinforcement learning (RL). In Chapter 4, we introduce MMRC, which investigates how environment generation can enhance the performance of recent foundational Large Language Models (LLMs) by leveraging a novel multimodal actor-critic framework. MMRC addresses the challenges faced by LLMs in web navigation tasks, particularly the difficulties in processing long HTML observations and grounding model predictions from visual inputs into actionable events. By presenting webpage elements as multiple-choice questions, MMRC facilitates grounding and tackles hallucinations. Experiments on the Mind2Web dataset demonstrate that the proposed actor-critic formulation of MMRC outperforms actor-only baselines and surpasses the previous best results obtained by Gemini 1.0 Pro Vision. Additionally, MMRC's use of a large foundational model as the actor, paired with a smaller fine-tuned critic, highlights a promising approach for enhancing the performance of general-purpose models in specific tasks.

Together, Scenic4RL, CLUTR, and MMRC introduce innovative approaches that demonstrate how environment formulation and generation can significantly enhance the design and performance of autonomous agents across a diverse range of realistic and challenging sequential decision-making problems. Scenic4RL enables systematic, human-guided environment generation for traditional deep neural network-based RL agents, allowing for the creation of complex and varied training environments. CLUTR addresses sample inefficiency and generalization challenges by automatically generating environments through an unsupervised curriculum learning algorithm, improving the stability and effectiveness of RL training. MMRC extends these concepts to the realm of Large Language Models (LLMs), offering a multimodal actor-critic framework that improves web navigation by effectively grounding model predictions in actionable events. These methods collectively provide robust solutions for enhancing both traditional RL agents and cutting-edge LLMs, showcasing the critical role of environment generation in advancing autonomous decision-making systems. We will discuss some of our learning and conclude in Chapter 5.

Chapter 2

Scenic4RL: Programmatic modeling and generation of real-time strategic soccer environments

The capability of a reinforcement learning (RL) agent heavily depends on the diversity of the learning scenarios generated by the environment. Generation of diverse realistic scenarios is challenging for real-time strategy (RTS) environments. The RTS environments are characterized by intelligent entities/non-RL agents cooperating and competing with the RL agents with large state and action spaces over a long period of time, resulting in an infinite space of feasible, but not necessarily realistic, scenarios involving complex interaction among different RL and non-RL agents. Yet, most of the existing simulators rely on randomly generating the environments based on predefined settings/layouts and offer limited flexibility and control over the environment dynamics for researchers to generate diverse, realistic scenarios as per their demand. To address this issue, for the first time, we formally introduce the benefits of adopting an existing formal scenario specification language, SCENIC, to assist researchers to *model* and *generate* diverse scenarios in an RTS environment in a flexible, systematic, and programmatic manner. To showcase the benefits, we interfaced SCENIC to an existing RTS environment Google Research Football (GRF) simulator and introduced a benchmark consisting of 32 realistic scenarios, encoded in SCENIC, to train RL agents and testing their generalization capabilities. We also show how researchers/RL practitioners can incorporate their domain knowledge to expedite the training process by intuitively modeling stochastic programmatic policies with SCENIC.

2.1 Introduction

Deep reinforcement learning (RL) has emerged as a powerful method to solve a variety of sequential decision-making problems, including board games [84, 82], video games [61, 92], and robotic manipulation [47]. These successes rely heavily on widely-used simulation environments [4, 8] and benchmarks [13, 21, 89]. However, regardless of a long history of RL benchmarks, the existing

RL environments/simulators are insufficient to properly train, test, and benchmark RL algorithms for real-time strategy (RTS) environments such as Starcraft [93], Dota2 [5], and soccer [54], due to their lack of support for modeling diverse scenarios involving sophisticated interactive behaviors.

These RTS environments are characterized by unique characteristics that require special support for modeling. The environments involve intelligent entities/non-RL agents co-operating and competing with the RL agents with large state and action spaces over a long horizon. This opens up extremely diverse strategies consisting of numerous interactive behaviors. Yet, most of the existing simulators rely on randomly generating the environments based on predefined settings/layouts and offer limited flexibility and control to the researchers over the environment dynamics to generate diverse realistic scenarios. As a result, RL research faces at least two fundamental challenges: (i) the lack of diverse and realistic training data often leads to lack of generalization [14, 13, 56, 55], and (ii) the lack of flexibility and control over the environment dynamics makes it hard to generate realistic evaluation scenarios to comprehensively test generalization in these complex RTS environments.

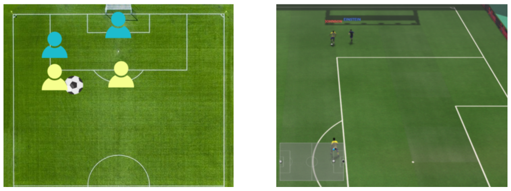
To address this issue, for the first time to the best of our knowledge, we introduce the benefits of adopting an existing formal scenario specification language, SCENIC, to assist researchers to *model* and *generate* diverse realistic scenarios in an RTS environment in a flexible, systematic, and programmatic manner [28, 27, 26, 77]. Each SCENIC program represents a Markov Decision Process (MDP) and provides high-level syntax and semantics, backed by its own compiler, to intuitively and quickly model diverse and complex interactive scenarios to train RL agents and test their generalization capabilities. Furthermore, it allows researchers/RL practitioners to incorporate their domain knowledge into the training process by generating offline data with stochastic programmatic policies written in high-level intuitive syntax of SCENIC. To demonstrate the benefits, we interfaced SCENIC to an existing RTS environment, Google Research Football (GRF) [54].

Our contributions are as follows:

- For the first time, we introduce the benefits of adopting a scenario specification language to

  (1) flexibly model interactive scenarios to train RL agents,
  (2) test their generalization capability, and
  (3) program stochastic RL policies to generate demonstration data.

- We open-sourced our SCENIC's interface to GRF environment along with our 32 scenarios, 5 stochastic policies, and libraries encoded in SCENIC to assist researchers to build upon them to easily model diverse and sophisticated scenarios.



(a) a bird-eye view of the scenario

(b) a snapshot of GRF environment

```
builder.SetBallPosition(0.7, -0.28)
builder.SetTeam(Team.e_Left)
builder.AddPlayer(-1.0, 0.0, e_PlayerRole_GK)
builder.AddPlayer(0.7, 0.0, e_PlayerRole_CB)
builder.AddPlayer(0.7, -0.3, e_PlayerRole_CB)
builder.SetTeam(Team.e_Right)
builder.AddPlayer(-1.0, 0.0, e_PlayerRole_GK)
builder.AddPlayer(-0.75, 0.1, e_PlayerRole_CB)
```

(c) GRF's scenario program

```
behavior egoBehavior(player, destinationPoint):
        do Uniform(ShortPassTo(player), dribbleToAndShoot(destinationPoint))

behavior attackMidBehavior()
  try:
    do HoldPosition()
  interrupt when self.owns_ball is True:
    do AimGoalCornerAndShoot()
  interrupt when (distance to nearestOpponentPlayer(self)) < 5:
    do dribble_evasive_zigzag()

LeftGK
attack_midfielder = LeftAM on left_penalty_arc_region,
                facing north,
                with behavior attackMidBehavior()
ego = LeftLM ahead of attack_midfielder by Range(5, 10),
    facing toward attack_midfielder,
    with behavior egoBehavior(attack_midfielder, left_penaltyBox_center)

Ball ahead of ego by 0.1
RightRB left of ego by Range(3,5)
RightGK
```

(d) SCENIC program of generalized pass-and-shoot scenario with distribution over players' initial condition and behaviors

Figure 2.1: Programs encoding the Google Research Football's (GRF) pass-and-shoot scenario

2.2 Related Work

Environment Generation in RL

In literature, several techniques have been adopted to generate a rich variation of learning scenarios, primarily to promote, or ensure generalization. Techniques such as changing background with natural videos [101] and introducing sticky actions [57] have been attempted, but are not robust enough. To ensure generalization, [55] and [76] generated training and testing scenarios by randomly sampling from different regions of parameter space. Similar to supervised learning, the use of separate train and test sets have also been adopted [67, 13, 14, 46], typically using techniques such as Procedural Content Generation [37], which has traditionally been used to automatically generate levels in video games. However, most of these focus on discrete domain, typically the dataset generation process is opaque, and it can be difficult to quantify, or reason about how different (or similar) these train and test sets are, since the generation process often uses random numbers to generate different configurations.

On the contrary, a few manually scripted scenario benchmarks are proposed with respect to a few RTS RL environments with limitations. For StarCraft [93], only two benchmark scenarios [91, 74] have been proposed. Both of these scenarios model different initial states but leave the behavior generation to either a learned RL agent or AI bots that are provided by the StarCraft environment, which are considered as blackbox agents. As a result, a sophisticated modeling and control over the behaviors of non-RL agents to create specific types of scenarios is not possible, severely restricting the diversity of the scenarios. For soccer domain, [86] presented one benchmark scenario on keepaway tactical scenario and later extended to more general half-field offense scenario [35]. They provided a library of APIs relating to behaviors (e.g. mark player, defend goal) of players, which helps users to model scenarios. However, SCENIC provides further benefits that are not covered in this work. SCENIC provides high-level syntax and semantics to (i) easily write spatial relations for intutively modeling initial states, (ii) assign distributions over both initial states and behaviors to generate variations of environments for robust training and testing generalization, and (iii) specify priorities over interaction conditions over behaviors to model more sophisticated types of higher level behavior (for more detail, refer to Section *Modeling Scenarios with* SCENIC).

Formal Scenario Specification Languages for Environment Modeling and Generation

A few scenario specification languages have been proposed in the autonomous driving domain including SCENIC. Paracosm language [58] models dynamic scenarios with reactive and synchronous model of computation. The Measurable Scenario Description Language (M-SDL) [25] shares common features as SCENIC to model interactive scenarios. In contrast, however, SCENIC provides a much higher-level, probabilistic, declarative way of modeling. Furthermore, unlike other scenario specification languages, SCENIC has demonstrated its generality over different domains such as autonomous driving, robotics, and aviation [27, 26, 28, 77]. For these reasons, we chose SCENIC in this paper for demonstration of benefits that a scenario specification language can provide to RL.

2.3 Background

Google Research Football Simulator

The Google Research Football (GRF) simulator [54] provides a realistic soccer environment to train and test RL agents. The setting, the rules, and the objective of the environment are the same as defined by Fédération Internationale de Football Association [23]. The environment setup is as the following. All the players on the field are controlled by (1) GRF's built-in, rule-based AI bots and (2) RL agents. The simulator dynamically determines which of the RL team players are to be controlled by RL agents based on their vicinity to the ball. GRF provides 11 offense scenarios to train and test RL agent performance and it provides trained RL agent checkpoints for a subset of its scenarios.

Scenario Specification Language: SCENIC

SCENIC [26, 27] is an object-oriented, probabilistic programming language whose syntax and semantics are designed to intuitively *model* and *generate* scenarios. A SCENIC program represents an abstract scenario, which models a *distribution* over initial states and behaviors of players in the scenario. For each scenario generation, an initial state is sampled from the program at the beginning of a simulation and interactive behaviors are sampled during simulation runtime. Therefore, with a single SCENIC program, users can generate a distribution of concrete scenarios.

SCENIC requires action and model libraries, which are imported and compiled with a user's SCENIC program for execution. The action library defines the action space which is determined by the simulator. The model library defines objects and their attributes (e.g. position, heading). We can assign prior distributions over these attributes. For example, a goalkeeper's position can be uniformly randomly distributed over the penalty box region. If a user simply instantiates a goalkeeper in a SCENIC program but does not specify any condition over its attributes, then they are sampled from the prior distributions by default. These prior distributions can be overwritten in the user's SCENIC program.

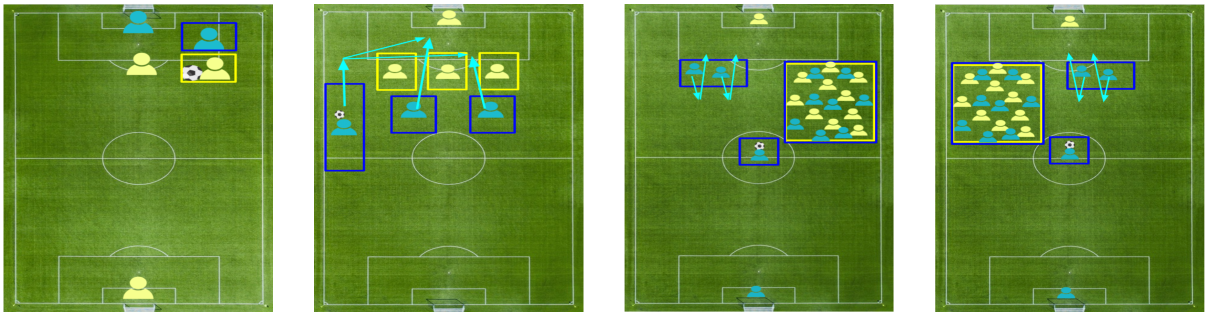
2.4 Scenario Specification Language for RL

Benefits of Scenario Specification Language for RL

The objective of this paper is to introduce the benefits of the use of scenario specification language for modeling and generating scenarios, specifically for RTS environments for RL. Using a scenario specification language whose syntax and semantics are carefully designed to intuitively model scenarios have the following benefits:

1. **Easily Model Interactive Environments on *User-demand* to Train and Test RL Agents:** The intuitive syntax and semantics, which abstracts away the implementation details and allows users to reason solely at high-level semantics, makes it easy to model complex spatial relations among multiple agents, their behaviors and conditions on how these behaviors should interact. It should be noted that, it requires a considerable amount of research and engineering effort to design and implement a formal scenario modeling language and its compiler from scratch.

2. **Program Stochastic Policies:** These programmed agents can serve two purposes: (i) allow developers to incorporate domain knowledge, e.g., generate demonstration data for offline training and (ii) provide performance baseline for trained RL agents.



(a) generalization test scenario for the scenario in Fig. 2.1a

(b) 3 vs 3 left mid-fielder crosses to either player in penalty box

(c) 11 vs 11 open player scenario

(d) mirrored Fig. 2.2c scenario

Figure 2.2: Examples of a new defense scenarios with specific assigned behaviors (a), a test scenario to assess generalization (b), and two full game scenarios (c,d) we used for training and testing. The RL team is yellow and the opponent, blue. The assigned opponent behaviors are highlighted with light blue arrows. Uniformly random distribution is assigned over a specific region for each player. These regions are highlighted boxes.

3. **Interpretability and Transparency**: The intuitive syntax and semantics make scenario programs interpretable and transparent. Therefore, users can reason about the difference/similarity of train and test environments by comparing their scenario programs.

4. **Reusability of Existing Scenarios**: The interpretability of scenario programs facilitates easy modification or re-use of existing Scenic programs, models, and behaviors to quickly model new scenarios. This facilitates building a community around designing and sharing scenario programs, by building upon each other's scenarios.

Modeling Scenarios with Scenic

Formally, a scenario is a Markov Decision Process (MDPs) [87] defined as a tuple $(\mathcal{S}, \mathcal{A}, p, r, \rho_0)$, with $\mathcal{S}$ denoting the state space, $\mathcal{A}$ the action space, $p\left(s'|s, a\right)$ the transition dynamic, $r\left(s, a\right)$ the reward function, and ρ_0 the initial state distribution. Given the state and action spaces as defined by the GRF environment, a Scenic program defines (i) the initial state distribution, (ii) the transition dynamics (specifically players' behaviors), and (iii) the reward function. Hence, users can exercise extensive control over the environment with Scenic.

Modeling Initial State Distribution Users can intuitively specify initial state distributions with Scenic's high-level syntax that resembles natural English. For example, refer to the full Scenic program in Fig. 2.1(d) which describes a more generalized version of GRF's Pass and Shoot scenario as visualized in Fig. 2.1(a,b). In line 12-22, the initial state distribution is specified. The Scenic syntax for modeling spatial relations among players are highlighted in yellow.

```
behavior receiveCrossAndShoot(destinationPoint):
    do MoveToPosition(destinationPoint)
    do HoldPosition() until self.owns_ball
    do dribbleToAndShoot(yellowPenaltyBox_centerPoint)
    do HoldPosition()

behavior crossToPlayers(list_of_players):
    destinationPoint = Point on region_in_penaltyBox
    do MoveToPosition(destinationPoint)
    do HighPassTo(Uniform(*list_of_players))
    do HoldPosition()
```

Figure 2.3: A snippet of a SCENIC program specifying behaviors for players Fig. 2.2b

In addition, SCENIC supports about 20 different syntax to support modeling complex spatial relations [27]. Rather than having to hand-code positions for a concrete scenario as in the GRF's scenario 2.1(c), users can much more intuitively and concisely model a distribution of initial states. Here, `Left` represents the yellow team, `Right` the blue, and the two following abbreviated capital letters indicate the player role.

Modeling Transition Dynamics One can flexibly modify transition dynamics of the environment by specifying the behaviors of non-RL players using SCENIC. Take the same example SCENIC program in Fig. 2.1(d) as above. Line 1-10 models two new behaviors. A behavior can invoke another behavior(s) with syntax `do`, succinctly modeling a behavior in a hierarchical manner. Users can assign distribution over behaviors as in line 2. The interactive conditions are specified using try/interrupt block as in line 5-10. Semantically, the behavior specified in the try block is executed by default. However, if any interrupt condition is satisfied, then the default behavior is paused and the behavior in the interrupt block is executed until completion and then the default behavior resumes. These interrupts can be nested with interrupt below has higher priority. In such case, the same semantics is consistently applied.

Rewards SCENIC has a construct called `monitor`, which can be used to specify reward functions. The reward conditions in the `monitor` is checked at every simulation step and updates the reward accordingly.

Termination Conditions Users can also specify termination conditions which are monitored at every simulation time step.

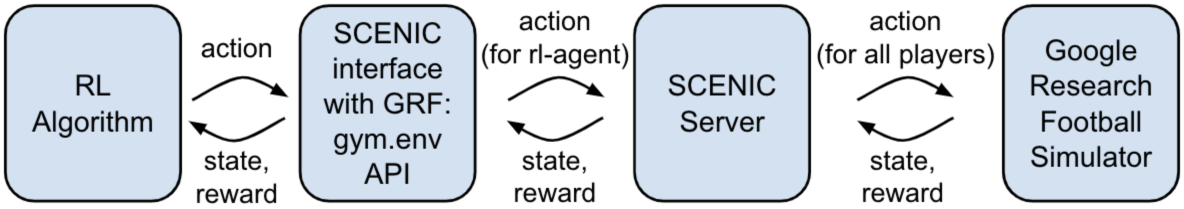


Figure 2.4: Interface Architecture between SCENIC and GRF

On Interfacing SCENIC to a Simulator

Interfacing SCENIC to other simulators is straight-forward. In fact, SCENIC is already interfaced with five other simulators [17] in domains such as autonomous driving, aviation, and robotics. To interface SCENIC with a simulator, one needs define the model, action, and behavior libraries. These libraries expedites modeling complex scenarios by helping users re-use the set of models, actions, and behaviors in the libraries, rather than having to write a scenario from scratch.

The *model* library defines the state space. It defines players with distribution over their initial state according to their roles and GRF's AI bot is assigned by default to all player behavior. These prior distribution over the initial state and behavior can be overwritten in the SCENIC program. The model library also defines region objects such as goal and penalty box regions as well as directional objects in compass directions. The *action* library defines the action space as determined by the GRF simulator. These action space consists of movement actions in eight compass directions, long/short/high pass, shoot, slide, dribble, and sprint.

The *behavior* library consists of behaviors and helper functions that represent widely used basic skills in soccer. These behaviors include give-and-go, evasive zigzag dribble to avoid an opponent's ball interception, dribbling to a designated point and shooting, shooting towards the left or right corner of the goal, etc. Additionally, the behavior library also include useful helper functions such as identifying nearest opponent or teammate, whether there is an opponent near the running direction of a dribbler, etc. Please refer to our open-sourced repository for more details.

Interface Architecture

Figure 2.4 shows an overview of our overall architecture. The architecture can be divided into two parts: i) RL interface, through which the RL algorithms interact with SCENIC and ii) the SCENIC Server, which executes a SCENIC program and governs the simulation by interacting with the underlying simulator. We follow the widely used OpenAI Gym API [8] as our interface, which allows our interface to be used seamlessly with all the existing standard RL frameworks.

For each simulation/episode, the SCENIC server first samples an initial state from the SCENIC program to start a new scenario in the GRF simulator and updates its internal model of the world (e.g., player and ball positions). From then on, a round of communication occur between the RL algorithm and SCENIC server, with the RL interface at the middle. At each timestep, the gym

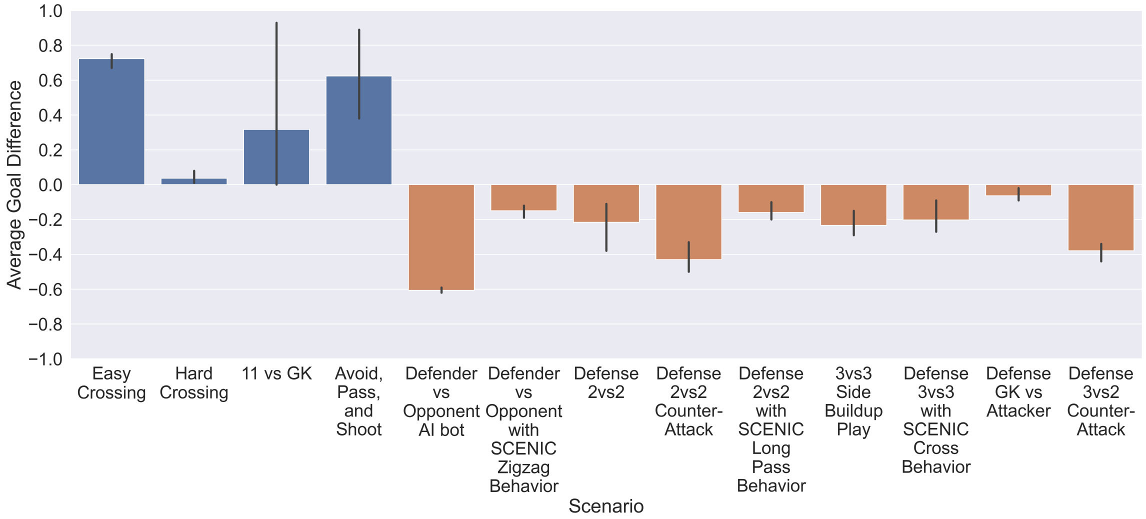


Figure 2.5: Average Goal Difference of PPO agents on the proposed mini-game scenario benchmark. The error bars represent 95% bootstrapped confidence intervals

interface takes in the action(s) for the RL agent and passes them to the SCENIC server. The SCENIC server in turn computes actions for all the remaining non-RL players—the players not controlled by the RL agent—and then executes all these actions (of both the RL and non-RL players) in the simulator. The SCENIC server then receives the observation and reward from the simulator, updates the internal world state, and then passes them back to the RL algorithm. This interaction goes on till any terminating conditions as specified in the scenario script is satisfied.

2.5 Evaluation

In this section, we demonstrate four use cases of SCENIC in RL. First, we present and benchmark a set of 13 realistic mini-game scenarios encoded in SCENIC with a varying level of difficulty. Second, we test the generalization capabilities of the trained RL agents on unseen, yet intuitively similar scenarios. Next, we show how developers can “debug" their agents for failure scenarios of interest. At last, we show how probabilistic SCENIC policies can be used to generate offline data and endow domain knowledge into the learning process for faster training, which we believe to be very important for applying RL in practice.

Experimental Setup

We run PPO [75] on a single GPU machine (NVIDIA T4) with 16 parallel workers on Amazon AWS. Unless otherwise specified, all the PPO training are run for 5M timesteps and repeated for 10 different seeds. All the evaluation has been done for 10000 timesteps. For all the experiments, we use the stacked Super Mini Map representation for observations —a 4x72x96 binary matrix representing positions of players from both team, the ball, and the active player—and the scores as rewards, i.e., $+1$ when scoring a goal and -1 upon conceding, from [54]. Similar to the academy scenarios from [54], we also terminate a game when one of the following happens: either of the team scores, ball goes out of the field, or, the ball possession changes. For further details, including hyperparameters and network architecture, we refer readers to the Supplementary Materials (Section *Details on Experimental Setup and Training*).

Mini-game Scenario Benchmark

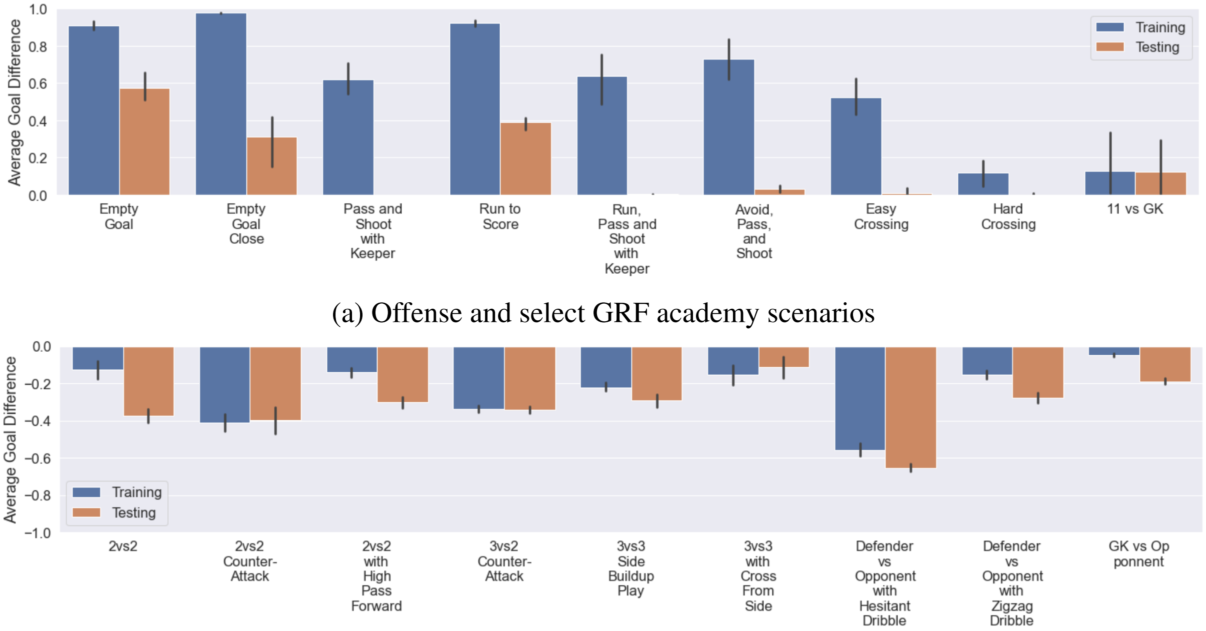
Training an RL agent to solve a full soccer game involving 22 players is very challenging and may take days even with distributed algorithms. For example, [54] showed even the easy version of GRF's 11 vs 11 game cannot be solved with 50M samples. To allow researchers to iterate their ideas with a reasonable amount of time and compute, we present a set of 13 mini-game scenarios. All these scenarios are inspired from common situations occurring in real soccer games but involves fewer number of players to make them amenable to be faster training.

Nine of our proposed mini-game scenarios are defense scenarios, which are nice complement to GRF's offense-only scenarios (refer to Sec. 2.3), along with four new offense scenarios. Most of these scenarios are initialized from a distribution, rather than fixed locations. By default all the opponent players are controlled by GRF's built-in AI bot (refer to Sec. 2.3). However, for the scenarios where the AI bot does not exhibit our desired behavior, we model the opponent behaviors using SCENIC. For example, in the 3vs3 cross scenario as shown in Fig. 2.2b, the opponent AI bots tried to pass the ball around instead of crossing. Therefore, we modelled and assigned behaviors such that the blue player on the leftmost side of the field would run up the field and cross the ball. Meanwhile, the two blue players in the center run into the penalty box area to receive the cross and shoot. These modelled behaviors are shown in Fig. 2.3.

We benchmark our mini-game scenarios by training agents with PPO. Figure 2.5 shows the average goal differences for all the scenarios. For these mini-game scenarios, we end the game if one of the teams score. Hence, the goal difference can range between -1 to +1. For the offense scenarios, a well trained agent is supposed to score consistently achieving an average goal difference close to +1. On the other hand, a well-trained agent should achieve a goal difference close to 0 for successfully defending the opponents in the defense scenarios. From the graph it can be seen that the proposed scenarios offer a varied levels of difficulties. For example, PPO consistently achieves goal difference of around 0.5 for the EASY CROSSING scenario, but barely learns anything for HARD CROSSING. In case of the defense scenarios, the results also show a varied range of difficulty, GK VS OPPONENT scenario being be easiest.

Testing for Generalization

We provide scripts to test generalization of all of our 13 new benchmark scenarios along with 5 scenarios provided by GRF. We changed the distribution over the initial state while keeping the formation of players and their behaviors in each scenario intact. For example, for testing generalization of an RL agent trained in the Pass and Shoot scenario (Fig. 2.1a), we instantiated the yellow and the blue players on the symmetric right side of the field instead of the left and kept the other initial state distribution the same (Figure 2.2a).



(a) Offense and select GRF academy scenarios

(b) Defense scenarios

Figure 2.6: Evaluation of PPO agents' generalization against varying initial conditions. For most of the academy and offense scenarios we observe a significant drop in performance. However, for several defense scenarios the difference in train and test scenarios is not that significant.

Fig. 2.6 compares the trained agents' performance in training and test scenarios. As expected, we observe a noticeable drop of performance in most of the GRF's academy and offense scenarios (Fig. 2.6a). For example, the Pass and Shoot scenario (Figure 2.1a), which achieved around 0.6 in training, failed to generalize for the test scenario. However, for the defense scenarios, the drop in performance was not as noticeable. We conjecture that this distinction comes from the differences in the offense and defense training scenarios, where the defense scenarios tend to contain larger distribution over the initial state than those of the offense scenarios (refer to Supplement). Consequently, larger variations of scenarios introduced during training may have contributed to better generalization for defense scenarios.

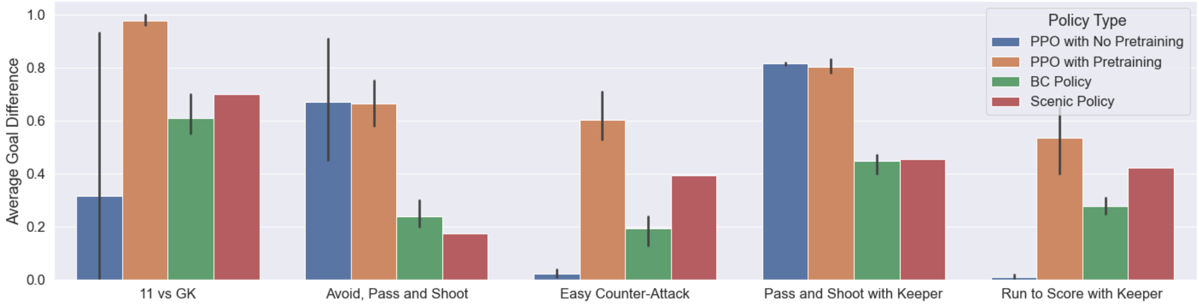


Figure 2.7: Performance of PPO agents trained with and without any demonstration data, along with the performance of corresponding behavior-cloned and SCENIC policies. We see significantly better performance on three of the scenarios, while the rest two achieves comparable performance, highlighting the usefulness of the proposed SCENIC policies.

Debugging Agents on 11v11 Failure Scenario

For this experiment, we evaluate and debug an RL checkpoint provided by GRF, which was trained on their 11 vs 11 easy stochastic scenario, i.e., easy version of their full-game scenario. This agent achieves an impressive average goal difference of 6.99 per full-game[1], scoring up to 14 goals in the training scenario during our experiments. We modelled a scenario, as visualized in Figure 2.2c, to test the agent's ability to quickly perceive open teammates near the opponent goal to advance the ball forward and score—a crucial skill for soccer. When we assigned GRF's built-in AI bots to control the open players on the left side of the field, the players ran straight toward the ball, instead of taking advantage of the closeness to the opponent goal without being marked. Hence, we modelled a behavior for open players in SCENIC so that they would stay close to the goal while abiding by the offside rule.

Although obvious to humans, the trained checkpoint performs poorly in this scenario with an average goal difference of 0.1. To 'debug' the agent, we then fine-tune the agent on a 'mirrored' scenario, as shown in Figure 2.2d, with PPO for 5M timesteps. The fine-tuned agent improved noticeably on the original scenario, achieving an average goal difference of 0.67. This showcases the usefulness of SCENIC to easily model and generate scenarios of interest using one's domain knowledge, which may have been difficult with blackbox agents (e.g. built-in AI bots, or trained RL agents), to test and debug certain capabilities of an RL agent.

Facilitating Training with Probabilistic SCENIC Policies

In the section, we show how RL practitioners can incorporate their domain knowledge by writing probabilistic SCENIC policies for faster training. We wrote simple semi-expert RL policies for five different scenarios, where the agent suffers to learn, and generated 8K samples of demonstration data for per scenario. To facilitate training on those scenarios, we first pre-train an agent via

[1] Evaluated on 100K timesteps

behavior cloning with the generated offline data and then fine-tune the agent using PPO for 5M timesteps. All the experiments were repeated for three different seeds. Figure 2.7 compares the training performance of these agents against the agents that were trained with PPO only. We notice that, even with such a low volume of demonstration data, we can train much better agents and can solve scenarios which were otherwise unsolved. The experimental results thus suggests, with stochastic SCENIC policies we can generate rich quality demonstration data to substantially enhance training performance, which can be particularly useful in practice for environments like GRF which requires a heavy compute resource.

