Université de Montréal

Enhancing Agent Learning through World Dynamics Modeling

par

Zhiyuan Sun

Department of Computer Science
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Discipline

August 30, 2024

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Enhancing Agent Learning through World Dynamics Modeling

présenté par

Zhiyuan Sun

a été évalué par un jury composé des personnes suivantes :

Jian-Yun Nie
(président-rapporteur)

Bang Liu
(directeur de recherche)

Glen Berseth
(membre du jury)

Résumé

Le développement rapide de l'intelligence artificielle (IA), allant des modèles comme BERT aux modèles de fondation à grande échelle, illustre la croissance exponentielle de la taille et des capacités des modèles, stimulée par les avancées en puissance de calcul et la disponibilité des données. Les modèles de fondation, qui tirent parti de l'apprentissage auto-supervisé sur d'énormes ensembles de données non étiquetées, ont montré une polyvalence remarquable dans une large gamme de tâches, du traitement du langage à la représentation des connaissances. Cependant, leur dépendance à des données de grande envergure, principalement issues d'Internet, introduit un « écart de connaissances »—un décalage entre les connaissances généralisées acquises pendant l'entraînement et les connaissances spécialisées nécessaires pour des domaines spécifiques. Cet écart est principalement causé par des informations insuffisantes, trompeuses ou superficielles disponibles lors de l'entraînement, ce qui peut mener à des sorties peu fiables, surtout dans des contextes de données rares ou de mauvaise qualité.

Pour relever ce défi, nous introduisons le cadre Discover, Verify, and Evolve (**DiVE**). DiVE est conçu pour améliorer la compréhension des modèles de fondation en les dotant de connaissances profondes et adaptées aux tâches en aval.

Le cadre fonctionne en trois étapes :

- **Découvrir l'information** : Extraire des informations pertinentes et utiles pour pallier le manque de données qui limite la compréhension des modèles dans des domaines spécialisés.
- **Vérifier l'information** : Valider les informations recueillies afin de filtrer les inexactitudes et les biais, garantissant ainsi que seules des connaissances fiables sont retenues.
- **Faire évoluer l'information** : Affiner et développer les informations vérifiées pour obtenir des connaissances plus approfondies, améliorant ainsi la capacité du modèle à traiter des requêtes complexes et à performer avec précision dans des tâches spécialisées.

En s'attaquant aux causes profondes de l'écart de connaissances, DiVE aide les modèles de fondation à passer d'une compréhension générale à une expertise spécialisée, comblant le fossé entre formation et application. Cette approche améliore la précision des modèles à travers les domaines et renforce leurs capacités de prise de décision. Dans cette thèse, nous démontrons l'efficacité de DiVE à travers des évaluations empiriques, soulignant son potentiel à améliorer l'adaptabilité et la robustesse des modèles de fondation dans des scénarios réels.

Mots-clés: Agents basés sur LLM, Ancrage linguistique et planification, Apprentissage incarné.

Abstract

The rapid evolution of artificial intelligence (AI) from models like BERT to large-scale foundation models illustrates the exponential growth in model sizes and capabilities, driven by advances in computational power and data availability. Foundation models, which leverage self-supervised learning on vast, unlabelled datasets, have shown remarkable versatility across a wide range of tasks, from language processing to knowledge representation. However, their reliance on large-scale, predominantly internet-sourced data introduces a "knowledge ga"—a mismatch between the generalized knowledge acquired during training and the specialized knowledge required for specific domains. This gap is primarily caused by insufficient, misleading, or superficial information available during training, which can lead to unreliable outputs, especially in low-data or poor-data settings.

To address this challenge, we introduce the Discover, Verify, and Evolve (**DiVE**) framework. DiVE is designed to enhance the understanding of foundation models by equipping them with deep, tailored knowledge about downstream tasks. The framework operates in three stages:

- **Discover the Information**: Extract relevant and useful information to address the lack of data that limits the model's understanding of specialized domains.
- **Verify the Information**: Validate the gathered information to filter out inaccuracies and biases, ensuring only reliable knowledge is retained.
- **Evolve the Information**: Refine and expand on verified information to gain deeper insights, improving the model's ability to handle complex queries and perform accurately in specialized tasks.

By addressing the root causes of the knowledge gap, DiVE helps foundation models transition from general understanding to specialized expertise, bridging the gap between training and application. This approach enhances model accuracy across domains and improves decision-making capabilities. In this thesis, we demonstrate the efficacy of DiVE through empirical evaluations, highlighting its potential to enhance the adaptability and robustness of foundation models in real-world scenarios.

Keywords: LLM-based Agents, Language Grounding and Planning, Embodied Learning

Contents

List of tables

List of figures

Liste des sigles et des abréviations

LLMs	Large Language Models
FMs	Foundation Models
DiVE	**Di**scover, **V**erify, and **E**volve
SLAM	Simultaneous Localization and Mapping
VLN	Vision-and-Language Navigation
$\mathcal{K}_{\text{data}}$	LLMs' embedded knowledge
$\mathcal{K}_{\text{target}}$	Essential knowledge for the target domain
$\mathcal{K}_{\text{rel}}$	LLMs' knowledge toward the target domain
$\mathcal{K}_{+}$	LLM's accurate knowledge toward the target domain
$\mathcal{K}_{-}$	LLM's inaccurate knowledge toward the target domain

R	Recall
P	Precision
D	Depth
$\mathcal{W}$	World dynamics
H	Human demonstrations
$\tilde{\tau}_i$	Verbalized observation
π	LLM-based agent
$\mathcal{S}$	State space
$\mathcal{A}$	Action space
$\mathcal{P}$	Probability distribution over the action space
o_t	Current observation
s_t	Current state
a_t	Current action

SG	Sub-goal sequence
sg_i	Current sub-goal
st	All sub-tasks
st_t	Current subtask
$\mathcal{I}$	Strategies
$\tilde{\mathcal{I}}$	Strategy candidates

Remerciements

Embarking on this research journey was undoubtedly challenging, but it would not have been possible without the support of those who guided and encouraged me along the way. I would like to express my deepest gratitude to my supervisor, Bang Liu, for his guidance. I am also profoundly thankful to my family, as well as Xingdi (Eric) Yuan, Marc-Alexandre Côté, and Glen Berseth, for their support and advice throughout this journey. Additionally, I extend my gratitude to my lab mates at MILA, Haochen Shi, and others, for their support. Lastly, I am grateful to Glen Berseth and Jian-Yun Nie for being on my thesis committee.

While my research focuses on foundational models, the support I received highlighted what these models cannot do—and probably never will.

Chapter 1

Introduction

The scaling from BERT [**1**], which was trained on a few GPUs, to foundation models [**2, 3, 4, 5**], which require thousands of GPUs, illustrates how model sizes have grown exponentially, along with their capabilities.

1.1. The Foundation Models

Before the advent of foundation models, the concept of a single model excelling across various domains seemed improbable. Traditional machine learning models [**6, 7**] were typically designed and optimized for specific tasks using specialized datasets [**8, 9, 10**] in a supervised learning manner. Tasks such as image recognition or object detection required distinct models, each meticulously trained on carefully curated data. This approach resulted in a landscape of highly specialized, task-specific solutions, with limited generalizability to other tasks or domains.

Foundation models, however, break this mold by leveraging self-supervised learning on vast amounts of unlabelled data. This approach enables them to acquire a broad and deep understanding of language and knowledge. Unlike traditional models that rely solely on labeled datasets, foundation models learn from the structure and patterns within the data itself. By training on large-scale datasets that encompass a diverse range of topics and formats—often sourced from the internet—these models develop a comprehensive understanding of language, concepts, and even world knowledge.

1.2. The Knowledge Gap

Foundation models have demonstrated remarkable capabilities across a wide range of tasks [**11, 12**]. However, they often encounter challenges in low-data and poor-data settings.

Due to their autoregressive nature, these models have a tendency to "memorize" the training data. When only limited, misleading, or superficial information is available for a specific domain, the models may produce unreliable outputs. This occurs because they lack the contextual understanding needed to distinguish between high-quality, relevant information and inaccurate or incomplete data.

We formalize these challenges as a "knowledge gap", which refers to the discrepancy between what the model has learned during training and what is required for success in a particular target domain. In other words, the knowledge gap represents the mismatch between the model's general understanding and the specialized knowledge necessary for a specific task.

The underlying causes of this knowledge gap can be categorized into three main areas:

- **Insufficient Information from the Internet**: Foundation models are primarily trained on large-scale datasets sourced from the internet, which may not include detailed or specialized information about niche domains.
- **Misleading Information on the Internet**: The vast amount of data available online contains a significant proportion of inaccurate or biased information. During training, models may inadvertently learn from these unreliable sources, incorporating false or misleading knowledge that can negatively impact their performance in specialized contexts.
- **Superficial Information from the Internet**: Even when information is relevant and accurate, it is frequently presented at a superficial level. Models trained predominantly on internet data may develop only a shallow understanding of topics, lacking the depth required to address complex queries or make nuanced decisions.

In this thesis, we aim to bridge this knowledge gap by developing methods that enhance foundation models' understanding of downstream tasks.

1.3. DiVE

To address these challenges, we introduce the framework Discover, Verify, and Evolve (**DiVE**). DiVE not only learn the knowledge about the downstream tasks but also evolves deep knowledge tailored for downstream domains. By equipping foundation models with this comprehensive knowledge, DiVE bridges the understanding gap between foundation models and downstream domains, thereby enhancing their decision-making capabilities.

To tackle the root causes of the knowledge gap, DiVE consists of three core components, forming a robust framework that strengthens foundatio models' ability to understand and navigate complex domains:

- **Discover the Information**: Leveraging the demonstrations, our goal is to uncover as much useful information as possible. This step is crucial to address the issue of insufficient information present in the training data, ensuring a comprehensive understanding of the domain.
- **Verify the Information**: In this phase, we focus on validating the gathered information, rigorously filtering out any unreliable or inaccurate data. By retaining only verified and trustworthy information, we build a solid and dependable knowledge base.
- **Evolve the Information**: With a foundation of reliable information, we aim to further develop and refine this knowledge. This involves evolving the information into deeper, more complex insights, fostering a profound understanding of the downstream tasks and enhancing the model's ability to perform accurately in specialized contexts.

Together, these components aim to provide comprehensive and deep knowledge about downstream tasks to foundation models, thereby bridging the potential knowledge gap.

Chapter 2

Literature Reviews

In this chapter, we provide the comprehensive literature reviews about the landscape of embodied agent that are related to theDiVE and the high-level novelty proposed within DiVE.

The literature review is mainly composed with three components:

(1) LLMs' Surprising Reasoning Ability

(2) Building an Embodied Agent

(3) Novelty ofDiVE Comparing with Prior Works

2.1. LLMs' Surprising Reasoning Ability

By training autoregressively in a left-to-right sequence, where each token is predicted based on the preceding ones from an internet-scale corpus, LLMs seem to gain a nuanced understanding of language and the world it represents [**2, 13, 5, 4, 14**]. This understanding allows LLMs to perform decently in tasks that require reasoning ability, such as question-answering, planning, and mathematics.

Although LLMs demonstrate impressive performance by absorbing vast amounts of internet knowledge, their reasoning abilities can be further optimized through various post-hoc methods without accessing or altering their underlying weights.

2.1.1. Enhancing Reasoning through Prompt Engineering

Incorporating the phrase "Let's think step by step" and providing examples of sequential solutions helps guide LLMs to break tasks into smaller subtasks, enhancing their reasoning abilities [**15**]. The Chain of Thought (CoT) approach significantly improves reasoning, and the self-consistency method further refines this by generating multiple reasoning paths and selecting the most consistent one through majority voting [**16**]. Building on this, the Tree of Thoughts (ToT) method explores solution paths in a tree-like structure, allowing the model to evaluate and choose the most promising branches [**17**]. The Graph of Thoughts (GoT) extends this concept by enabling LLMs to consider multiple reasoning paths simultaneously in a graph-based structure, leading to more comprehensive solutions [**18**].

