Survey on Large Language Model-Enhanced Reinforcement Learning: Concept, Taxonomy, and Methods

Yuji Cao, Huan Zhao, *Member, IEEE*, Yuheng Cheng, Ting Shu, Yue Chen, *Member, IEEE*, Guolong Liu, *Member, IEEE*, Gaoqi Liang, *Member, IEEE*, Junhua Zhao, *Senior Member, IEEE*, Jinyue Yan, and Yun Li, *Fellow, IEEE*

***Abstract*— With extensive pretrained knowledge and high-level general capabilities, large language models (LLMs) emerge as a promising avenue to augment reinforcement learning (RL) in aspects, such as multitask learning, sample efficiency, and high-level task planning. In this survey, we provide a comprehensive review of the existing literature in LLM-enhanced RL and summarize its characteristics compared with conventional RL methods, aiming to clarify the research scope and directions for future studies. Utilizing the classical agent–environment interaction paradigm, we propose a structured taxonomy to systematically categorize LLMs' functionalities in RL, including four roles: information processor, reward designer, decision-maker, and generator. For each role, we summarize the methodologies, analyze the specific RL challenges that are mitigated and provide insights into future directions. Finally, the comparative analysis of each role, potential applications, prospective opportunities, and challenges of the LLM-enhanced RL are discussed. By proposing this taxonomy, we aim to provide a framework for researchers to effectively leverage LLMs in the RL field, potentially accelerating RL applications in complex applications, such as robotics, autonomous driving, and energy systems.**

***Index Terms*— Large language models (LLMs), LLM-enhanced reinforcement learning (RL), multimodal RL, RL, vision-language models (VLMs).**

Received 30 March 2024; revised 23 September 2024 and 29 October 2024; accepted 7 November 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 72331009, Grant 72171206, and Grant 92270105; in part by Guangdong Power Grid Company under Grant GDKJXM20231024; in part by the Shenzhen Key Lab of Crowd Intelligence Empowered Low-Carbon Energy Network under Grant ZDSYS20220606100601002; in part by the Shenzhen Natural Science Fund under Grant GXWD20231128112434001; in part by the Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS); in part by the PolyU Direct Grant under Grant P0047700, Grant P0043885, and Grant P0051105; in part by the CUHK Strategic Partnership Award for Research Collaboration under Grant 4750467; and in part by the CUHK Direct Grant for Research under Grant 4055228. *(Corresponding authors: Huan Zhao; Junhua Zhao.)*

Yuji Cao and Yue Chen are with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong, SAR, China (e-mail: yjcao@mae.cuhk.edu.hk; yuechen@mae.cuhk.edu.hk).

Huan Zhao and Jinyue Yan are with the Department of Building Environment and Energy Engineering, The Hong Kong Polytechnic University, Hong Kong, China (e-mail: huan-paul.zhao@polyu.edu.hk; j-jerry.yan@polyu.edu.hk).

Yuheng Cheng and Junhua Zhao are with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, China, and also with the Center for Crowd Intelligence, Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS), Shenzhen 518129, China (e-mail: yuhengcheng@link.cuhk.edu.cn; zhaojunhua@cuhk.edu.cn).

Ting Shu is with the National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen 518000, China (e-mail: cqushuting@gmail.com).

Guolong Liu is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: guolong.liu@ntu.edu.sg).

Gaoqi Liang is with the School of Mechanical Engineering and Automation, Harbin Institute of Technology, Shenzhen 518055, China (e-mail: lianggaoqi@hit.edu.cn).

Yun Li is with Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China, and also with i4AI Ltd., WC1N 3AX London, U.K. (e-mail: Yun.Li@ieee.org).