2.6 Description of Proposed Scenarios and Policies

In this section we provide brief descriptions of all of the scenarios in our dataset. To see our SCENIC programs, please refer to our attached README pdf file for the pathways to our scenarios.

On Mini and Full Game Scenarios

In general, all our scenarios have the following three termination conditions: (i) ball goes off the field, (ii) change in ball possession across teams, (iii) one of the team scores. If any of these conditions are satisfied, then the scenario will terminate in simulation.

Offense Scenarios

In all of our six offense scenarios as shown in Fig. 2.8 and 2.9, we explicitly modelled the initial state distribution in using SCENIC and implicitly specified the behaviors to environment players by assigning the rule-based AI bots provided by Google Research Football (GRF) to control all non-RL players.

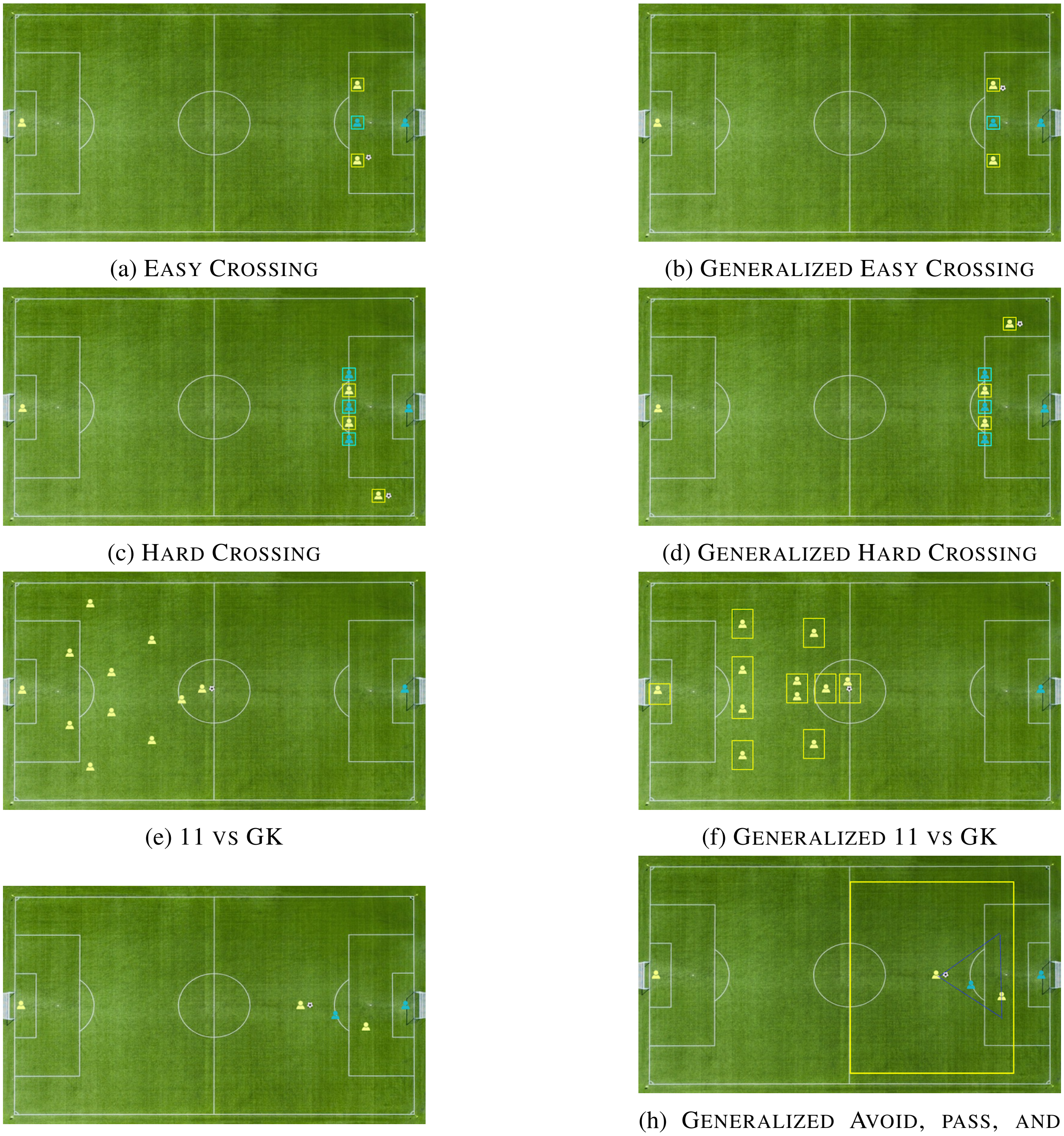
Hard Crossing: A very common scenario in real soccer games: 2 of our players along are guarded by 3 of the opponent players, in an interleaved manner, along the line of the penalty box. Another of our player at the edge of the field is attempting a cross.

11 vs GK: Our team, with a full lineup of eleven players in a traditional 4-4-2 formation, needs to score against the opponent goalkeeper.

Avoid, Pass, and Shoot: Two of our players, one starting on the middle of the right half and the other inside the penalty box, tries to score. One opponent defender starts between our players to intercept direct pass.

Easy Crossing: An easy crossing scenario involving two of our players against opponent defender and goalkeeper in the penalty box.

11 vs 11 with Open Players: A full game scenario where there are two unmarked players near the opponent goal. This is to test how wide "vision" an RL agent has in identifying unmarked players near the oppnent goal.



(a) Easy Crossing

(b) Generalized Easy Crossing

(c) Hard Crossing

(d) Generalized Hard Crossing

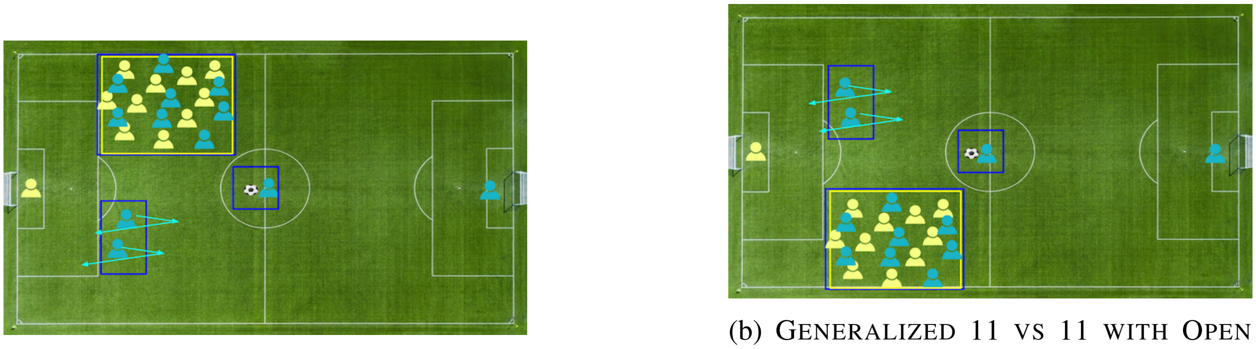
(e) 11 vs GK

(f) Generalized 11 vs GK

(g) Avoid, pass, and Shoot

(h) Generalized Avoid, pass, and Shoot

Figure 2.8: New offense benchmark scenarios (left images) and corresponding generalized test scenarios (right images) in our dataset. The highlighted boxes represent the regions over which players' initial positions are uniformly randomly distributed. The opponent is in blue and the RL team in yellow.



(a) 11 VS 11 WITH OPEN PLAYERS

(b) GENERALIZED 11 VS 11 WITH OPEN PLAYERS

Figure 2.9: New offense benchmark scenarios (left images) and corresponding generalized test scenarios (right images) in our dataset. The highlighted boxes represent the regions over which players' initial positions are uniformly randomly distributed. The opponent is in blue and the RL team in yellow.

Defense Scenarios

Like the offense scenarios, we assigned rule-based AI bots provided by GRF by default to control non-RL players in many of our defense scenarios as shown in Fig. 2.10, 2.11, 2.12. However, if the AI bots do not exhibit expected behavior for our modelled scenarios, we specified non-RL players' behaviors in SCENIC. For these scenarios with specified behaviors in SCENIC, their behaviors are highlighted with light blue arrows.

Goalkeeper vs Opponent: This scenario is designed to train an RL agent to be a defensive goalkeeper when it has to face an opponent one-on-one.

Defender vs Opponent with Hesitant Dribble: The opponent dribbles, stop, then dribbles again in a repeated manner.

Defender vs Opponent with Zigzag Dribble: This opponent aggressively evades the defender with zigzag dribble towards the goal and shoots.

2 vs 2: Typical, 2 vs 2 setting where two defenders are already in place to fend off the two opponents near the penalty area with the ball.

2 vs 2 Counterattack: An opponent attacking midfielder is already advanced deep into the left side of the field. The opponent right midfielder behind either short passes the ball to the attacking midfielder or dribbles up the field.

2 vs 2 High Pass Forward: An opponent attacking midfielder is already advanced deep into the left side of the field. The opponent right midfielder quickly advances the ball to the attacking midfielder via high pass.

3 vs 2 Counterattack: The defender near the penalty box is temporarily outnumbered by the opponent players due to a sudden counterattack.

3 vs 3 Cross from Side: The opponent player on the side crosses the ball to either of the teammates in the middle who are running towards the penalty box to receive the ball. [2]

3 vs 3 Side Build Up Play: Instead of crossing, the opponent player on the side builds up a play by short passing to its teammates.

Testing Generalization

For all the new benchmark scenarios in our dataset as well as for the selected five GRF's scenarios, we generalized those scenarios to test the generalizability of the trained RL agent. Our test scenarios are juxtaposed to corresponding scenarios in Fig. 2.8, 2.10, 2.11, 2.12. We modelled these test scenarios by either (i) adding distribution over the initial state or (ii) creating a symmetric opposite formation.

Semi-Expert Stochastic Policies

We selected five scenarios from GRF's and our benchmark scenarios. The selected GRF's scenarios are shown in Fig. 2.13. The following are the brief descriptions of policies encoded in SCENIC for each scenario.

Pass and Shoot with a Goal Keeper: We randomly choose one of the two policies for the RL agent. In the first policy, the player dribbles to the penalty area and shoots once inside it. In the second policy, the player passes the ball to the teammate, who will then dribble towards the goal and shoot.

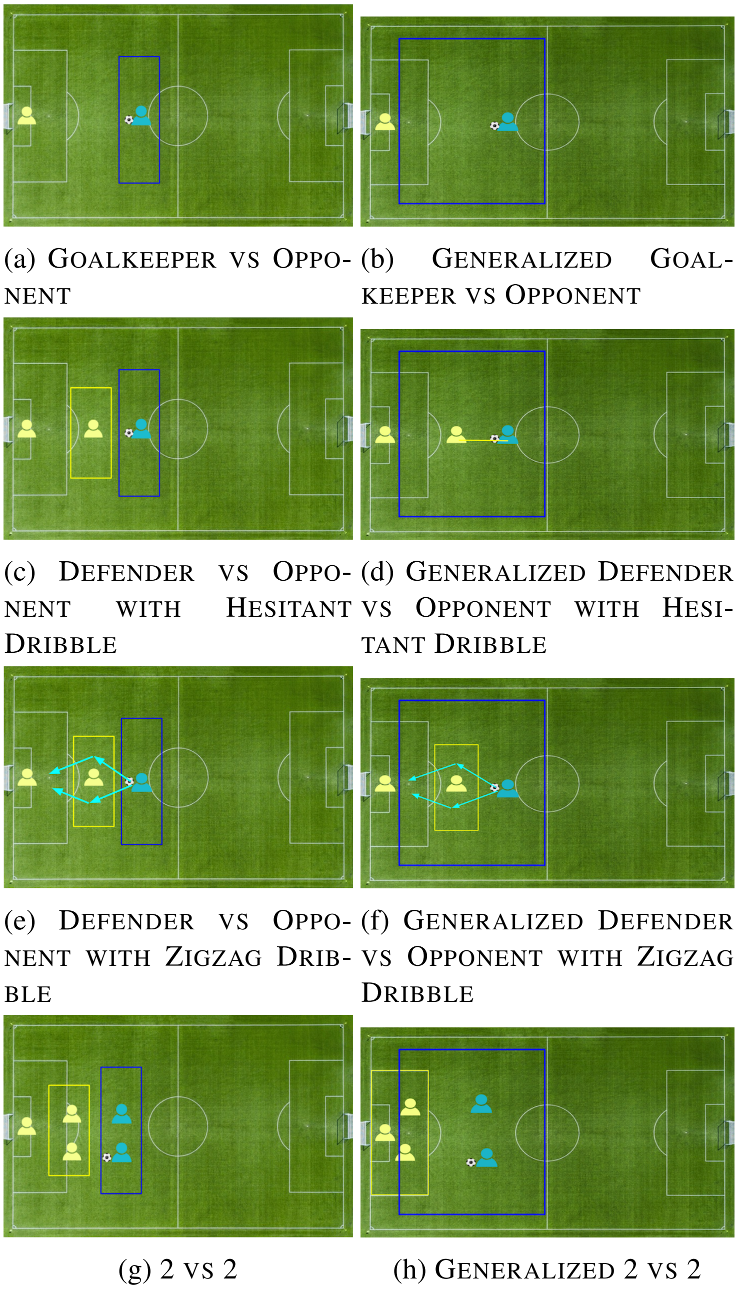
Easy Counterattack: The first player will pass the ball to the right midfielder. Then, the player with the ball will run into the penalty area, and if there is an opponent player on the way, the player will pass the ball to the nearest teammate. The player will shoot at a corner of the goal once inside the penalty area.

Run to Score with a Goal Keeper: The player with the ball will first sprint towards the goal and turn slightly to either left or right randomly to evade the opponent goalkeeper's interception. Once the player bypasses the goalkeeper, or is inside the penalty area, the player will shoot.

Avoid Pass and Shoot: The player decides to go towards 3 suitable regions of scoring: left edge/ middle/ right edge of the right goal post, by keeping as much distance possible to the opponent defender. At each time step it decides one of the three destination location. It first predicts its next position for all the three suitable destinations and pick the direction which keeps it farthest of the opponent. If the player comes near the defender it passes the ball to its teammate and if it can successfully go near the goal post, it attempts shooting.

11 vs Goal Keeper: The player with the ball runs towards the right goalpost, if it reaches near the goal post it attempts to shoot. If the opponent goal keeper comes near (i.e. within seven meters) our player before it can reach near the right goal post, it stops running and shoots immediately.

[2] The DEFENSE 3VS3 WITH CROSS scenario doesn't conclude a game upon a change in ball possession, unlike other scenarios



(a) Goalkeeper vs Opponent

(b) Generalized Goalkeeper vs Opponent

(c) Defender vs Opponent with Hesitant Dribble

(d) Generalized Defender vs Opponent with Hesitant Dribble

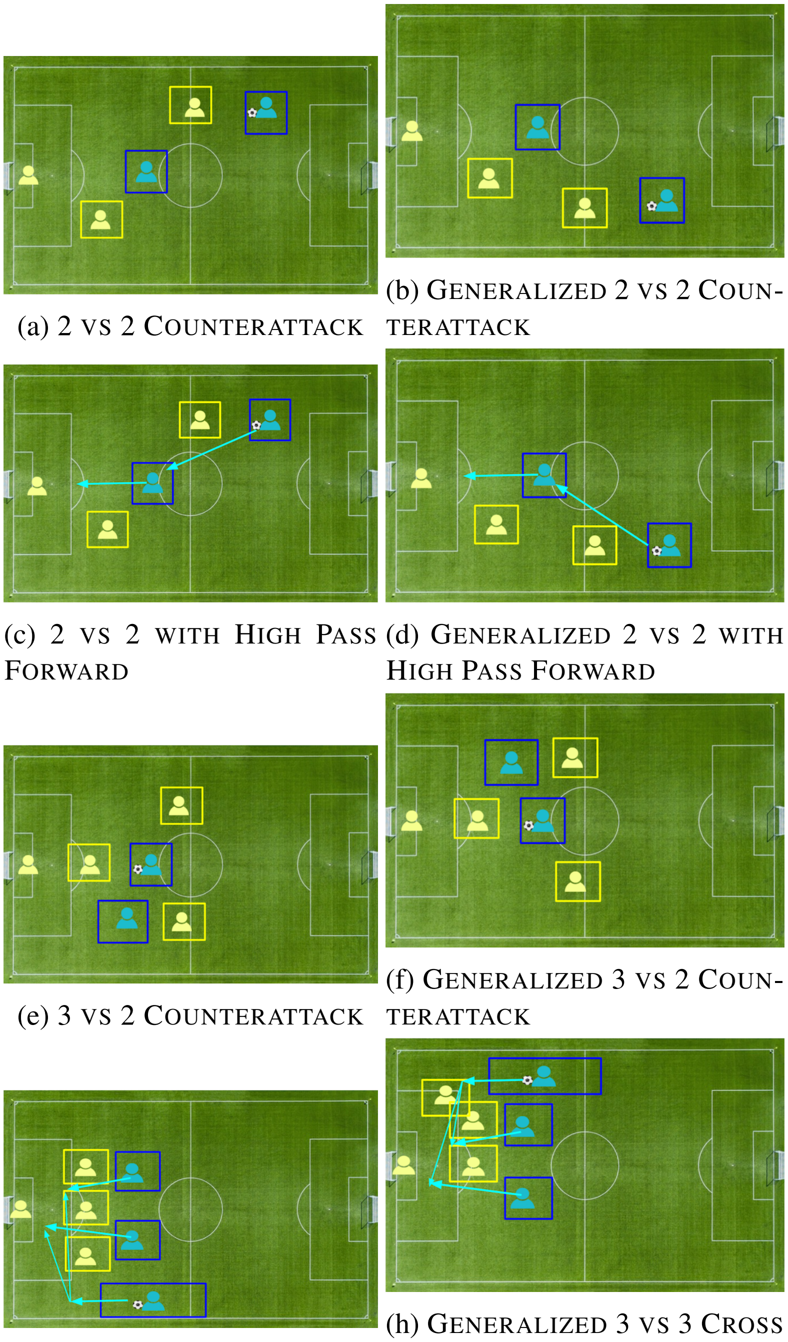
(e) Defender vs Opponent with Zigzag Dribble

(f) Generalized Defender vs Opponent with Zigzag Dribble

(g) 2 vs 2

(h) Generalized 2 vs 2

Figure 2.10: New defense benchmark scenarios (left images) and corresponding generalized test scenarios (right images) in our dataset. The highlighted boxes represent the regions over which players' initial positions are uniformly randomly distributed. The opponent is in blue and the RL team in yellow.



(a) 2 vs 2 Counterattack

(b) Generalized 2 vs 2 Counterattack

(c) 2 vs 2 with High Pass Forward

(d) Generalized 2 vs 2 with High Pass Forward

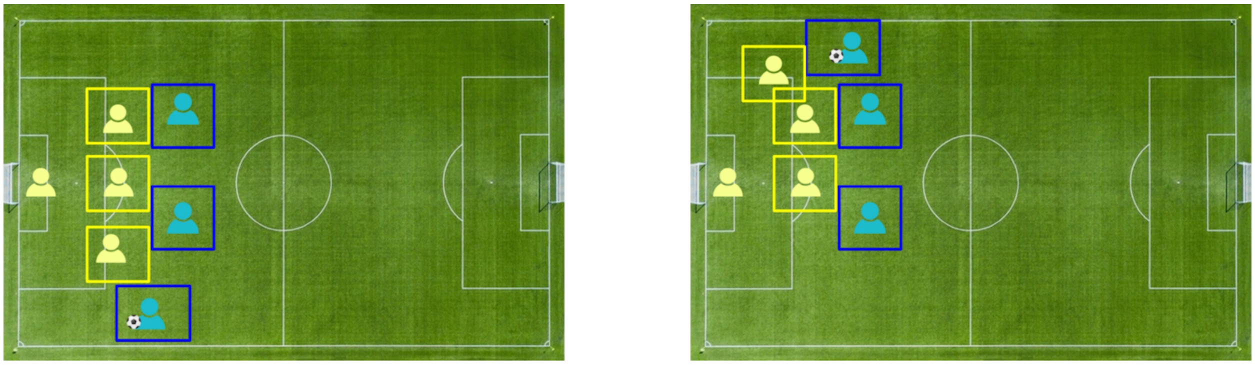
(e) 3 vs 2 Counterattack

(f) Generalized 3 vs 2 Counterattack

(g) 3 vs 3 Cross from side

(h) Generalized 3 vs 3 Cross from side

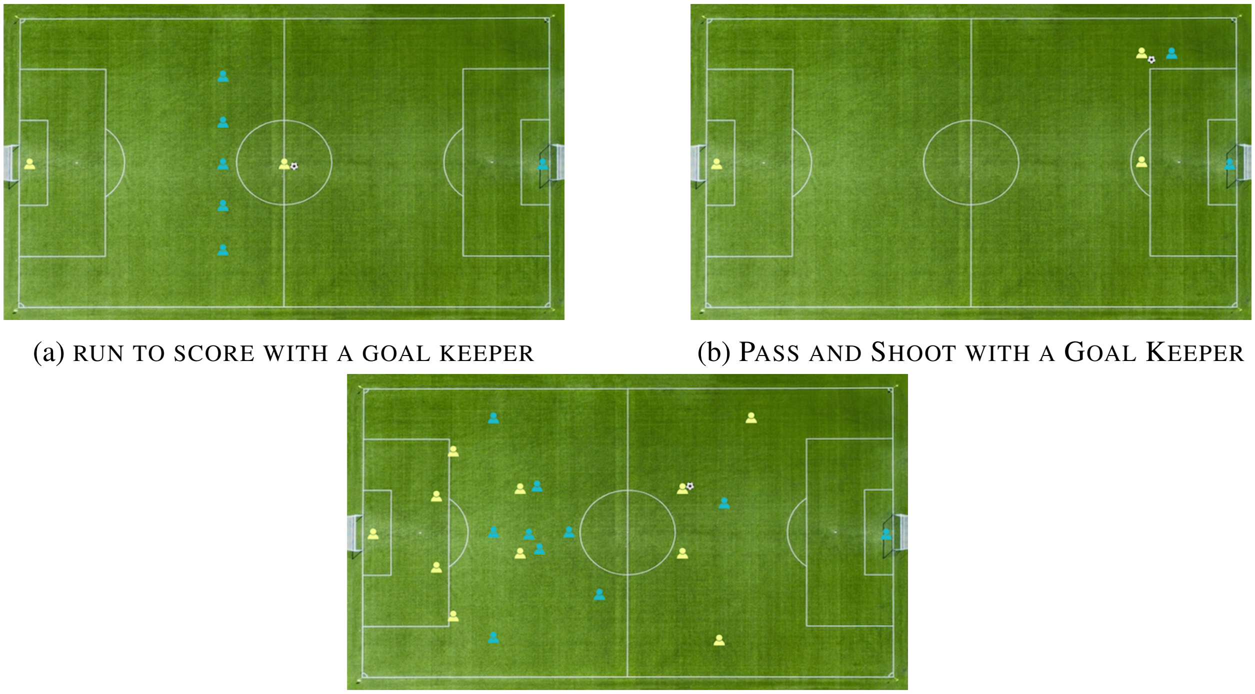
Figure 2.11: New defense benchmark scenarios (left images) and corresponding generalized test scenarios (right images) in our dataset. The highlighted boxes represent the regions over which players' initial positions are uniformly randomly distributed. The opponent is in blue and the RL team in yellow.



(a) 3 vs 3 side build up play

(b) Generalized 3 vs 3 side build up play

Figure 2.12: New defense benchmark scenario (left images) and corresponding generalized test scenarios (right images) in our dataset. The highlighted boxes represent the regions over which players' initial positions are uniformly randomly distributed. The opponent is in blue and the RL team in yellow.



(a) run to score with a goal keeper

(b) Pass and Shoot with a Goal Keeper

(c) easy counterattack

Figure 2.13: Google Research Football environment's scenarios for which we wrote semi-expert RL policies

2.7 On Our Scenic Libraries

Users can quickly model scenarios by referencing models, actions, and behaviors from the libraries that we open-sourced along with our interface. To see our library codes, please refer to our attached README pdf file for the pathways to these libraries.

Model Library

The model library defines three categories of objects. First, it defines different regions of the field such as penalty box area. Second, it defines the `Ball`. Lastly, it defines the `Player`. There are two types of player objects which inherit this `class Player`: Left and Right players. The left players represent the RL team, and the right, the opponent. Within each team, the players are further classified into different roles. The naming convention is “the team + role abbreviated in two letters." For example, the left team’s goalkeeper is defined as texttLeftGK. Likewise, for the right team players.

Action Library

This library defines the action space of any players. It consists of twelve different actions such as `Pass`, `Shoot`, `Dribble`, `Sprint`, `Slide`, etc. These actions can be referenced in the Scenic script using the syntax, *take*. For example, to take sliding action, users can write `take Slide()` in their programs.

Behavior Library

The behavior library consists of basic soccer skills that we modelled in Scenic. This library consists of helper functions defined using the syntax, `def`, and behaviors, which reference those helper functions, are defined with the syntax, `behavior`. For brevity, we refer the reviewers to our annotated library code.

2.8 Details on Experimental Setup and Training

We use the OpenAI Baselines’ [20] implementation of PPO. The training was run for 5M timesteps with 16 parallel workers. All of our experiments are run on g4dn.4xlarge instances on Amazon AWS: a machine with a single NVIDIA T4 gpu, 16 virtual cores and 64GB RAM.

Network architecture & Hyperparameters For the PPO training, we first experimented with the network architecture and hyperparameters from [54] and was able to reproduce their result. [54] did an extensive search to select their hyperparameters and hence we decided to use the same for our experiments. The architecture we used from [54] is similar to the architecture introduced in [22], with the exception of using four big blocks instead of three.

Table 2.1 provides specific values of the hyperparameters used in the PPO experiments.

Parameter	**Value**
Action Repetitions	1
Clipping Range	.115
Discount Factor (γ)	0.997
Entropy Coefficient	0.00155
GAE (λ)	0.95
Gradient Norm Clipping	0.76
Learning Rate	0.00011879
Number of Actors	16
Optimizer	Adam
Training Epochs per Update	2
Training Min-batches per Update	4
Unroll Length/n-step	512
Value Function Coefficient	0.5

Table 2.1: Training Parameters for PPO.

The parameters for behavior cloning is shown in Table 2.2. For the GRF academy scenarios the behavior cloning algorithm is run for 16 epochs while for the offense scenarios it was run for 5 epochs.

Parameter	**Value**
Learning Rate	3e-4
Batch Size	256
Optimizer	Adam
Epsilon(Adam)	1e-5

Table 2.2: Training Parameters for Imitation Learning.

2.9 Interface details and Reproducibility

Our interface follows the widely used OpenAI Gym API [8]. For sample usage, we refer readers to our code that is submitted along with this supplement. The code contains necessary scripts, and the attached README pdf file contains detailed description of all our API with examples and a link to a google drive which contains all our trained checkpoints and training logs.

2.10 Performance

As we are adding an additional layer over the GRF simulator, we wanted to measure how much overhead we are adding over the base GRF simulator. We selected five GRF academy scenarios and ran a simulation of 20K timesteps with a random policy both in the GRF simulator and in our interface. The simulation was ran sequentially, i.e., no parallelism was used. Across the scenarios, the GRF simulator took an average of 74.28 seconds for executing a simulation of 20K timesteps, while our interface took 222.07 seconds, showing a 2.99x drop in speed. Some of this overhead is inevitable however, we believe there are ways to speed up . First, as we change the initial state every episode/simulation: we update the Python scenario file used by the GRF simulator for each episode/simulation. We plan to modify GRF interface to avoid such disk-access each simulation to speed up among other performance improvements. The scenarios we used for the experiments are namely: i) Empty Goal, ii) Empty Goal Close, iii) Pass and Shoot with Keeper, iv) Run, Pass, and Shoot with Keeper, and v) Run to Score with Keeper.

2.11 Conclusion & Future Work

We introduced and demonstrated the benefits of adopting a scenario specification language to train RL agents and test their generalization capabilities in various realistic scenarios generated by SCENIC programs, which succinctly capture distributions of initial states and behaviors. We also showcased modeling domain knowledge via stochastic SCENIC policies by generating demonstration data to facilitate training in GRF, a complex real-time strategy environment. We hope our work could gather an interest to support systematic modeling of RTS environments.

Acknowledgements

The research work presented in this chapter is performed in collaboration with Edward Kim, Qiancheng Wu, Kimin Lee, Prof. Ion Stoica, Prof. Pieter Abbeel, Prof. Alberto Sangiovanni-Vincentelli, and Prof. Sanjit A Seshia from UC Berkeley. It has been published in Proceedings of the 36th AAAI Conference on Artificial Intelligence [3].

This work was supported in part by National Science Foundation grants CNS-1730628, CNS-1545126 (VeHICaL), CNS-1739816, and CCF-1837132, by DARPA contracts FA8750-16-C0043 (Assured Autonomy) and FA8750-20-C-0156 (Symbiotic Design of Cyber-Physical Systems), by Berkeley Deep Drive, by Toyota through the iCyPhy center, by the Toyota Research Institute, and by the Berkeley Artificial Intelligence Research (BAIR) Commons program.

Chapter 3

CLUTR: Curriculum Learning via Unsupervised Task Representation Learning

Reinforcement Learning (RL) algorithms are often known for sample inefficiency and difficult generalization. Recently, Unsupervised Environment Design (UED) emerged as a new paradigm for zero-shot generalization by simultaneously learning a task distribution and agent policies on the generated tasks. This is a non-stationary process where the task distribution evolves along with agent policies; creating an instability over time. While past works demonstrated the potential of such approaches, sampling effectively from the task space remains an open challenge, bottlenecking these approaches. To this end, we introduce CLUTR: a novel unsupervised curriculum learning algorithm that decouples task representation and curriculum learning into a two-stage optimization. It first trains a recurrent variational autoencoder on randomly generated tasks to learn a latent task manifold. Next, a teacher agent creates a curriculum by maximizing a minimax REGRET-based objective on a set of latent tasks sampled from this manifold. Using the fixed-pretrained task manifold, we show that CLUTR successfully overcomes the non-stationarity problem and improves stability. Our experimental results show CLUTR outperforms PAIRED, a principled and popular UED method, in the challenging CarRacing and navigation environments: achieving 10.6X and 45% improvement in zero-shot generalization, respectively. CLUTR also performs comparably to the non-UED state-of-the-art for CarRacing, while requiring 500X fewer environment interactions. We open source our code at https://github.com/clutr/clutr.

3.1 Introduction

Deep Reinforcement Learning (RL) has shown exciting progress in the past decade in many challenging domains including Atari [62], Dota [6], Go [83]. However, deep RL is also known for its sample inefficiency and difficult generalization—performing poorly on unseen tasks or failing altogether with the slightest change [15, 3, 100]. While, Curriculum Learning (CL) algorithms have shown to improve RL sample efficiency by adapting the training task distribution, i.e., the curriculum [70, 66], recently a class of Unsupervised CL algorithms, called Unsupervised Environment Design (UED) [19, 45] has shown promising zero-shot generalization by automatically generating the training tasks and adapting the curriculum simultaneously.

UED algorithms employ a teacher that generates training tasks by sampling the free parameters of the environment (e.g., the start, goal, and obstacle locations for a navigation task) and can either be adaptive or random. Contemporary adaptive UED teachers, i.e., PAIRED [19] and REPAIRED [45], are implemented as RL agents with the free task parameters as their action space. The teacher agent aims at generating tasks that maximize the student agent's regret, defined as the performance gap between the student agent and an optimal policy. Inspite of promising zero-shot generalization, adaptive teacher UEDs are still sample inefficient.

This sample inefficiency is attributed primarily to the difficulty of training a regret based RL teacher [68]. First, the teacher receives a sparse reward only after specifying the full parameterization of a task; leading to a long-horizon credit assignment problem. Additionally, the teacher agent faces a combinatorial explosion problem if the parameter space is permutation invariant—e.g., for a navigation task, a set of obstacles corresponds to factorially different permutations of the parameters[1]. Most importantly, the teacher needs to simultaneously learn a task manifold–from scratch–to generate training tasks and navigate this manifold to induce an efficient curriculum. However, the teacher learns this task manifold implicitly based on the student regret and as the student is continuously co-learning with the teacher, the task manifold also keeps evolving over time. Hence, the simultaneous learning of task manifold and curriculum results in an instability over time and makes it a difficult learning problem.

To address the above-mentioned challenges, we present Curriculum Learning via Unsupervised Task Representation Learning (CLUTR). At the core of CLUTR, lies a hierarchical graphical model that decouples task representation learning from curriculum learning. We develop a variational approximation to the UED problem and employ a Recurrent Variational AutoEncoder (VAE) to learn a latent task manifold, which is pretrained unsupervised. Unlike contemporary adaptive-teachers, which builds the tasks from scratch one parameter at a time, the CLUTR teacher generates tasks in a single timestep by sampling points from the latent task manifold and uses the generative model to translate them into complete tasks. The CLUTR teacher learns the curriculum by navigating the pretrained and fixed task manifold via maximizing regret. By utilizing a pretrained latent task-manifold, the CLUTR teacher can train as a contextual bandit – overcoming the long-horizon credit assignment problem – and create a curriculum much more efficiently – improving stability at no cost to its effectiveness. Finally, by carefully introducing bias to the training corpus (such as sorting each parameter vector), CLUTR solves the combinatorial explosion problem of parameter space without using any costly environment interactions.