Further enhancing reasoning, Program-Aided Language Models (PAL) use programmatic logic to break down complex tasks, improving problem-solving [**19**]. The ReAct method integrates reasoning with actions, allowing LLMs to dynamically adjust based on feedback [**20**]. Multimodal Chain of Thought extends CoT to handle both text and visual information, enabling multimodal reasoning [**21**]. Additionally, systems like [**22**] enhance reasoning by focusing on multiple dimensions of information simultaneously.

In parallel, methods like KATE optimize few-shot learning by selecting the most relevant examples [**23**]. Other approaches refine prompt generation and selection: AutoPrompt uses a gradient-guided search to automate prompt creation [**24**], while LM-BFF leverages prompt ensembling for better performance [**25**]. Prefix-Tuning and Prompt Tuning optimize prompts with minimal parameter adjustments, improving task performance with reduced computational costs [**26, 27**]. PTR combines manual templates with search algorithms for effective prompt selection [**28**], and [**29**] shows that LLMs can act as human-level prompt engineers, generating optimal prompts using their internal reasoning.

2.1.2. Augmenting LLMs with External Tool Integration

Incorporating external tools like code interpreters, physical simulators, and APIs significantly enhances the capabilities of LLMs, enabling them to handle complex tasks that require extensive computation and memory, areas where LLMs often face challenges. By delegating these tasks to external tools, LLMs can also benefit from external feedback, leading to continuous improvement.

[**30**] highlight the benefits of integrating physical simulators with LLMs, enabling these models to perform grounded reasoning. By interacting with simulated environments and interpreting the results, LLMs can better understand outcomes, leading to improved performance. Similarly,[**31, 19, 32**] demonstrate how LLMs, when combined with code interpreters, can execute structured programs to solve tasks involving logical deduction and computation. In these cases, the LLMs generate high-level code, while the code interpreters manage the necessary low-level logic. On the other hand, [**33**] introduce a framework in

which LLMs autonomously determine when and how to invoke external tools. Trained to select the appropriate API, decide when to call it, pass the correct arguments, and incorporate the results into future predictions, LLMs can handle a broader range of tasks without human intervention. Without fine-tuning LLMs, [**34**] demonstrate how LLMs can integrate multiple tools in a coordinated, multi-step reasoning process. This approach enables LLMs to solve intricate problems by automatically generating intermediate reasoning steps as a program.

2.2. Building an Embodied Agent

The development of embodied agents represents a significant leap in the field of artificial intelligence, where the goal is to create systems capable of perceiving, reasoning, and acting within complex environments. These agents are not merely passive observers; they must interact with their surroundings, make decisions based on sensory inputs, and adapt to dynamic conditions in real-time.

Building such agents requires a deep integration of various components, including perception, planning, learning, and control. However, the challenges involved in constructing these agents vary significantly depending on the nature of the embodied environments in which they operate. Different environments demand distinct strategies and technologies, influencing how agents are designed and trained. In the following subsections, we will explore the role of embodied environments, examine approaches that utilize domain-specific knowledge, and discuss the integration of LLMs to enhance the capabilities of embodied agents.

2.2.1. Understanding and Designing Embodied Environments

Embodied environments are crucial for the development and evaluation of embodied agents, as they provide the simulated or real-world contexts within which these agents operate. These environments are designed to replicate the complexities of the real world, allowing agents to learn, adapt, and demonstrate capabilities that are transferable to real-world tasks.

By simulating daily tasks [**35, 36, 37**], these environments focus on activities such as cleaning, cooking, and organizing, which require a deep understanding of both language and the physical world. Building on the ALFRED environment [**35**], ALFWorld [**38**] extends this concept by aligning text-based inputs with embodied environments. This alignment enables agents to interact with their surroundings in more meaningful ways, effectively bridging the gap between language and action while eliminating the need for visual input.

On the other hand, Minecraft-based environments [**39, 40**] provide a distinctive platform for developing and testing embodied agents within open-ended, highly interactive worlds. These environments capitalize on the versatility and complexity of Minecraft, where agents can engage in a broad array of tasks, from resource gathering to construction and exploration. Unlike more narrowly focused environments, Minecraft-based platforms offer a vast sandbox that fosters creativity and problem-solving, enabling agents to develop generalized skills that are applicable across various real-world scenarios.

In a more simplified setting, environments like Crafter [**41**] and MiniGrid [**42**] provide controlled, modular platforms that are ideal for studying specific aspects of agent behavior. Crafter is a lightweight yet challenging environment that emphasizes resource management and survival in a procedurally generated 2D world. Inspired by elements of Minecraft but with a simplified interface, Crafter tasks agents with collecting resources, crafting tools, and surviving in a hostile environment filled with dangers like wild animals and hunger. The environment's focus on long-term planning, resource optimization, and survival strategies helps agents develop robust decision-making skills that can be generalized to more complex, real-world situations. MiniGrid is a grid-based, partially observable environment that offers a variety of tasks, from simple object manipulation to more complex goal-directed behavior. MiniGrid's emphasis on partial observation and memory-based challenges makes it particularly valuable for studying how agents can learn to navigate and make decisions with limited information.

2.2.2. Leveraging Domain-Specific Knowledge in Embodied Agents

In the realm of embodied agents, training approaches that leverage domain-specific knowledge have been developed to enable agents to interact effectively with their environment. These methods can be broadly categorized into self-supervised, planning-based, sequence-to-sequence, and reinforcement learning approaches.

Self-supervised methods utilize pretext tasks to provide additional training supervision without requiring extra annotations. For example, [**43**] proposed tasks such as predicting depth views and detecting loop closures to aid navigation. These auxiliary tasks accelerate learning and enhance visual perception, as demonstrated by [**44**], which improved visual encoding and segmentation. Additionally, self-supervised learning has been employed to encourage exploration through curiosity-driven objectives.

Planning-based methods often involve building maps of unknown environments while addressing localization challenges. SLAM techniques [**45**], such as Neural SLAM [**46**], integrate deep learning to enhance localization capabilities. Active Neural SLAM [**47**] introduces hierarchical planning structures, achieving state-of-the-art performance in navigation challenges.

Sequence-to-sequence methods, initially proposed by [**48**] for vision-and-language navigation (VLN), face challenges with stability and generalization. To address these issues, RPA [**49**] introduced a "lookahead" module for future state prediction. Further improvements include data augmentation and the integration of reinforcement learning with imitation learning, as seen in [**50**]. Advanced models like the Regretful Agent and frameworks incorporating asynchronous search have been developed to enhance navigation by allowing agents to backtrack and correct mistakes. Recent research has also explored the role of natural language in embodied AI, emphasizing the importance of language understanding for better generalization and navigation performance.

Reinforcement learning methods train agents to navigate end-to-end without explicit environment modeling, using objectives such as imitation learning and reinforcement learning. For instance, [**50**] were among the first to employ deep learning for feature matching and deep reinforcement learning for policy prediction, improving generalization to new environments. The Successor Representation approach further enhanced interaction with objects by considering object states and scene descriptions. To improve temporal encoding, [**51**] introduced an LSTM layer, and other efforts have focused on augmenting visual inputs with instance features and diverse visual representations.

2.2.3. Integrating LLMs for Enhanced Embodied Intelligence

The embedded knowledge within LLMs holds significant promise for developing embodied agents capable of tackling complex tasks that often require multi-hop reasoning. An embodied agent using LLMs is defined as an autonomous system that integrates language processing to navigate and interact within its environment. These agents leverage LLMs to interpret instructions, execute multi-step reasoning, and adapt to dynamic tasks with enhanced flexibility and sophistication.

Naively prompting LLMs for task decomposition, as demonstrated in [**52**], shows that LLMs can effectively extract actionable knowledge without requiring prior training on specific tasks. This approach leverages the LLM's inherent ability to understand and process complex instructions, enabling agents to break down tasks into manageable steps in a zero-shot manner. Building on this foundation, [**53**] further explored how LLMs can be guided to not only decompose tasks but also describe, explain, and plan actions within intricate environments. Moreover, the introduction of pre-defined skill libraries [**54, 55**] has shown that LLMs can evolve new and advanced skills from existing ones, significantly enhancing overall agent performance.

In addition to skill evolution, recent work [**56, 57**] has introduced mechanisms for LLMs to store state-aware memory and retrieve relevant information during the decision-making process. This memory integration allows agents to maintain a coherent understanding of their environment over time, making decisions that are more informed and contextually appropriate. Further enhancing the capabilities of embodied agents, [**58**] explored the translation of natural language instructions into executable robotic commands, which underscores the importance of LLMs in bridging the gap between human language and robotic action.

Recent research also emphasizes the role of LLMs in enabling agents to understand and manipulate objects in physical environments. For instance, [**59**] investigates how LLMs can guide agents to learn from interaction data, resulting in more effective and adaptive behaviors in dynamic settings. Similarly, [**60**] explores the potential of LLMs in robot learning, where agents are trained to perform complex tasks by interpreting high-level commands and executing precise actions.

On the other hand, prompting LLMs for a denser reward function, as explored in [**61, 62, 63**], has shown promising results in refining the reinforcement learning process for embodied agents. These approaches utilize the LLM's ability to understand nuanced task descriptions and generate more granular reward signals, guiding agents toward more precise and effective behaviors.

2.3. Novelty of DiVE Comparing with Prior Works

Research Focus: Most prior methods [**57, 56, 64, 55**] have concentrated on enhancing the performance of LLMs by leveraging past trajectories and environment feedback through self-reflection. These approaches operate under the assumption that LLMs possess a comprehensive understanding of the downstream environment. However, our work diverges by addressing the knowledge gap in LLMs through learning the dynamics of the downstream environment. By learning these dynamics, such as associations like “grass can be found near trees, water, or paths” and “grass is not associated with diamonds, coal, or iron” and practical rules like “you can walk directly through grass”, we aim to improve LLMs’ general understanding of their operational environment before they are deployed.

Methodology Difference: Given that LLMs can produce hallucinations, the discovered dynamics might contain inaccuracies, as highlighted in our results. Therefore, unlike methods [**57, 56, 55**] that rely on LLMs to score experiences or use ground-truth data to filter incorrect rules, our approach emphasizes the importance of verifying dynamics before applying them. Valid dynamics, once confirmed, should not conflict with sampled trajectories, and LLMs can effectively identify such inconsistencies. Furthermore, as the number of trajectories grows, so does the number of discovered dynamics. Our method merges all verified dynamics, retaining only reliable ones and reducing the size of the dynamics library, whereas previous methods see an increase in redundant tokens as trajectories accumulate.

Structured Learning from the Trajectories: We introduce a structured learning approach that acquires environmental knowledge from trajectories using a bottom-up process, starting with basic actions and objects, progressing to subtasks, and culminating in subgoals. This structured learning not only ensures the acquisition of necessary knowledge at each level but also addresses the challenge of long-term planning by extracting essential planning information. In contrast, prior method [**57, 56, 64, 55**] often assume that LLMs already possess comprehensive knowledge at every level.

Evolve for State-Aware Strategies: Finally, while previous approaches [**57, 56, 64, 55**] typically retrieve the most similar experiences or append all experiences, our method asks LLMs to derive a strategy for the current state in real-time, using all discovered and verified dynamics. This approach, as demonstrated in our experiments, not only improves performance but also uncovers novel and insightful strategies, such as using collected stones to build a protective barrier around a safe location during nighttime—a strategy not previously present in the knowledge library.