Digital Object Identifier 10.1109/TNNLS.2024.3497992

I. Introduction

REINFORCEMENT learning (RL) is a powerful learning paradigm that focuses on control and decision-making, where an agent learns to optimize a specified target through trial-and-error interactions with the environment. Traditional RL methods, however, often struggled with high-dimensional state spaces and complex environments [1]. The integration of deep learning techniques with RL, known as deep RL, has led to significant breakthroughs. In 2015, deep Q-networks (DQNs) [2] marked a turning point, demonstrating human-level performance on Atari games using raw pixel inputs. Subsequent innovations, such as proximal policy optimization (PPO) [3] and soft actor–critic [4] have further expanded the capabilities of deep RL. In different realms, deep RL algorithms have achieved promising performance, such as real-time strategy games [5], [6], board games [7], [8], energy management [9], and imperfect information games [10], [11]. Concurrent advancements in natural language processing (NLP) and computer vision (CV) [12], [13], have fostered new RL paradigms, such as language-conditional RL [14], which uses natural language to instruct agents, and vision-based RL [15], where agents learn from high-dimensional visual inputs.

The integration of language and vision capabilities into deep RL has introduced new challenges, as the agent has to learn the high-dimensional features and a control policy jointly. To reduce the burden of visual feature learning, Stooke et al. [16] decoupled representation learning from RL. To handle language-involved tasks, a survey [17] called for potential uses of NLP techniques in RL. Nevertheless, the capabilities of language models were limited at that time and the following four challenges still have not been addressed.

1) *Sample Inefficiency:* Language and vision tasks involve large, complex state–action spaces, making it challenging for RL agents to learn effective policies. Moreover, agents must understand tasks and connect them to

corresponding states, necessitating even more extensive interactions [18], [19], [20].

2) *Reward Function Design:* In language and vision tasks, designing effective reward functions is particularly challenging. These functions must capture subtle linguistic nuances and complex visual features, significantly increasing the complexity of an already difficult process. Moreover, aligning rewards with high-level task objectives in these domains often requires domain expertise and extensive trial-and-error [1], [21], [22], [23].
3) *Generalization:* RL agents often overfit to training data, especially in vision-based environments, leading to poor performance when deployed in states with interventions (e.g., added noise). Agents must learn invariant features robust to such interventions, enabling generalization across varied linguistic contexts and visual scenes. However, the complexity of these domains makes extracting such features and adapting to new environments particularly challenging [24], [25].
4) *Natural Language Understanding*: Deep RL faces difficulties in NLP and understanding scenarios, where the nuances and complexities of human language present unique challenges that are not adequately addressed by current RL methodologies [17].

The field of NLP has undergone a revolutionary transformation since the introduction of the Transformer architecture in 2017 [12]. This breakthrough paved the way for the development of large language models (LLMs), with landmark models, such as bidirectional encoder representations from transformers (BERT) [26], generative pre-trained transformer (GPT) [27], and more recent iterations such as GPT-3 [28] and PaLM [29] marking significant milestones. The emergence of these LLMs has demonstrated powerful capabilities across various real-world applications, including medicine [30], chemical research [31], energy system [32], [33], [34], and embodied control in robotics [35]. These models have not only advanced the field of NLP but also shown remarkable potential in tackling complex, multidisciplinary challenges. Compared with small language models, LLMs have emergent capabilities that are not present in small language models [36], such as in-context learning [37] and reasoning ability [38]. In addition, leveraging the vast amounts of training data, pretrained LLMs are equipped with a broad spectrum of world knowledge [39]. Benefiting from these capabilities, the applications of language models have been shifted from language modeling to task-solving, ranging from basic text classification and sentiment analysis to complex high-level task planning [40] and decision-making [41], [42].

With emergent capabilities, the potential of LLMs to address the inherent challenges of RL has recently gained popularity [43], [44]. The capabilities of LLMs, particularly in natural language understanding, reasoning, and task planning, provide a unique approach to solving the above-mentioned RL issues. For sample inefficiency, Lin et al. [45] proposed a framework, where LLMs can be employed to improve the sample efficiency of RL agents by providing rich, contextually informed predictions or suggestions, thereby reducing the need for extensive environment interactions. For reward function design, LLMs can aid in constructing more nuanced and effective reward functions, enhancing the learning process by offering a deeper understanding of complex scenarios [46]. For generalization, Chakraborty et al. [47] proposed a framework that leverages language-based feedback for improving the generalization of RL policy in unseen environments. For natural language understanding, Pang et al. [48] used LLMs to translate complex natural language-based instructions to simple task-specified languages for RL agents. These works show that LLM is a promising and powerful role that can contribute to the longstanding RL challenges.