While CLUTR can be integrated with any adaptive teacher UEDs, we implement CLUTR on top of PAIRED—one of the most principled and popular UEDs. Our experimental results show

[1] Consider a 13x13 grid for a navigation task, where the locations are numbered from 1 to 169. Also consider a wall made of four obstacles spanning the locations: {21, 22, 23, 24}. This wall can be represented using any permutation of this set, e.g., {22, 24, 23, 21}, {23, 21, 24, 22}, resulting in a combinatorial explosion.

that CLUTR outperforms PAIRED, both in terms of generalization and sample efficiency, in the challenging pixel-based continuous CarRacing and partially observable discrete navigation tasks. For CarRacing, CLUTR achieves 10.6X higher zero-shot generalization on the F1 benchmark [45] modeled on 20 real-life F1 racing tracks. Furthermore, CLUTR performs comparably to the non-UED attention-based CarRacing SOTA [88], outperforming it in nine of the 20 test tracks while requiring 500X fewer environment interactions. In navigation tasks, CLUTR outperforms PAIRED in 14 out of the 16 unseen tasks, achieving a 45% higher solve rate.

In summary, we make the following contributions:

1. We introduce CLUTR, a novel adaptive-teacher UED algorithm derived from a hierarchical graphical model for UEDs, that augments the teacher with unsupervised task-representation learning

2. CLUTR, by decoupling task representation learning from curriculum learning, solves the long-horizon credit assignment and the combinatorial explosion problems faced by regret-based adaptive-teacher UEDs such as PAIRED.

3. Our experimental results show CLUTR significantly outperforms PAIRED, both in terms of generalization and sample efficiency, in two challenging domains: CarRacing and navigation.

3.2 Related Work

Unsupervised Curriculum Design

[19] was the first to formalize UED and introduced the minimax regret-based UED teacher algorithm, PAIRED, with a strong theoretical robustness guarantee. However, gradient-based multi-agent RL has no convergence guarantees and often fails to converge in practice [59]. Pre-existing techniques like Domain Randomization (DR) [42, 73, 90] and minimax adversarial curriculum learning [63, 69] also fall under the category of UEDs. DR teacher follows a uniform random strategy, while the minimax adversarial teachers follow the maximin criteria, i.e., generate tasks that minimize the returns of the agent. POET [95] and Enhanced POET [94] also approached UED, before PAIRED, using an evolutionary approach of a co-evolving population of tasks and agents.

Recently, [45] proposed Dual Curriculum Design (DCD): a novel class of UEDs that augments UED generation methods (e.g., DR and PAIRED) with replay capabilities. DCD involves two teachers: one that actively generates tasks with PAIRED or DR, while the other curates the curriculum to replay previously generated tasks with Prioritized Level Replay (PLR) [44]. [45] shows that, even with random generation (i.e., DR), updating the students only on the replayed level (but not while they are first generated, i.e., no exploratory student gradient updates as PLR) and with a regret-based scoring function, PLR can also learn minimax-regret agents at Nash Equilibrium and call this variation Robust PLR. It also introduces REPAIRED, combining PAIRED with Robust PLR. [68] introduces ACCEL, which improves on Robust PLR by allowing edit/mutation of the tasks with an evolutionary algorithm. Currently, random-teacher UEDs outperform adaptive-teacher UED methods.

While CLUTR and other PAIRED-variants actively adapt task generation to the performance of agents, other algorithms such as PLR generate tasks from a fixed-random task distribution, resulting in two categories of UED methods, i) adaptive teacher/generator based UEDs and ii) random-generator based UEDs. The existing adaptive-teacher UEDs are variants of PAIRED, which try to improve PAIRED from different aspects, but are still susceptible to the instability due to an evolving task-manifold. Unlike other PAIRED variants, CLUTR introduces a novel variational formulation with a VAE-style pretraining for task-manifold learning to solve this instability issue and can be applied, also potentially improve, any adaptive-teacher UEDs. On the other hand, random-generator UEDs focus on identifying or, prioritizing which tasks to present to the student from the randomly generated tasks, and is orthogonal to our proposed approach.

For recent advancements on supervised curriculum learning and alternate curriculum objectives, we refer the readers to [40, 53, 11].

Representation Learning

Variational Auto Encoders [52, 72, 38] have widely been used for their ability to capture high-level semantic information from low-level data and generative properties in a wide variety of complex domains such as computer vision [71, 31, 103, 104], natural language [7, 41], speech [12], and music [43]. VAE has been used in RL as well for representing image observations [49, 99] and generating goals [65]. While CLUTR also utilizes similar VAEs, different from prior work, it combines them in a new curriculum learning algorithm to learn a latent task manifold. [24] also proposed a curriculum learning algorithm, however, for latent-space goal generation using a Generative Adversarial Network.

3.3 Background

Unsupervised Environment Design (UED)

As formalized by [19] UED is the problem of inducing a curriculum by designing a distribution of concrete, fully-specified environments, from an underspecified environment with free parameters. The fully specified environments are represented using a Partially Observable Markov Decision Process (POMDP) represented by $(A, O, S, \mathcal{T}, \mathcal{I}, \mathcal{R}, \gamma)$, where A, O, and S denote the action, observation, and state spaces, respectively. $\mathcal{I} \rightarrow O$ is the observation function, $\mathcal{R} : S \rightarrow \mathbb{R}$ is the reward function, $\mathcal{T} : S \times A \rightarrow \Delta(S)$ is the transition function and γ is the discount factor. The underspecified environments are defined in terms of an Underspecified Partially Observable Markov Decision Process (UPOMDP) represented by the tuple $\mathcal{M} = (A, O, \Theta, S^{\mathcal{M}}, \mathcal{T}^{\mathcal{M}}, \mathcal{I}^{\mathcal{M}}, \mathcal{R}^{\mathcal{M}}, \gamma)$. Θ is a set representing the free parameters of the environment and is incorporated in the transition function as $\mathcal{T}^{\mathcal{M}} : S \times A \times \Theta \rightarrow \Delta(S)$. Assigning a value to $\vec{\theta}$ results in a regular POMDP, i.e., UPOMDP + $\vec{\theta}$ = POMDP. Traditionally (e.g., in [19] and [45]) Θ is considered as a trajectory of environment parameters $\vec{\theta}$ or just θ—which we call task in this paper. For example, θ can be a concrete navigation task represented by a sequence of obstacle locations. We denote a concrete environment generated with the parameter $\vec{\theta} \in \Theta$ as $\mathcal{M}_{\vec{\theta}}$ or simply $\mathcal{M}_{\theta}$. The value of a policy π in $\mathcal{M}_{\theta}$ is defined as $V^{\theta}(\pi) = \mathbb{E}[\sum_{t=0}^{T} r_t \gamma^t]$, where r_t is the discounted reward obtained by π in $\mathcal{M}_{\theta}$.

PAIRED

PAIRED [19] solves UED with an adversarial game involving three players [2]: the agent π_P and an antagonist π_A, which are trained on tasks generated by the teacher $\tilde{\theta}$. PAIRED objective is: $max_{\tilde{\theta},\pi_P} min_{\pi_A} U(\pi_P, \pi_A, \tilde{\theta}) = \mathbb{E}_{\theta \sim \tilde{\theta}}[\textsc{Regret}^{\theta}(\pi_P, \pi_A)]$. Regret is defined by the difference of the discounted rewards obtained by the antagonist and the agent in the generated tasks, i.e., $\textsc{Regret}^{\theta}(\pi_P, \pi_A) = V^{\theta}(\pi_A) - V^{\theta}(\pi_P)$. The PAIRED teacher agent is defined as $\Lambda : \Pi \rightarrow \Delta(\Theta^T)$, where Π is a set of possible agent policies and Θ^T is the set of possible tasks. The teacher is trained with an RL algorithm with U as the reward while, the protagonist and antagonist agents are trained using the usual discounted rewards from the environments. [19] also introduced the flexible regret objective, an alternate regret approximation that is less susceptible to local optima. It is defined by the difference between the average score of the agent and antagonist returns and the score of the policy that achieved the highest average return.

3.4 Curriculum Learning via Unsupervised Task Representation Learning

In this section, we formally present CLUTR as a latent UED and discuss it in details.

Formulation of CLUTR

At the core of CLUTR is the latent generative model representing the latent task manifold. Let's assume that R is a random variable that denotes a measure of success over the agent and antagonist agent and z be a latent random variable that generates environments/tasks, denoted by the random variable E. We use the graphical model shown in Figure 3.1 to formulate CLUTR. Both E and R are observed variables while z is an unobserved latent variable. R can cover a broad range of measures used in different UED methods including PAIRED and DR (Domain Randomization). In PAIRED, R represents the REGRET.

We use a variational formulation of UED by using the above graphical model to derive the following ELBO for CLUTR, where VAE(z, E) denotes the VAE objective:

[2] In the original PAIRED paper, the primary student agent was named *protagonist*. However, in this paper, we generally refer to it simply as the *agent*, except in a few instances where using the term *protagonist agent* provides greater clarity.

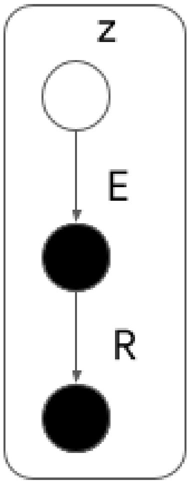


Figure 3.1: Hierarchical Graphical Model for CLUTR

$$ELBO \approx \textsc{Regret}(R, E) - \text{VAE}(z, E) \tag{3.1}$$

We share the details of this derivation in Section 3.6 of the Appendix. The above ELBO (Eq.3.1) defines the optimization objective for CLUTR, which can be seen as optimizing the VAE objective with a regret-based regularization term and vice versa. As previously discussed, it is difficult to train a UED teacher while jointly optimizing for both the curriculum and task representations. Hence, we propose a two-level optimization for CLUTR. First, we pretrain a VAE to learn unsupervised task representations, and then in the curriculum learning phase, we optimize for regret to generate the curriculum while keeping the VAE fixed. In Section 3.5, we empirically show that this two-level optimization performs better than the joint optimization of Eq.3.1, i.e., finetuning the VAE decoder with the regret loss during the curriculum learning phase.

Unsupervised Latent Task Representation Learning

As discussed above, we use a Variational AutoEncoder (VAE) to model our generative latent task-manifold. Aligning with [19] and [45], we represent task θ, as a sequence of integers. For example, in a navigation task, these integers denote obstacle, agent, and goal locations. We use an LSTM-based Recurrent VAE [7] to learn task representations from integer sequences. We learn an embedding for each integer and use cross-entropy over the sequences to measure the reconstruction error. This design choice makes CLUTR applicable to task parameterization beyond integer sequences, e.g., to sentences or images. To train our VAEs, we generate random tasks by uniformly sampling from Θ^T, the set of possible tasks. Thus, we do not require any interaction with the environment to learn the task manifold. Such unsupervised training of the task manifold is practically very useful as interactions with the environment/simulator are much more costly than sampling.

Algorithm 1 CLUTR

```
1: Pretrain VAE with randomly sampled tasks from $\Theta$
2: Randomly initialize Agent $\pi^P$, Antagonist $\pi^A$, and Teacher $\tilde{\Lambda}$;
3: repeat
4:     Generate latent task vector: $z \sim \mathcal{Z}$ from the teacher
5:     Create POMDP $M_\theta$ where $\theta = \mathcal{G}(z)$ and $\mathcal{G}$ is the VAE decoder function
6:     Collect Agent trajectory $\tau^P$ in $M_\theta$. Compute: $U^\theta(\pi^P)$ = $\sum_{i=0}^{T} r_t \gamma^t$
7:     Collect Antagonist trajectory $\tau^A$ in $M_\theta$. Compute: $U^\theta(\pi^A)$ = $\sum_{i=0}^{T} r_t \gamma^t$
8:     Compute: $\textsc{Regret}^\theta(\pi^P, \pi^A) = U^\theta(\pi^A) - U^\theta(\pi^P)$
9:     Train Agent policy $\pi^P$ with RL update and reward $R(\tau^P) = U^\theta(\pi^P)$
10:    Train Antagonist policy $\pi^A$ with RL update and reward $R(\tau^A) = U^\theta(\pi^A)$
11:    Train Teacher policy $\tilde{\Lambda}$ with RL update and reward $R(\tau^{\tilde{\Lambda}}) = \textsc{Regret}$
12: until not converged
```

Furthermore, we sort the input sequences, fully or partially, when they are permutation invariant, i.e., essentially represent a set. By sorting the training sequences, we avoid the combinatorial explosion faced by other adaptive UED teachers.

CLUTR

Now we describe CLUTR, which is outlined in Algorithm 1. As discussed in Section 3.4, CLUTR follows a two-stage optimization of Eq. 3.1. First, the VAE is pretrained to learn the latent task-manifold $\mathcal{Z}$ (Line 1) and kept fixed during the curriculum learning phase—the loop spanning Line 3 to 12. Similar to existing adaptive-teacher UED methods, CLUTR learns a curriculum employing an adversarial game where the agent π_P and the antagonist π_A solve environments generated by the teacher $\tilde{\Lambda}$. However, unlike the exisiting adaptive-teachers which directly generate the task parameters θ, CLUTR teacher is a latent task designer/generator. Defined as $\Lambda : \Pi \rightarrow \Delta(\mathcal{Z})$, CLUTR teacher samples latent task vectors z from the latent task-manifold $\mathcal{Z}$, where Π is a set of possible agent policies (Line 4). We then create an environment with the concrete task parameters $\theta = \mathcal{G}(z)$ using the VAE decoder $\mathcal{G} : \mathcal{Z} \rightarrow \Theta$ (Line 5). The agent and the antagonist then navigate these environments. These trajectories are collected (Line 6 and 7) and the agent and the antagonist are updated using the usual discounted rewards from the environments (Line 9-10). To learn the curriculum, CLUTR teacher is trained using the same regret-based objective as PAIRED: $\textsc{Regret}(R, E) = \textsc{Regret}^\theta(\pi_P, \pi_A)$ (Line 8 and 11). In our implementation, we used the Proximal Policy Optimization [75] algorithm for updating the teacher and the student agents. As we notice, CLUTR is outlined similar to PAIRED, but with two critical updates to incorporate the latent space in Line 4 and 5.

Now we discuss a couple of additional properties of CLUTR compared to other adaptive-teacher UEDs, i.e., PAIRED and REPAIRED. First, CLUTR teacher samples from the latent space $\mathcal{Z}$ and thus generates a task in a single timestep. Note that this is not possible for other adaptive UED teachers, as they operate on parameter space and generate one task parameter at a time,

Algorithm	Task Representation Learning	Teacher Model	UED Method	Replay Method
DR	-	Random	DR	-
PLR				PLR
Robust PLR				Robust PLR
ACCEL			DR + Evolution	
PAIRED	Implicit via RL	Learned	Regret	-
REPAIRED				Robust PLR
CLUTR	Explicit via Unsupervised Generative Model			-

Table 3.1: A comparative characterization of contemporary UED methods

conditioned on the state of the partially-generated task so far. Furthermore, Adaptive-teacher UEDs typically observe the state of their partially generated task to generate the next parameters. Hence, they require designing different teacher architectures for environments with different state space. CLUTR teacher architecture, however, is agnostic of the problem domain and does not depend on their state space. Hence, the same architecture can be used across different environments.

CLUTR in the Context of Contemporary UED Method Landscape

As discussed in Section 3.2, contemporary UED methods can be characterized by their i) teacher type (random/fixed or, learned/adaptive) and, ii) the use of replay. To clearly place CLUTR in the context of contemporary UEDs, we discuss another important aspect of curriculum learning algorithms: how the task manifold is learned. The random-generator UEDs (e.g., DR, PLR) do not learn a task manifold. Regret-based adaptive-teachers, i.e., PAIRED and REPAIRED, learn an implicit (e.g., the hidden state of the teacher LSTM) task-manifold—from scratch—but it is not utilized explicitly. It is trained via RL, based on the regret estimates of the tasks they generate. Hence, these task-manifolds depend on the quality of the estimates, which in turn depends on the overall health of the multi-agent RL training. Furthermore, they do not take into account the actual task structures. In contrast, CLUTR introduces an explicit task-manifold modeled with VAE, that can represent a local neighborhood structure capturing the similarity of the tasks, subject to the parameter space being used. Hence, similar tasks (in terms of parameterization) would be placed nearby in the latent space. Intuitively this local neighborhood structure should facilitate the teacher to navigate the manifold effectively. The above discussion illustrates that CLUTR along with PAIRED and REPAIRED form a category of UEDs that generates tasks based on a learned task-manifold, orthogonal to the random generation-based methods, while CLUTR being the only one utilizing an unsupervised generative task manifold. Table 3.1 summarizes the similarity and differences.

3.5 Experiments

In this section, we evaluate CLUTR in two challenging domains: i) Pixel-Based Car Racing with continuous control and dense rewards, and ii) partially observable navigation tasks with discrete control and sparse rewards. We compare CLUTR primarily with PAIRED to analyze its impact on improving adaptive-teacher UED algorithms, experimenting with two commonly used regret objectives: standard and flexible. As discussed in Section 3.2 and 3.4, there are other random-generator and adaptive-teacher UEDs employing techniques complimentary or orthogonal to our approach. For completeness, we compare CLUTR with such existing UED methods in Section 3.6.

We then empirically investigate the following hypotheses:

H1: Simultaneous learning of latent task manifold and curriculum degrades performance (Section 3.5)

H2: Training VAE on sorted data solves the combinatorial explosion problem. (Section 3.5)

At last, we analyze CLUTR curriculum in multiple aspects while comparing it with PAIRED to have a closer understanding. Full details of the environments, network architectures, training hyperparameters, VAE training and further details are discussed in the Appendix.

CLUTR Performance on Pixel-Based Continuous Control CarRacing Environment

The CarRacing environment [45, 9] requires the agent to drive a full lap around a closed-loop racing track modeled with Bézier Curves [64] of up to 12 control points. Both CLUTR and PAIRED were trained for 2M timesteps for flexible regret objective and for 5M timesteps for the standard regret objective experiments. We train the VAE on 1 million randomly generated tracks for 1 million gradient updates. Note that only one VAE was trained and used for all the experiments (10 independent runs, both objectives). We evaluate the agents on the F1 benchmark [45] containing 20 test tracks modeled on real-life F1 racing tracks. These tracks are significantly out of distribution than any tracks that the UED teachers can generate with just 12 control points. Further details on the environment, network architectures, VAE training, and detailed experimental results with analysis can be found in Section 3.6, 3.6, 3.6, 3.6 of the Appendix, respectively.

Figure 3.2 shows the mean return obtained by CLUTR and PAIRED on the full F1 benchmark, on. We independently experimented with both the standard and flexible regret objectives. We notice that PAIRED performs miserably with standard regret in these tasks. However, implementing CLUTR or changing to the flexible regret objective, improves the performance considerably. Furthermore, CLUTR with flexible regret results in much better performance, comparable to the non-UED attention-based SOTA for CarRacing [88], despite not using a self-attention policy and training on 500X fewer environment interactions, while outperforming it on nine of the 20 F1 tracks (See Table 3.4 in Appendix). We also note, CLUTR improves PAIRED irrespective of the choice of the regret objectives: achieving 10.6X and 82% higher returns with standard and flex-

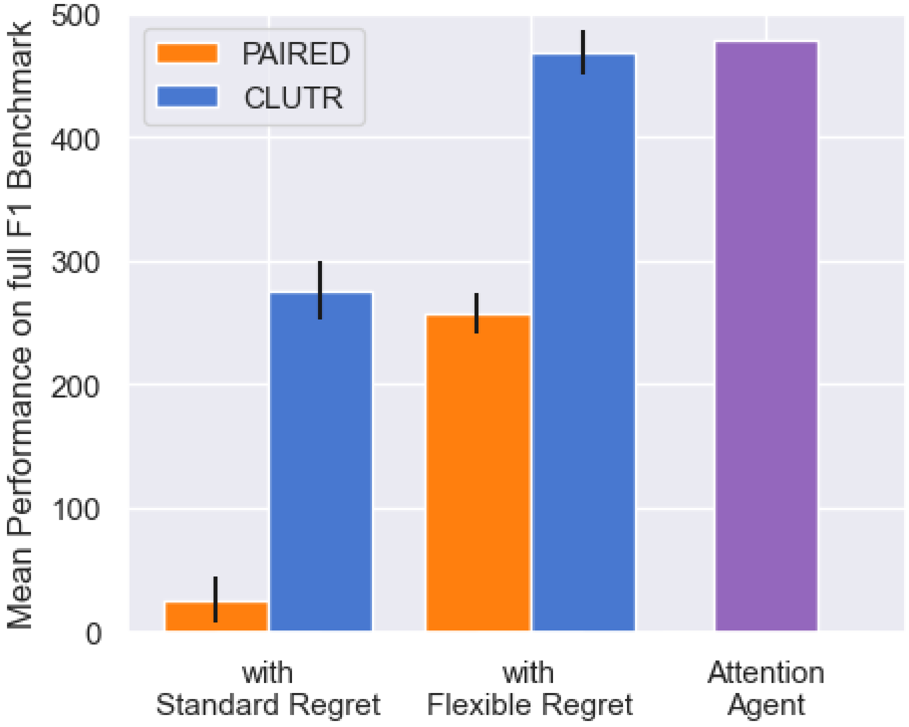
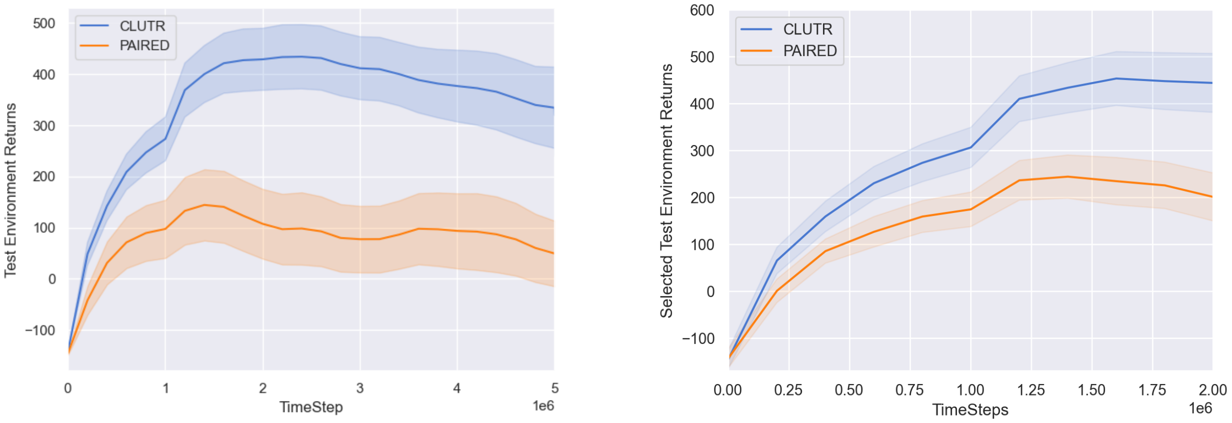


Figure 3.2: Comparison on the F1 Benchmark comprising 20 tracks modeled on real-life F1 racing tracks collected from 10 independent runs. CLUTR achieves 10.6X and 82% higher returns than PAIRED with standard and flexible regret objectives, respectively. CLUTR also performs comparably to the attention-based non-UED CarRacing SOTA, while requiring 500X fewer environment interactions.

ible regret objectives, respectively and outperforming PAIRED on each of the 20 F1 tracks (See Table 3.4). Figure 3.3 illustrate the agents' generalization capabilities during training, by periodically evaluating them on a subset of three unseen F1 tracks: Singapore, Germany, and Italy, which are selected aligning with [45]. Based on these environments, CLUTR shows significantly better trends of sample efficiency, achieving better generalization with significantly fewer environment interactions compared to PAIRED. Furthermore, CLUTR (with flexible regret) emerges as the best adaptive-teacher UED for CarRacing outperforming the other adaptive-teacher UED: REPAIRED and random-generator UEDs: DR, and PLR by 58%, 38% and 16%, repectively. CLUTR is also the only adaptive-teacher UED that outperforms the random-teacher UED methods. CLUTR falls short (by 14%) only to Robust PLR—a random generator dual-curriculum UED with replay and stop-gradient capabilities—a method fundamentally different than ours or, PAIRED. Further discussion with detailed performance and comparison can be found in Section 3.6.



(a) with Standard Objective

(b) with Flexible Objective

Figure 3.3: Zero-shot generalization over the course of training by periodic evaluation on a subset of three F1 tracks: Singapore, Germany, and Italy. CLUTR indicate significantly better sample efficiency than PAIRED.

CLUTR Performance on Partially Observable Navigation Tasks on MiniGrid

We now compare CLUTR with PAIRED on the popular MiniGrid environment, originally introduced by [10] and adopted by [19] for UEDs, for both standard and flexible regret objectives. In these navigation tasks, an agent explores a grid world to find the goal while avoiding obstacles and receives a sparse reward upon reaching the goal. For flexible regret experiment, we generated 10 million random grids to train the VAE, with the obstacle locations sorted, and the number of obstacles uniformly varying from zero to 50, aligning with [19]. The standard regret experiment uses a similar but smaller dataset of 1 million grids. Note that the results reported in the original PAIRED paper are obtained after 3 billion timesteps of training, while we train PAIRED and CLUTR for 250M and 500M timesteps (5 independent runs), for flexible and standard regret objectives, respectively. We evaluate on a testset of 16 novel navigation tasks from [19].

Figure 3.4 shows the mean solve rate obtained by CLUTR and PAIRED on the test dataset. CLUTR improves PAIRED irrespective of the choice of the regret objectives: 45% and 35% higher solve rate than PAIRED outperforming on 14 and 13 individual test grids out of 16 (See Figure 3.26 and Figure 3.21 in Section 3.6 for details), with standard and flexible regret objectives, respectively. Figure 3.5 plot solve rate on all the 16 test grids during training for flexible objective and a subset of four grids, namely, Sixteen Rooms, Sixteen Rooms with Fewer Doors, Labyrinth, and Large Corridor, for standard objective. We see CLUTR, though showing an initial dip for flexible objective, shows better sample efficiency by achieving a higher solve rate earlier than PAIRED.

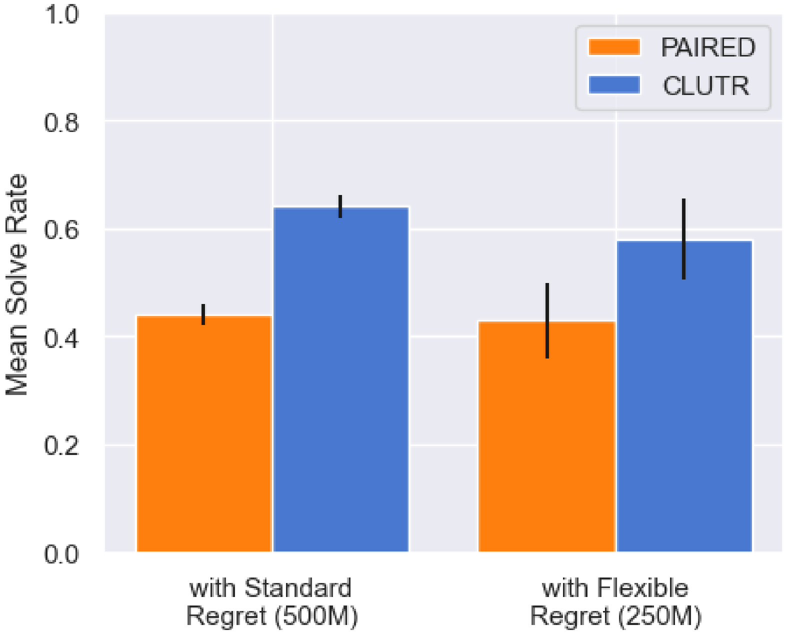
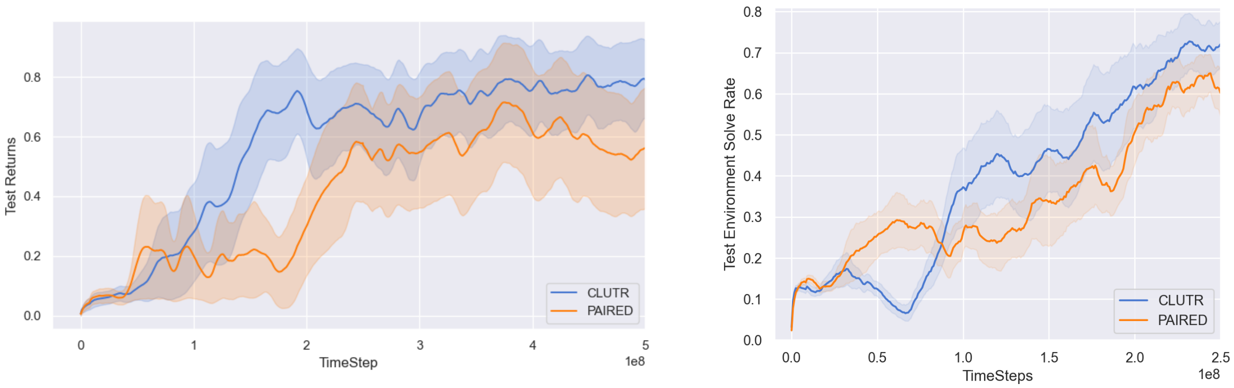


Figure 3.4: Mean solve rate on the test dataset comprising 16 novel nagivation tasks from 5 independent runs. CLUTR achieves 45% and 35% higher solve rate than PAIRED, with standard and flexible regret objectives, respectively.



(a) with Standard Objective

(b) with Flexible Objective

Figure 3.5: Agent solved rate on the 16 unseen grids from [19] during training. CLUTR shows better sample efficiency and generalization than PAIRED. The results show an average of 5 independent runs.

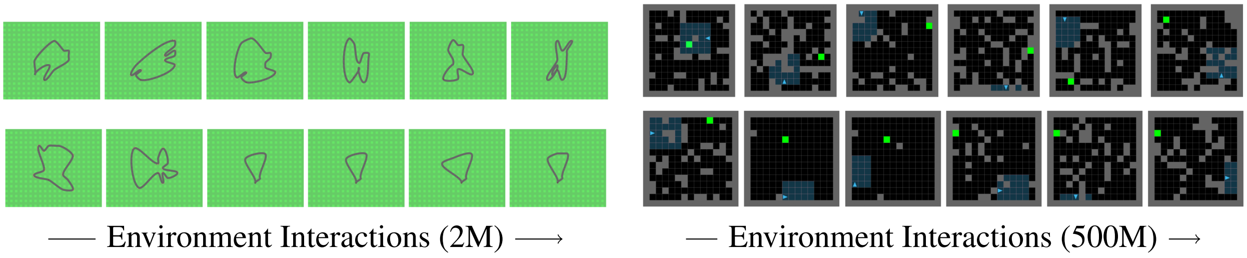

—— Environment Interactions (2M) ——→
— Environment Interactions (500M) —→

Figure 3.6: Example tracks(left) and grids(right) generated by CLUTR(top) and PAIRED(bottom) uniformly sampled at different stages of training. The training progresses from left to right. PAIRED seems to generate over simplified tasks for substantial amount of time hampering agent learning. CLUTR generates interesting tasks throughout.

Learning Task Manifold and Curriculum: Joint vs Two-staged Optimization

We hypothesized that learning the task representation and the curriculum simultaneously results in a difficult learning problem due to the non-stationarity of the process. To test this, we conduct an experiment in which we allow finetuning our pretained decoder with the regret loss during the curriculum learning phase. This experiment, namely 'CLUTR with Decoder Finetuning', shows a 29% performance drop in the CarRacing domain with the standard regret objective (Figure 3.7). Similarly, we see a drop of 10% in case of flexible regret further justifying our hypothesis (See Section 3.6 for details). As a side note, the smaller drop in the later case indicates that flexible objective mitigates some of the instability problem too. Finally, even with decoder finetuning, CLUTR achieves 7.6X and 65% improvement over PAIRED, for standard and flexible regret respectively—indicating the benefits of pretrained decoupled latent task space. The above experimental results thus empirically validates our hypothesis that keeping the pretrained task manifold fixed during curriculum learning helps solving the instability problem.

Impact of Sorting VAE Training Data on Solving Combinatorial Explosion

We hypothesized that training a VAE on sorted sequences can solve the combinatorial explosion problem. To test this, we conduct an experiment, 'CLUTR with Shuffled VAE', in which we train CLUTR with an alternate VAE—trained 5X longer on a non-sorted and 10X bigger version of the original dataset. This experiment shows a 31% performance drop in the CarRacing domain as seen in Figure 3.7, empirically validating our hypothesis. On another note, CLUTR with Shuffled VAE still shows a 7.3X improvement over PAIRED. This indicates that, even when the task manifold is 'suboptimal', a fixed and pretrained task-manifold, i.e., the decoupling of task representation and curriculum learning, helps solving the learning instability and combinatorial explosion problem faced by PAIRED. Further details of this experiment are discussed in Section 3.6 of the Appendix.