Chapter 3

Enhancing Agent Learning through World Dynamics Modeling

In this chapter, we present the motivation behind DiVE, describe its methodology in detail, and evaluate its performance through a series of experiments.

3.1. Introduction

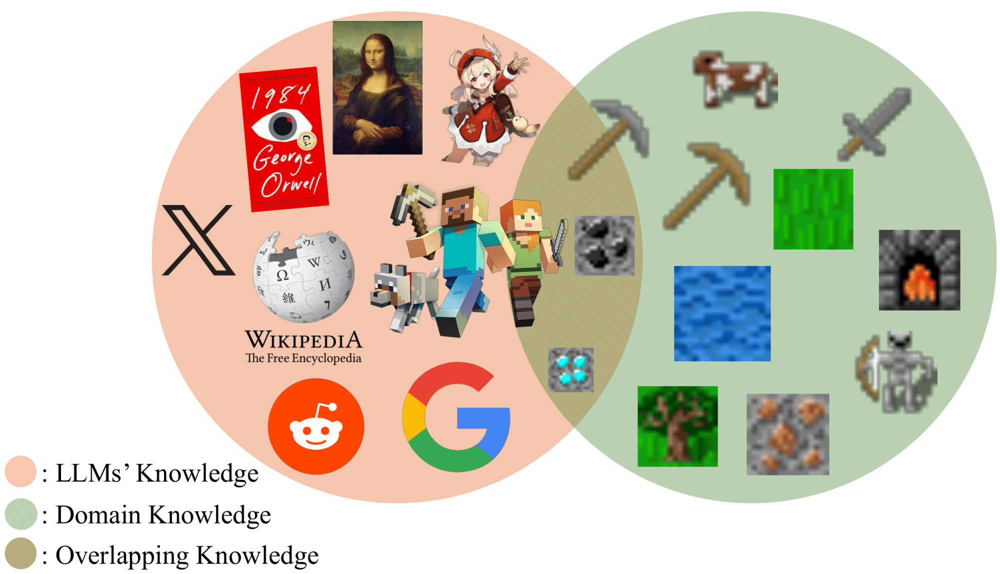


Fig. 3.1. The gap in knowledge between LLMs and the target domain.Even though LLMs are trained on vast internet-scale data, they may still not possess the required domain-specific expertise.

Large Language Models (LLMs) acquire a sophisticated understanding of world dynamics by sequentially absorbing vast amounts of information from the internet [**2, 3, 65, 13**]. This deep comprehension enables them to excel in tasks such as question answering [**11, 66, 12**], planning [**67, 65, 52**], and commonsense reasoning [**20, 17, 18**]. However, achieving this level of proficiency hinges on the diversity, precision, and comprehensiveness of the training data, ensuring that it captures the necessary domain-specific information.

A deficiency in such data can lead to a knowledge gap between LLMs and the targeted domain, as shown in Figure 3.1. Since LLMs tend to replicate the most common patterns in their training datasets [**68**], it is crucial that domain-specific data is not only available online but also appears frequently enough to be learned. For emerging or niche downstream tasks [**41, 42**], meeting this requirement can be challenging, often resulting in suboptimal decision-making by LLMs.

Effective decision-making requires access to high-quality, domain-specific information, which can be difficult to find and validate within the vast expanse of internet data [**69**]. The process of identifying and validating this information against authoritative domain knowledge is both labor-intensive and error-prone. Therefore, equipping LLMs with the detailed and accurate knowledge necessary for effective decision-making remains a significant challenge. Even when LLMs have a general understanding of a domain, optimal decision-making requires deep, context-specific knowledge. By providing such detailed information, LLMs' comprehension of both the environment and the current state can be significantly enhanced. For example, mastering the game of Go requires not just an understanding of its basic rules but also strategies tailored to specific board situations. This specialized knowledge can vary not only across different domains but also within different states of the same domain, making it impractical to compile offline.

To address these challenges, we introduce the framework **Di**scover, **V**erify, and **E**volve (**DiVE**). Drawing on the concept of World Models [**70, 71, 72**], DiVE not only identifies and verifies fundamental world dynamics from demonstrations but also evolves state-specific knowledge for downstream domains. By equipping LLMs with this comprehensive knowledge, DiVE bridges the understanding gap between LLMs and downstream domains, thereby improving their decision-making capabilities.

- **The Discoverer**: This component iteratively uncovers the environment's dynamics through provided demonstrations. Starting with the basic dynamics of actions and objects, it progresses to identifying the steps of subtasks, and ultimately determines the topological sequence of subgoals. This curriculum learning approach ensures both efficiency and a comprehensive understanding of the environment.

- **The Verifier**: To maintain the integrity of the knowledge base, this component filters out inaccuracies resulting from the LLMs' tendency to generate erroneous information, ensuring that only accurate and reliable data is retained.
- **The Evolver**: This component explores state-specific dynamics, such as strategic plays or contextual decisions. By doing so, it equips the agent with actionable knowledge tailored to the current state, thus enhancing real-time decision-making.

In the Crafter environment [**41**], DiVE learns comprehensive and accurate dynamics from demonstrations, guiding the agent through the decision-making process by developing in-depth knowledge. This guidance enables the agent to outperform strong baselines, including the best-reported systems, and to achieve rewards on par with human players. We also present quantitative and qualitative analyses that provide insights into DiVE's behavior, demonstrating the effectiveness of the proposed framework.

In summary, our main contribution is a general framework that autonomously learns world dynamics from demonstrations and aids LLMs in decision-making by evolving contextual knowledge. This approach not only enhances LLMs' capabilities in specific domains but also provides a flexible foundation for creating agents capable of mastering complex tasks across various environments through continuous learning and refinement.

3.2. Problem Formulation

We acknowledge that precisely measuring the overlap between the LLM's knowledge and the target domain is challenging. Therefore, we adopt these concepts from Information Retrieval and related fields [**73**] to guide our system's design.

3.2.1. The Knowledge Gap

In this study, we consider a scenario where a pre-trained LLM-based agent is tasked with solving challenges in the Crafter environment. We define $\mathcal{K}_{\text{data}}$ as the set of knowledge encapsulated in the LLM through its training (including both pre-training and fine-tuning), and $\mathcal{K}_{\text{target}}$ as the comprehensive set of knowledge required for success in the Crafter environment.

For the LLM to be effective, we aim to maximize $\mathcal{K}_{\text{rel}}$, the subset of $\mathcal{K}_{\text{data}}$ relevant to $\mathcal{K}_{\text{target}}$. Additionally, we strive for $\mathcal{K}_{\text{rel}}$ to contain more accurate knowledge $\mathcal{K}_{+}$ than inaccurate knowledge $\mathcal{K}_{-}$, with $\mathcal{K}_{\text{rel}} = \mathcal{K}_{+} \cup \mathcal{K}_{-}$.

Thus, we establish three desirable properties for our system design:

- **Recall** $R = \frac{|\mathcal{K}_+|}{|\mathcal{K}_{\text{target}}|}$ quantifies the extent to which the LLM's knowledge covers the knowledge required to solve tasks in the target domain. A low recall generally indicates a significant distribution gap between the LLM's training data and the target domain.
- **Precision** $P = \frac{|\mathcal{K}_+|}{|\mathcal{K}_{\text{rel}}|}$ assesses the accuracy of the LLM's knowledge when applied to the target domain.
- **Depth** D evaluates the levels of abstraction within the knowledge representations, ranging from basic game mechanics to high-level strategies.

3.2.2. Challenges of Fine-tuning

In an ideal scenario, addressing the knowledge gap in a large language model (LLM) would involve fine-tuning the model to adapt it to a specific target domain. This approach capitalizes on the model's inherent ability to learn and internalize domain-specific nuances by re-training it on data relevant to that domain. The advantages of such an approach include improved performance on specialized tasks, such question-answering [**74, 75, 76**], aligning with human preference [**77, 78, 79**], grounding LLMs [**80**].

However, this fine-tuning process is not without its challenges, which often make it impractical:

- **Necessity of Abundant Annotated Data**: Fine-tuning an LLM requires a significant amount of annotated data from the target domain to achieve meaningful improvements. For many specialized fields, such annotated data is scarce or costly to produce. Domains like medical research or legal analysis require expert annotation, which is both time-consuming and expensive [**81, 82**]. Without adequate data, the model may fail to generalize well to the domain, leading to suboptimal performance.
- **Significant Computational Resources**: Fine-tuning large models requires substantial computational resources, including powerful GPUs or TPUs and large memory capacities, which significantly increases both the cost and time needed for training. This challenge is particularly acute for smaller organizations or research teams that may not have access to such resources [**76, 75**]. To address these challenges, techniques like low-rank adaptation (LoRA) have been proposed, which reduce the number of trainable parameters and thus lower the computational burden while still achieving competitive performance in domain-specific tasks [**76**]. However, as model

size often correlates with performance, the increasing scale of modern LLMs further exacerbates the demand for computational resources, making efficient fine-tuning even more challenging.

- **Catastrophic Forgetting**: During the fine-tuning, there is the risk of catastrophic forgetting, where the model loses previously acquired knowledge while adapting to the new domain [**83**]. This phenomenon occurs because the fine-tuning process often involves updating the model's weights based on new domain-specific data, which can inadvertently overwrite the representations and knowledge learned from the original, broader dataset. This trade-off can be particularly problematic in scenarios where the model needs to retain general knowledge while also performing well in a specialized domain [**84**]. Strategies such as elastic weight consolidation (EWC) [**85**] have been proposed to address this issue, helping to preserve previous knowledge while allowing for domain-specific adaptation [**83**]
- **Restricted Access to Model Weights**: An increasing number of large language models are becoming closed-source [**2, 14, 3, 65**], limiting access to their underlying weights. This restriction prevents researchers and developers from fine-tuning these models for specific domains or tasks [**86**]. Consequently, fine-tuning must rely on alternative approaches such as API-based prompt engineering or the use of smaller, open-source models, which may not achieve the same performance as direct fine-tuning. This trend presents a significant barrier to fine-tuning efforts.

To address the knowledge gap and challenges associated with direct fine-tuning of LLMs, we propose DiVE, a method designed to bridge this gap by learning environment dynamics. Further details on this approach are provided in Section 3.4.

3.3. Environment

In this section, we provide a brief overview of the environments used for evaluation. DiVE is assessed within hierarchical achievement environments, specifically Crafter [**41**] and MiniGrid [**42**].

(a) Crafter Environment

(b) MiniGrid Environment

Fig. 3.2. Screenshots of the Crafter and MiniGrid Environments.

3.3.1. Hierarchical Achievement Environments

Environments characterized by hierarchical achievements [**41, 42**], denoted as $C_i \in \mathcal{C}$, function interactively. At each time step t, an agent observes $o_t \in \mathcal{O}$ and must choose an action $a_t \in \mathcal{A}$. Subsequently, the environment generates a new observation o_{t+1}, reflecting the resulting state. Within any given environment C_i, the achievement structure is represented as a directed acyclic graph (DAG), $\mathcal{G} = (V, E)$.

Here, each vertex $v \in V$ corresponds to an individual achievement, while each edge $(u, v) \in E$ signifies that achievement v is contingent upon the prior achievement u. Achieving v requires unlocking all its prerequisite achievements. The overarching objective is to maximize the number of achievements unlocked, adhering to these dependency relationships.