Despite the advancements in the domain of integrating LLMs into the RL paradigm, there is currently a notable absence of comprehensive review in this rapidly evolving area. In addition, though various methods are proposed to integrate LLMs into the RL paradigm, there is no unified framework for such integration. Our survey paper seeks to fill these gaps by providing an extensive review of the related literature, defining the scope of the novel paradigm called *LLM-enhanced RL*, and further proposing a taxonomy to categorize the functionalities of LLMs in the proposed paradigm.

A. Contributions

This survey makes the following contributions.

1) *LLM-Enhanced RL Paradigm:* This article presents the first comprehensive review in the emerging field of integrating LLM into the RL paradigm. To clarify the research scope and direction for future works, we define the term *LLM-enhanced RL* to encapsulate this class of methodologies, summarize the characteristics, and provide a corresponding framework that clearly illustrates: a) how to integrate LLMs in classical agent–environment interaction and b) the multifaceted enhancements that LLMs offer to the conventional RL paradigm.
2) *Unified Taxonomy:* Further classifying the functionalities of LLMs in the LLM-enhanced RL paradigm, we propose a structured taxonomy to systematically categorize LLMs within the classical agent–environment paradigm, where LLMs are classified as information processors, reward designers, decision-makers, and generators. By such a categorization, a clear view of how LLMs integrate into the classical RL paradigm is offered.
3) *Algorithmic Review:* For each role of LLM, we review emerging works in this direction and discuss different algorithmic characteristics from the perspective of capabilities. Based on this foundation, future applications, opportunities, and challenges of LLM-enhanced RL are analyzed to provide a potential roadmap for advancing this interdisciplinary field.

B. Text Organization

The remaining sections are organized as follows. Section II provides the foundational knowledge of both RL and LLM. Section III presents the concept of LLM-enhanced RL and provides its overall framework. Following this, Sections IV–VII offer an in-depth analysis of LLMs within the RL context, exploring their roles as information processor, reward designer, decision-maker, and generator, respectively. Last, Section VIII discusses the application, opportunities,

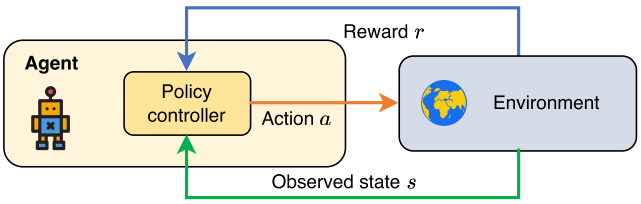

Fig. 1. Classical RL paradigm.

and challenges of LLM-enhanced RL. Finally, Section IX concludes the survey.

II. Background

In this section, we provide a concise overview of the classical RL paradigm and related challenges. Next, we explore the prevailing trend in RL—specifically, the fusion of multimodal data sources, including language and visual information. Following this, we offer an introductory background on LLMs and outline the key capabilities that can enhance the RL learning paradigm.

A. Background of Reinforcement Learning (RL)

1) Classical RL: In a classical RL paradigm shown in Fig. 1, the agent interacts with an environment through a trial-and-error process to maximize the specified rewards in the trajectory. In each step, the agent takes an action a based on the observed state s from the environment. By optimizing the policy π (action controller), the agent maximizes the cumulative rewards. Such an optimization problem is usually formalized through the concept of Markov decision process (MDP), defined by the quintuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$. Here, $\mathcal{S}$ denotes a set comprising all possible states, $\mathcal{A}$ denotes a set of all possible actions, $\mathcal{T}$ represents the state transition probability function $\mathcal{T}$:$\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, $\mathcal{R}$ is a reward function $\mathcal{R}$:$\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, and γ (with $0 \leq \gamma \leq 1$) is the discount factor. The objective in RL is to optimize the policy $\pi(a|s)$ such that the cumulative returns $\sum_{k=0}^{\infty} \gamma^k r_{k+1}$ is maximized.

2) Challenges of RL: While RL algorithms have made remarkable performance in recent years [2], [7], [49], there are still several longstanding challenges that limit the real-world applicability of RL.