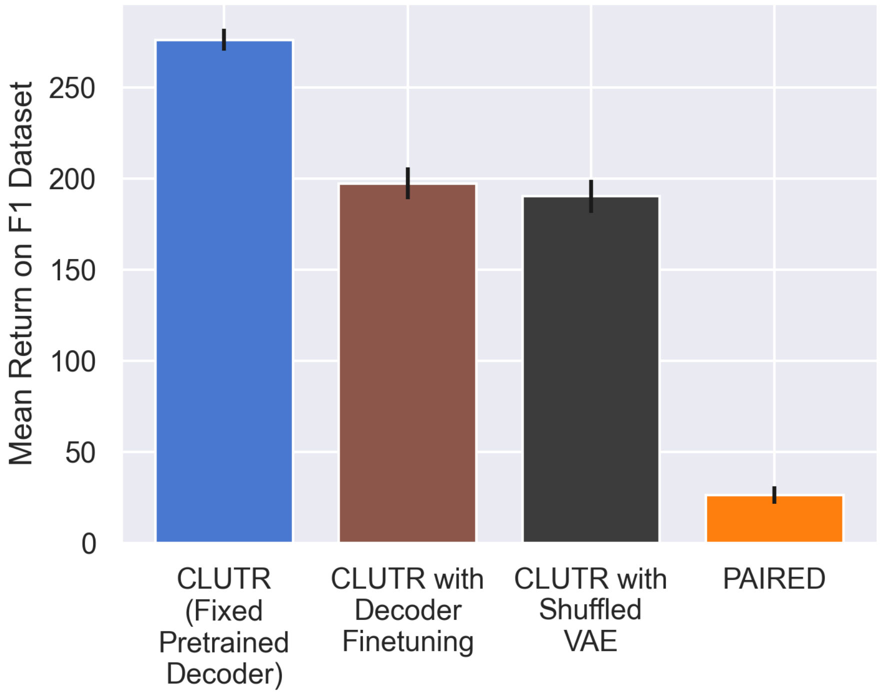
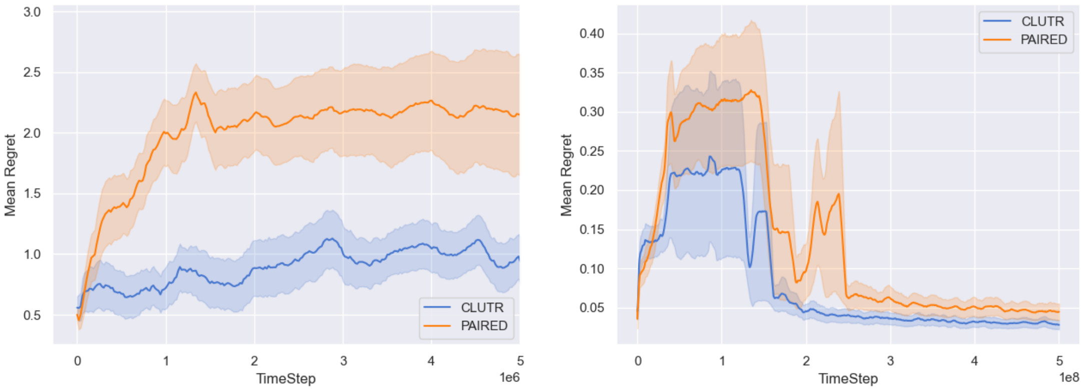


Figure 3.7: Impact of i) joint vs two-staged optimization of the task manifold and ii) using a 'Shuffled' VAE, trained on a larger shuffled dataset. The leftmost column shows the default CLUTR performance—i.e., using a pretrained decoder (VAE) trained on sorted training data, kept fixed during the curriculum learning phase—with standard regret objective for CarRacing. Allowing the decoder to finetune with the regret loss results in a 29% performance drop and the use of Shuffled VAE shows a drop of 31%. These performance drops empricially justify our hypotheses **H1** and **H2**. Also, CLUTR with decoder finetuning and Shuffled VAE still outperform PAIRED, with 7.6X and 7.3X better returns, respectively.

Analysis of the Curriculum: CLUTR vs PAIRED

In Section 3.5 and 3.5 we discussed how CLUTR outperforms PAIRED, both in terms of sample efficiency and generalization, suggesting CLUTR induces a significantly more effective curriculum than PAIRED. For better understanding of CLUTR curriculum, in Figure 3.8 we analyze the mean regret—the performance gap between the agent and the adversary—on the teacher-generated curricula for both CarRacing and navigation tasks.

CLUTR and PAIRED show similar regret patterns, which is not surprising as both optimize regret using the same criteria. However, CLUTR converges to a smaller regret value; faster than PAIRED. From a curriculum learning perspective, smoother training is expected with tasks that are 'slightly' harder than the agent can already solve or, can obtain 'slightly' better returns. In practice, both the agent and the antagonist are trained in the same training data and context e.g., the same



(a) Mean Regret - CarRacing

(b) Mean Regret - Navigation

Figure 3.8: Mean standard regret during training. CLUTR shows a smaller regret value indicating a smaller performance gap between the agent and the antagonist, compared to PAIRED.

hyper-parameters, architecture, differing only by their random initial weights. Hence, a lower regret implies that the teacher is generating tasks at the frontier of the agents' capabilities, which are either slightly harder than the agent should be able to solve (because antagonist is solving them) or, the tasks in which antagonist is performing slightly better. On the other hand, higher regret values can result from generating tasks which are biased towards the strength or, idiosyncracy of only one of the agents, which might not be useful for generalization. In fact, PAIRED has shown to over exploit the relative strength of the antagonist for CarRacing ([45]), inducing curriculum showing high regret but poor generalization. Furthermore, a high regret can also imply the antagonist becoming significantly better than agent, which may lead to the teacher not having enough incentive to generate novel and diverse tasks, harming agent learning. Hence the lower regret value, might indicate that CLUTR is identifying the frontier of agents' capabilities better than PAIRED and thus inducing a more effective curriculum for training the student agents, as supported by the empirical performance.

Figure 3.6 shows snapshots of CLUTR and PAIRED generated curriculums as training progress. We notice, PAIRED generates over-simplified tasks for substantial amount of time, which might hamper its generalization and sample efficiency. On the other hand, CLUTR doesnt seem to start with overly-simplistic tasks, rather generates tasks with a wide range of difficulty throughout. Section 3.6 shares detailed analysis supporting the above observation and further insights.

3.6 Additional Details of CLUTR

CLUTR Objective Derivation

We use a hierarchical graphical model to formulate the latent environment design problem. Let's assume that R is a random variable that denotes a measure of success defined using the protagonist and antagonist agents and z be a latent random variable. We use the graphical model in Figure 3.1 where z generates an environment E and R is the success defined over E. Both E and R are observed variables while z is an unobserved variable. R covers a broad range of measures used in different UED methods including PAIRED and DR (Domain Randomization). In PAIRED, R represents the REGRET as the difference of returns between the antagonist and protagonist agents and it depends on the environments that the agents are evaluated on.

We use a variational formulation of UED by using the above graphical model. We first define the variational objective as the KL-divergence between an approximate posterior distribution and true posterior distribution over latent variable z,

$$\begin{aligned} D_{KL}(q(z)||p(z|R,E)) &= E_{z\sim q(z)}[\log q(z)] - E_{z\sim q(z)}[\log p(z|R,E)] \\ &= E_{z\sim q(z)}[\log q(z)] - E_{z\sim q(z)}[\log p(R,E,z)] + \log p(R,E) \end{aligned}$$

where both R and E are given.

Next, we write the ELBO,

$$
\begin{aligned}
ELBO &= E_{z\sim q(z)}[\log p(R, E, z)] - E_{z\sim q(z)}[\log q(z)] \\
&= E_{z\sim q(z)}[\log p(R|E)p(E|z)p(z)] - E_{z\sim q(z)}[\log \ q(z)] \\
&= E_{z\sim q(z)}[\log p(R|E)] + E_{z\sim q(z)}[\log p(E|z)] + E_{z\sim q(z)}[\log p(z)] - E_{z\sim q(z)}[\log q(z)] \\
&= \log p(R|E) + E_{z\sim q(z)}[\log p(E|z)] - E_{z\sim q(z)}[\log \frac{q(z)}{p(z)}] \\
&= \log p(R|E) + E_{z\sim q(z)}[\log p(E|z)] - D_{KL}(q(z)||p(z)) \\
&= \log p(R|E) - \text{VAE}(z, E)
\end{aligned}
$$

We can also induce an objective that includes minimax REGRET. Let R be distributed according to an exponential distribution, $p(R|E) \propto exp(\text{REGRET}(\pi_P, \pi_A|E))$,

we derive,

$$
ELBO \approx \text{REGRET}(R, E) - \text{VAE}(z, E)
$$

where the normalizing factor is ignored.

Robustness Guarantees

CLUTR essentially proposes including a pretrained latent space within the teacher/generator. From the teacher's perspective, the difference is while the PAIRED teacher starts from randomly initialized weights, CLUTR starts from the pretrained weights. Thus, CLUTR does not impose new assumptions on possible teacher policies. Furthermore, CLUTR does not change any other specifics of the underlying PAIRED algorithm. Hence, CLUTR holds the same theoretical robustness guarantees provided by PAIRED.

In practice, both CLUTR and PAIRED deviate from these theoretical guarantees. For example, both algorithms approximate the regret value, which is the case for other regret-based UEDs such as Robust PLR and REPAIRED ([45]). Also, the robustness guarantee depends on reaching the Nash equilibrium of the multiagent adversarial game. However, gradient-based multi-agent RL has no convergence guarantees and often fails to converge in practice([59]). We also note that, by introducing the latent space, CLUTR VAE might not have access to the full task space due to practical limitations on training, e.g., the training dataset not having all possible tasks. However, when the decoder is allowed to be finetuned, CLUTR will have access to the full task space, similar to PAIRED. Our empirical results (discussed in Section 3.5) suggest that keeping the pretrained decoder fixed performs better than finetuning it, so we kept it fixed for our main experiments. We also want to mention, when the flexible objective is used, CLUTR (and PAIRED) does not hold the robustness guarantee as it changes the dynamics of the underlying game between the teacher and the agents, even though flexible regret works better in practice.

Training Details

Environment Details

Car Racing: The CarRacing environment was originally proposed by OpenAI Gym [9], and later has been reparameterized by [45] with Bézier Curves([64]) for UED algorithms. This environment requires the agents to drive a full lap around a closed-loop track. The track is defined by a Bézier Curve modeled with a sequence of upto 12 arbitrary control points, each spaced within a fixed radius $B/2$ of the center of the $B \times B$ field. This sequence of control points can uniquely identify a track, subject to a set of predefined curvature constraints [45]. The control points are encoded in a 10×10 grid—a discrete downsampled version of the racing track field. Each control point hence is a integer denoting a cell of the grid and the cell coordinates are upscaled to match the original scale of the field afterwards. This ensures no two control points are too close together, preventing areas of excessive track overlapping. The track consists of a sequence of L polygons and the agent receives a reward of $1000/L$ upon visiting each unvisited polygon and a penalty of -0.1 at each time step to incentivize completing the tracks faster. Episodes terminate if the agent drives too far off-track but is not given any additional penalty. The agent controls a 3 dimensional continuous action space corresponding to the car's steer: torque $\in [-1.0, 1.0]$, gas: acceleration $\in [0, 0, 1.0]$, and brake: deceleration $\in [0.0, 1.0]$. Each action is repeated 8 times. The agent receive a $96 \times 96 \times 3$ RGB pixel observation. The top 84×96 portion of the frame contains a clipped, egocentric, bird's eye view of the horizontally centered car. The bottom 12×96 segment simulates a dashboard visualizing the agent's latest action and return. Snapshots of the test track in the F1 benchmark are shown in Figure 3.9.

Minigrid: The environment is partially observable and based on [10] and adopted for UED by [19]. Each navigation task is represented with a sequence of integers denoting the locations of the obstacles, the goal, and the starting position of the agent: on a 15×15 grid similar to [19]. The grids are surrounded by walls on the sides, making it essentially a 13×13 grid. [19] parameterizes the locations using integers. Each task is a sequence of 52 integers, while the first 50 numbers denote the location of obstacles followed by the goal and the agent's initial location. The sequences may contain duplicates to allow the generation of navigation tasks with fewer than 50 obstacles. Snapshots of the test grids used in our paper are shown in Figure 3.10.

Network Architectures

All the student and teacher agents are trained with PPO [75].

Student Architecture: For CarRacing, we use the same student architecture as [45]. The architecture consists an image embedding module composed of 2D Convolutions with square kernels of sizes 2,2,2,2,3,3, stride lengths 2,2,2,2,1,1 and channel outputs of 8, 16, 64, 128, 256 stacked together. The image embedding is of size 256 and is passed through a Fully Connected (FC) layer of 100 hidden units and then passed through ReLU activations. This embedding is then passed through two FC with 100 hidden neurons, and then a softplus layer, and finally added to 1 for the beta distribution used for the continuous action space. Further details can be found in [45].

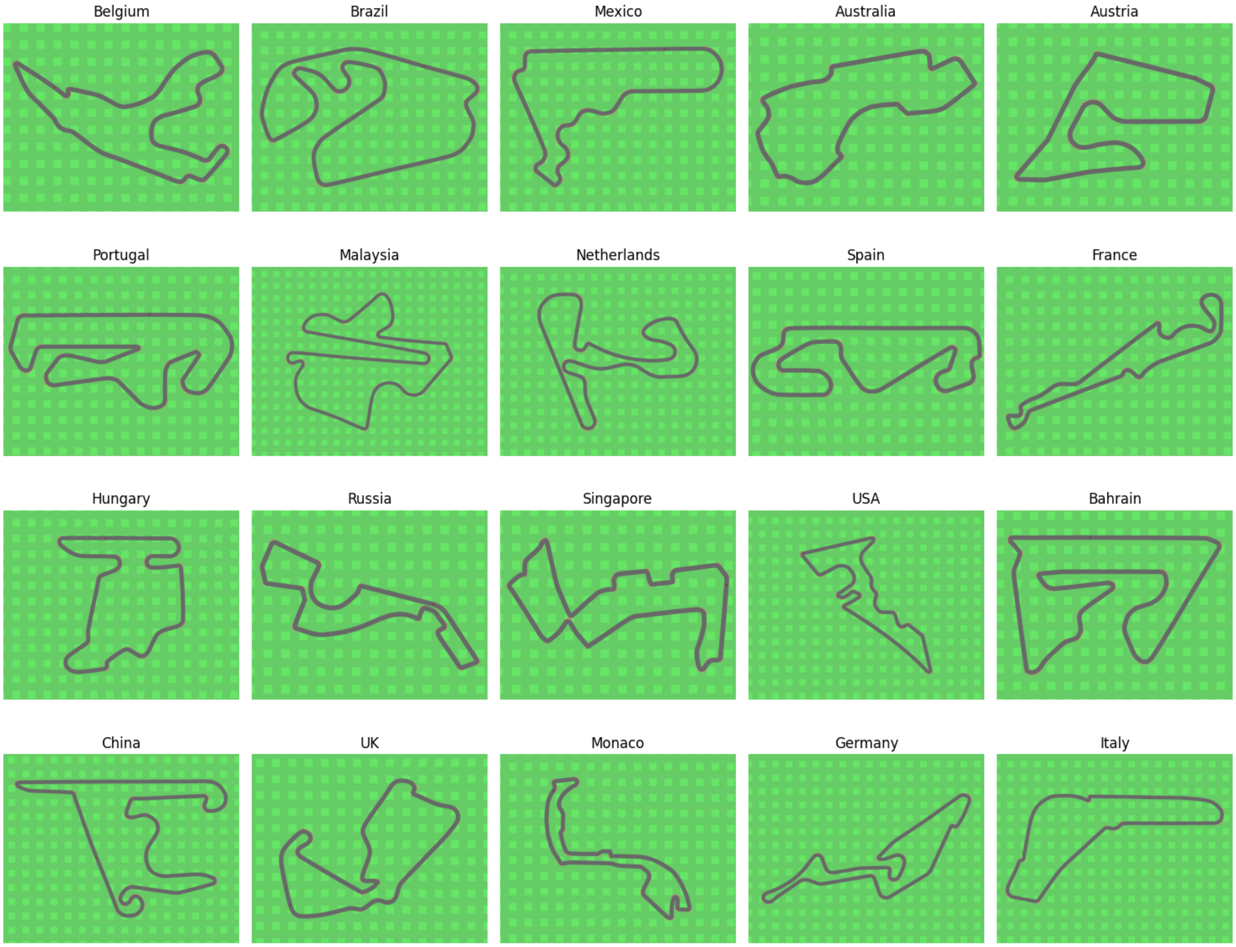


Figure 3.9: Snapshots of the test tracks in F1 benchmark

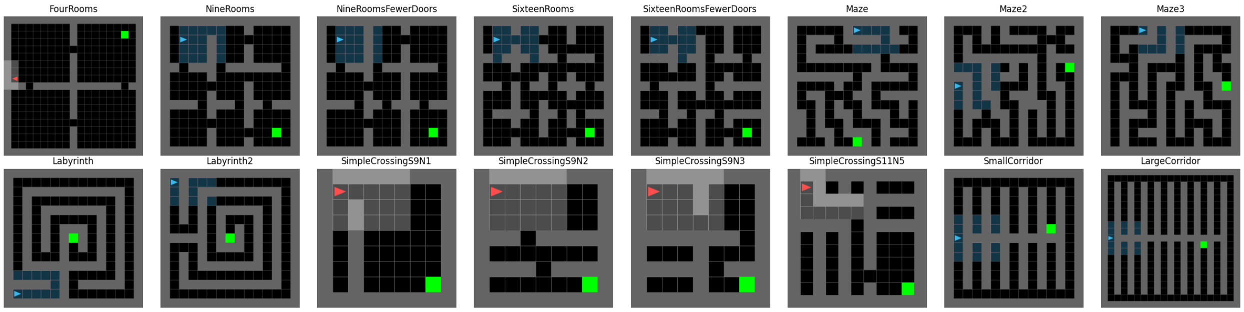


Figure 3.10: Snapshots of the test grids for MiniGrid

For navigation tasks, we use the same student architecture as [19]. The observation is a tuple with a $5 \times 5 \times 3$ grid observation and a direction integer in $[0 - 3]$. The grid view is fed to a convolutional layer with kernels of size 3 with 16 filters and the direction integer is passed through a FC with 5 units. This is followed by an LSTM of size 256, and then to two FC layers with 32 units, which connect to the policy outputs. The value network uses the same architecture.

Teacher Architecture: For CarRacing, CLUTR teacher takes a random noise and generates a continuous vector, i.e., the latent task vector. We pass the random noise through a feed-forward network with one hidden layer of 8 neurons as the teacher. The output of this layer is fed through two separate fully-connected layers, each with a hidden size of 8 and an output dimension equal to the latent space dimension, followed by soft plus activations. We then add 1 to each component of these two output vectors, which serve as the α and β parameters respectively for the Beta distributions used to sample each latent dimension. In all of our experiments, we used a 64-dimensional latent task space.

For Minigrid experiments with flexible regret objective, we use a similar architecture as CarRacing described above, except the hidden layer consists of 10 neurons, instead of eight. For Minigrid experiments with standard regret objective (which is discussed later in Section 3.6), we use the network architecture used in [19] but only take a noise input. As this adversary network generates discrete actions, we scale them to real numbers before feeding into the VAE decoder.

VAE architecture: We use the architecture proposed in [7]. We use a word-embedding layer of size 300 with random initialization. The encoder comprises a conditional 'Highway' network followed by an LSTM. The Highway network is a two-staged network stacked on top of each other. Each stage computes $\sigma(x) \odot f(G(x)) + (1 - \sigma(x)) \odot Q(x)$, where x is the inputs to each of the highway network stages, G and Q is affine transformation, $\sigma(x)$ is a sigmoid non-linearization, and $\odot$ is element-wise multiplication. G and Q are feed-forward networks with a single hidden layer with equal input and output dimensions of 300, equal to the word-embedding output dimension. We use ReLU activation as f. The highway network is followed by a bidirectional LSTM with a single layer of 600 units. The LSTM outputs are passed through linear layer of dimension 64 to get the VAE mean and log variance. The mean vectors are passed through a hyperbolic tangent activation. For CarRacing (both Flexible and Standard Objective experiments) and navigation (only Standard Objective) tasks the output of the hyperbolic tangent activation is linearly scaled in $[-4, 4]$. No such scaling is done for the MiniGrid experiments with Flexible Regret Objective. The decoder takes in latent vectors of dimension 64 and passes through a bidirectional LSTM with two hidden layers of size 800 and follows it by a linear layer with size equaling the parameter vector dimension.

Hyperparameters

All our agents are trained with PPO [75]. We did not perform any hyperparameter search for our experiments. The CarRacing experiments used the same parameters used in [45] and the Minigrid experiments used the parameters from [19]. The VAE used for CarRacing and Minigrid standard objective experiments (Section 3.6) were trained using the default parameters from [7]. For the VAE used in the Minigrid flexible objective experiments, which we presented in the main text of the paper, we used a reconstruction weight of 1000 and ran the training for 10M steps to incorporate the larger dataset. The detailed parameters are listed in Table 3.2 and Table 3.3.

The flexible objective blurs the distinction between the agent and the antagonist. Hence, we designate the agent achieving the higher average training return during the last 10 steps as the primary student agent and the other one as antagonist.

Parameter	Value
Batch Size	32
Number of Training Steps	1000000
Reconstruction Weight	79
Latent Variable Size	64
Word Embedding size	300
Maximum Sequence Length	52
Encoder Activation	Hyperbolic Tangent
Learning Rate	0.00005
Dropout	0.3

Table 3.2: Hyperparameters for training the Task VAE

Parameter	CarRacing	MiniGrid
γ	0.99	0.995
λ_{GAE}	0.9	0.95
PPO rollout length	125	256
PPO epochs	8	5
PPO minibatches per epoch	4	1
PPO clip range	0.2	0.2
PPO number of workers	16	32
Adam learning rate	3e-4	1e-4
Adam ϵ	1e-5	1e-5
PPO max gradient norm	0.5	0.5
PPO value clipping	no	yes
Return normalization	yes	no
Value loss coefficient	0.5	0.5
Student entropy coefficient	0	0
Action Repeat	8	-

Table 3.3: Hyperparameters for PAIRED and CLUTR PPO training.

VAE Training Data

For CarRacing, we follow the same parameterization as [45]: each track is defined with a sequence of up to 12 integers denoting control points of a Bézier Curve. . Each control point is represented with an integer. We generate 1M random sorted integer sequences of fixed length 12 with duplicates—which enables generating tracks defined with less than 12 control points.

For navigation tasks we use the parameterization of [19], generating upto 50 obstacles for each task for a 15×15 grid, surrounded by walls, effectively an active area of 13×13. Hence, each location is numbered in 1 to 169. Every number except the last two of the sequence represent obstacle locations, and the last two represent the goal and agent location, respectively. To generate training data, we uniformly generate 1M and 10M sequences of variable length between 2 and 52 (inclusive), for the standard and flexible regret objective, respectively. We note that, the obstacle locations, though represented as a sequence, essentially is a set. The parameter vector is thus partially permutation invariant. As we discussed in 3.4, due to this permutation invariance, conteporary adaptive-teacher UEDs, e.g., PAIRED and REPAIRED, face combinatorial explosion. CLUTR addresses this by sorting the obstacle locations of this parameter-vector dataset.

Details on Compute Resources

We have conducted our experiments in cloud machines from Amazon EC2 - Secure Cloud Services (https://aws.amazon.com/) and Google Cloud Platform (GCP) - Google Cloud (https://cloud.google.com/). We used a single NVIDIA T4 GPUs for our experiments with machines having 8(16) and 16(32) physical(virtual) cores, 64GB and 128 GB Memory for CarRacing and Minigrid experiments. A typical 500M Minigrid training of CLUTR ran with a speed of around 800-900 environment interactions per second, taking around 6-8 days, with 32 parallel workers. CarRacing experiments ran on around 90-110 environment interactions per second with 16 parallel processes.

Detailed Experimental results on CarRacing

Detailed Comparison on Full F1 dataset

Figure 3.11 and Table 3.4, compares CLUTR with contemporary random-generator UED methods, REPAIRED, and the attention based SOTA. It is to be noted that, CLUTR and PAIRED with flexible regret objective was trained for 2M timesteps. All the other UED methods, along with CLUTR and PAIRED with standard regret was trained for 5M timesteps.

We notice that, each of the random-teacher UEDs outperform all the other adaptive-teacher UEDs, with the exception of CLUTR with flexible regret objective. PAIRED performs miserably in its basic form, however performs significantly better when coupled with extended capabilties e.g., by using flexible regret objective, or by introducing replay and stop-gradient capabilities (i.e., REPAIRED). However, they they still fall short to the random-teacher UEDs. This indicates that adaptive-teacher UEDs face significant difficulty in this domain.

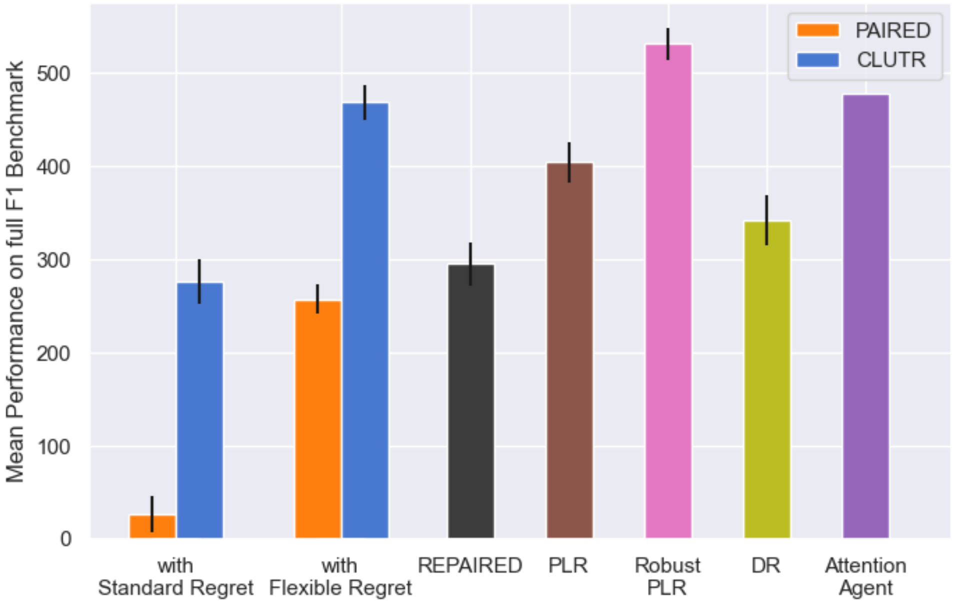


Figure 3.11: Comparison on the F1 Benchmark comprising 20 tracks modeled on real-life F1 racing tracks. CLUTR (with flexible regret) emerges as the best adaptive-teacher UED for CarRacing and being the only adaptive-teacher UED to outperform some of the random-generator UEDs. Each of the other adaptive-teacher UEDs (REPAIRED, PAIRED with flexible regret, CLUTR with standard regret) are outperformed by all of the random-generator UEDs (DR, PLR, Robust PLR). CLUTR outperforms the adaptive-teacher PAIRED and REPAIRED by 82% and 58%, respectively, while outperforming Domain Randomization and PLR, by 38% and 16%, repectively. It only falls short to Robust PLR by 14%. The results show mean and standard error of 10 independent runs.

CLUTR with flexible regret emerges as the best adaptive-teacher UED despite being trained only for 2M timesteps. It achieves an impressive 18X higher zero-shot generalization than PAIRED with standard regret and outperforms REPAIRED by 58%.

CLUTR with flexible regret is the only adaptive-teacher UED to outperform other random-teacher UEDs. CLUTR outperforms Domain Randomization and PLR, by 38% and 16%, repectively. It only falls short to Robust PLR by 14%. Nonetheless, CLUTR shows competitive results compared to Robust PLR, showing comparable results in seven out of the 20 test tracks and outperforming in the Netherlands track. CLUTR also outperforms the non-UED SOTA on the full F1 dataset. It outperforms the Attention Agent on nine out of the 20 tracks and shows comparable performance in another one.

Figure 3.12 compares how different UEDs perform during training by periodically evaluating them on three tracks from the F1 benchmark: Singapore, Germany, and Italy. CLUTR (with flexible regret) shows better generalization and sample efficiency than all the other UEDs, except

Track	DR	PLR	Robust PLR	REPAIRED	PAIRED	CLUTR	PAIRED	CLUTR	Attention
					Standard Regret		Flexible Regret (2M)		Agent
Australia	484 ± 29	545 ± 23	**692 ± 15**	414 ± 27	100 ± 22	429 ± 28	342 ± 29	**683 ± 20**	826
Austria	409 ± 21	442 ± 18	**615 ± 13**	345 ± 19	92 ± 24	309 ± 19	316 ± 23	507 ± 19	*511*
Bahrain	298 ± 27	411 ± 22	**590 ± 15**	295 ± 23	-35 ± 19	225 ± 24	183 ± 28	414 ± 20	*372*
Belgium	328 ± 16	327 ± 15	**474 ± 12**	293 ± 19	72 ± 20	315 ± 14	309 ± 17	429 ± 15	668
Brazil	309 ± 23	387 ± 17	**455 ± 13**	256 ± 19	76 ± 18	244 ± 16	237 ± 16	363 ± 18	*145*
China	115 ± 24	84 ± 20	**228 ± 24**	7 ± 18	-101 ± 9	33 ± 19	23 ± 21	**254 ± 28**	344
France	279 ± 32	290 ± 35	**478 ± 22**	240 ± 29	-81 ± 13	266 ± 30	158 ± 24	**498 ± 31**	*153*
Germany	274 ± 23	388 ± 20	**499 ± 18**	272 ± 22	-33 ± 16	195 ± 26	286 ± 26	404 ± 20	*214*
Hungary	465 ± 32	533 ± 26	**708 ± 17**	414 ± 29	98 ± 29	325 ± 32	327 ± 31	630 ± 24	769
Italy	461 ± 27	588 ± 20	**625 ± 12**	371 ± 25	132 ± 24	439 ± 31	451 ± 27	**639 ± 16**	798
Malaysia	236 ± 25	283 ± 20	**400 ± 18**	200 ± 17	-26 ± 17	174 ± 23	192 ± 21	**426 ± 22**	*300*
Mexico	458 ± 33	561 ± 21	**712 ± 12**	415 ± 30	67 ± 31	387 ± 31	391 ± 30	627 ± 19	*580*
Monaco	268 ± 28	360 ± 32	**486 ± 19**	256 ± 26	-28 ± 18	234 ± 30	125 ± 28	**460 ± 29**	835
Netherlands	328 ± 26	418 ± 21	419 ± 25	307 ± 21	70 ± 20	302 ± 27	306 ± 24	**488 ± 21**	*131*
Portugal	324 ± 27	407 ± 15	**483 ± 13**	265 ± 21	-49 ± 13	299 ± 24	149 ± 19	**462 ± 20**	606
Russia	382 ± 30	479 ± 24	**649 ± 14**	419 ± 25	51 ± 21	319 ± 25	337 ± 24	497 ± 23	732
Singapore	336 ± 29	386 ± 22	**566 ± 15**	274 ± 21	-35 ± 14	229 ± 18	192 ± 21	382 ± 19	*276*
Spain	433 ± 24	482 ± 17	**622 ± 14**	358 ± 24	134 ± 24	373 ± 15	414 ± 19	496 ± 15	759
UK	393 ± 28	456 ± 16	**538 ± 17**	380 ± 22	138 ± 25	396 ± 18	339 ± 18	471 ± 19	729
USA	263 ± 31	243 ± 28	**381 ± 33**	120 ± 25	-119 ± 11	27 ± 29	67 ± 29	238 ± 31	*-192*
Mean	342 ± 27	404 ± 22	**531 ± 17**	295 ± 23	26 ± 19	276 ± 24	257 ± 16	468 ± 21	478

Table 3.4: Comparison between CLUTR and other UED algorithms on each track of the F1 benchmark. We note that CLUTR and PAIRED with flexible regret were trained for 2M timesteps. All the other UEDs were run for 5M timesteps. The boldface denotes SOTA among UED algorithms. Italic in the Attention Agent column means it is comparable/outperformed by our best-performing model CLUTR with Flexible Regret. CLUTR also outperformed the non-UED SOTA on 9 out of the 20 tracks. CLUTR outperforms PAIRED, Domain Randomization, PLR, and REPAIRED and only falls short of Robust PLR.

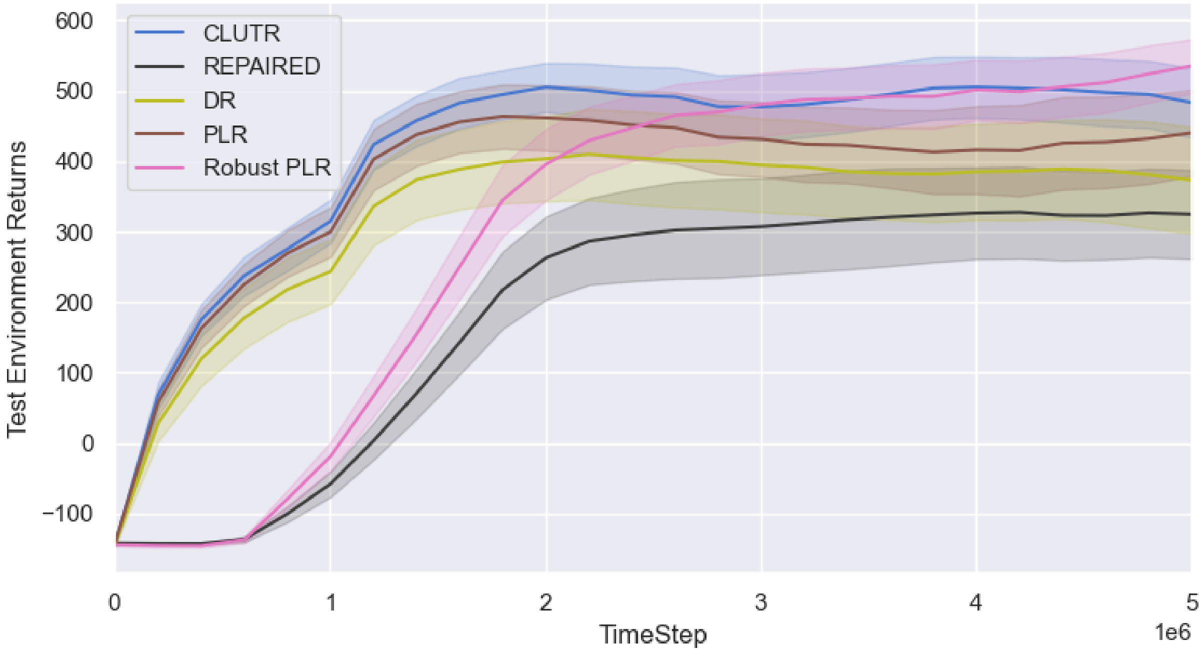


Figure 3.12: Comparison of mean agent returns on three tracks: Singapore, Germany, and Italy. Based on this subset of tracks, CLUTR (with flexible regret) shows better generalization than all the other UEDs, except Robust PLR. CLUTR was ahead of Robust PLR till around 3M timesteps, followed by both curves following each other closely, and near the very end Robust PLR surpassed CLUTR.