3.3.2. The Crafter Environment

We employ the Crafter environment as a benchmark for evaluating an agent's performance in contexts involving hierarchical achievements. The Crafter environment [**41**] presents a complex, long-horizon challenge where the ultimate objective is to collect a diamond, a task that typically requires the agent to take hundreds of steps. The environment is structured around a technology tree, a hierarchical set of achievements that the agent must unlock in sequence to progress toward the final goal. Therefore, the agent must strategically determine the topological order of these achievements, ensuring that all prerequisite achievements are completed before advancing to higher-level tasks.

The environment is further complicated by its survival elements. The agent must manage limited resources, such as food and tools, while also defending itself against hostile creatures that can appear at any moment. Resource management is critical, as failing to maintain sufficient food levels or allowing tools to degrade can result in the agent's failure to survive. The combination of these survival requirements and the hierarchical nature of the technology tree creates a highly challenging environment where the agent must balance immediate needs with long-term objectives.

3.3.3. The MiniGrid Environment

In addition to the Crafter environment, we utilize the MiniGrid environment as another benchmark for evaluating an agent's performance, specifically in settings that emphasize efficient exploration, reasoning, and memory. MiniGrid is a collection of lightweight, grid-based environments designed to test an agent's ability to navigate and solve tasks in a partially observable setting. Each grid world is composed of a series of rooms, corridors, and objects, with the agent needing to find an optimal path to complete the task at hand.

MiniGrid environments are particularly notable for their focus on minimal visual complexity, which shifts the emphasis toward the agent's cognitive abilities rather than sensory processing. This makes MiniGrid an ideal environment for testing the underlying reasoning and planning capabilities of an agent, especially those involving hierarchical decision-making. The environment typically presents the agent with a series of rooms connected by doors, with keys and other objects placed within. The agent must explore the grid, locate keys, open doors, and reach a designated goal location, all while operating under partial observability where only a portion of the grid is visible at any given time.

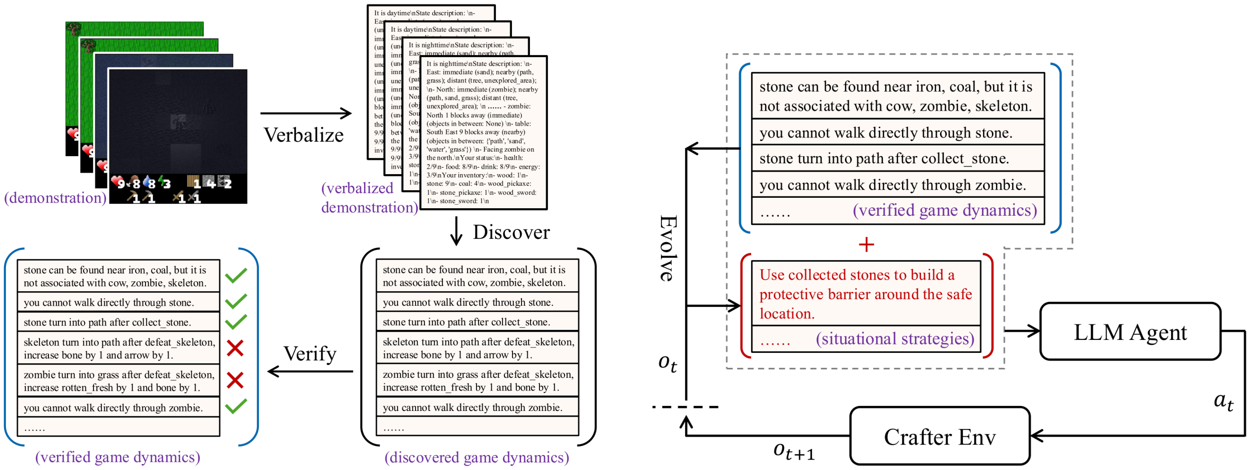


Fig. 3.3. The overall pipeline of DiVE. **Left**: Learning basic game dynamics from offline demonstrations (Section 3.4). We highlight the incorrect game dynamics identified by the Verifier (marked with ×), which indicate that the LLM may be hallucinating false facts, possibly due to memorization of Minecraft data. **Right**: Learning situational strategies through online interactions (Section 3.4). For clarity, the verbalization process is omitted in the right figure.

3.4. Methodology

To address the knowledge gap between LLMs and downstream environments, we introduce our framework, DiVE. This framework is designed to bridge this gap by focusing on three essential properties: recall, precision, and depth. Notably, DiVE achieves this without requiring extensive data collection from the target domain. Instead, it employs a prompt-based approach that allows the model to effectively learn world dynamics $\mathcal{W}$ from the environment.

As illustrated in Figure 3.3, DiVE begins by leveraging a small set of human demonstration trajectories, represented as $H = \{\tau_i\}_{i=0}^N = \{\{o_t^i, a_t^i, r_t^i\}_{t=0}^{|\tau_i|}\}_{i=0}^N$. Each trajectory τ_i consists of a sequence of observations o_t^i, actions a_t^i, and rewards r_t^i over time steps t. To make these observations more interpretable by the LLM, each observation o_t is converted into the language space using a Verbalizer function, resulting in $\tilde{o}_t = \text{Verbalizer}(o_t)$. This transformation converts the human demonstration trajectories into language-encoded trajectories $\tilde{\tau}_i = \{\tilde{o}_t^i, a_t^i, r_t^i\}_{t=0}^{|\tau_i|}$.

In the subsequent phase, the Discoverer component of DiVE extracts a set of world dynamic candidates, denoted as $\tilde{\mathcal{W}} = \{\tilde{W}_+, \tilde{W}_-\}$, from the human demonstrations H. Here,

$\tilde{W}_+$ includes the correct world dynamics, while $\tilde{W}_-$ comprises the incorrect or misleading world dynamics.

In essence, we prompt LLMs with trajectories to uncover game dynamics at varying levels of abstraction:

- **Action-level dynamics**: We prompt LLMs with transitions (two consecutive frames and the action in between) to identify action-level dynamics.
  - *Discovered example*: Stone turns into a path after collecting stone.
- **Object-level dynamics**: We use transitions involving specific objects to prompt LLMs to discover dynamics related to those objects.
  - *Discovered example*: Stone can be found near iron, coal, and diamond.
- **Subtask-level dynamics**: We prompt LLMs with a sequence of transitions that complete a subtask, helping to uncover subtask-level dynamics.
  - *Discovered example*: Step 1 for collecting stone: Locate a stone block within immediate reach by navigating the environment.
- **Subgoal-level dynamics**: We prompt LLMs with a sequence of completed subtasks within a trajectory to reveal subgoal-level dynamics.
  - *Discovered example*: Subgoal 1: Collect 4 units of wood while maintaining adequate levels of health, food, drink, and energy.

It is empirically inevitable to encounter incorrect dynamics $\tilde{W}_-$ within the set $\tilde{\mathcal{W}}$. This can occur due to limitations in the backbone LLM, which may struggle to extract meaningful knowledge from the trajectory data or might generate hallucinated, inaccurate information. To mitigate this, we introduce the Verifier component, which filters out potentially invalid or conflicting world dynamic candidates from $\tilde{\mathcal{W}}$. The valid dynamics that remain after this filtering process are denoted as $\mathcal{W}$.

Once a reliable set of world dynamics $\mathcal{W}$ is established, the Evolver component is employed to derive advanced game-playing strategies $\mathcal{I}$. These strategies are formulated based on the world dynamics $\mathcal{W}$ and the language descriptions of the observations $\tilde{o}_t$.

The final decision-making process for selecting primitive actions $a_t \in \mathcal{A}$ is hierarchically decomposed into several planning tasks. These tasks involve determining sub-goals (SG), sub-tasks (ST), and actions ($\mathcal{A}$). The planning procedure is further guided by both the world dynamics $\mathcal{W}$ and the strategies $\mathcal{I}$. In situations where $\mathcal{W} \neq \varnothing$, the properties of recall (R), precision (P), and depth (D) are guaranteed to improve as follows:

$$R: \frac{|\mathcal{K}_{+}|}{|\mathcal{K}_{\text{target}}|} \xRightarrow[H]{\text{Discoverer}} \frac{|\mathcal{K}_{+}| + |\mathcal{W}|}{|\mathcal{K}_{\text{target}}|} \tag{3.4.1}$$

$$P: \frac{|\mathcal{K}_{+}|}{|\mathcal{K}_{\text{rel}}|} \xRightarrow[\tilde{\mathcal{W}},H]{\text{Verifier}} \frac{|\mathcal{K}_{+}| + |\mathcal{W}|}{|\mathcal{K}_{\text{rel}}| + |\mathcal{W}|} \tag{3.4.2}$$

$$D: \varnothing \xRightarrow[\text{Verifier}]{\text{Discoverer}} \mathcal{W} \xRightarrow[\mathcal{W},H]{\text{Evolver}} \mathcal{I} \cup \mathcal{W} \tag{3.4.3}$$

DiVE is composed of two major components: learning basic game dynamics from offline demonstrations (Figure 3.3, left) and learning situational strategies from online interactions (Figure 3.3, right).

3.4.1. Learning from Offline Demonstrations

The objective of the offline dynamics learning process is to align a Large Language Model's (LLM's) understanding with the foundational rules of the Crafter world by acquiring knowledge of its dynamics, represented as $\mathcal{W}$. This acquired understanding acts as a prior to inform subsequent decision-making processes. Unlike methods that derive world dynamics from human-authored game manuals or handbooks [**64, 87**], which may be impractical in many real-world situations, our approach focuses on learning world dynamics $\mathcal{W}$ directly from observed experiences H.

Curriculum Learning

Given the varying levels of complexity involved in understanding the dynamics of different elements within Crafter, we adopt a curriculum learning strategy [**88**]. This strategy involves a sequential learning process that progresses from simpler to more complex dynamics, thereby enhancing the learning process. Specifically, we propose a method for learning the dynamics of each element within a task decomposition hierarchy, denoted as $TD = \{\mathcal{A}+(O), ST, SG\}$, where O represents the set of objects in Crafter. Our approach begins with elements of lower abstraction, such as actions $a \in \mathcal{A}$, and advances to elements of higher abstraction, such as sub-goals $sg_i \in SG$. The sequence of sub-goals, $SG = [sg_1, sg_2, \ldots]$, follows an ordered progression required to unlock achievements in the achievement graph $\mathcal{G} = (V, E)$, where SG is determined by performing a topological sort on $\mathcal{G}$, with each sg_i corresponding to a vertex in V. We utilize the Discoverer to extract this order from human demonstrations H. Subtasks ST are defined as the nodes in the achievement graph G, i.e., $ST = V$, and completing a sub-goal sg_i may require accomplishing several sub-tasks multiple times. This structured approach ensures a logical progression through the tasks, enhancing both the learning and application of Crafter's dynamics.

Discover

The Discoverer is designed to identify dynamic candidates $\tilde{\mathcal{W}}$ related to the elements within the task decomposition hierarchy TD. A single dynamics discovery step for an element $E \in TD$ involves the following three main steps:

(1) **Construction of a Semantic Experience Bank (B^E)**: For each element E, a semantic experience bank B^E is created from human demonstrations H. This bank stores experiences that have been transformed from H into a suitable granularity for analyzing dynamics related to E. The transformation process involves chunking and summarizing the verbalized human demonstrations $\tilde{H}$ to capture essential semantic details.

(2) **Sampling of Relevant Experiences**: For each attribute of an instance $e \in E$, a subset of experiences B_e^E that are relevant to the instance e is sampled from B^E.

(3) **Identification of Dynamic Candidates**: A dynamic candidate $\tilde{w}$ is identified from the subset B_e^E by recognizing patterns that are consistent across all experiences within B_e^E.

The action-level semantic experience bank, denoted as $B^{\mathcal{A}}$, stores transition tuples derived from verbalized human demonstrations $\tilde{H}$, specifically $B^{\mathcal{A}} = \{\{\tilde{o}_t, a_t, \tilde{o}_{t+1}\}_i\}_{i=1}^{|B^{\mathcal{A}}|}$. Similarly, the object-level semantic experience bank B^O collects individual observations related to specific object interactions from $\tilde{H}$, stored as $B^O = \{\tilde{o}_i\}_{i=1}^{|B^O|}$. The sub-task-level semantic experience bank B^{ST} aggregates trajectory segments representing the completion of sub-tasks, formatted as $B^{ST} = \{\{\tilde{o}_t, \ldots, a_{tst}, \tilde{o}_{tst+1}\}_i\}_{i=1}^{|B^{ST}|}$, where tst denotes the time step at which a sub-task $st \in ST$ is completed. For sub-goals, the sub-goal-level experience bank B^{SG} records sequences of sub-tasks leading to the completion of sub-goals, expressed as $B^{SG} = \{\{st_t, \ldots, st_{tsg}\}_i\}_{i=1}^{|B^{SG}|}$, where tsg is the time step at which the sub-goal sg is achieved.

For action-level dynamics learning, the relevant experiences $B_a^{\mathcal{A}}$ are compiled by randomly sampling transition tuples from $B^{\mathcal{A}}$ where the action a is successfully executed. A similar approach is applied to the dynamic learning for other elements within the task decomposition hierarchy TD. For action-level dynamics discovery, we identify the prerequisites and outcomes of each action (e.g., The action MakeWoodPickaxe requires 1 wood). For an example, see Figure 3.4. For object-level dynamics, we concentrate on the co-occurrence relationships between objects and their occurrences over time. The attribute set for a sub-task typically includes the general steps required to complete the task, along with its prerequisites, outcomes, and termination conditions. The only attribute of interest for a sub-goal is its correct position within the sub-goal sequence SG.

Algorithm 1: Pseudo-code for the Discoverer Algorithm

Input: Verbalized human demonstrations $\tilde{H}$, current element $E \in TD$ for discovering, dynamics of E's parents $\{\mathcal{W}_V\}_{V \in \text{Parents}(E)}$

Constant: Discovery steps s, action space $\mathcal{A}$, object set O, semantic trajectory transformation function T^E, dynamic distillation function D^E, semantic sampling function S_E.

Result: $\tilde{\mathcal{W}}_E$

```
W̃_E ← {};
B^E ← T^E(H̃) ;                     /* B^E is the semantic experience bank. */
for k ← 1 to s do
    for e ∈ E do
        for x ∈ n.attributes do
            B^E_e ← S_E(n, B^E);
            y ← D^E(x, B^E_e, {W_V});
            W̃_E ← W̃_E ∪ {y}
        end
    end
end
```

Dynamic discovery processes are susceptible to various sources of noise, such as confounding factors, hallucinations by the language model (LLM), and the LLM's limitations in extracting meaningful knowledge from trajectory data. To mitigate these issues, we introduce a dynamic verifier designed to filter out noisy dynamic candidates from the set $\tilde{\mathcal{W}}$.

In the Crafter environment, the world dynamics are unique.
You are tasked with discovering the unique dynamics of actions within this environment.
Given the state transitions described as follows: {partial description}, identify the materials used in the inventory for executing this action.
To discover the materials used in the inventory, consider the following aspects:

- List all the materials used not gained for each state transition and their corresponding quantities;
- List all the common materials used in all state transitions and their corresponding quantities.

If no materials are used in common, simply output as "None".
materials are: wood, stone, coal, iron, diamond, sapling.
Output as many pre-conditions as you can think of in the following format: {output format}

Fig. 3.4. An example prompts for the Discoverer to Discover the Action's attribute

Verifier

For each candidate $w \in \tilde{\mathcal{W}}$, the verifier begins by selecting a subset of relevant semantic experiences B_e^E from the associated semantic experience bank B_E. Here, w represents a dynamic candidate linked to a specific attribute of an instance $e \in E$, where $E \in \mathcal{A} + (O), ST, SG$ corresponds to an element of the task decomposition hierarchy TD. The verification process then evaluates whether w consistently holds across all experiences within B_e^E and does not contradict any established dynamics. Candidates that fail to meet these criteria are discarded, while those that pass are classified as verified dynamics, represented as $\mathcal{W}$. See Figure 3.5 for an example.

3.4.2. Learning Situational Strategies

To effectively integrate the learned world dynamics, denoted as $\mathcal{W}$, into the Crafter environments, we deploy an agent based on a Large Language Model (LLM), represented by the function $\pi : \mathcal{S} \times \mathcal{W} \rightarrow \mathcal{P}(\mathcal{A})$. Here, $\mathcal{S}$ refers to the state space, $\mathcal{A}$ represents the action space, and $\mathcal{P}$ is the probability distribution over the action space. Instead of directly mapping the world dynamics $\mathcal{W}$ and the current state observation o_t to the action a_t, we address the challenge of long-horizon planning by employing an online strategy learning approach. This method decomposes the planning process into three key tasks: sub-goal planning, sub-task planning, and action planning.

In the crafter environment, you have discovered unique dynamics and need to verify the pre-conditions for the action '{action}'.
Given the state transitions before and after taking the action {action}, described as follows {sampled descriptions}
You are asked to verify the immediate objects pre-conditions for the action 'action' based on the discovered dynamics.
Precondition: You only need {precondition} within immediate distance.
For the verification, you need to consider the followings:

- List all the objects within immediate distance before executing the action for each state transition only from the object list.
- Identify and list the objects that are present in all state transitions within the immediate distance before executing the action as common immediate objects, only from the object list.
- The common object **must** be present in all state transitions' immediate objects.
- Determine if there are objects mentioned in the pre-condition that are not in the common immediate objects.
- If there are objects mentioned in the pre-condition that are not in the common immediate objects, then it is invalid; otherwise, it is valid.

Object list: [coal, cow, diamond, furnace, iron, lava, skeleton, stone, table, tree, water, zombie, plant]
Output in this format: {output format}

Fig. 3.5. An example prompts for the Verifier to Verify the Action's attribute

Sub-goal Planning

Given a sequence of sub-goals, $SG = [sg_1, sg_2, \ldots]$, derived from human demonstrations H and treated as fixed, we employ a simple heuristic for sub-goal planning. As each sub-goal is achieved, the current sub-goal is updated to the next uncompleted sub-goal in the sequence SG.

Given the following details:

- Subtask description
- Current observation
- Initial observation
- Previous executed actions

you are asked to decide whether the subtask should be terminated or not.
For deciding whether to terminate the subtask, consider:

- The previous action, provided it was executed successfully.
- The difference between the initial and current observations, including the inventory changes.

The subtask should be terminated, and the output should be 'True' only if any of its termination conditions are met.
Otherwise, if none of the termination conditions are met, the subtask should continue running, and the output should be 'False'.
Justify whether the termination conditions are met or not first, and then provide the termination decision. Output in this format: {output format}.

Fig. 3.6. The prompt for the subtask planning

Sub-task Planning

For a given current sub-goal sg_i, we utilize an LLM-based sub-task planner. This planner assesses and ranks all potential sub-tasks $st \in ST$ based on the learned world dynamics $\mathcal{W}$, the verbalized current observation $\tilde{o}t$, and the most recently completed sub-task $stt-1$. The sub-task with the highest rank is selected as the current sub-task st_t. To ensure correct execution, the completion of a sub-task st depends on satisfying a specific termination condition. This condition is determined by querying the LLM using the current verbalized observation, the observation at the start of the sub-task, and the termination criteria for the current sub-task. The detailed prompt can be found in 3.6.

Evolve Situational Strategies

Beyond mastering the basic rules of the Crafter environment (i.e., world dynamics $\mathcal{W}$), we delve into the development of sophisticated game-playing strategies grounded in these dynamics. While learning the world dynamics is straightforward, the strategy space is often too expansive to explore exhaustively. To tackle this challenge, we propose evolving these dynamics into strategies $\mathcal{I}$ through an online learning approach. This approach narrows the search space by conditioning not only on the world dynamics $\mathcal{W}$ but also on the verbalized current observation $\tilde{o}_t$ and the sub-task st_i. This focused method generates strategies that are more relevant and adaptive to the current game scenarios compared to those derived through offline methods.

To achieve this, we have developed an LLM-based Evolver that formulates strategy candidates $\tilde{\mathcal{I}}$ using deductive reasoning based on $\mathcal{W}$. Specifically, the Evolver generates these strategy candidates by applying rules of inference, such as modus ponens. The generated strategy candidates $\tilde{\mathcal{I}}$ are then evaluated for their validity and ranked for their usefulness by an LLM-based critic. Finally, the candidates deemed valid and useful are incorporated into the situational strategy set $\mathcal{I}$. A detailed demonstration can be found **??**.

Action Planning

The action selection process is conducted in two main steps:

(1) Invalid Action Masking: In this step, any actions that are not viable given the current game conditions are masked out. This masking is based on the verified world dynamics $\mathcal{W}$ and the current observation $\tilde{o}_t$.

(2) Action Selection: From the remaining valid actions, a specific primitive action a is selected. This choice is informed by multiple factors, including the current sub-task st_i, the verbalized current observation $\tilde{o}_t$, the world dynamics $\mathcal{W}$, a windowed history of previously planned actions and observations, and the derived strategies $\mathcal{I}$.

Given the following details:

- Primitive dynamics
- Current subtask
- Current observation

You are asked to identify the difficulties in completing the current subtask and provide 3 primitive or evolved dynamics to solve each of these difficulties.
Instructions for identifying difficulties:

- List the objects required to complete the subtask, specify their locations, and explain where to find them if they are not in the current observation.
- Outline all possible obstacles that may be encountered along the way.

Instructions for evolving advanced dynamics:

- Do not introduce any new objects that are not part of the primitive dynamics.
- The evolved dynamics should not contradict the primitive dynamics.
- If a difficulty cannot be resolved by existing dynamics, evolve new and advanced dynamics by combining only existing dynamics using deductive reasoning.

Instructions for providing deductive reasoning steps:

- For each evolved dynamics, provide the used primitive dynamics.
- For the deductive reasoning steps, provide the steps to combine the primitive dynamics to evolve the advanced dynamics and the rule of inference used (Modus Ponens, Modus Tollens, ...)

Last, for each situation and dynamics should be general and do not contain details about specific locations about the objects.
Output in the following format:{output format} and leaves 'None' for deductive reasoning steps if the dynamics are primitive.

Fig. 3.7. The prompt for evolving situational strategies

3.5. Experiment

In this section, we first outline our experimental setup, covering the evaluation metrics and the baseline models used for comparison (Section 3.5.1). Following that, we present the primary results (Section 3.5.4). Moreover, we perform a series of controlled experiments and provide both quantitative and qualitative analyses to gain deeper insights into DiVE (Sections 3.5.5 and 3.5.6).

3.5.1. Environment Settings

Crafter

Crafter is an open-world survival games set on 64×64 grid maps. These maps are populated with a variety of materials, including tree, stone, coal, as well as entities such as cow, zombie and skeleton, which spawn semi-randomly. The games feature an achievement graph comprising 22 unique achievements spread across 7 levels. The agent perceives its surroundings through a local 7×9 window and is also aware of its current status within the game environment.

The action space includes {up, down, left, right, make_X, place_X, defeat_X, sleep}. Additionally, the agent is required to follow the technology tree to advance its crafting abilities.