Robust PLR. CLUTR showed better performance than Robust PLR till alomost 3M timesteps, after that CLUTR and Robust PLR curves followed each other closely, and near the very end Robust PLR surpasses CLUTR.

CLUTR with flexible regret loss

Training Returns: Figure 3.13 plot mean return on the training tasks for both the student agents. CLUTR student agents show close performance, while PAIRED students show a bigger gap of performance between them. Closely competing agents can indicate the training tasks being slightly harder than the agents can currently solve.

Learning task manifold and curriculum: Joint vs Two-staged Optimization: In Section 3.5, we empirically justified our hypothesis that learning the task representation and the curriculum simultaneously results in a difficult learning problem due to the non-stationarity of the process—using the standard regret objective. In this section we repeat the experiment with the flexible regret objective. In Figure 3.14, we see a 10% drop in the performance when the decoder was allowed to finetune with regret loss, further justifying our hypothesis. As a side note, the

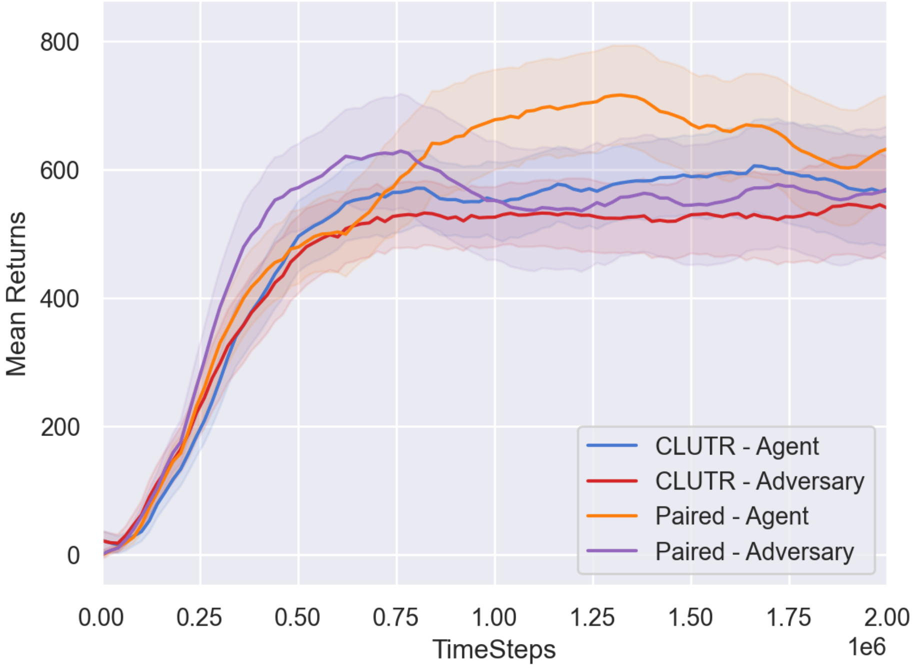


Figure 3.13: Mean return on the training tasks for both the student agents. CLUTR student agents show close performance, while PAIRED students show a bigger gap of performance between them. Closely competing agents can indicate the training tasks being slightly harder than the agents can currently solve, resulting in a smoother curriculum

smaller drop compared to standard regret objective indicates that flexible objective mitigates some of the instability problem too. Finally, even with decoder finetuning, CLUTR achieves a 65% improvement over PAIRED indicating the benefits of pretrained decoupled latent task space.

CLUTR with standard regret loss

We train CLUTR with the standard regret loss for 5M timesteps. Figure 3.15 compares the impact of standard/flexible regret loss on the regret and agent returns during training. With standard regret loss, CLUTR shows a lower regret value, but shows similar pattern. The CLUTR agent achieves better returns with flexible loss throughout the training.

Figure 3.16 compares the mean regret and agent training returns with PAIRED. CLUTR with standard loss shows much lower regret than PAIRED (Figure 3.16a). Figure 3.16b shows that the CLUTR agents compete closely, while PAIRED antagonist achieves much higher returns than the PAIRED agent which leads to higher regret returns for the teacher agent but results in a weak student agent. To test the Zero-shot generalization, we evaluate CLUTR with the standard loss on the full F1 benchmark. Figure 3.17 shows CLUTR with standard regret loss outperforms PAIRED in all the 20 test tracks. This implies that CLUTR outperforms PAIRED irrespective of the choice

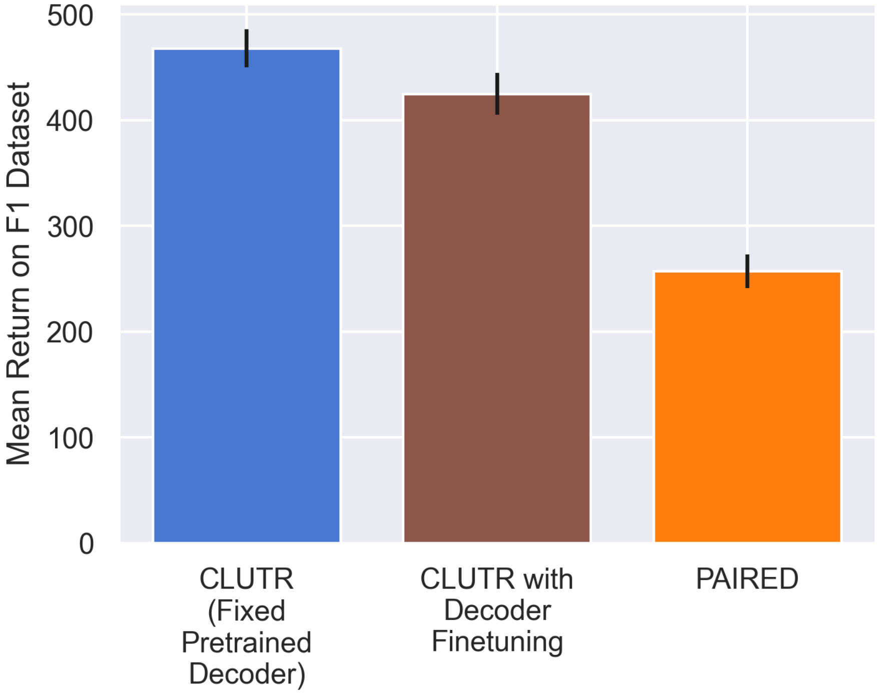
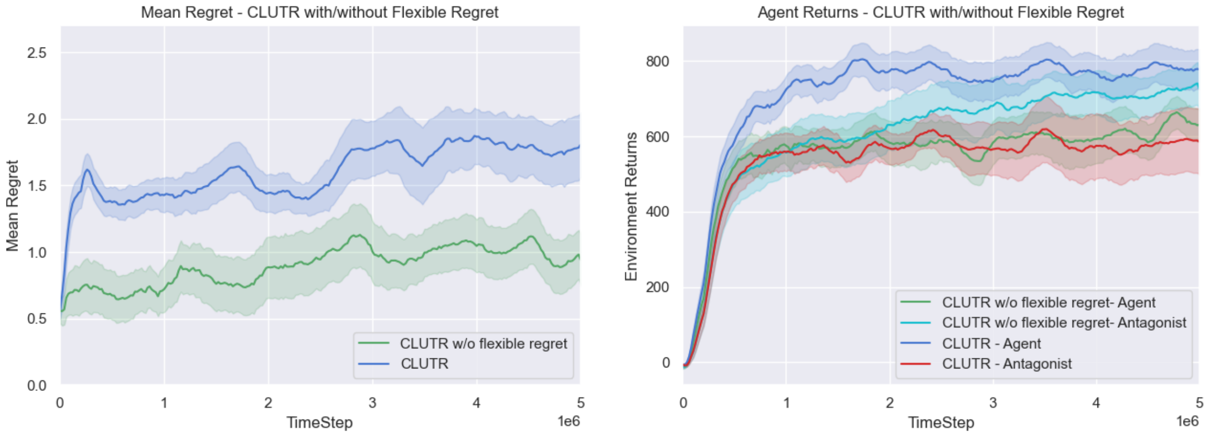


Figure 3.14: Impact of joint vs two-staged optimization of the task manifold. The leftmost column shows the default CLUTR performance—i.e., using a pretrained decoder kept fixed during the curriculum learning phase—with flexible regret objective in the CarRacing domain. Decoder finetuning, i.e., when the decoder is allowed to finetune with the regret loss, results in a 10% performance drop. This performance drop empricially justify our choice of using a pretrained and fixed VAE to solve learning instability.

of the loss function (standard/flexible). Figure 3.18 compares the sample efficiency of CLUTR with the standard regret loss with PAIRED by evaluating the agents on four selected tracks (Vanilla, Singapore, Germany, Italy) during training. It can be seen that CLUTR, even without the regret loss, outperforms PAIRED significantly. We note that these test environments were not used in any way, neither during training CLUTR (and PAIRED) nor while designing it.

As mentioned in [45] PAIRED overexploits the relative strengths of the antagonist over the protagonist agent and generates a curriculum that gradually reduces the task complexity. However, CLUTR overcomes this and generates a curriculum where the agent and the antagonist closely compete (Figure 3.16b) and shows a robust generalization on the unseen F1 benchmark.



(a) Mean Regret - Car Racing - with vs without flexible regret loss

(b) Returns on UED generated Car Racing tracks - with vs without flexible regret loss

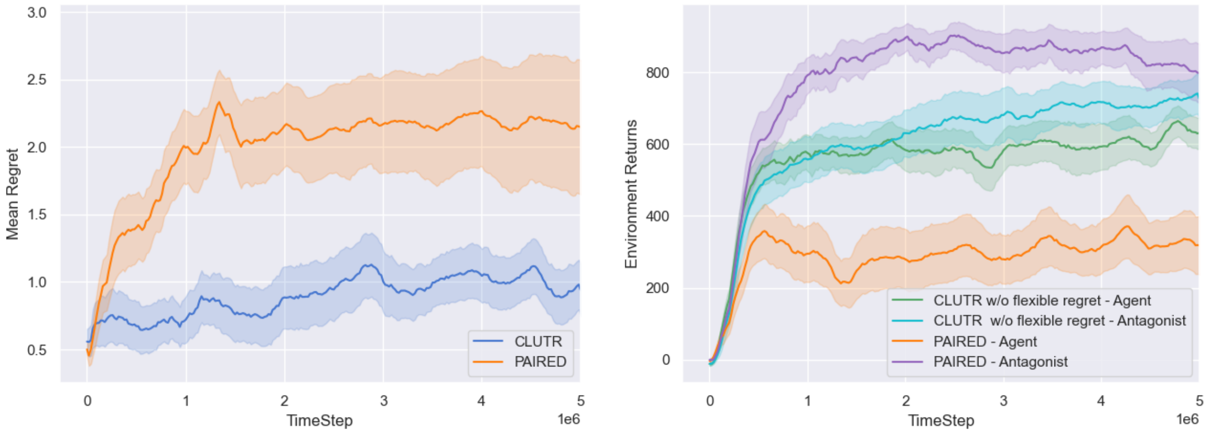
Figure 3.15: Mean Regret and agent returns during training CLUTR (with flexible regret) vs CLUTR with standard PAIRED regret approximation.

Extended Analysis on Impact of sorting training data for VAE training

The non-sorted dataset was generated by shuffling each track of the original VAE training dataset 10 different times, resulting in a 10X bigger dataset (10M tracks). It was trained for 5X longer for 5M training steps. We planned on training for 10M gradient steps (10X than the original VAE) but stopped at 5M as it converged much sooner. We ran both CLUTR and CLUTR-shuffled, i.e., CLUTR with a VAE trained on non-sorted data up to 5M timesteps. CLUTR-shuffled shows inferior performance and also signs of unlearning compared to CLUTR. Figure 3.19 shows detailed experiment results.

Impact of Task Representation Learning

In this section, we discuss the impact of the learned task representation on performance. In Section 3.5, we showed that if we finetune the VAE decoder during curriculum learning, the overall performance drops significantly (Figure 3.7). To get a better understanding, in Figure 3.20, we plot how much the performance deviates as the VAE decoder changes during the training process. The curve in red shows the deviation of the decoder from its pretrained weights as it is fine-tuned during the training. We estimate the deviation as the L2 distance between the finetuned and the pretrained decoder weights. The green curve shows the performance drop from CLUTR (with standard loss). To estimate the performance drop, we periodically evaluate both CLUTR and CLUTR with Finetuned VAE, on the selected test tracks during training. From the figure, we observe that, as the decoder weights are finetuned, they become increasingly different from the initial pretrained



(a) Mean Regret - Car Racing

(b) Returns on UED generated Car Racing tracks

Figure 3.16: Mean Regret and agent returns during training CLUTR with standard PAIRED regret loss (i.e., without the flexible regret). CLUTR shows a smaller regret value(i.e., closely competing agent and antagonist), indicating a better UED curriculum.

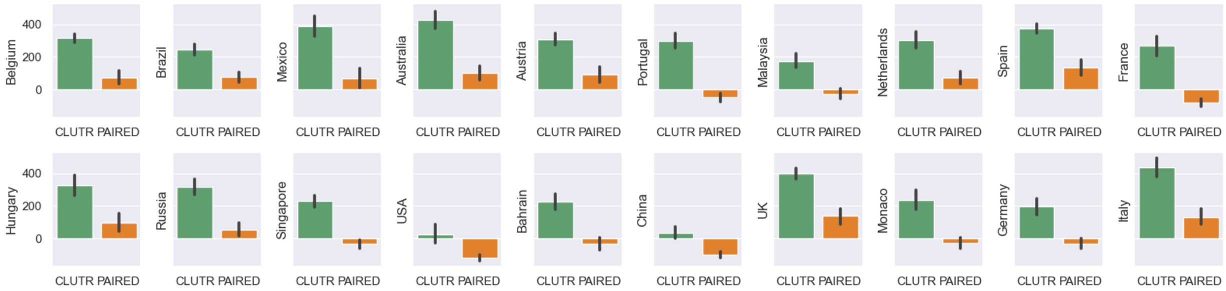


Figure 3.17: Zero-shot generalization of both PAIRED and CLUTR (with the standard regret loss) agents after 5M timesteps on the full F1 benchmark. CLUTR with the standard regret loss outperforms PAIRED on every track. For each track, we test the agents on 10 different episodes and the error bar denotes the standard error.

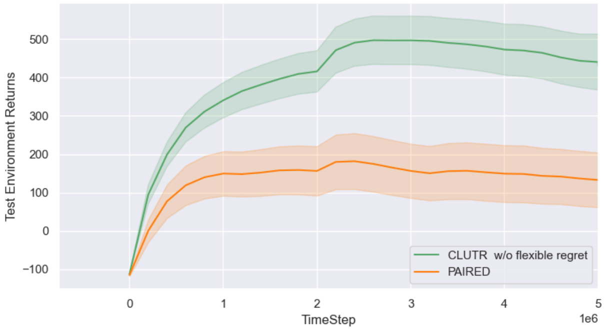
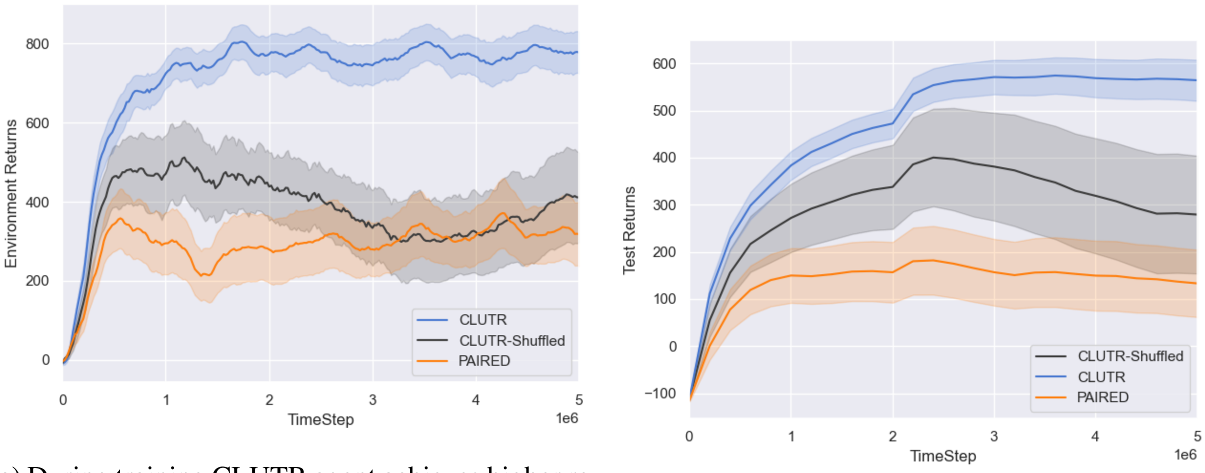


Figure 3.18: Test Returns on Selected Tracks (Vanilla, Singapore, Germany, and Italy) of CLUTR with standard PAIRED regret loss alongside PAIRED performance.



(a) During training CLUTR agent achieves higher returns while, CLUTR-shuffled agent shows lower returns. CLUTR-Shuffled agent's return is also less stable showing a decrease and increase.

(b) CLUTR achieves higher and more stable mean returns on the selected tracks. CLUTR-Shuffle shows signs of unlearning.

Figure 3.19: Analysis of sorting training data for VAE. Trained on shuffled data, CLUTR-Shuffled performs inferior compared to CLUTR and shows signs of unlearning.

weights. At the same time, the overall performance gap from CLUTR also increases. This suggests that the pretrained VAE weights are crucial for better performance.

Furthermore, the quality of the learned representation depends on the quality of the data they are trained on. In section 3.5, we showed that a VAE trained on a non-sorted dataset significantly deteriorates the performance (Figure 3.7). This further suggests that the learned representation has a significant impact on performance. We also want to note that both of these variations (CLUTR with Finetuned VAE and the CLUTR with Shuffled VAE) perform much better than PAIRED,

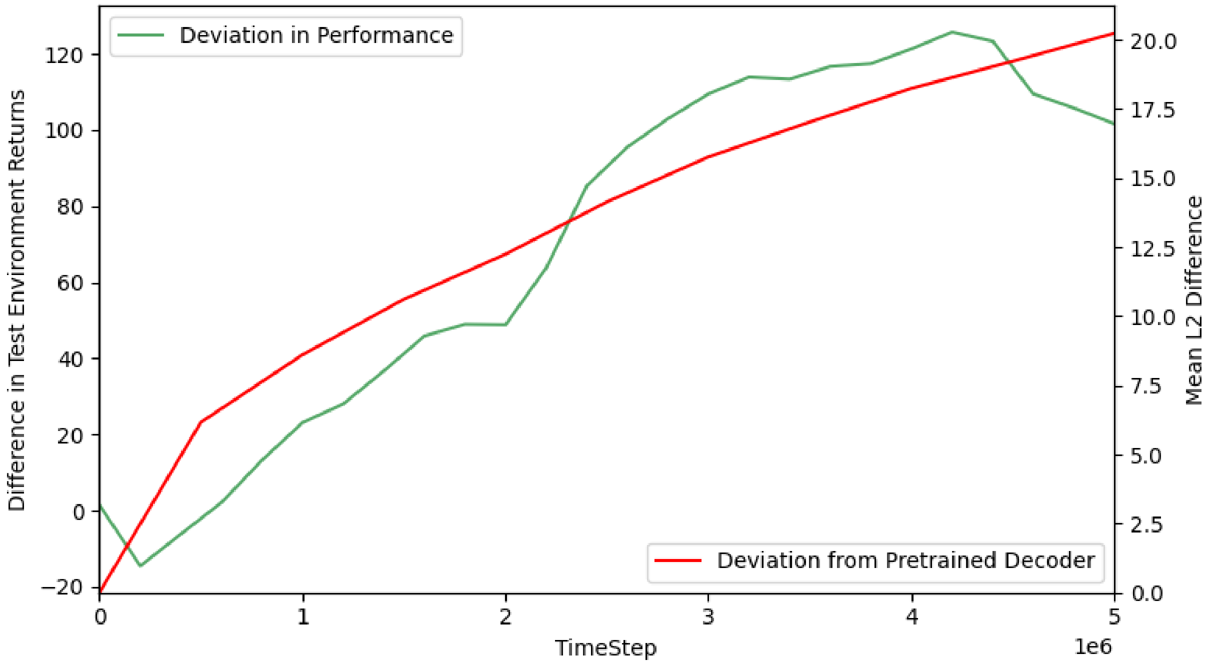


Figure 3.20: Impact of pretrained decoder weights on performance. The red curve plots the deviation of the decoder from its pretrained weights as it is finetuned. The green curve shows the performance drop from CLUTR with the standard loss. These curves suggest that pretrained weights are crucial for performance.

which suggests that, though CLUTR's performance depends on the representation, with a reasonable representation, it can still perform better than PAIRED.

Detailed Experimental results on MiniGrid

CLUTR with flexible regret objective

To train the CLUTR VAE, we generated 10 million random grids, with the obstacle locations sorted, and the number of obstacles uniformly varying from zero to 50, aligning with [19]. We train both CLUTR and PAIRED using the flexible regret objectives.

Figure 3.21 shows zero-shot generalization performance of CLUTR and PAIRED on the 16 unseen navigation tasks from [19], in terms of the percent of environments the agent solved, i.e., solved rate. CLUTR achieves a 1.35X better generalization solving 58% of the unseen grids, than PAIRED which solves 43% of the unseen grids. It can also be seen that CLUTR outperforms PAIRED on 13 out of the 16 test navigation tasks.

Figure 3.22 compared the mean peforamnce of CLUTR, PAIRED, and REPAIRED. REPAIRED outperforms both PAIRED and CLUTR. We note that, REPAIRED and CLUTR are both

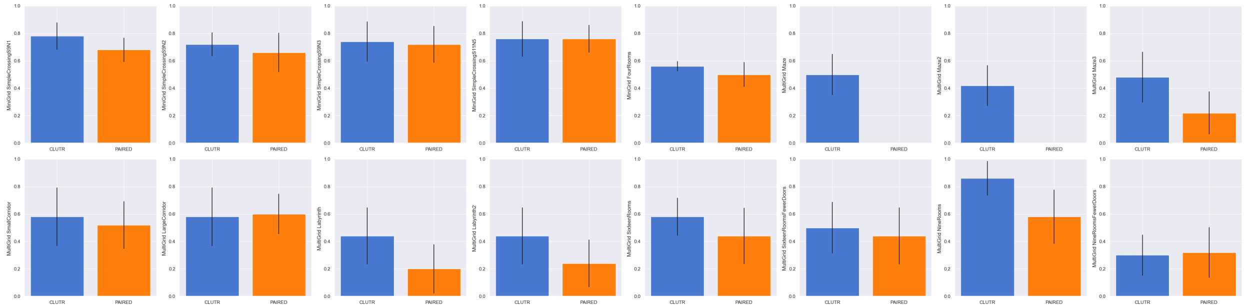


Figure 3.21: Zero-shot generalization of CLUTR and PAIRED, in terms of percent of the environments solved. CLUTR achieves a higher solved rate than PAIRED in 13 out of the 16 tasks. We evaluate the agents with 10 independent episodes on each task. Error bars denote the standard error.

improvement towards PAIRED. However, REPAIRED involves a dual-curriculum methods, with two different teachers adopting replay capabilities with disabling exploratory gradients. On the other hand CLUTR is a much simpler method, and can also be augmented with REPAIRED too.

Training Returns: Figure 3.23 plot mean return on the training tasks for both the student agents. CLUTR student agents show close performance, while PAIRED students show a bigger gap of performance between them.

CLUTR with standard regret objective

Training Returns: Figure 3.24 plot mean return on the training tasks for both the student agents. CLUTR student agents show close performance, while PAIRED students show a bigger gap of performance between them initiallly at the beginning.

Performance: Figure 3.26 shows zero-shot generalization performance of CLUTR and PAIRED on 16 unseen navigation tasks from [19] based on the percent of environments the agent solved, i.e., solved rate. CLUTR achieves superior generalization solving 64% of the unseen grids, a 45.45% improvement over PAIRED, which achieves a 44% solve rate. From figure 3.26 it can be seen CLUTR outperforms PAIRED achieving a higher mean solve rate on 14 out of the 16 unseen navigation tasks. Figure 3.25 shows solved rates on four selected grids (Sixteen Rooms, Sixteen Rooms with Fewer Doors, Labyrinth, and Large Corridor) during training. CLUTR shows better sample efficiency, as well as generalization than PAIRED.

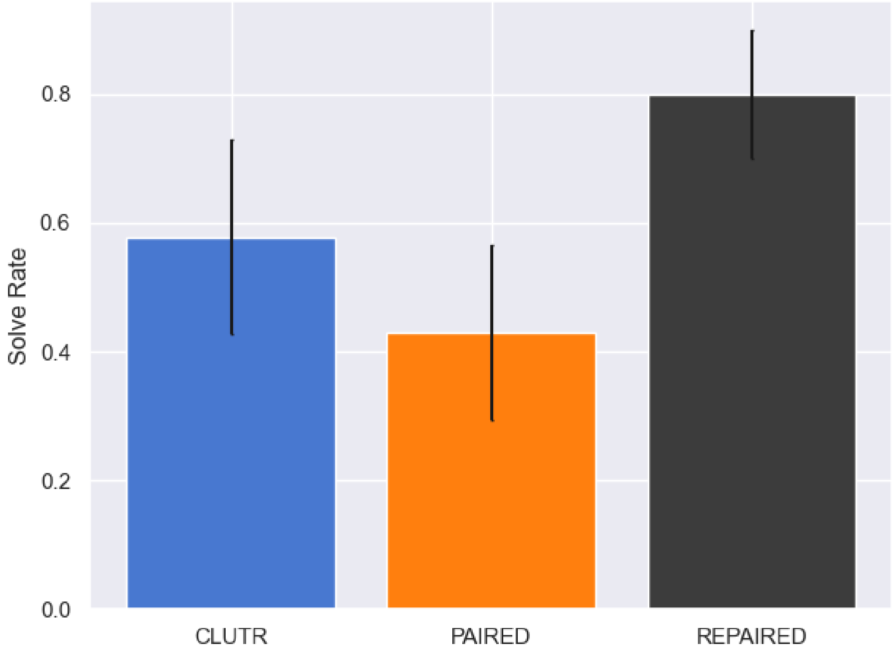


Figure 3.22: Mean solve rate on Minigrid testset. REPAIRED outperforms both CLUTR and PAIRED.

Comparison with Other UED Methods

Comparison with Domain Randomization: Since, CLUTR VAE is trained on uniformly random samples, for completeness we compare CLUTR's result with Domain Randomization (DR) baseline in Figure 3.27. Similar to our flexible regret objective experiments, we trained DR for 250M timesteps with up to 50 obstacles. Our results show that CLUTR significantly outperforms DR with a 29% higher solve rate, while DR exhibits only marginal improvement over PAIRED.

Comparison with ACCEL: ACCEL [68] outperforms CLUTR in the MiniGrid domain, as shown in Figure 3.28. We would like to mention that ACCEL was trained using 60-block settings, whereas CLUTR and PAIRED were trained using 50-block settings. We would further note that ACCEL and CLUTR are fundamentally different approaches with distinctly different training settings and techniques. While ACCEL uses a dual-curriculum with a random generator/teacher, level-replay with stop-gradient [45], and an evolutionary algorithm for task editing; CLUTR focuses on improving adaptive-teacher UED algorithms. Therefore, a direct comparison between the two methods would require a more careful consideration of their respective training methodologies, strengths, and limitations.

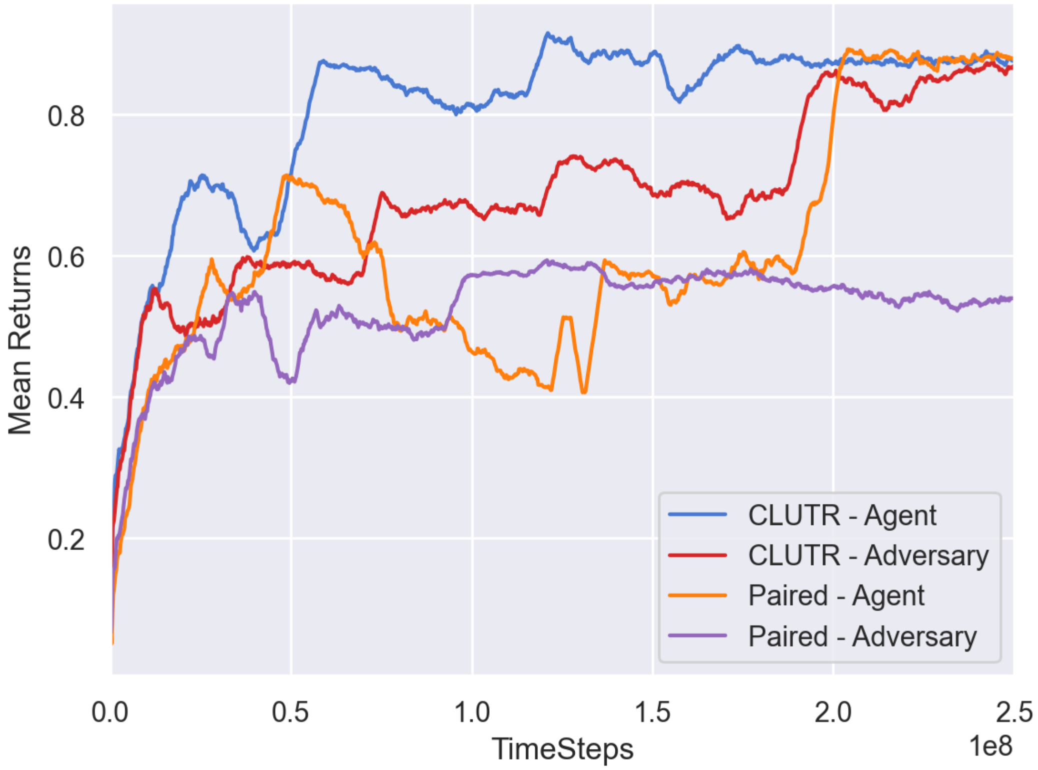


Figure 3.23: Mean return on the training tasks for both the student agents. CLUTR student agents show close performance, while PAIRED students show a bigger gap of performance between them. Closely competing agents can indicate the training tasks being slightly harder than the agents can currently solve, resulting in a smoother curriculum

Curriculum Analysis

Curriculum Snapshot

In this section, we visually inspect the curriculum generated by CLUTR and PAIRED, with snapshots of tasks generated by these methods during different stages of the training (Figure 3.29). We illustrate one common mode of failure/ineffectiveness shown by PAIRED: The curriculum starts with arbitrarily complex tasks, which none of the agents can solve at the initial stage of training. After a while, PAIRED starts generating rudimentary degenerate tasks. While kept training, PAIRED eventually gets out of the degenerative local minima, and the curriculum complexity starts to emerge. On the other hand, CLUTR does not show such degeneration and generates seemingly interesting tasks throughout.

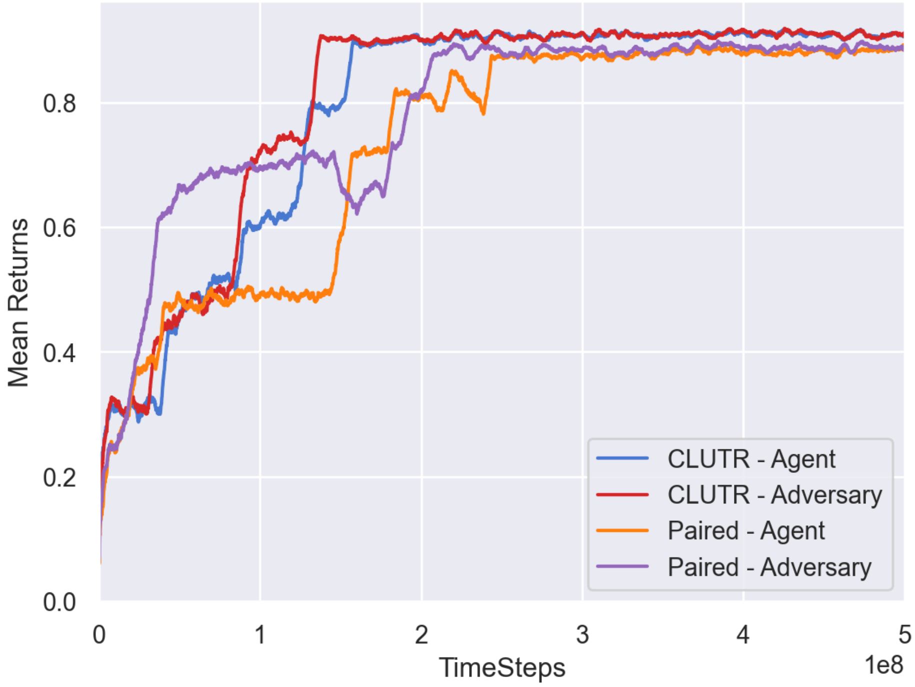


Figure 3.24: Mean return on the training tasks for both the student agents. CLUTR student agents show close performance, while PAIRED students show a bigger gap of performance between them initiallly at the beginning.

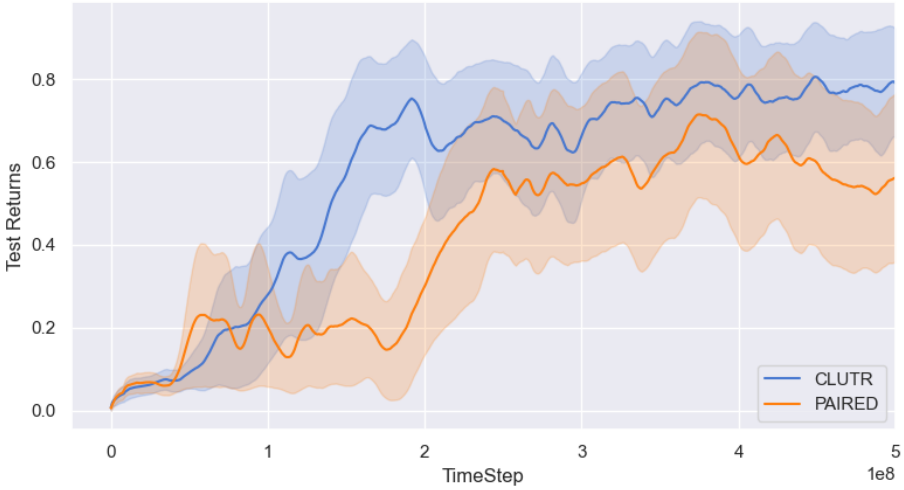


Figure 3.25: Agent solved rate on selected grids during training. CLUTR shows better sample efficiency and generalization than PAIRED. The results show an average of 5 independent runs.

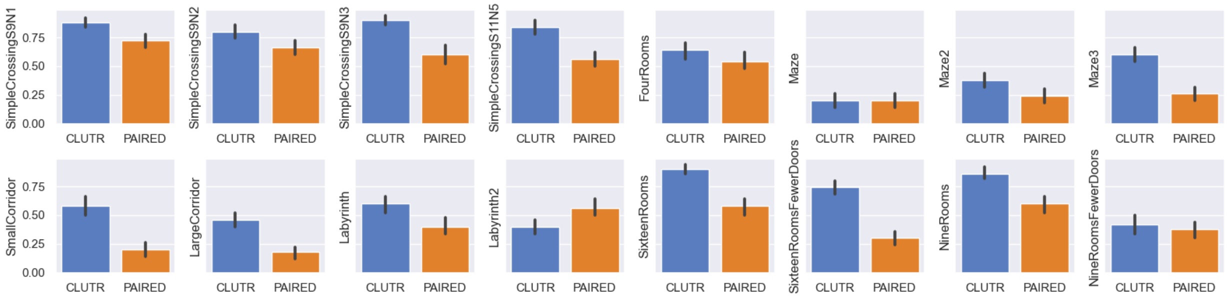


Figure 3.26: Zero-shot generalization of CLUTR and PAIRED, in terms of percent of the 14 solved. CLUTR achieves a higher solved rate than PAIRED in 14 out of the 16 unseen tasks. We evaluate the agents with 100 independent episodes on each task. Error bars denote the standard error.

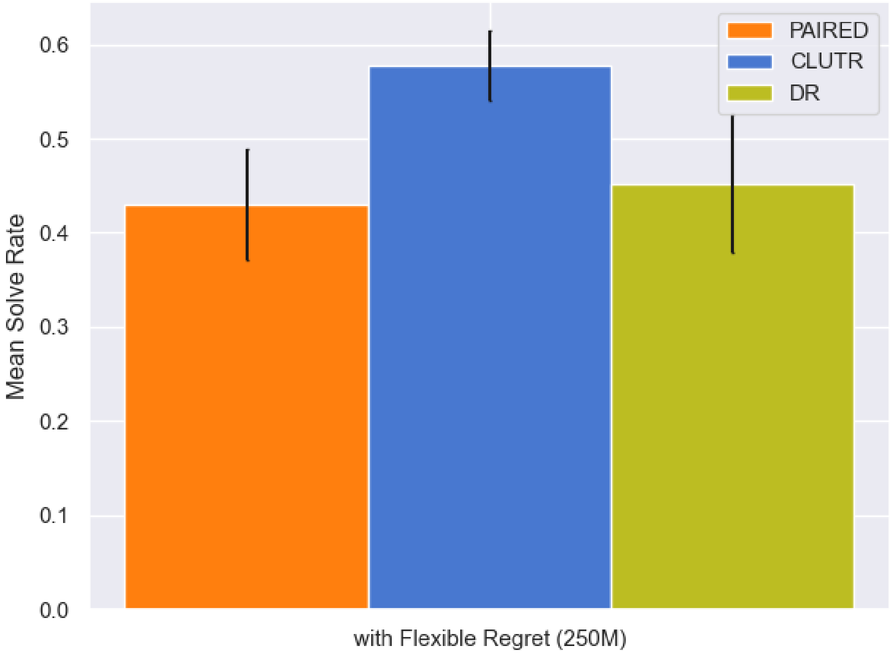


Figure 3.27: Comparison of CLUTR (and PAIRED) with Domain Randomization(DR) baseline. CLUTR outperforms DR with a 29% higher solve rate.

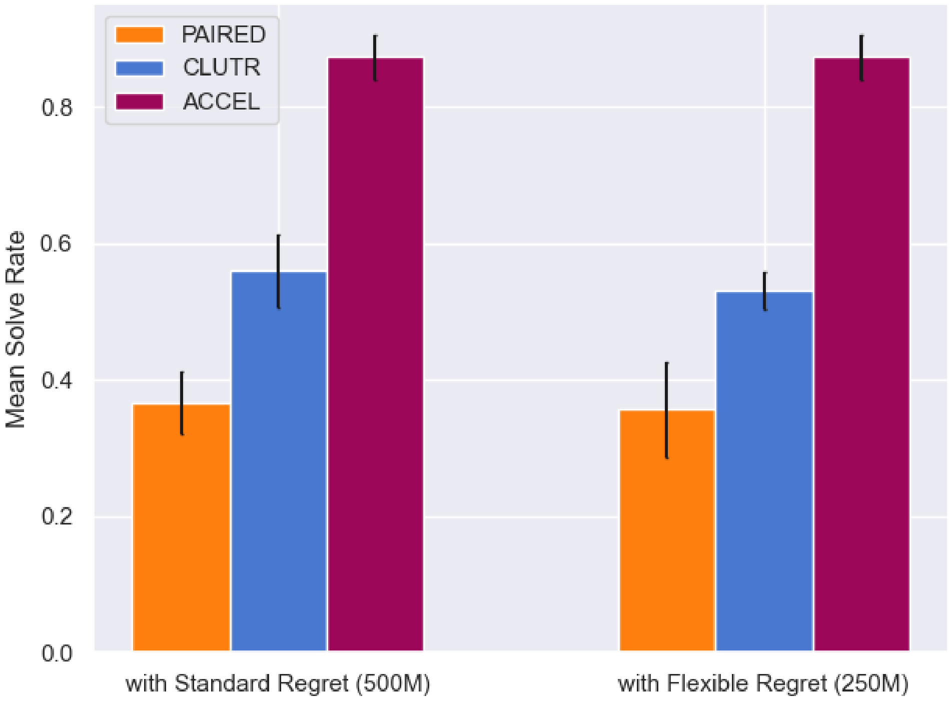


Figure 3.28: Comparison of CLUTR (and PAIRED) with ACCEL. ACCEL outperforms both CLUTR and PAIRED. However, we note that ACCEL is a fundamentally different approaches with distinctly different training settings and techniques.

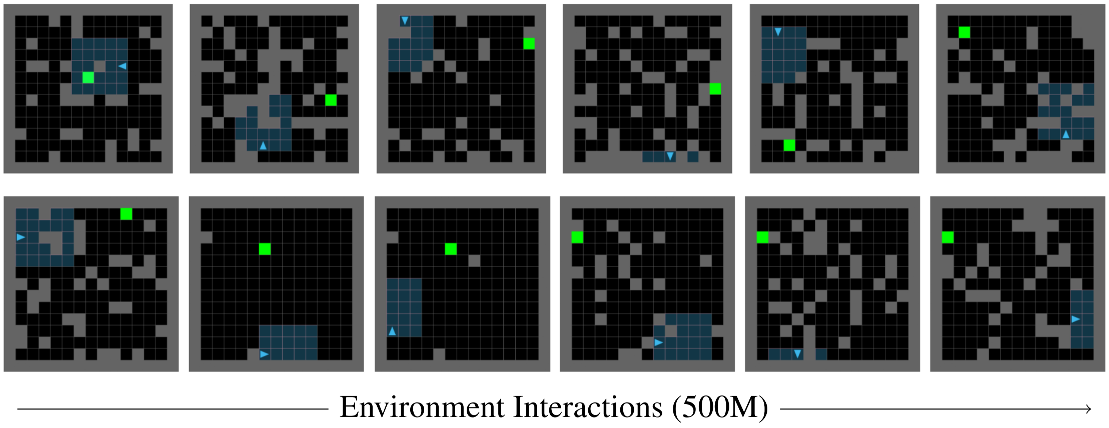
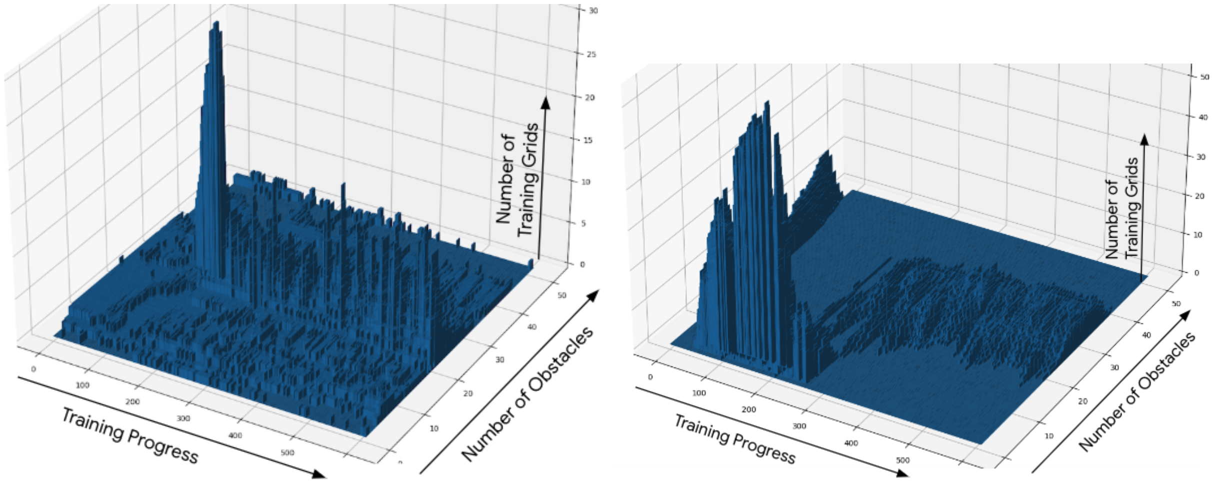


Figure 3.29: Example grids (right) generated by CLUTR (top) and PAIRED (bottom) uniformly sampled at different stages of training. The training progresses from left to right.

CLUTR vs PAIRED

Figure 3.30 shows 3D Histograms showing the frequency of the generated grids against the total number of obstacles they contain. PAIRED starts with a high number of obstacles and then degenerates quickly into grids with very few numbers of obstacles and stays similar for a significant number of steps. Eventually, the number of obstacles increases sharply, converging into a band of around 20 to 40 obstacles on average. On the other hand, in CLUTR, the number of obstacles starts flat, centers around a peak around the middle but still with a wide interval for some number of steps, and the peak drops slightly while the interval stays almost the same. After the 'convergence', PAIRED rarely generates grids with fewer or more obstacles than the band it converges to. On the contrary, CLUTR still generates grids with few or many blocks, which might help to address unlearning or improve the agents on grids with more obstacles, respectively. The above observations illustrate that we can achieve a more efficient curriculum learning without making the problem too easy early or without focusing on a narrow interval with a flat distribution later. Instead, we can start with a wide interval and gradually focus on a peak around the middle without making the interval very narrow.

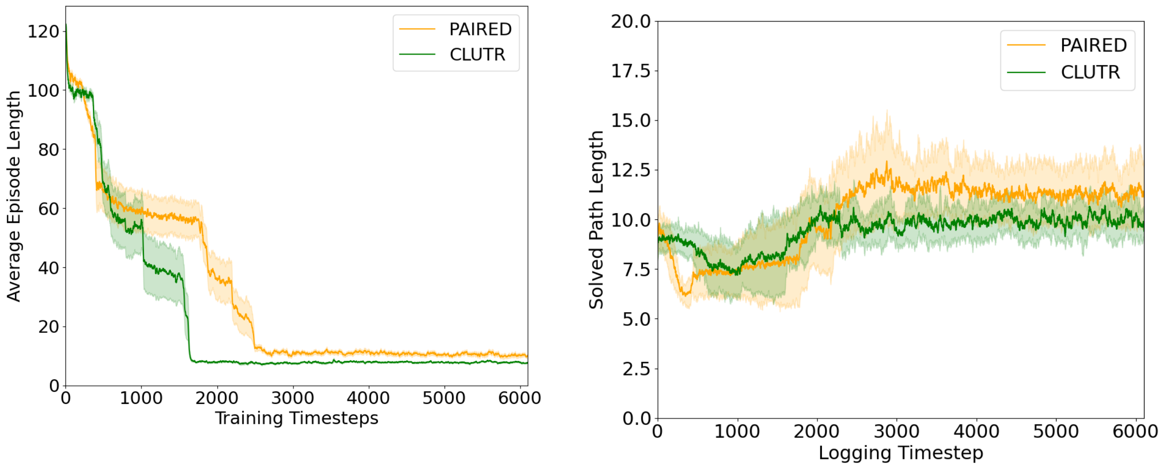


(a) CLUTR

(b) PAIRED

Figure 3.30: 3D Histograms showing the frequency of the generated grids against the total number of blocks they contain. Both PAIRED and CLUTR converge to a similar band of grids. However, CLUTR converges much faster.

Figure 3.31a shows the average episode lengths of both CLUTR and PAIRED. The curves show both methods start with long episodes—indicating at the beginning, the agents do not solve the training grids consistently, and many of the episodes end due to timeout. As the agents learn, the episodes become shorter for both methods until they converge to a small value. However, CLUTR converges sooner than PAIRED.



(a) Average length of the training episodes. CLUTR converges sooner than PAIRED to a shorter episode length.

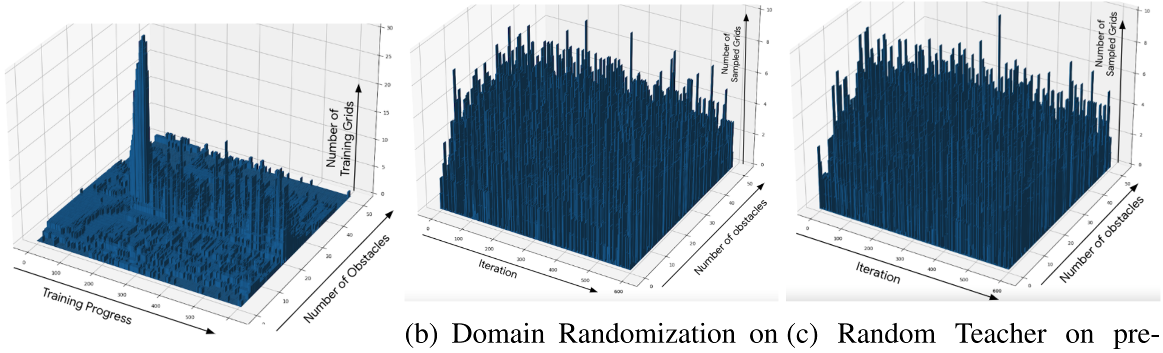
(b) Average solution length of the solved training tasks.

Figure 3.31: Comparison of CLUTR and PAIRED curriculum based on properties of the generated grids.

We also compare the average solution length of the solved training grids. Both PAIRED and CLUTR show a similar pattern. However, PAIRED converges to a larger value than CLUTR. This might indicate that CLUTR is solving the environments more efficiently. This might also mean that CLUTR is solving some easier tasks (e.g., fewer obstacles, as we noticed from Figure 3.30) even after convergence lowering its average solved path length slightly.

CLUTR curriculum vs. Random Latent Curricula

We further compare CLUTR curriculum with two different domain randomized curriculums. First we compare CLUTR curriculum with a uniform random (i.e., Domain Randomization) curriculum on the latent space by repeatedly sampling the trained VAE (the same VAE used by CLUTR) with a uniform random distribution. Second, we generate a curriculum generated by a random teacher acting on the pretrained latent space. The random teacher uses the same architecture and intialization procedure as the original CLUTR teacher it is being compared to. Figure 3.32 shows the comparison characterizing the grids by the number of obstacles they contain similarly as the previous section. As expected, we can see that the DR and random teacher curriculum generates grids with obstacles ranging from 0 to 50 without showing any pattern, showing significant difference in the curricula generated by CLUTR and the domain randomized baselines.



(a) CLUTR

(b) Domain Randomization on the pretrained Latent Space

(c) Random Teacher on pretrained Latent Space

Figure 3.32: 3D Histograms showing the frequency of the CLUTR generated grids against the total number of blocks they contain vs. Domain Randomization on the latent space vs. A random teacher curriculum on the pretrained latent space. The figures clearly show that CLUTR generates a curriculum significantly different from random curriculums.

CLUTR vs Domain Randomized Environments

To further compare how CLUTR generated grids, differ from Domain Randomized grids: we trained a PCA on a combined set of grids generated from both the methods and projected them into a 2D space. The resulting plot (Figure 3.33) shows that the projections of CLUTR-generated grids form a distinct pattern in the embedded space, while DR-generated grids are all clustered together. The projections are almost entirely disjoint, indicating that the two sets of grids exhibit distinctively different distributions or patterns of variations. This observation suggests that CLUTR and DR generate fundamentally different types of grids. We also not that, he color of the grids intensifies linearly as the training progresses, e.g., grids generated at the early stage of training are of lighter intensity.

Analysis of the Latent Task Manifold

Visualization of Training Progress using the Latent Space: We trained a 2D t-SNE model on a set of latent vectors, which are sampled during training in the MiniGrid domain. We divided the training into 10 equally sized phases and sampled approximately 20K latent vectors from the CLUTR teacher at each phase, resulting in a total of 200K (approximately) latent vectors. We trained the t-SNE model over this entire latent-vector dataset but plotted the embeddings separately in Figure 3.34 for each phase to visualize the evolution of latent vectors during training.

We observe that, early in the training ($< 30\%$), as the protagonist agent is not trained well yet, the teacher easily finds a region towards the far right where the REGRET is maximum. As the protagonist agent improves, the teacher begins exploring new regions (at around $30 - 40\%$) to maximize the REGRET again, leading to a shift in the embeddings towards the far left (up to around 60%). After around 60% training steps, both the antagonist and protagonist agents learn

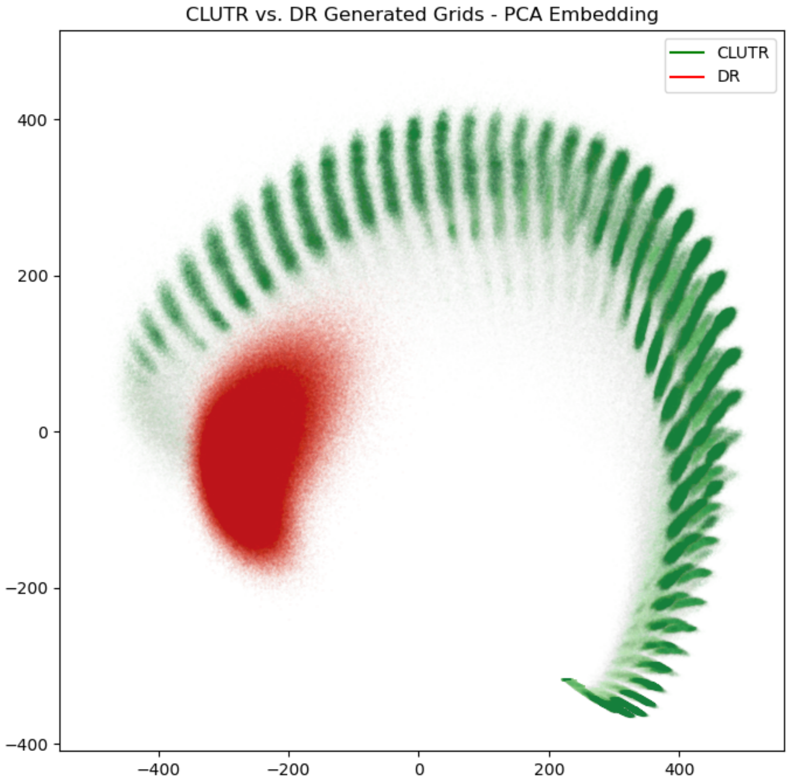


Figure 3.33: PCA embedding of the combined set of grids generated by CLUTR and Domain Randomization. CLUTR-generated grids form a distinct pattern in the embedded space, while DR-generated grids are all clustered together indicating that these methods generate distiinctively different set of grids.

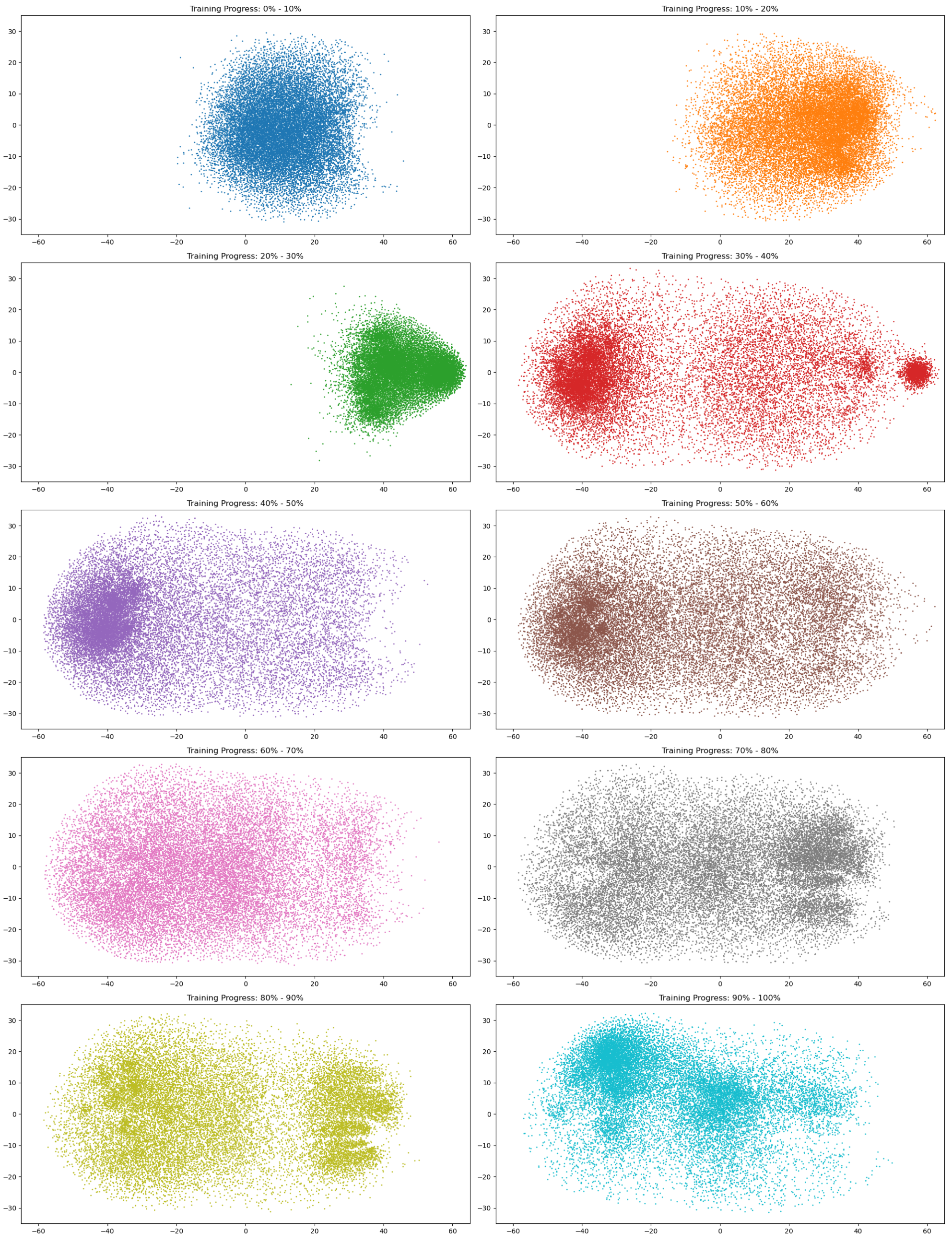


Figure 3.34: t-SNE embedding of the generated tasks during different phase of training. During the initial phase of training the teacher moves from the central region to far right and then moves to far left. We hypothesize, as the protagonist agent is not well-trained during the intial phase, the teacher easily finds regions in the latent space to maximize the REGRET, however as the traing progresses and the agent learns better, the teacher converges its search into a wider region.

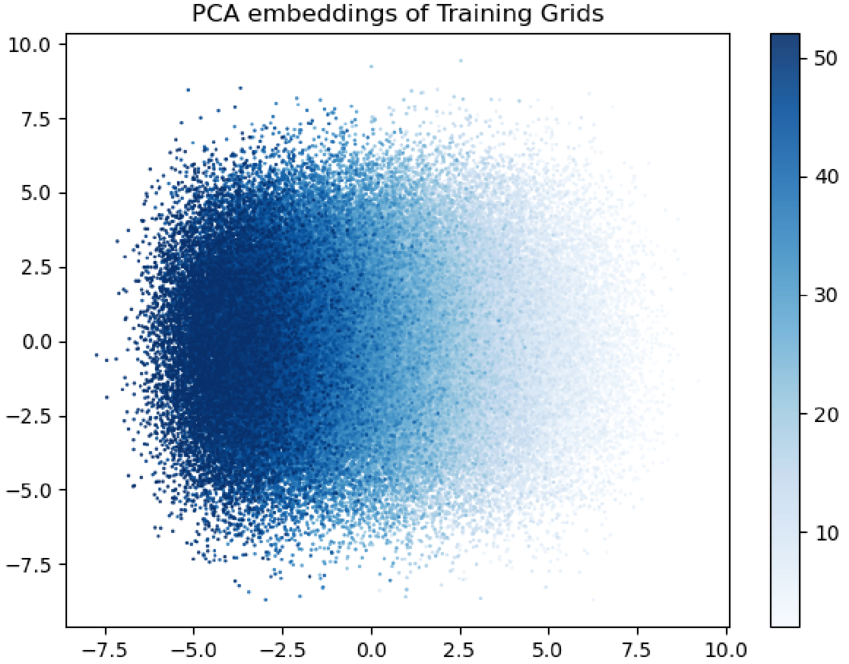


Figure 3.35: PCA Embedding of VAE training dataset. The color intensity represents the number of obstacles in a grid, as indicated by the color bar on the right.

well and the REGRET gets close to zero; embeddings also become relatively wider, and training starts converging.

Structure of the Latent Space: To further investigate the structure of the latent space, we trained a 2D PCA model on a set of latent vectors generated with the VAE from 100K grids sampled uniformly from the VAE training dataset. The resulting 2D embeddings are displayed in Figure 3.35, with their colors transitioning from light to dark blue as the number of obstacles increases from 0 to 50. We observe that the latent vectors show a smooth and gradual pattern in the PCA embedded space as the number of obstacles increases.

Additionally, we constructed a sample maze one obstacle at a time, obtained their latent representation from the VAE, and plotted their 2D PCA embedding using the same PCA model. The incremental construction of the maze is shown in Figure 3.36 and the corresponding embeddings, transitioning from light to dark green as more obstacles are added, are shown in Figure 3.37. The latent vectors form a clear and smooth trajectory in the embedding space as the maze grows.

The above analysis indicate that CLUTR VAE learns a smooth manifold in terms of different grid properties, e.g., number of obstacles and structure.

Linear Interpolation in the Latent Space: To grow a sense of the latent task manifold, we linearly interpolate in the latent space between an empty grid and a 15x15 version of the FourRoom grid (shown in Figure 3.39). Figure 3.38 visualizes the interpolation results. We first get the

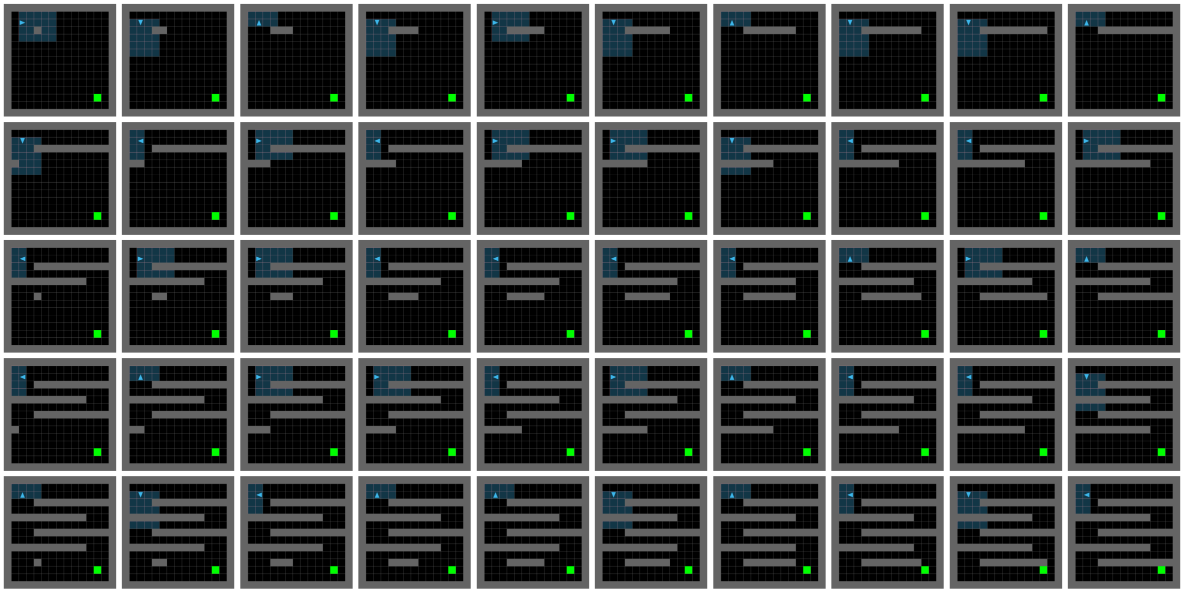


Figure 3.36: An example grid constructed by adding one obstacle at a time (from top left to bottom right). The correcponding 2D PCA embedding can be found in Figure 3.37.

latent vectors of the empty grid and the target FourRoom task using the VAE encoder. We then linearly interpolate 23 equidistant points between them. At last, we reconstruct the grids from these vectors using our decoder. From Figure 3.38 we see that, as we interpolate in the latent space, the reconstructed grid incrementally adds more obstacles and the grids start to look more like the FourRoom target grid. We note that the reconstruction is not perfect. We also note that the increase in the number of obstacles is not uniform, e.g., the first 5 reconstructed grids are all empty grids, and more obstacles are added near the target point. Overall, this experiment provides an insight that the latent space holds a useful structure, which CLUTR teacher utilizes to generate the curriculum.

3.7 Conclusion: Limitations and Future Work

In this work, we introduce CLUTR, an unsupervised latent space adaptive-teacher UED method that augments adaptive UED teachers with a pretrained latent task manifold to decouple task representation learning from curriculum learning. CLUTR first trains a recurrent VAE from random tasks to learn the latent task manifold and then employs a regret-based adaptive-teacher to induce the curriculum. Through this decoupling, CLUTR solves the long-horizon credit assignment and the combinatorial explosion problems faced by regret-based adaptive-teacher UED methods. Our experimental results show strong empirical evidence supporting the effectiveness of our proposed approach.