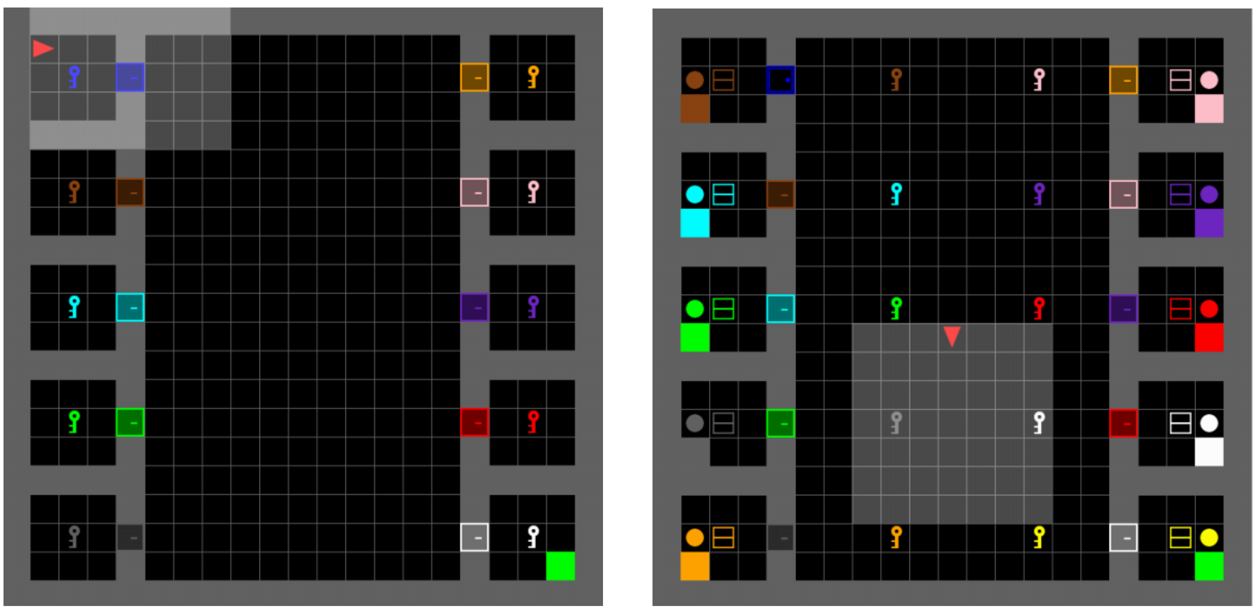


Fig. 3.8. The MiniGrid environment configurations. **Left**: The MiniGrid-UnlockDoors Environment. **Right**: The MiniGrid-UnlockGoals Environment.

MiniGrid-UnlockGoals

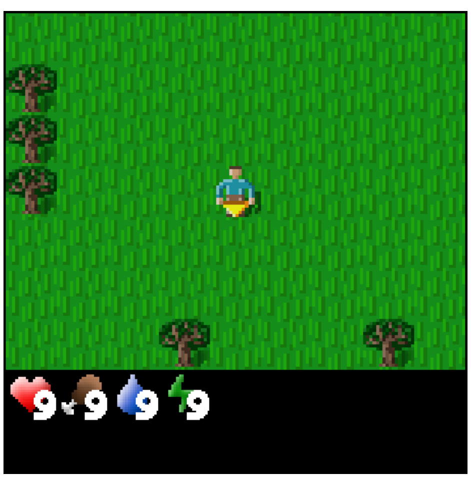
UnlockGoals is a 21x21 MiniGrid environment consisting of 10 rooms, as illustrated on the left in Figure 3.8. Each room contains 1 box, 1 ball, 1 subgoal, and 1 locked door. The agent can only see the contents of a room after unlocking it with a key. Each subgoal requires either a ball or a box to unlock. The agent's goal is to unlock all 10 subgoals. Note that the color mapping between doors and keys is randomly shuffled, as is the mapping between subgoals and the required colored balls or boxes.

The action space is {pick_X_key, pick_X_box, pick_X_ball, unlock_X_door, unlock_X_subgoal, drop_key, drop_box, drop_ball}. Meanwhile, the agent has a single space in its inventory, i.e., it can only take one item at a time.

MiniGrid-UnlockDoors

UnlockDoors is a 21x21 MiniGrid environment with 10 rooms and 10 keys, in different colors,, as illustrated on the right in Figure 3.8.. Each room can be accessed through a door that needs to be unlocked first by a key. An agent's goal is to unlock all 10 doors in a row in a specific order to reach the final room. Note the color mapping between doors and keys is randomly shuffled (but stays unchanged throughout the game), so the agent cannot rely on commonsense to guess the mapping.

The action space is {pick_X_key, unlock_X_door, drop_key, go_to_goal}, we bypass the need for navigation due to time constraints. Note the agent has a single space in its inventory, i.e., it can only take one key at a time.



It is daytime

State description:

- East: immediate (grass); nearby (unexplored area)

  ......

Closest:

- grass: North 1 blocks away (immediate) (objects in between: None)

  .....

Your status:

- health: 9/9

  ......

You have nothing in your inventory.

Fig. 3.9. An example of the Verbalizer's output in the Crafter environment.

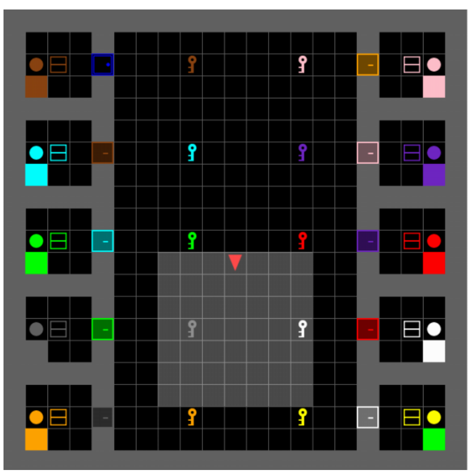
3.5.2. Verbalizer

Verbalizer for Crafter Environment

The verbalizer's text description for the Crafter environment provides the following details: the nearest object of each type within the accumulated observations, the objects positioned between these closest objects, the objects located in each direction, and the current status and inventory of the agent, as shown in Figure 3.9

Verbalizer for MiniGrid Environment

The verbalizer's text description for the MiniGrid environment provides the following details: the keys that can be picked up, the doors, the box and the squares that are not behind the locked door., as shown in Figure 3.10



You observe:

A green key is on the ground
A blue key is on the ground
A pink key is on the ground.
......
Door blue is locked.
Door grey is locked.
Door black is locked.
......

Fig. 3.10. An example of the Verbalizer's output in the MiniGrid environment.

3.5.3. Evaluation Metrics

Evaluation Metrics For Crafter Environment

Building upon previous research [**41**], we evaluate the agent using two primary metrics: reward and score. The reward metric assigns +1 for each new achievement unlocked (e.g., crafting a wooden pickaxe or placing a furnace) and ±0.1 for every health point gained or lost. The score metric, on the other hand, aggregates the success rates across all achievements. To account for the varying levels of difficulty associated with each achievement, it is calculated by averaging the success rates in log-space, also known as the geometric mean. This approach ensures that the score reflects the challenges of the achievements without requiring prior knowledge of their difficulty.

Evaluation Metrics For MiniGrid Environment

For the MiniGrid environment [**42**], we evaluate the agent's performance based on two criteria: effectiveness and efficiency. The agent's effectiveness is measured by the number of target objectives that have been achieved. For example, in the UnlockDoors environment, this refers to the number of doors unlocked, while in the UnlockGoals environment, it refers to the number of goals completed. Efficiency is evaluated by the total number of steps taken to successfully complete each environment.

3.5.4. Overall Results

Crafter Environment

Table 3.1 highlights that DiVE significantly outperforms all other baseline methods across two key evaluation metrics. Specifically, DiVE surpasses the previous state-of-the-art (SOTA) LLM-based method, SPRING[1], by a remarkable margin, demonstrating a 337.8% relative improvement in score and a 110.1% enhancement in reward. Furthermore, DiVE also exceeds the prior SOTA RL-based approach, DreamerV3, with a 21.4% absolute increase in score and a 2.8-point absolute improvement in reward. Impressively, DiVE achieves reward levels comparable to those of domain-expert human players, despite utilizing only 10 human demonstrations.

MiniGrid Environment

In the **UnlockGoals** environment, DiVE successfully unlocked all subgoals across all 5 trials, with an average of 179.6 steps per trial and a standard deviation of 37.7. In contrast, CoT with memory failed in all 5 trials, even within a 400-step budget. Additionally, when the mapping was not randomized, CoT with memory still failed in all 5 trials under the same 400-step limit.

In the **UnlockDoors** environment, DiVE successfully reached the goal in all 5 trials, averaging 35.3 steps per trial with a standard deviation of 8.3. By comparison, CoT with memory failed to reach the goal within the 100-step limit in all 5 trials. However, when tested on

[1]Results were obtained by executing the official SPRING code using the same LLM configuration (GPT-4o) and environment random seeds as DiVE, ensuring a fair comparison.

Methods	Score	Reward
Component analysis		
DiVE	$\mathbf{35.9 \pm 3.2}$**%**	$\mathbf{14.5 \pm 2.4}$
w/o E	21.1 ± 9.7%	11.3 ± 4.3
w/o V	9.8 ± 1.0%	10.1 ± 0.7
w/o V&E	11.5 ± 4.9%	8.3 ± 3.8
w/o D&V&E	0.9 ± 0.1%	2.5 ± 1.3
CoT	1.3 ± 0.3%	2.5 ± 0.5
CoT + D&V	3.6 ± 0.9%	3.9 ± 2.3
Dynamics from distinct sources		
DiVE	$\mathbf{35.9 \pm 3.2}$**%**	$\mathbf{14.5 \pm 2.4}$
w/o D&V+S†	15.7 ± 5.3%	8.9 ± 5.1
w/o D&V&E+S†	12.1 ± 4.6%	8.7 ± 3.0
w/o D&V&E+H†	34.2 ± 2.8%	14.5 ± 0.9

Tableau 3.1. The impact of various components on performance is analyzed. The abbreviations D, V, and E represent Discover, Verifier, and Evolver, respectively. S† refers to the manually defined dynamics used in the SPRING paper, while H† denotes human-annotated dynamics.

UnlockDoors without randomizing the mapping between keys and doors, CoT with memory succeeded in 3 out of 5 trials within the 100-step budget.

3.5.5. Quantitative Analysis

Ablation study for Crafter Environment

Performance We conducted an ablation study to clarify the contribution of individual components to the overall performance of our method. The first section of Table 3.1 presents various variants of DiVE designed to evaluate the effectiveness of each component. The substantial performance gap between DiVE and its variant without the Evolver component highlights the critical role of the Evolver in developing effective gameplay strategies based on world dynamics $\mathcal{W}$, thereby enhancing the agent's capabilities in the Crafter environment. Similarly, the observed performance drop in the variant without the Verifier component emphasizes the necessity of formulating strategies $\mathcal{I}$ based on accurate world dynamics $\mathcal{W}$.

Additionally, the further performance decline in the version lacking both the Verifier and Evolver components demonstrates their complementary roles—where the Verifier ensures precision in dynamics and the Evolver facilitates strategy development.

When DiVE is stripped of the Discoverer, Verifier, and Evolver components, its performance reverts to the CoT baseline. This outcome indicates that decomposing the task according to the hierarchy $\mathcal{H}$ without incorporating domain knowledge $\mathcal{K}_{\text{target}}$ provides no benefit. Moreover, the notable performance difference between the CoT + D&V variant and DiVE without the Evolver component further illustrates that an LLM-based agent struggles with long-horizon planning tasks without task decomposition, thereby underscoring the importance of the decomposition hierarchy $\mathcal{H}$.