Even though CLUTR and other regret-based UEDs empirically show good generalization on

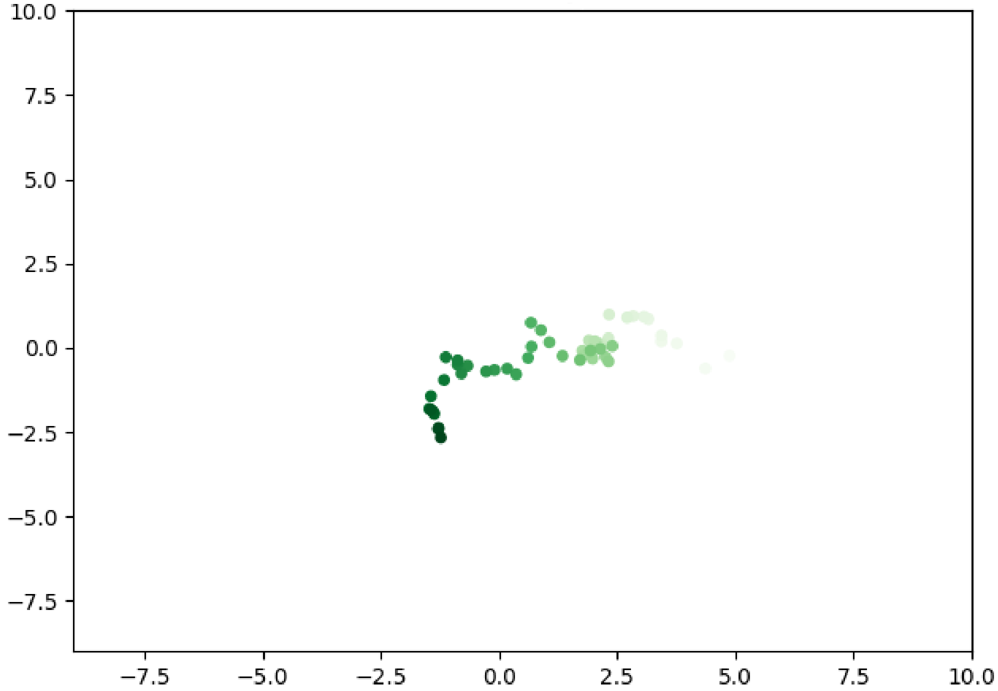


Figure 3.37: PCA emneddings of the grids—constructed by adding one obstacle at a time— shown in Figure 3.36. The color intensity increases with the number of obstacles. We observe a clear and smooth trajectory in the embedding space formed by the latent vectors, indicating the smooth and incremental properties of the latent space.

human-curated complex transfer tasks, they rarely can generate human-level task structures during training. An interesting direction would be to enable UED algorithms to generate realistic tasks. Furthermore, as these methods rely significantly on the design of parameter-space, it would be interesting to investigate how these methods scale on the higher dimensional environments. Another important direction would be to reduce the gap between the theoretical and practical aspects of regret-based multi-agent UED algorithms, which are subject to the quality of regret estimates and multi-agent RL training. At last, random generator algorithms like Robust PLR or even, DR have been shown to perform better than adaptive-teacher approaches like CLUTR or PAIRED. An interesting direction would be to investigate the conditions/environments under which a random generator performs better than an adaptive generator and vice versa. At last, we are excited about latent-space curriculum design and hope our work will encourage further research in this domain.

Acknowledgments

The research work presented in this chapter is performed in collaboration with Izzeddin Gur and Aleksandra Faust from Google DeepMind; Jasper Emhoff, Nathaniel Alexis, Prof. Pieter Abbeel,

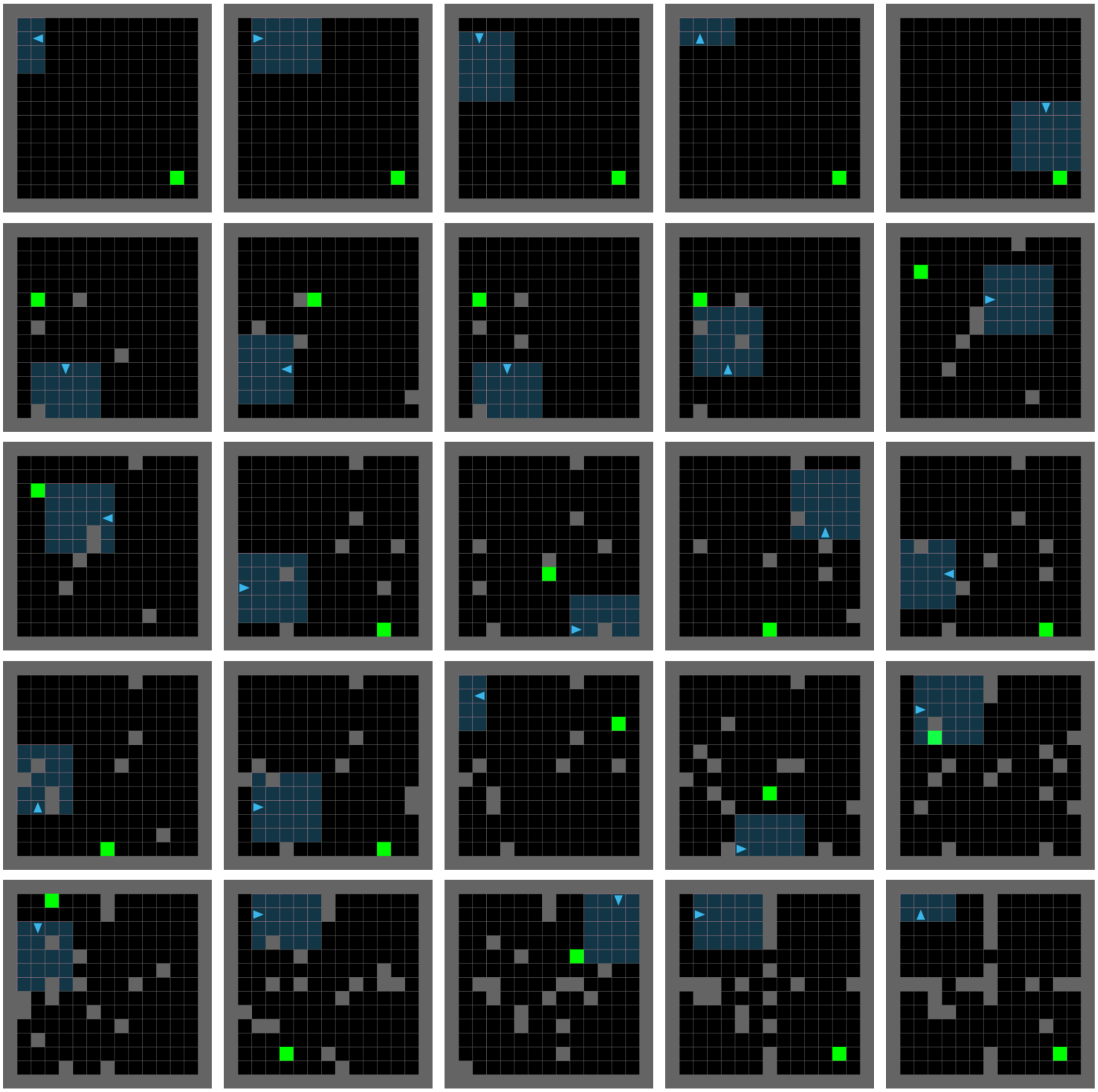


Figure 3.38: A linear interpolation between an empty grid and 15x15 version of the Four-Room grid (Figure 3.39) in the latent space. The grids are organized from top-left to bottom-right in row-major order.

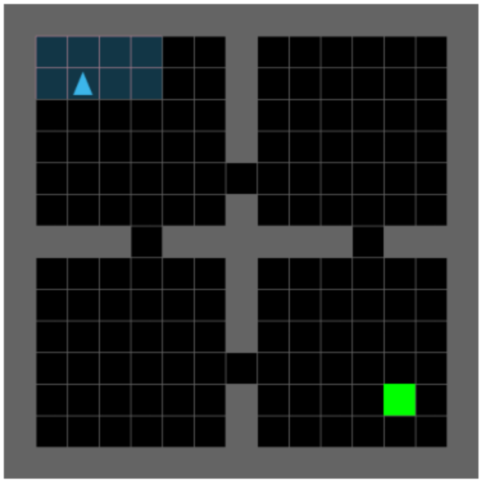


Figure 3.39: 15x15 FourRooms

and Prof. Ion Stoica from UC Berkeley. It has been published in Proceedings of the 40th International Conference on Machine Learning, PMLR [2].

This project is a collaboration under the Google-BAIR (Berkeley Artifcial Intelligence Research) Commons program. It was supported in part by NSF CISE Expeditions Award CCF-1730628, in addition to gifts from Astronomer, Google, IBM, Intel, Lacework, Microsoft, Mohamed Bin Zayed University of Artificial Intelligence, Nexla, Samsung SDS, Uber, and VMware. We gratefully acknowledge Natasha Jaques for their valuable feedback. We also acknowledge Raymond Chong, Adrian Liu, and Sarah Bhaskaran for the valuable discussions.

Chapter 4

MMRC: Multimodal Reasoning and Critique for Web Navigation

With the rapid advancements in Multimodal Large Language Models, these models are increasingly being used to solve sequential decision tasks out-of-the-box, showcasing promising reasoning and generalization capabilities. Several techniques such as few-shot prompting, chain-of-thought, and actor-critic framework have been developed to enhance their reasoning and planning abilities. However, these techniques often fall short for web navigation in practice. For web navigation, the input prompts or observations typically include HTML content and, occasionally, screenshots of the webpage. The HTML documents can be arbitrarily long, even for current long-context foundational models, impacting both performance and cost. Contemporary methods often focus on summarizing the HTML to a more manageable size, while methods that use visual input struggle to ground model predictions in actionable events on the webpage. To address these challenges, we introduce MMRC: Multimodal Reasoning and Critique for Web Navigation, which employs a novel environment formulation with a multimodal actor-critic framework. The actor and critic interact in a loop where the actor suggests an action, the critic analyzes it, and the actor's observation (i.e., prompt) is revised if the critic rejects the actor's prediction. We address the grounding problem by presenting short descriptions of webpage elements as a multiple-choice question for the actor to select from. Our experiments on the Mind2Web dataset—a benchmark consisting of 137 real-world websites spanning 31 domains—demonstrate that MMRC outperforms actor-only baselines by up to 11.33% and surpasses the previous best results obtained by Gemini 1.0 Pro Vision by up to 24.17%. Additionally, we show that employing a large foundational model (Gemini 1.5 Pro) as the actor with a smaller, fine-tuned Phi-3 critic model can partially outperform GPT-4V, highlighting a promising direction for improving the performance of general-purpose foundational models in target tasks with smaller fine-tuned critics.

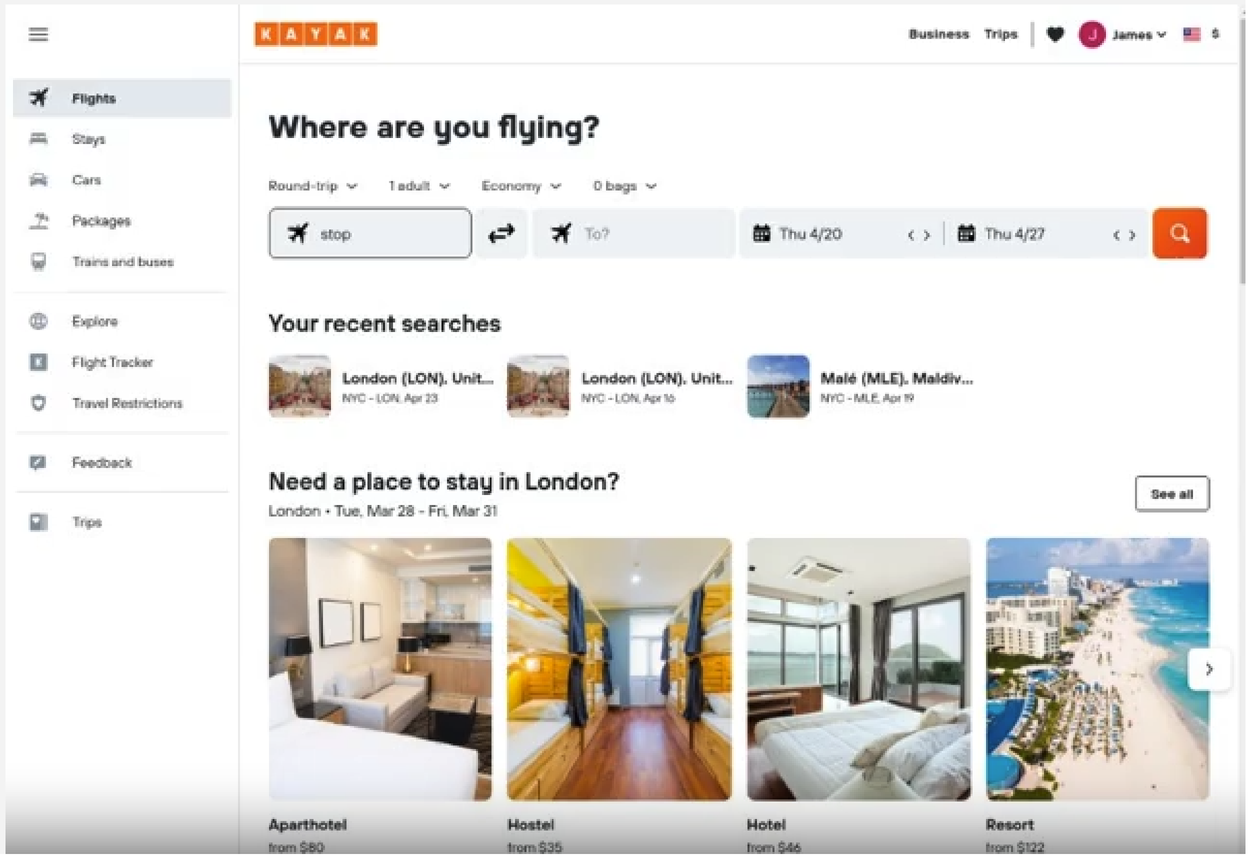


Figure 4.1: Sample task from Mind2Web [18]: 'Book the cheapest hotel in le maraise neighborhood in paris with 2 room for 3 adult on march 27th to april 2nd.' on an Airlines website

4.1 Introduction

Large Language Models (LLM) and Multimodal Large Language Models have seen tremendous advancements in recent years and they have been used to solve a wide variety of sequential decision-making problems such as interactive embodied environments [81], mathematical and verbal reasoning tasks [16, 97], gaming [30], web navigation [105, 18, 32], and computer control [106]. Several techniques such as few-shot prompting, chain-of-thought prompting [96], actor-critic methods [48, 102, 50], and Reflexion [80], have been proposed to enhance the reasoning and planning capabilities of these models. However, such techniques can often fall short for real-world web navigation.

Web navigation is a sequential decision-making task where agents navigate websites based on user-specified tasks. Figure 4.1 shows a sample navigation task from the Mind2Web [18] benchmark. Contemporary methods typically work with textual input, i.e., the HTML of the web pages. However, HTML documents can be arbitrarily long, necessitating summarization techniques to keep the input prompts to a manageable length [18, 32]. On the other hand, while a few methods work with visual inputs, grounding the LLM outputs to actionable events on the webpage remains a challenge. Moreover, the actor-critic framework, a promising approach for sequential decision-making problems, has only been applied to textual inputs [102, 48], and in the domain of web navigation, it has only been used on simple benchmarks [50].

To address these challenges, we introduce MMRC: Multimodal Reasoning and Critique for Web Navigation, a novel multi-modal actor-critic algorithm. MMRC employs one actor and one critic, both multimodal in nature. The actor and critic interact in a loop where the actor suggests an action, the critic analyzes it, and the actor's observation (i.e., prompt) is revised if the critic rejects the actor's prediction. We tackle the grounding and hallucination problems by presenting short textual descriptions of the webpage elements as multiple-choice options for the actor to select, alongside a screenshot of the current webpage. By introducing a multimodal critic, MMRC addresses the grounding problem without requiring complex image manipulation or HTML processing components. We evaluate our method on the Mind2Web dataset [18], a popular and realistic web navigation benchmark consisting of 137 real-world websites spanning 31 domains, divided into three distinct test datasets: Cross Task, Cross Website, and Cross Domain, each evaluating generalization from different perspectives.

In summary, our contributions are as follows:

- We introduce MMRC, the first multimodal actor-critic algorithm for web navigation.
- We present an augmented environment formulation for the actor-critic framework.
- MMRC achieves improvements of up to 7.56%, 11.33%, and 4.85% over actor-only baselines in Cross Task, Cross Website, and Cross Domain splits, respectively, in step success rate.
- We demonstrate that the performance of a general-purpose foundational model can be improved by using a small critic fine-tuned on the target task.
- MMRC surpasses the previous best results obtained by Gemini 1.0 Pro Vision by 4.54%, 11.6%, and 24.17% on the Cross Task, Cross Website, and Cross Domain splits, respectively.

4.2 Related Work

Web Navigation

Most contemporary works in web navigation use the HTML document as the primary input [85, 51, 33]. However, raw HTML can often be arbitrarily long, resulting in a high number of input tokens, even for current long-context models. To address this issue, contemporary methods employ some form of summarization of the raw HTML. For example, MindAct [18] uses a small language model to filter out a subset of elements (50 in their experiments) and generates a series of multiple-choice questions (MCQs), each corresponding to 5 out of those 50 elements. In each MCQ, MindAct generates an HTML snippet related to the 5 elements, leading to a series of at least 10 MCQ queries to the actor. [32] introduces HTML-T5, a pretrained LLM for long HTML documents, and proposes a planning strategy that decomposes and summarizes long HTML documents into task-relevant snippets. On the other hand, several methods use visual information, i.e., the webpage screenshots [78, 39, 29, 79]. SeeAct [105] adopts a similar strategy to MindAct but incorporates visual information as well. To address the grounding problem, it experiments with several techniques, including image annotations, multi-round textual MCQs similar to MindAct, and manual grounding by humans using the GPT-4V foundational model. In contrast, MMRC poses only one actor query with short textual representations of all the filtered elements, eliminating the need for multiple rounds of MCQ actor queries or HTML snippet processing like MindAct and SeeAct.

Actor-Critic Methods

The actor-critic framework has been extensively employed across various domains to address complex decision-making and control problems. Notably, SAC [34] is one of the most popular RL algorithms following an actor-critic framework. Several recent methods have explored the use of actor-critic formulations using Large Language Models (LLMs), where one model acts as the actor to make predictions, and another model serves as the critic to evaluate these predictions. LLaMAC [102] uses a centralized critic to coordinate multiple actors, facilitating collaboration and iterative reasoning in multi-agent systems, solving system resource allocation and robot grid transportation problems. LLM-ARC [48] generates logic programs with the actor, which the critic evaluates and provides feedback on to enhance logical reasoning in complex natural language reasoning tasks. Prospector [50] generates a set of trajectories with its actor, while the critic evaluates the entire trajectories to rank them, achieving impressive performance on ALFWorld [81], an interactive decision-making benchmark for embodied reasoning in solving household tasks, and WebShop [98], an online shopping environment. All of these contemporary methods work only in the text domain. On the other hand, MMRC works with both visual and text inputs in the arguably more complex domain of web navigation on real-world web pages.

4.3 Background

Environment Formulation for Web Navigation

Given a website $\mathcal{W}$ and an intent or task $\mathcal{T}$ (e.g., "*Book the cheapest flight from NYC to SFO on 25th July*") a web navigation problem can be defined as an POMDP (S, A, T, R, S_0, O, Z), where:

- **State Space (S):** Each state is a webpage w and is defined by the tuple (H, I, M) where, H and I corresponds to the HTML and screenshot of the webpage, respectively. M corresponds to previous actions taken by the actor agent.
- **Observation Space (O) and Observation Function (Z):** Similar to contemporary LLM agents, the observation is a prompt P with multiple components. Commonly used obser-

vations in contemporary methods include the entire HTML, the screenshot, the modified screenshot (e.g., bounding boxes), summary information (e.g., MCQs with HTML snippets), or a combination.

- **Action Space (A):** The actions typically refer to a browser event provided by the website environment. For the Mind2Web dataset, each action is defined as a triplet of three necessary variables for a browser event (e, ϕ, v). $e \in \mathcal{E}$ denotes the target webpage element to act upon, such as a button or textbox. $\mathcal{E}$ is the set of elements within the webpage $\mathcal{W}$. The operation $\phi \in \Phi$ is the operation to be performed on the target element, with $\mathcal{O}$ corresponding to operations such as, `Click`, `Type`, `Select`. The variable v specifies additional value needed for certain operations (e.g., 'NYC' for a `Type` operation).

- **Transition Dynamics** $T(s'|s, a)$**:** The transition function is defined by the website $\mathcal{W}$ determining the landing webpage $s' = w'$ when an action a is executed on the webpage $s = w$.

- **Reward Function (R):** The reward function $R(s, a)$ is a sparse reward denoting success (i.e., 1) when the task $\mathcal{T}$ is successfully completed and 0 otherwise.

- **Initial State Distribution ($\mathbf{S_0}$):** The distribution over all websites and tasks.

4.4 Method: MMRC

Typical web navigation agents follow a single agent workflow, as shown in Figure 4.2. $MMRC$ employs a simple two-agent workflow with a modified environment formulation, where an actor model predicts an action and a critic model analyzes the predicted action and determines if that would be executed on the webpage or, should the actor revise its prediction. This actor-critic interaction is repeated up to a predefined maximum limit. Figure 4.3 shows the modified agentic workflow used in $MMRC$. In the next sections, we discuss $MMRC$ in detail.

MDP Formulation for Web Navigation

MMRC formulates web navigation as a POMDP with modified transition function $T'((s', c)|s, a))$ and observation function $Z'(s, c)$ employing two different agent/policies π_{Actor} and π_{Critic}. The primary aspect of this formulation is as follows: the actions a generated by the actor policy $\pi_{\text{Actor}}(a|o = Z'(s, c = \emptyset)$ is not executed readily in the environment. On the contrary, it is first passed to the critic policy $\pi_{\text{Critic}}(c|s, a)$, which given the current state, decides whether the action should be executed or not. In case the critic decides against executing the suggested action a, the actor policy is queried again with a modified observation $Z'(s, c)$ incorporating the critic's response. This is repeated until the critic accepts the suggested action or a stopping criterion (e.g., a maximum number of actor queries) is met.

Formally, MMRC formulates web navigation as a POMDP $(S, A, T', R, S_0, O, Z', N)$ where N is the max number of critic steps, with the following transition dynamics and state functions:

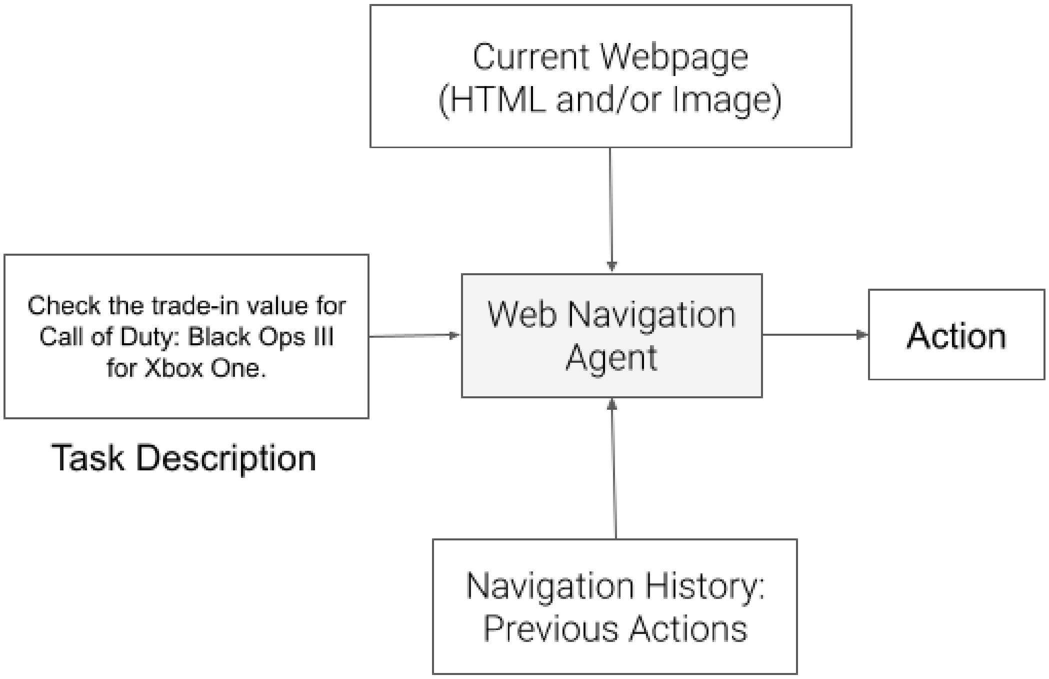

Current Webpage
(HTML and/or Image)
Check the trade-in value for Call of Duty: Black Ops III for Xbox One.
Task Description
Web Navigation Agent
Action
Navigation History: Previous Actions

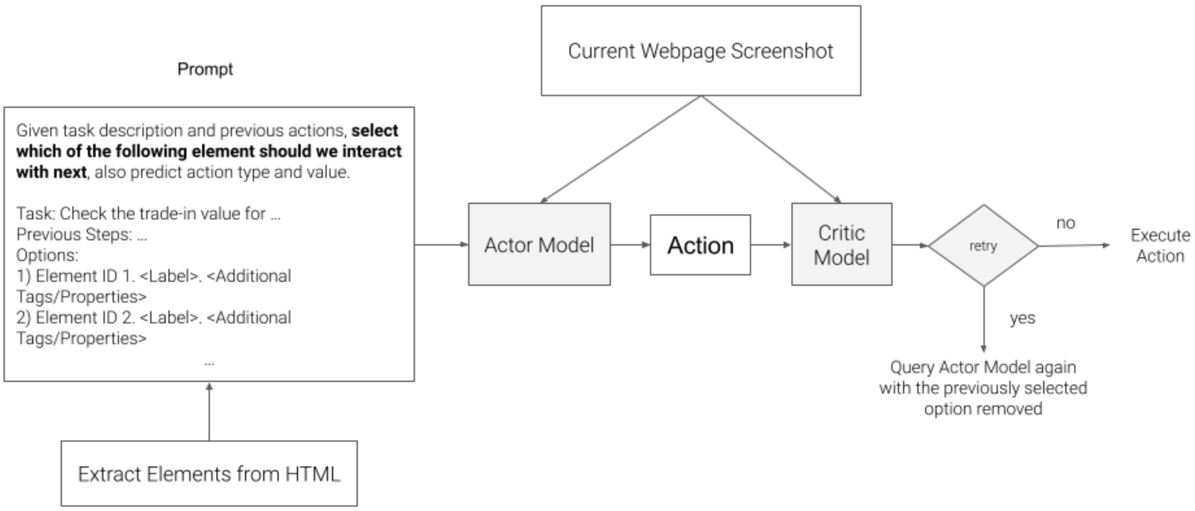
Figure 4.2: The typical agentic workflow used in web navigation.

Transition Dynamics $T'((s', c)|s, a)$**:**

$$T'((s', c)|s, a) = \begin{cases} (T(s, a), \emptyset) & \text{if critic } \pi_{\text{Critic}}(c|s, a) \text{ accepts action } a \text{ or} \\ & N \text{ critic steps have been executed} \\ (s, c) & \text{otherwise} \end{cases}$$

Observation Function $Z'(s, c)$:

$$Z'(s, c) = \begin{cases} \text{observation } Z(s) \text{ incorporating critic response } c & \\ \qquad\qquad \text{if the critic } \pi_{\text{Critic}}(c|s, a) \text{ rejects action } a, & \\ Z(s) & \text{otherwise.} \end{cases}$$



Given task description and previous actions, **select which of the following element should we interact with next**, also predict action type and value.

Task: Check the trade-in value for ...
Previous Steps: ...
Options:
1) Element ID 1. <Label>. <Additional Tags/Properties>
2) Element ID 2. <Label>. <Additional Tags/Properties>
...

Extract Elements from HTML
Current Webpage Screenshot
Actor Model
Action
Critic Model
retry
no
Execute Action
yes
Query Actor Model again with the previously selected option removed

Figure 4.3: Agentic Workflow used in MMRC involving critic with modified envirnment formulation.

MMRC Actor and Critic Details

$MMRC$ employs Multimodal Large Language models as actors and critics. The rationale behind using multimodal input for web navigation is that webpage HTML can be arbitrarily long, which can cause significant inference costs. On the other hand, webpage screenshots can significantly reduce the number of input tokens needed. However, the main challenge working with visual inputs is grounding the actions to the underlying HTML to automate the navigation [105].

The main essence of the actor and critic agents is how the observation space, i.e., the prompts, is designed. Listing 4.1 shows the template used for the $MMRC$ actor.

Listing 4.1: MMRC Actor Prompt

```
{The webpage screenshot}
You are an automated web navigation assistant: Given a task,
   a webpage screenshot, previous steps, and a list of
   elements from the webpage, you need to suggest which
   element from the provided list to act upon in the current
   step.
Task: {The Task Description}
Previous Steps: {History of the previous actions taken}
Current Step Options:
You must act upon exactly one of the following webpage
   elements, where each element is formatted as:
ID. [element type] label <Additional properties separated by
   semicolon>(optional).
{Enumerated List of Interactive Elements, represented with
   short textual description}
Based on the screenshot, previous steps, and the provided
   options, summarize how much progress has already been made
   and suggest which element from the list to act upon in the
   current step. An action can be one of [CLICK, TYPE, SELECT
   ], where SELECT means selecting an option from dropdown/
   radio buttons.
Structure your suggestion strictly in the following format:
progress_made: Brief sentence explaining the current progress
    towards completing {task_description}.
next_step: Suggest the next step based on the provided list
   of elements with brief explanation/rationale
element_id: Exact ID of the selected element from the
   provided list.
action: CLICK/TYPE/SELECT
value: In case of TYPE, value is the text we should type. For
    SELECT, value corresponds to the option we are selecting.
   For CLICK value is "None".
```

The $MMRC$ actor prompt consists of four components:

1. The Task Description $\mathcal{T}$
2. History of the previous actions taken M
3. The webpage screenshot I
4. A list of textual descriptions of the interactive elements from the webpage, which are presented as a MCQ in the prompt. The actor selects one of the choices.

The textual representation for each element is generated from the underlying HTML using their type, text label, and attribute values, truncated to a maximum length, e.g., 100. For example, a <a> hyperlink element (with tag a) to trip.com would be described as Hyperlink [Trip.com]. By using short textual descriptions we restrict the actor model to pick elements only present on the webpage and help solve the grounding problem and hallucinations.

Once the actor model generates a prediction, the critic is asked to analyze it, using the following prompt shown in Listing 4.2. If the critic suggests taking a different action, the actor is provided with a revised prompt with the previous precited MCQ option removed.

Listing 4.2: MMRC Critic Prompt

```
{The webpage screenshot}
You are an automated web navigation evaluator. Given a task,
   a webpage screenshot, previous navigation steps, and a
   suggested next step: decide whether we should execute the
   suggested next step or, consider a different step to
   complete the task.
Task: {The Task Description}
Previous Steps:
{History of the previous actions taken}
Next Step Suggestion:
{Precited Action}
Based on the previous steps taken and the current state of
   the website, suggest whether we should execute the
   suggested next step to complete the task ({The Task
   Description}).
Structure your suggestion strictly in the following format:
decision: yes/no
explanation: Brief sentence explaining why we should execute/
   not execute the suggested next step.
```

MMRC Algorithm

Algorithm 2 shows the MMRC algorithm. For a website $\mathcal{W}$ and a task $\mathcal{T}$, MMRC interacts with the webpage until a $TERMINATION\_CONDITION$ is met. For each step, MMRC generates the set of all elements, represented as their textual representation as described before. Then, MMRC works with only a subset of those elements using the $FILTER$ method, which is discussed in detail in the experiments section. The actor-critic loop runs for at most N iterations. First, the actor agent π_{Actor} predicts an action based on $\mathcal{O}$, the webpage screenshot I. The set $\mathcal{P}$ stores the predicted action. After that, the critic agent π_{Critic} analyzes the action and decides whether to resample the actor. In case the critic rejects the predicted action, it is removed from the set of options $\mathcal{O}$. Once the loop terminates, an action is selected from the set $\mathcal{P}$ using the method $SELECT\_ACTION$ and executed in the environment. Usually, the selected action is the action

Algorithm 2 MMRC

```
Input: Website $\mathcal{W}$, task $\mathcal{T}$, and maximum critic steps $N$
$t \leftarrow 0$
$M \leftarrow \emptyset$
while $TERMINATION\_CONDITION$ is not met do
    $\mathcal{O} \leftarrow$ Set of textual representations of the elements of the current webpage HTML $H$
    $\mathcal{O} \leftarrow FILTER(\mathcal{O})$
    $\mathcal{P} \leftarrow \emptyset$
    for $i \leftarrow 1$ to $N$ do
        $a \leftarrow$ Predict next action with $\pi_{\text{Actor}}$ using prompt template listed in Listing 4.1
        $\mathcal{P} \leftarrow \mathcal{P} \cup \{a\}$
        $d \leftarrow$ Critique the predicted action $a$ using $\pi_{\text{Critic}}$
        if $d$ is True then
            break
        end if
        $\mathcal{O} \leftarrow \mathcal{O} - \{a\}$
    end for
    $a_{sel} \leftarrow SELECT\_ACTION(\mathcal{P})$
    Execute $a_{sel}$ on the webpage $w$
    $M \leftarrow M \cup \{a_{sel}\}$
    $t \leftarrow t + 1$
end while
```

the critic accepts. In the case, the critic rejects all the predicted actions, multiple strategies can be adopted. In our experiments, we select the first action actor predicted in case none of the predictions are accepted by the critic.

4.5 Experiments

In this section, we describe our experiments and discuss the following questions and/or hypotheses:

1. **H1:** The enhanced environment formulation with critic improves the generalization of web agents.

2. **H2:** A finetuned critic can result in better generalization compared to a general-purpose foundational model.

3. **H3:** How does MMRC perform compared to other multimodal methods?

4. **H4:** What is the impact of the number of iterations of the actor-critic interaction loop?