Source of Dynamics We also examined how DiVE performs when using world dynamics derived from different sources. The second section of Table 3.1 shows that DiVE significantly outperforms variants utilizing dynamics $S^{\dagger}$, as employed in the SPRING paper [**64**]. This improvement suggests that the learned dynamics $\mathcal{W}$ offer greater benefits than $S^{\dagger}$, potentially because $S^{\dagger}$ lacks some of the valuable details found in $\mathcal{W}$. We will explore this aspect further in upcoming case studies. Additionally, the performance difference between methods using $S^{\dagger}$ with and without the Evolver component highlights the importance of strategy evolution, which is closely tied to the quality of the world dynamics employed.

In addition to dynamics learned from human demonstrations and the manually defined dynamics used in SPRING, we also explored a third source: human-annotated dynamics. Our findings show that DiVE performs comparably to variants using human-annotated dynamics, demonstrating the robustness and effectiveness of DiVE's dynamic learning approach.

Quality of learned dynamics Directly quantifying the desired properties of learned dynamics is inherently challenging. This difficulty arises from our inability to precisely measure how much domain-relevant information, denoted as $\mathcal{K}_{\text{rel}}$, is contained within a large language model (LLM), or to accurately determine the exact amount of knowledge, $\mathcal{K}_{\text{target}}$, required for a specific domain. To address this, we use human-annotated dynamics, $H^{\dagger}$, as a silver standard for $\mathcal{K}_{\text{target}}$. By doing so, we can estimate the precision and recall of the learned dynamics, $\mathcal{W}$, which allows us to effectively track and evaluate the progress of LLM-based dynamic learning.

Recall is calculated as $R = \frac{|\mathcal{W} \cap H^{\dagger}|}{|H^{\dagger}|}$, and precision is calculated as $P = \frac{|\mathcal{W} \cap H^{\dagger}|}{|\mathcal{W}|}$. Figure 3.11 illustrates that both the discovered dynamics, $\tilde{\mathcal{W}}$, and the verified dynamics, $\mathcal{W}$, exhibit an

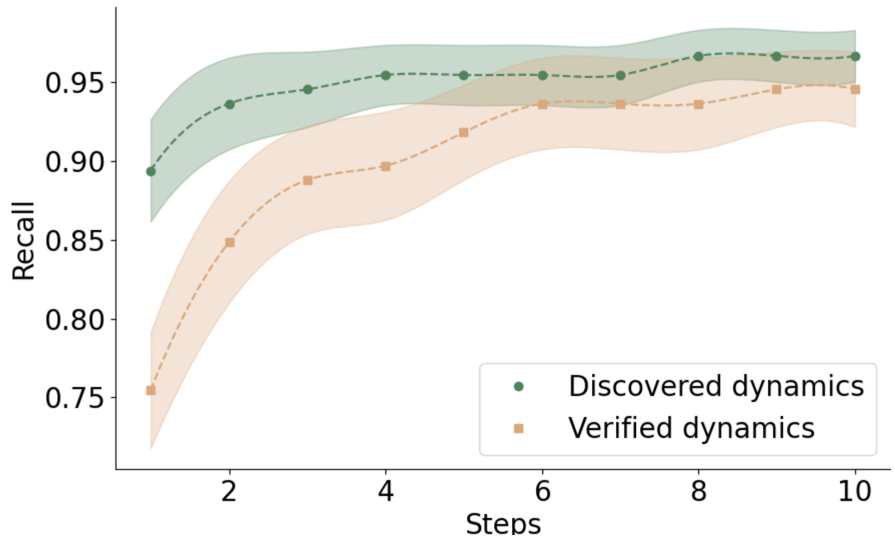


Fig. 3.11. Recall of learned dynamics across discovery steps, presented with mean values and standard deviations in the Crafter environment.

increase in recall as discovery steps progress. This trend indicates that the richness of the learned dynamics improves with more discovery steps. Additionally, the recall gap between $\tilde{\mathcal{W}}$ and $\mathcal{W}$ narrows as discovery steps increase, suggesting that the Verifier is successful in filtering out "noisy" dynamic candidates while retaining those that generalize well across different trajectory segments.

Effectiveness of verifier To evaluate whether the Verifier effectively retains correct world dynamic candidates, $\tilde{\mathcal{W}}^+$, while discarding inaccurate ones, $\tilde{\mathcal{W}}^-$, we analyze the precision of both the discovered dynamics, $\tilde{\mathcal{W}}$, and the verified dynamics, $\mathcal{W}$. As illustrated in Figure 3.12, the precision of the verified dynamics consistently and significantly surpasses that of the discovered dynamics. This result highlights the Verifier's success in identifying and eliminating inaccurate dynamic candidates, thereby affirming its crucial role in enhancing the reliability of the dynamics used in decision-making processes.
Additionally, as shown in Figure 3.12, precision increases rapidly after the initial verification step and then stabilizes for the remaining steps.

Ablation study for MiniGrid Environment
In the UnlockGoals experiment without randomized dynamics, as shown in Table 3.2, CoT fails due to a knowledge gap between the LLMs and the environment, particularly because

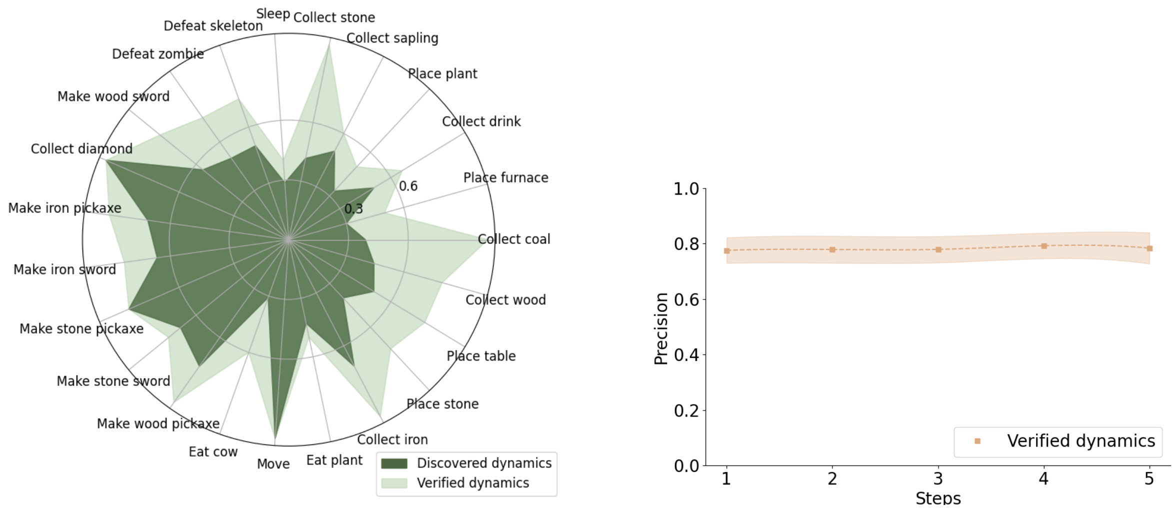


Fig. 3.12. Precision of the learned dynamics in the Crafter environment.
Left: Precision of individual instances before and after verification. **Right**: Precision across verification steps, shown with mean values and standard deviations.

Environment	**Method**	**Success Rate**	**Steps**
UnlockGoals			
	DiVE	**100.0%**	**179.6 ± 37.7 steps**
	CoT	0.0%	-
Non-Random	CoT	0.0%	-
UnlockDoors			
	DiVE	**100%**	**35.3 ± 8.3 steps**
	CoT	0.0%	-
Non-Random	CoT	60.0%	179.6 ± 24.1 steps

Tableau 3.2. Performance comparison of DiVE and CoT with Memory in the UnlockGoals and UnlockDoors environments. The success rate is shown as the percentage of successful trials out of 5, with the average number of steps per trial and the standard deviation in parentheses. A dash ('-') indicates that the method did not reach the goal within the maximum allowed steps, so no valid step count is available.

unlocking a subgoal requires specific objects like a box or ball. In contrast, DiVE successfully unlocked all subgoals across all trials, even with randomized mappings. This suggests that DiVE is effective at discovering the necessary environment dynamics, thereby bridging the knowledge gap between LLMs and the environment. These results align with our findings in the Crafter and UnlockDoor environments.

In the UnlockDoors environment, as shown in Table 3.2, CoT performs reasonably well on the UnlockDoor task when mappings are not randomized. However, its performance significantly declines with randomized mappings. This indicates that LLMs may struggle when there is a disparity between their embedded knowledge and the actual dynamics of the environment. On the other hand, DiVE consistently achieved the goal across all trials, demonstrating its ability to enhance LLMs' understanding of downstream environments and overcome potential knowledge gaps. This finding is also consistent with our experiments in the Crafter environment.

3.5.6. Qualitative Analysis

Correctness	**Outcome**
Correct (✓)	none
Confounder (✗)	1 health
In-domain Hallucination (✗)	1 wood
Out-domain Hallucination (✗)	1 bone

Tableau 3.3. The dynamics that govern the outcome of defeating a zombie

DiVE For Crafter Environment

Source of Errors in Learned Dynamics The accuracy of the learned and verified dynamics can be categorized into two distinct outcomes: correct or incorrect. Errors typically stem from three sources: confounders, in-domain hallucinations, or out-of-domain hallucinations. Table 3.3 illustrates these concepts using an example of defeating a zombie.

An error due to confounders occurs when a simultaneous increase in health points is observed at the moment the zombie is defeated. The Discoverer incorrectly attributes this health boost to the act of defeating the zombie. In the case of in-domain hallucinations, the error arises when the system mistakenly infers that wood can be increased by defeating the zombie, despite the fact that such an increase is impossible. Lastly, out-of-domain hallucinations

Methods	**Dynamics**
SPRING	Place stone requires stones
D&V	Place stone requires **1** stone and **faces paths, grass, sand, water, and lava**
D&V&E	Place stone to block zombies and skeletons, preventing them from reaching the player

Tableau 3.4. Comparing SPRING and DiVE on place stone's dynamics

involve the incorrect identification of dynamics that pertain to objects not present in the observation or even within the Crafter environment.

Learned Dynamics Comparison In comparison to the dynamics presented by SPRING, as shown in Table 3.4, we found that the dynamics generated by DiVE are not only more accurate but also more detailed. For instance, in the context of placing a stone, SPRING merely identified that stones are needed. In contrast, DiVE precisely determined that placing a stone requires exactly one stone and identified the specific facing condition necessary for successful placement.

Dynamics about Grass:

- grass can be found near ['tree', 'water', 'path']
- grass is not associated with ['diamond', 'coal', 'iron']
- You can walk directly through grass.
- grass can only be used for: ['collect sapling', 'eat plant']

Dynamics about Coal:

- coal can be found near ['stone', 'iron', 'diamond']
- coal is not associated with ['grass', 'cow', 'skeleton']
- You cannot walk directly through coal.
- coal turn into path after collect coal
- coal can only be used for: ['make iron pickaxe', 'make iron sword', 'collect coal']
- coal can be collected by ['wood pickaxe']

Fig. 3.13. Examples of Learned Dynamics in the Crafter Environment

With this detailed information, as illustrated in Figure 3.13, the Evolver can infer more complex dynamics, such as the potential for the placed stone to act as a barrier between the agent and dangerous creatures.

DiVE For MiniGrid Environment

Similarly, in the MiniGrid-based environments, UnlockDoors and UnlockGoals, DiVE has successfully learned the essential dynamics needed for effective decision-making by LLMs, as shown in the Figure 3.14. Unlike the Crafter environment, which presents sources of error, these simpler environments did not exhibit any detectable errors.