Dataset

All our experiments are conducted on the Mind2Web dataset [18]. Mind2Web is a real-world offline dataset with over 2K complex tasks with human-annotated actions from 137 websites. The test dataset is divided into three categories, to evaluate the agent's generalization across different domains, websites, and tasks. Cross-Task, Cross-Website, and Cross-Domain contain 252 tasks from 69 websites, 177 tasks from 10 websites, and 912 tasks from 72 websites, respectively. The websites cover a wide variety of domains such as Airlines, Car rental, Hotel, Housing, Shopping, Health, Government, and Music. Mind2Web is an offline dataset with human-annotated ground truth. Each action in Mind2Web is comprised of a (Target Element, Operation, Value) tuple. The target element is an interactable element of the webpage associated with an id. The operations can be Click, Type, and Select. For the Type and Select options, the Value field contains the text to type or the value of the option from the dropdown. For Click, value is set to NULL.

Evaluation: The performance of web agents on the Mind2Web dataset is measured using three metrics: i) Element Accuracy, ii) Operation F1, and iii) Step Success Rate (Step SR) following [18]. Element accuracy measures if the predicted element matches one of the human-labeled ground truth elements. One of the major limitations of this metric is, there might be more than one way, or one order of sequence that can solve the task. However, this evaluation only considers those trajectories that were followed by the human annotators. Operation F1 measures the F1-score based on the text representation of the (Operation, Value) tuple. Step Success Rate measures the percentage of steps that exactly matches the ground truth, and combines the previous two metrics.

MMRC Agent and Critic Models

For our experiments, we use two different foundation models as actors and critics: i) Gemini 1.5 Pro and ii) Gemini 1.0 Pro Vision. We also experiment with the Phi-3 Multimodal model [1] as a critic—finetuned on the target task. However, the Phi-3 model was finetuned only on the textual predictions of the Mind2Web train dataset, the vision module was kept intact. More specifically, the Phi-3 model was finetuned on the textual representation of the (Target Element, Operation, Value) tuple, only focusing on the planning part without any finetuning to better understand the visual inputs. We finetuned it for one epoch with a batch size of 64 using LoRA with fp16, LoRA Rank 128, LoRA alpha 256, and learning rate 2e-4. Further details of the phi-3 training can be found in the Hateful Memes section of the official Phi-3 CookBook [60].

MMRC Baseline

To examine the impact of our environment formulation with critic, we define MMRC-Baseline. The baseline algorithm follows the algorithm delineated at Algorithm 3, which is very similar to the main MMRC algorithm (Algorithm 2), except the actor-critic interaction loop is not present.

Algorithm 3 MMRC-Baseline

```
Input: Website $\mathcal{W}$, task $\mathcal{T}$, and maximum critic steps $N$
$t \leftarrow 0$
$M \leftarrow \emptyset$
while $TERMINATION\_CONDITION$ is not met do
    $\mathcal{O} \leftarrow$ Set of textual representations of the elements of the current webpage HTML $H$
    $\mathcal{O} \leftarrow FILTER(\mathcal{O})$
    $a \leftarrow$ Predict next action with $\pi_{\text{Actor}}$ using prompt template listed in Listing 4.1
    Execute $a$ on the webpage $w$
    $t \leftarrow t + 1$
end while
```

Details on the $FILTER$ and $SELECT\_ACTION$ Methods and other Algorithm Parameters

For the $FILTER$ method used in Algorithm 2 nad Algorithm 3, we follow the candidate selection method used in [18]. They use a small LM that filters out elements that are unlikely to be the target element. [18] finetuned the base version of DeBERTa [36] with 86M parameters for candidate selection on the training split of Mind2Web dataset. It achieves 88.9%, 85.3%, and 85.7% Recall@50 on Cross Task, Cross Website, and Cross Domain test splits.

In all our experiments, we keep 50 elements using the $FILTER$ method to generate the MCQ presented to the actor. However, unlike [18] we present all the 50 options in a single prompt, while [18] perform a multi-round MCQ, where they generate multiple MCQ pormpts for its actor with 5 options in each prompt, resulting in a minimum of 10 actor rounds. [18] also uses a different format of MCQ where they generate an HTML snippet with the elements corresponding to the 5 MCQ options. In our case, we generate short text representations, without requiring any short HTML snippet.

In all our experiments, $SELECT\_ACTION$ selects the action accepted by the critic or, in case all of the actor predictions are rejected by the critic, it selects the first prediction made by the actor.

H1: Can MMRC improve web navigation?

In this section, we examine the impact of MMRC's enhanced environment formulation with multimodal critic, by comparing its performance with MMRC-baseline. Table 4.1 compares the Step Success Rate of MMRC-baseline and MMRC. In all our experiments we see an improvement in performance with MMRC, validating the positive impact of using a critic agent. Specifically, we observe an improvement of up to 7.56%, 11.33%, and 4.85% over baselines in Cross Task, Cross Website, and Cross Domain splits.

For experiments with Gemini 1.5 Pro, we have used two different models as critics i) Gemini 1.5 Pro itself and ii) The Phi-3 Multimodal model [1]. Note that, the Phi-3 model was finetuned only on the textual predictions of the Mind2Web train dataset, the vision module was kept intact. Table 4.1 shows that Gemini 1.5 Pro achieves better improvement over performance, compared to the Phi-3 critic. This can be explained by the finetuning nature of our critic model, which only focused on the textual planning part without any finetuning to better understand the visual inputs. Gemini 1.5 Pro is a much larger model, so it is understandable that it will perform better reasoning compared to Phi-3. In Table 4.2, we observe that MMRC beats the baseline consistently across different metrics. For experiments with Gemini 1.0 Pro Vision, we see relatively a higher improvement compared to the Gemini 1.5 Pro actor. This suggests MMRC approach might be more effective for weaker foundation models and needs further investigation.

Base Actor	Method Name	Cross Task		Cross Website		Cross Domain	
		Step SR	Relative Improvement (%)	Step SR	Relative Improvement (%)	Step SR	Relative Improvement (%)
Gemini 1.5 Pro	Baseline - No Critic	28.83		26.89		28.87	
	MMRC with Phi-3 Vision Critic	29.72	3.09	27.38	1.82	30	3.91
	MMRC with Gemini 1.5 Pro Critic	30.69	6.45	28.56	6.21	30.27	**4.85**
Gemini 1.0 Pro Vision	Baseline - No Critic	19.05		15.44		21.37	
	MMRC with Gemini 1.0 Pro Vision Critic	20.49	**7.56**	17.19	**11.33**	22.35	4.59

Table 4.1: Relative improvement of MMRC over baseline actor. For the experiments with Gemini 1.5 Pro and Phi-3 Vision, we allowed atmost three iteration of actor-critic interation, while with Gemini 1.0 Pro Vision we allowed upto five iterations. We observe by using a critic agent MMRC achieved upto 7.56%, 11.33%, and 4.85% improvement over baselines that only uses actor.

H2: Can a critic, finetuned on the target task, yield better performance than a foundational model critic?

Table 4.2 shows all three metrics for all our experiments. For the experiments with Gemini 1.5 Pro, we observe that Phi-3 Vision critic achieves better Operation F1 in the Cross Website and Cross Domain splits. This shows that smaller critics finetuned on target tasks can achieve better results than much larger foundational models, while we hypothesize the lower improvement over the baseline for Element Accuracy might be attributed to not finetuning the visual module of Phi-3

Vision. This suggests using foundational models as actors and a custom smaller fine-tuned model on the target task might improve performance and a direction worth exploring in further detail.

Base Actor	Method Name	Cross Task			Cross Website			Cross Domain		
		Element Accuracy	Operation F1	Step SR	Element Accuracy	Operation F1	Step SR	Element Accuracy	Operation F1	Step SR
Gemini 1.5 Pro	Baseline - No Critic	38.54	71.65	28.83	37.1	71.78	26.89	36.63	74.5	28.87
	MMRC with Phi-3 Critic	39.36	71.7	29.72	33.1	**75.9**	27.38	38.2	**74.3**	30
	MMRC with Gemini 1.5 Pro Critic	**39.88**	**72.15**	**30.69**	**39.55**	72.75	**28.56**	**38.57**	73.7	**30.27**
Gemini 1.0 Pro Vision	Baseline - No Critic	22.3	72.04	19.05	18.79	70.26	15.44	24.7	73.83	21.37
	MMRC with Gemini 1.0 Pro Vision Critic	23.59	72.77	20.49	20.39	71.21	17.19	25.58	74.08	22.35

Table 4.2: Detailed evaluation metrics for all our experiments. MMRC outperforms the baseline in every evaluation metric, except for Element Accuracy in Cross Website split with Phi-3 critic. Phi-3 critic, which was finetuned on the training dataset, while keeping its cisual core intact, obtains better Operation F1 than Gemini 1.0 Pro, suggesting that a finetuned smaller model as critic can improve performance over a baseline and general-purpose foundational critic.

H3: How does MMRC perform compared to other multimodal methods?

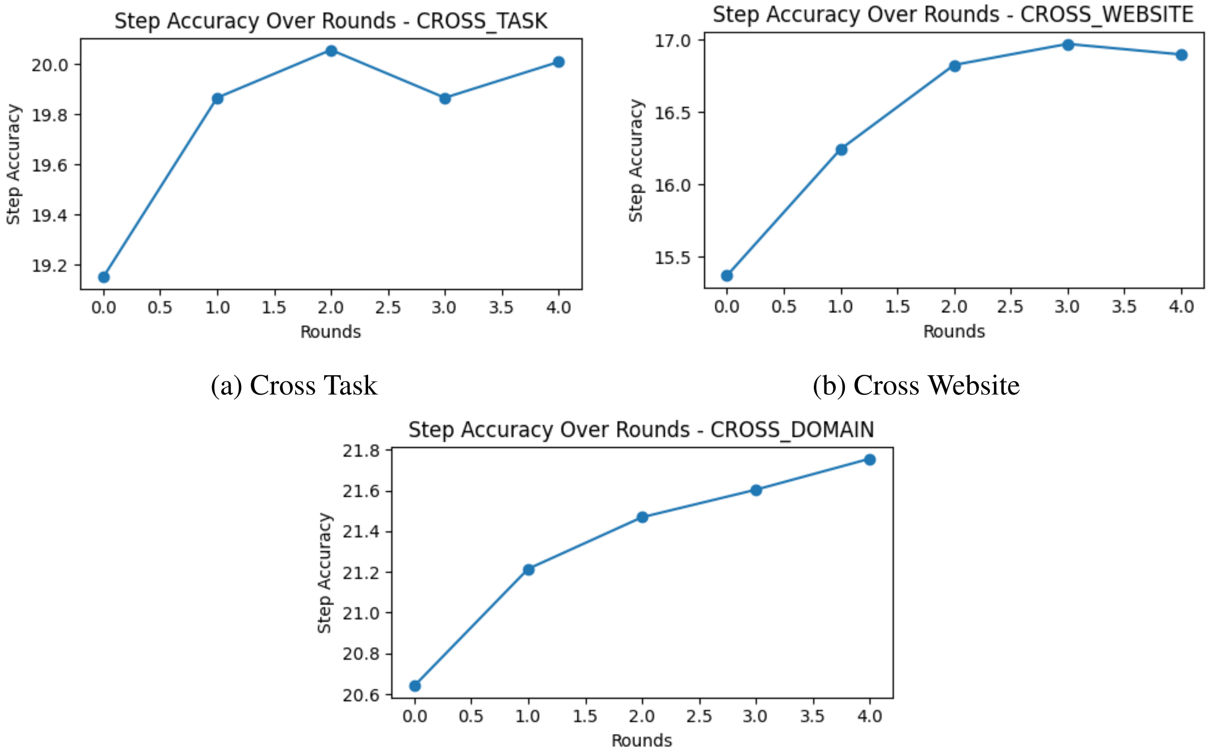
In Table 4.3, we show the results of some of the contemporary methods working on the Mind2Web dataset with visual inputs. SeeAct [105] with GPT-4V and a Human Oracle achieves the best performance. However, as the name suggests, the GPT-4V prediction was grounded manually by human supervision—which is the most challenging aspect of working with the visual inputs [105]. Excluding that, we observe that MMRC improves the previous best step-success-rate obtained by Gemini 1.0 Pro Vision by 4.54%, 11.6%, and 24.17% respectively on the Cross Task, Cross Website, and Cross Domain splits. We also notice MMRC with Gemini 1.5 Pro actor and Phi-3 Vision critic, outperforms GPT-4V on Operation F1 for Cross Website and Cross Domain splits. This also strengthens our case for using a finetuned smaller model as critic.

Method Name	Cross Task			Cross Website			Cross Domain		
	Element Accuracy	Operation F1	Step SR	Element Accuracy	Operation F1	Step SR	Element Accuracy	Operation F1	Step SR
SeeAct - with Gemini 1.0 Pro Vision - MCQ	21.5	67.7	19.6	17.1	61.3	15.4	20.7	64.3	18
SeeAct - with GPT4 + MCQ	**46.4**	**73.8**	**40.2**	38	67.8	**32.4**	**42.4**	69.3	**36.8**
SeeAct - with GPT4 + Human Oracle	66.4	79.2	61.9	69.5	78.9	65	72.8	73.6	62.1
MMRC with Gemini 1.0 Pro Vision	23.59	72.77	20.49	20.39	71.21	17.19	25.58	74.08	22.35
MMRC with Gemini Pro 1.5 Actor and Phi-3 Vision Critic	39.36	71.7	29.72	33.1	**75.9**	27.38	38.2	**74.3**	30
MMRC with Gemini Pro 1.5	39.88	72.15	30.69	**39.55**	72.75	28.56	38.57	73.7	30.27

Table 4.3: Comparison between MMRC and contemporary multimodal approaches on Mind2Web. MMRC improve the previous best step-success-rate obtained by Gemini 1.0 Pro Vision by 4.54%, 11.6%, and 24.17% respectively on the Cross Task, Cross Website, and Cross Domain splits. Also, MMRC with Gemini 1.5 Pro actor and Phi-3 Vision critic, outperforms GPT-4V on Operation F1 for Cross Website and Cross Domain splits.

H4: What is the impact of the number of iterations of the actor-critic interaction loop?

In Figure 4.4, we evaluate the impact of the number of rounds of actor-critic interactions on MMRC with Gemini 1.0 Pro Vision. On the Y-axis, we plot Step Success Rate/Accuracy, and on the X-axis, we plot the number of rounds starting from 0. The performance initially increases with more actor-critic interactions, but after a certain number of interactions, the performance plateaus or declines.



(a) Cross Task

(b) Cross Website

(c) Cross Domain

Figure 4.4: Impact of the number of rounds of actor-critic interactions on MMRC with Gemini 1.0 Pro Vision. On the Y-axis, we plot Step Success Rate/Accuracy, and on the X-axis, we plot the number of rounds starting from 0. The performance initially increases with more actor-critic interactions, but after a certain number of interactions, the performance plateaus or declines.

4.6 Conclusion: Limitations and Future Work

In summary, we have presented a novel multimodal actor-critic framework for web navigation. Unlike previous methods that require complex HTML summarization and grounding techniques for visual input, we employ a simple MCQ-based grounding technique and critic-based reasoning to enhance web navigation on a complex, real-world website benchmark Mind2Web [18]. We also demonstrate promising improvements in the performance of general-purpose foundational models when used in conjunction with smaller critic models through task-specific fine-tuning.

In the future, we aim to fine-tune the critic model on both visual and textual ground truths across multiple benchmarks to investigate the impact of task-specific smaller critics on large general-purpose foundational models. In our current implementation, some HTML elements have textual descriptions that are not meaningful; for example, image elements without alternative texts are reduced to just '<img>'. When multiple elements share the same text and type, the actor cannot distinguish between them. Therefore, improving the textual descriptions could further enhance MMRC's performance.

Acknowledgments

The research work presented in this chapter is performed in collaboration with Izzeddin Gur, Hiroki Furuta, and Aleksandra Faust from Google DeepMind; Jai Sharma, Prof. Pieter Abbeel, and Prof. Ion Stoica from UC Berkeley. It is a collaboration under the Google-BAIR (Berkeley Artifcial Intelligence Research) Commons program.

Chapter 5

Conclusion and Future Work

In this dissertation, our objective is to develop techniques that enhance the generalization capabilities of autonomous agents by formulating and generating environments for a range of complex and realistic sequential decision-making problems. Chapter 2 delves into human-guided systematic environment generation using Scenic4RL. In Chapter 3, we introduce CLUTR, a novel unsupervised curriculum learning algorithm designed to automatically generate environments that address sample inefficiency and improve generalization for traditional deep neural network-based agents trained with Reinforcement Learning (RL). Chapter 4 presents MMRC, which explores how environment generation can enhance the performance of recent foundational Large Language Models (LLMs) in web navigation tasks through a novel multimodal actor-critic framework. Together, Scenic4RL, CLUTR, and MMRC introduce innovative approaches that demonstrate how environment formulation and generation can significantly enhance performance of autonomous agents.

Looking ahead, I plan to focus on advancing more effective reasoning and planning techniques for multimodal, multi-task autonomous assistants.

5.1 Advancements in Reasoning and Planning Techniques

Despite the recent remarkable progress of LLMs, there is still much to achieve in terms of reasoning and understanding. LLMs are primarily designed to predict the next token based on previously generated tokens. However, due to their training on vast amounts of data and strong language modeling capabilities, they exhibit strong surface-level reasoning. Nevertheless, new techniques are needed to achieve more powerful and generalizable reasoning. I aim to explore novel methods to further improve reasoning and planning, specifically focusing on multimodal reasoning techniques that effectively combine reasoning across multiple modalities. In the near future, I will concentrate on designing attachable, task-specific small head-agents that can be integrated with general-purpose foundational agents to enhance task-specific generation based on user needs—an approach similar to what we explored at a surface level in Chapter 4 with the fine-tuned critic. I believe that approaches utilizing a general-purpose multimodal foundational model or API, combined with multiple customized attachable heads to serve different tasks, will soon provide cost-effective generalization for multi-task autonomous assistants.

5.2 Environment/Data Generation for Multi-Task Self-Refining Agents

Current AI systems depend significantly on large, high-quality datasets, which are expensive, and data is often scarce for many practical applications. To address these limitations, there is a growing emphasis on developing autonomous learning frameworks. These frameworks enable AI to generate its own data or produce data for another agent through simulations or interactions with the environment. The capability of AI to autonomously generate data is particularly crucial for the development of task-specific agents. As we design AI systems for diverse and evolving tasks, the need for adaptable and self-sufficient data generation becomes more pronounced, allowing us to meet the ever-increasing and dynamic demands of novel tasks. Building upon my previous research, which focused on task-specific environment generation techniques, my objective is to enhance the generalization capabilities of AI agents. By improving these techniques, AI systems can better adapt to a wider range of tasks and scenarios, ultimately leading to more robust and versatile agents.

Bibliography

[1] Marah Abdin et al. *Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone*. 2024. arXiv: 2404.14219 [cs.CL]. URL: https://arxiv.org/abs/2404.14219.

[2] Abdus Salam Azad et al. "Clutr: Curriculum learning via unsupervised task representation learning". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 1361–1395.

[3] Abdus Salam Azad et al. "Programmatic Modeling and Generation of Real-Time Strategic Soccer Environments for Reinforcement Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 6. 2022, pp. 6028–6036.

[4] Marc G Bellemare et al. "The arcade learning environment: An evaluation platform for general agents". In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 253–279.

[5] Christopher Berner et al. *Dota 2 with Large Scale Deep Reinforcement Learning*. 2019.

[6] Christopher Berner et al. "Dota 2 with large scale deep reinforcement learning". In: *arXiv preprint arXiv:1912.06680* (2019).

[7] Samuel R Bowman et al. "Generating sentences from a continuous space". In: *arXiv preprint arXiv:1511.06349* (2015).

[8] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[9] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[10] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. *Minimalistic Gridworld Environment for Gymnasium*. https://github.com/Farama-Foundation/MiniGrid. 2018.

[11] Daesol Cho, Seungjae Lee, and H Jin Kim. "Outcome-directed Reinforcement Learning by Uncertainty & Temporal Distance-Aware Curriculum Goal Generation". In: *arXiv preprint arXiv:2301.11741* (2023).

[12] Jan Chorowski et al. "Unsupervised Speech Representation Learning Using WaveNet Autoencoders". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.12 (2019), pp. 2041–2053. DOI: 10.1109/TASLP.2019.2938863.

[13] Karl Cobbe et al. "Leveraging procedural generation to benchmark reinforcement learning". In: *International conference on machine learning*. 2020.

[14] Karl Cobbe et al. "Quantifying generalization in reinforcement learning". In: *International Conference on Machine Learning*. 2019.

[15] Karl Cobbe et al. "Quantifying generalization in reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1282–1289.

[16] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG]. URL: https://arxiv.org/abs/2110.14168.

[17] Edward Kim Daniel Fremont. *Scenic interfaced simulators*. https://scenic-lang.readthedocs.io/en/latest/simulators.html. Accessed: 2021-10-01. 2021.

[18] Xiang Deng et al. "Mind2web: Towards a generalist agent for the web". In: *Advances in Neural Information Processing Systems* 36 (2024).

[19] Michael Dennis et al. "Emergent complexity and zero-shot transfer via unsupervised environment design". In: *Advances in neural information processing systems* 33 (2020), pp. 13049–13061.

[20] Prafulla Dhariwal et al. *OpenAI Baselines*. https://github.com/openai/baselines. 2017.

[21] Yan Duan et al. "Benchmarking deep reinforcement learning for continuous control". In: *International conference on machine learning*. 2016.

[22] Lasse Espeholt et al. "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures". In: *International conference on machine learning*. PMLR. 2018, pp. 1407–1416.

[23] FIFA. *Laws of the Game*. https://ussoccer.app.box.com/s/xx3byxqgodqtl1h15865/file/850765570638. Accessed: 2021-10-01. 2021.

[24] Carlos Florensa et al. "Automatic goal generation for reinforcement learning agents". In: *International conference on machine learning*. PMLR. 2018, pp. 1515–1528.

[25] Foretellix. *Measurable Scenario Description Language*. https://www.foretellix.com/wp-content/uploads/2020/07/M-SDL_LRM_OS.pdf. Accessed: 2021-10-01. 2020.

[26] Daniel Fremont, Tommaso Dreossi, and et al. "Scenic: a language for scenario specification and scene generation". In: *PLDI*. ACM, 2019, pp. 63–78.

[27] Daniel Fremont, Edward Kim, and et al. "Scenic: A Language for Scenario Specification and Data Generation". In: *CoRR* abs/2010.06580 (2020). arXiv: 2010.06580. URL: https://arxiv.org/abs/2010.06580.

[28] Daniel J. Fremont et al. "Scenic: A Language for Scenario Specification and Data Generation". In: *Machine Learning Journal* (2022).

[29] Hiroki Furuta et al. *Multimodal Web Navigation with Instruction-Finetuned Foundation Models*. 2024. arXiv: 2305.11854 [cs.LG]. URL: https://arxiv.org/abs/2305.11854.

[30] Ran Gong et al. *MindAgent: Emergent Gaming Interaction*. 2023. arXiv: 2309.09971 [cs.AI]. URL: https://arxiv.org/abs/2309.09971.

[31] Ishaan Gulrajani et al. "Pixelvae: A latent variable model for natural images". In: *arXiv preprint arXiv:1611.05013* (2016).

[32] Izzeddin Gur et al. *A Real-World WebAgent with Planning, Long Context Understanding, and Program Synthesis*. 2024. arXiv: 2307.12856 `[cs.LG]`. URL: https://arxiv.org/abs/2307.12856.

[33] Izzeddin Gur et al. *Understanding HTML with Large Language Models*. 2023. arXiv: 2210.03945 `[cs.LG]`. URL: https://arxiv.org/abs/2210.03945.

[34] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International Conference on Machine Learning*. 2018.

[35] Matthew Hausknecht et al. "Half field offense: An environment for multiagent learning and ad hoc teamwork". In: *AAMAS Adaptive Learning Agents (ALA) Workshop*. Vol. 3. sn. 2016.

[36] Pengcheng He et al. *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. 2021. arXiv: 2006.03654 `[cs.CL]`. URL: https://arxiv.org/abs/2006.03654.

[37] Mark Hendrikx and et al. "Procedural content generation for games: A survey". In: *ACM Transactions on Multimedia Computing, Communications, and Applications*. Vol. 9. 2013.

[38] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: (2016).

[39] Wenyi Hong et al. *CogAgent: A Visual Language Model for GUI Agents*. 2023. arXiv: 2312.08914 `[cs.CV]`. URL: https://arxiv.org/abs/2312.08914.

[40] Peide Huang et al. "Curriculum Reinforcement Learning using Optimal Transport via Gradual Domain Adaptation". In: *arXiv preprint arXiv:2210.10195* (2022).

[41] Unnat Jain, Ziyu Zhang, and Alexander G Schwing. "Creativity: Generating diverse questions using variational autoencoders". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6485–6494.

[42] Nick Jakobi. "Evolutionary robotics and the radical envelope-of-noise hypothesis". In: *Adaptive behavior* 6.2 (1997), pp. 325–368.

[43] Junyan Jiang et al. "Transformer VAE: A Hierarchical Model for Structure-Aware and Interpretable Music Representation Learning". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 516–520. DOI: 10.1109/ICASSP40776.2020.9054554.

[44] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. "Prioritized level replay". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 4940–4950.

[45] Minqi Jiang et al. "Replay-guided adversarial environment design". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 1884–1897.

[46] Niels Justesen et al. "Illuminating generalization in deep reinforcement learning through procedural level generation". In: *arXiv preprint arXiv:1806.10729* (2018).

[47] Dmitry Kalashnikov et al. "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation". In: *Conference on Robot Learning*. 2018.

[48] Aditya Kalyanpur et al. *LLM-ARC: Enhancing LLMs with an Automated Reasoning Critic*. 2024. arXiv: 2406.17663 [cs.CL]. URL: https://arxiv.org/abs/2406.17663.

[49] Alex Kendall et al. "Learning to drive in a day". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8248–8254.

[50] Byoungjip Kim et al. *Prospector: Improving LLM Agents with Self-Asking and Trajectory Ranking*. 2024. URL: https://openreview.net/forum?id=YKK1jXEWja.

[51] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. *Language Models can Solve Computer Tasks*. 2023. arXiv: 2303.17491 [cs.CL]. URL: https://arxiv.org/abs/2303.17491.

[52] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[53] Pascal Klink et al. "Curriculum reinforcement learning via constrained optimal transport". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 11341–11358.

[54] Karol Kurach et al. "Google research football: A novel reinforcement learning environment". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 4501–4510.

[55] Kimin Lee et al. "Context-aware dynamics model for generalization in model-based reinforcement learning". In: *International Conference on Machine Learning*. 2020.

[56] Kimin Lee et al. "Network randomization: A simple technique for generalization in deep reinforcement learning". In: *International Conference on Learning Representations*. 2020.

[57] Marlos C Machado et al. "Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents". In: *Journal of Artificial Intelligence Research* 61 (2018), pp. 523–562.

[58] Rupak Majumdar et al. *Paracosm: A Language and Tool for Testing Autonomous Driving Systems*. 2019. arXiv: 1902.01084.

[59] Eric Mazumdar et al. *Policy-Gradient Algorithms Have No Guarantees of Convergence in Linear Quadratic Games*. 2019. DOI: 10.48550/ARXIV.1907.03712. URL: https://arxiv.org/abs/1907.03712.

[60] Microsoft. *Fine-Tuning Vision Models with Phi-3*. https://github.com/microsoft/Phi-3CookBook/blob/main/md/04.Fine-tuning/FineTuning_Vision.md. Accessed: 2024-08-08. 2024.

[61] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), p. 529.

[62] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[63] Jun Morimoto and Kenji Doya. "Robust reinforcement learning". In: *Neural computation* 17.2 (2005), pp. 335–359.

[64] Michael E Mortenson. *Mathematics for computer graphics applications*. Industrial Press Inc., 1999.

[65] Ashvin V Nair et al. “Visual Reinforcement Learning with Imagined Goals”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[66] Sanmit Narvekar et al. “Curriculum learning for reinforcement learning domains: A framework and survey”. In: *arXiv preprint arXiv:2003.04960* (2020).

[67] Alex Nichol et al. “Gotta learn fast: A new benchmark for generalization in rl”. In: *arXiv preprint arXiv:1804.03720* (2018).

[68] Jack Parker-Holder et al. “Evolving Curricula with Regret-Based Environment Design”. In: *arXiv preprint arXiv:2203.01302* (2022).

[69] Lerrel Pinto, James Davidson, and Abhinav Gupta. “Supervision via competition: Robot adversaries for learning tasks”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 1601–1608.

[70] Rémy Portelas et al. “Automatic curriculum learning for deep rl: A short survey”. In: *arXiv preprint arXiv:2003.04664* (2020).

[71] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. “Generating diverse high-fidelity images with vq-vae-2”. In: *Advances in neural information processing systems* 32 (2019).

[72] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic backpropagation and approximate inference in deep generative models”. In: *International conference on machine learning*. PMLR. 2014, pp. 1278–1286.

[73] Fereshteh Sadeghi and Sergey Levine. *CAD2RL: Real Single-Image Flight without a Single Real Image*. 2016. DOI: 10.48550/ARXIV.1611.04201. URL: https://arxiv.org/abs/1611.04201.

[74] Mikayel Samvelyan and et al. “The StarCraft Multi-Agent Challenge”. In: *Workshop on Deep Reinforcement Learning at the 33rd Conference on Neural Information Processing Systems*. 2019.

[75] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[76] Younggyo Seo et al. “Trajectory-wise Multiple Choice Learning for Dynamics Generalization in Reinforcement Learning”. In: *arXiv preprint arXiv:2010.13303* (2020).

[77] Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. “Toward Verified Artificial Intelligence”. In: *Communications of the ACM* 65.7 (2022), pp. 46–55.

[78] Peter Shaw et al. *From Pixels to UI Actions: Learning to Follow Instructions via Graphical User Interfaces*. 2023. arXiv: 2306.00245 [cs.LG]. URL: https://arxiv.org/abs/2306.00245.

[79] Peter Shaw et al. *From Pixels to UI Actions: Learning to Follow Instructions via Graphical User Interfaces*. 2023. arXiv: 2306.00245 `[cs.LG]`. URL: https://arxiv.org/abs/2306.00245.

[80] Noah Shinn et al. *Reflexion: Language Agents with Verbal Reinforcement Learning*. 2023. arXiv: 2303.11366 `[cs.AI]`. URL: https://arxiv.org/abs/2303.11366.

[81] Mohit Shridhar et al. *ALFWorld: Aligning Text and Embodied Environments for Interactive Learning*. 2021. arXiv: 2010.03768 `[cs.CL]`. URL: https://arxiv.org/abs/2010.03768.

[82] David Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (2018), pp. 1140–1144.

[83] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[84] David Silver et al. "Mastering the game of go without human knowledge". In: *Nature* 550.7676 (2017), p. 354.

[85] Abishek Sridhar et al. *Hierarchical Prompting Assists Large Language Model on Web Navigation*. 2023. arXiv: 2305.14257 `[cs.CL]`. URL: https://arxiv.org/abs/2305.14257.

[86] Peter Stone and et al. "Keepaway Soccer: From Machine Learning Testbed to Benchmark". In: *RoboCup 2005: Robot Soccer World Cup IX*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 93–105.

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

[88] Yujin Tang, Duong Nguyen, and David Ha. "Neuroevolution of self-interpretable agents". In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 2020, pp. 414–424.

[89] Yuval Tassa et al. "Deepmind control suite". In: *arXiv preprint arXiv:1801.00690* (2018).

[90] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[91] Alberto Uriarte and Santiago Ontanón. "A benchmark for starcraft intelligent agents". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 11. 2. 2015, pp. 22–28.

[92] Oriol Vinyals et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (2019), pp. 350–354.

[93] Oriol Vinyals et al. *StarCraft II: A New Challenge for Reinforcement Learning*. 2017. arXiv: 1708.04782 `[cs.LG]`.

[94] Rui Wang et al. "Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9940–9951.

[95] Rui Wang et al. "Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions". In: *arXiv preprint arXiv:1901.01753* (2019).

[96] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 `[cs.CL]`. URL: https://arxiv.org/abs/2201.11903.

[97] Zhilin Yang et al. *HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering*. 2018. arXiv: 1809.09600 `[cs.CL]`. URL: https://arxiv.org/abs/1809.09600.

[98] Shunyu Yao et al. *WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents*. 2023. arXiv: 2207.01206 `[cs.CL]`. URL: https://arxiv.org/abs/2207.01206.

[99] Denis Yarats et al. "Improving sample efficiency in model-free reinforcement learning from images". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 12. 2021, pp. 10674–10681.

[100] Amy Zhang, Nicolas Ballas, and Joelle Pineau. "A dissection of overfitting and generalization in continuous reinforcement learning". In: *arXiv preprint arXiv:1806.07937* (2018).

[101] Amy Zhang, Yuxin Wu, and Joelle Pineau. "Natural environment benchmarks for reinforcement learning". In: *arXiv preprint arXiv:1811.06032* (2018).

[102] Bin Zhang et al. *Controlling Large Language Model-based Agents for Large-Scale Decision-Making: An Actor-Critic Approach*. 2024. arXiv: 2311.13884 `[cs.AI]`. URL: https://arxiv.org/abs/2311.13884.

[103] Mingtian Zhang, Andi Zhang, and Steven McDonagh. "On the out-of-distribution generalization of probabilistic image modelling". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 3811–3823.

[104] Mingtian Zhang et al. "Improving VAE-based Representation Learning". In: *arXiv preprint arXiv:2205.14539* (2022).

[105] Boyuan Zheng et al. "Gpt-4v (ision) is a generalist web agent, if grounded". In: *arXiv preprint arXiv:2401.01614* (2024).

[106] Longtao Zheng et al. *Synapse: Trajectory-as-Exemplar Prompting with Memory for Computer Control*. 2024. arXiv: 2306.07863 `[cs.AI]`. URL: https://arxiv.org/abs/2306.07863.