Dynamics in the UnlockGoals environment

- before unlocking yellow door, the blue key needs to be in hand
- after unlocking yellow door, the yellow door is unlocked and reveals a grey box, orange ball, and pink goal
- before unlocking black subgoal, the green ball needs to be in hand
- after unlocking black subgoal, the black subgoal is unlocked

Dynamics in the UnlockDoors environment

- before picking blue key, the purple door needs to be unlocked
- after picking blue key, the blue key is now in hand
- before unlocking purple door, the green key needs to be in hand
- after unlocking purple door, the purple door is unlocked

Fig. 3.14. Examples of Learned Dynamics in the UnlockGoals and UnlockDoors Environment

Chapter 4

Conclusion and Future Works

By bridging the potential knowledge gap, DiVE achieves impressive performance. However, there are still improvements to be made and limitations to consider.

4.1. Conclusion

In this work, we introduce DiVE, a framework tailored for long-horizon interactive planning tasks. The framework operates as follows:

- Discoverer: Using a small set of demonstrations, a Discoverer identifies key world dynamics, such as the preconditions and outcomes of actions.
- Verifier: A Verifier then filters out any inaccurate dynamic candidates, ensuring the reliability of the knowledge base.
- Evolver: In an online setting, using the knowledge acquired offline, an Evolver learns and adapts situational strategies through interaction.

DiVE effectively bridges the gap between a pre-trained LLM and the target domain by autonomously acquiring hierarchical world dynamics and evolving contextual knowledge. Extensive experiments confirm its effectiveness. This work potentially paves the way for developing LLM-based frameworks that enable the mutual enhancement of dynamics learning and policy learning, all without the need for extensive human annotation.

4.2. Future Works

Here, we outline potential future research directions based on DiVE, focusing the knowledge gap between LLMs and downstream tasks.

4.2.1. Actively Exploration

The prompts in DiVE is designed to help LLMs focus on the specific properties that need to be discovered.
In the current work, DiVE-v2, we propose a general pipeline for the discoverer, as shown in Figure 4.1 that allow LLMs to actively explore the environment's dynamics.
The main idea is to allow LLM to initially discover dynamics in a free-form manner. Afterward, these dynamics can be grouped based on their semantic relationships. By prompting LLMs to explore elements that are least semantically similar to those already identified for **A**, we can redirect their focus. This method encourages the discovery of new dynamics that are less related to the existing ones for **A**, thereby expanding the range of dynamics identified for **A**.

> The agent has some knowledge about the element: element name.
> However, the agent wants to learn more details about this element.
> Specifically, other actions have these facts: dissimilar examples.
> The agent is curious about what similar facts this element have.
> To discover the similar facts:
> - The facts should be valid across the examples.
> - The facts should be similar to the examples.
> - The facts should be specific to the element, environment, tools and resources.
>
> Here are some examples of the element: sampled replay buffer.
> Please respond with the following template in JSON format: templates

Fig. 4.1. The general prompts

After two iterations of discovery, this approach achieves approximately 97% recall compared to the ground-truth dynamics in the Crafter environment.

4.2.2. The Need for Human Demonstrations

All dynamics are discovered and verified based on human demonstrations; however, collecting such demonstrations can be challenging or costly, especially in fields like robotics [**89, 90**] and medical imaging [**91**].

A generalist embodied agent should actively explore its environment and tasks to autonomously learn the necessary dynamics, rather than relying solely on human demonstrations. By taking this approach, the agent can adapt to a broader range of scenarios and tasks, developing a deeper understanding of its surroundings and the tasks it needs to perform. This autonomous learning enables the agent to handle more complex and varied situations effectively.

4.3. Future Works

To build a generalist problem-solving agent, the agent must possess both a broad understanding of the world and a deep knowledge of the specific problem at hand.

4.3.1. Foundation Models and Specialist Models

FMs can learn and capture fundamental patterns of the world by autoregressively predicting the next token on a large scale using internet data. However, adding more dimensions to the modality of FMs ironically reduces the amount of available data, which in turn limits the model's ability to deepen its understanding. As a result, while FMs can develop a broad understanding of the world, this understanding often remains general and superficial due to the constraints of expanding dimensionality.

By interacting with the environment millions of times, Specialist Models (SMs) can learn the dynamics of a specific domain and optimize their performance within that area of specialization. However, this understanding and performance are not easily generalizable to new environments, even if they are similar.

Thus, FMs possess a broad yet superficial understanding of the world, whereas SMs have a deep, specialized understanding of specific problems.

4.3.2. The Human Approach to Problem-Solving

As human beings, we possess an internal model—our common sense—that provides us with a fundamental understanding of the world. When confronted with a problem that falls outside the scope of our common sense, we may initially struggle. However, through repeated attempts, we can learn the mechanisms of the problem and gradually improve our performance by applying our common sense. During this trial-and-error process, we not only acquire the necessary techniques to tackle the problem but also broaden our common sense, preparing ourselves for similar challenges in the future. This creates a mutually reinforcing cycle, where our problem-specific techniques and common sense continuously enhance each other.

4.3.3. The Mutually Reinforcing Cycle

A generalist problem-solving agent should possess a similar mutually reinforcing cycle, enabling it to not only acquire the necessary mechanisms to handle the task at hand but also absorb this knowledge and broaden its general understanding of the world. To build such a mutually reinforcing cycle, FMs offer an initial, yet powerful, prior for the agent's understanding of the world. Based on this understanding, the agent can design SMs to solve specific tasks, even if initial performance is poor. By rolling out the trajectories of the SMs, the agent can deepen its understanding of the task and iteratively design better SMs. Throughout these iterations, the agent's understanding of the world should also broaden, enhancing its ability to handle a wider range of tasks.

Références

[1] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[4] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[5] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[6] Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[10] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[11] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[12] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

[13] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[14] The claude 3 model family: Opus, sonnet, haiku. *https://api.semanticscholar.org/CorpusID:268232499*, 2024.

[15] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[16] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[17] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[18] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690, 2024.

[19] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

[20] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[21] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023.

[22] Jason Weston and Sainbayar Sukhbaatar. System 2 attention (is something you might need too). *arXiv preprint arXiv:2311.11829*, 2023.

[23] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

[24] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

[25] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

[26] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[27] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[28] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192, 2022.

[29] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

[30] Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M Dai. Mind's eye: Grounded language model reasoning through simulation. *arXiv preprint arXiv:2210.05359*, 2022.

[31] Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. Language models of code are few-shot commonsense learners. *arXiv preprint arXiv:2210.07128*, 2022.

[32] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

[33] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

[34] Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*, 2023.

[35] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

[36] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, et al. igibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7520–7527. IEEE, 2021.

[37] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[38] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[39] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.

[40] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, 2022.

[41] Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

[42] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems*, 36, 2024.

[43] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[44] Xin Ye, Zhe Lin, Joon-Young Lee, Jianming Zhang, Shibin Zheng, and Yezhou Yang. Gaple: Generalizable approaching policy learning for robotic object searching in indoor environment. *IEEE Robotics and Automation Letters*, 4(4):4003–4010, 2019.

[45] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[46] Jingwei Zhang, Lei Tai, Ming Liu, Joschka Boedecker, and Wolfram Burgard. Neural slam: Learning to explore with external memory. *arXiv preprint arXiv:1706.09520*, 2017.

[47] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155*, 2020.

[48] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[49] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 37–53, 2018.

[50] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6629–6638, 2019.

[51] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018.

[52] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022.

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

[54] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[55] Kolby Nottingham, Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Sameer Singh, Peter Clark, and Roy Fox. Skill set optimization: Reinforcing language model behavior via transferable skills. *arXiv preprint arXiv:2402.03244*, 2024.

[56] Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. Clin: A continually learning language agent for rapid task adaptation and generalization. *arXiv preprint arXiv:2310.10134*, 2023.

[57] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642, 2024.

[58] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*, 2022.

[59] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[60] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[61] Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Automated dense reward function generation for reinforcement learning. *arXiv preprint arXiv:2309.11489*, 2023.

[62] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[63] Yecheng Jason Ma, William Liang, Hung-Ju Wang, Sam Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. Dreureka: Language model guided sim-to-real transfer. *arXiv preprint arXiv:2406.01967*, 2024.

[64] Yue Wu, So Yeon Min, Shrimai Prabhumoye, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Tom Mitchell, and Yuanzhi Li. Spring: Gpt-4 out-performs rl algorithms by studying papers and reasoning. *arXiv preprint arXiv:2305.15486*, 12, 2023.

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

[66] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[67] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.

[68] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

[69] Nouha Dziri, Sivan Milton, Mo Yu, Osmar Zaiane, and Siva Reddy. On the origin of hallucinations in conversational models: Is it the datasets or the models? In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5271–5285, Seattle, United States, July 2022. Association for Computational Linguistics.

[70] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pages 2451–2463. Curran Associates, Inc., 2018. `https://worldmodels.github.io`.

[71] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[72] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[73] Robert B Ash. *Information theory*. Courier Corporation, 2012.

[74] Jiawei Zheng, Hanghai Hong, Xiaoli Wang, Jingsong Su, Yonggui Liang, and Shikai Wu. Fine-tuning large language models for domain-specific machine translation. *arXiv preprint arXiv:2402.15061*, 2024.

[75] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

[76] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[77] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[78] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[79] Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

[80] Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning.

In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 3676–3713. PMLR, 23–29 Jul 2023.

[81] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

[82] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[83] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[84] Khurram Javed and Martha White. Meta-learning representations for continual learning. *Advances in neural information processing systems*, 32, 2019.

[85] Yijun Li, Richard Zhang, Jingwan Lu, and Eli Shechtman. Few-shot image generation with elastic weight consolidation. *arXiv preprint arXiv:2012.02780*, 2020.

[86] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[87] Yue Wu, Yewen Fan, Paul Pu Liang, Amos Azaria, Yuanzhi Li, and Tom M Mitchell. Read and reap the rewards: Learning to play atari with the help of instruction manuals. *Advances in Neural Information Processing Systems*, 36, 2024.

[88] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[89] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[90] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[91] Amber L Simpson, Michela Antonelli, Spyridon Bakas, Michel Bilello, Keyvan Farahani, Bram Van Ginneken, Annette Kopp-Schneider, Bennett A Landman, Geert Litjens, Bjoern Menze, et al. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *arXiv preprint arXiv:1902.09063*, 2019.

Annexe A

Collaborative Works

During my master's studies, I had the opportunity to collaborate with my lab mates, Haochen Shi and Dekun Wu. Together, we proposed the following works:

- OPEx: A Component-Wise Analysis of LLM-Centric Agents in Embodied Instruction Following
- Deciphering Digital Detectives: Understanding LLM Behaviors and Capabilities in Multi-Agent Mystery Games

A.1. OPEx

In this work, we introduce OPEx, a comprehensive framework for Embodied Instruction Following (EIF), focusing on three core components: Observer, Planner, and Executor. Our work addresses the gap in understanding how these components affect task performance by conducting extensive evaluations. We found that while LLM-centric designs significantly enhance EIF outcomes, visual perception and low-level action execution remain key challenges. Additionally, we explore a multi-agent dialogue strategy in a TextWorld environment, demonstrating that this approach further boosts performance.

A.2. ThinkThrice

In this study, we explore the use of LLMs in Jubensha, a Chinese detective role-playing game. We introduce the dataset tailored for Jubensha, featuring character scripts and game rules, to advance AI agent development in this complex narrative setting. Our unique multi-agent interaction framework enables AI agents to autonomously participate in the game. We developed innovative evaluation methods to assess these agents' mastery of case information and reasoning skills. By leveraging in-context learning techniques, we enhanced the agents' abilities in information gathering, murderer identification, and logical reasoning. Our experimental results confirm the effectiveness of our methods, offering a new perspective on LLM capabilities and setting a benchmark for evaluating LLM-based agents.