1) *Generalization in Unseen Environment:* Generalization in unseen environments remains a significant challenge in the field of RL [50]. The core issue lies in the ability of RL algorithms to transfer learned knowledge or behaviors to new, previously unseen environments. RL models are often trained in simulated or specific settings, excelling in those scenarios but struggling to maintain performance when faced with novel or dynamic conditions. This limitation hinders the practical application of RL in real-world situations, where environments are rarely static or perfectly predictable. Achieving generalization requires models to not only learn specific task solutions but also to understand underlying principles that can be adapted to a range of situations.
2) *Reward Function Design:* Reward function is the principal contributing factor to the performance of an RL agent. Despite their fundamental importance, reward functions are known to be notoriously difficult to design, especially in contexts involving sparse reward environments and complex scenarios [1]. In sparse reward settings, where feedback is limited, reward shaping becomes essential to guide agents toward meaningful behaviors; however, this introduces the risk of inadvertently biasing the agent toward suboptimal policies or overfitting to specific scenarios [51], [52]. Conversely, for complex tasks, high-performance reward functions usually require massive manual trial-and-error since most designed rewards are suboptimal [23] or lead to unintended behavior [53].
3) *Compounding Error in Model-Based Planning*: Model-based RL is prone to the issue of compounding errors during planning. As the prediction horizon extends, errors in the model's predictions accumulate, leading to significant deviations from optimal trajectories [54], [55]. This problem is particularly acute in complex environments with high-dimensional state spaces. With their advanced predictive capabilities and understanding of sequential dependencies, LLMs could help mitigate these errors, leading to more accurate and reliable planning in model-based RL.
4) *Multitask Learning:* Multitask RL faces several key challenges that limit its effectiveness. One major issue is managing varying task difficulties, where simpler tasks can overshadow learning of more complex ones, leading to negative transfer [56]. The task interference is another critical problem, as shared parameters or data between tasks can result in suboptimal performance on individual tasks [57]. Determining optimal parameter-sharing strategies is complex, as it must balance learning efficiency with task-specific requirements [58]. Sample efficiency remains a significant hurdle, with traditional data-sharing approaches not fully leveraging learned behaviors across tasks [59]. Finally, effectively transferring knowledge between tasks without negative interference is an ongoing challenge that impacts the agent's ability to accelerate learning across multiple objectives [60].

3) Multimodal Reinforcement Learning: With the advances in both CV and NLP, pattern recognition in vision and natural language has become increasingly powerful, and multimodal data has been involved in the RL paradigm recently. Visual data are commonly involved in the observation space of RL when agents receive image-based information from the environment, e.g., in applications such as robots [61] and video game control [2]. Compared with the visual data, natural languages are usually included when RL agents are given specific tasks when interacting with the environments. The use of natural languages in RL can be divided into the following two categories [17].

1) *Language-Conditional RL:* In language-conditional RL, the problem itself requires the agent to interact with the environment through language. Specifically, there are two ways to integrate natural language in RL: a) *task description*: the task or instruction is described in natural languages, e.g., instruction following, where the agents

learn to interpret the instructions first and then execute actions and b) *action space or observation space:* natural language is part of the state and action space, e.g., text games, dialog systems, and question and answering (Q&A). This class of RL leverages natural language as a direct component of the RL process, guiding the agent's actions and decisions within the language environment.

2) *Language-Assisted RL*: In language-assisted RL, natural language is used to facilitate learning but not as a part of problem formulation. Two usages of language-assisted RL are: a) *communicating domain knowledge*: the text containing task-related information can be helpful for agents. Therefore, wikis and manuals related to the environment can potentially assist the agents in such cases and b) *structuring policies*: structuring policies is to communicate information about the state or dynamics of the environment based on language instead of representations of the environment or models. In such cases, language can be leveraged to shape representations toward a generalizable abstraction, such as using "avoid hitting the wall" instead of representations of a policy. This approach represents a more indirect use of natural language, serving as a guide or enhancer to the primary RL tasks.

The integration of multimodal data challenges the RL paradigm since the agent has to simultaneously learn how to process complex multimodal data and optimize the control policy in the environment [16]. Issues such as natural language understanding [62] and visual-based reward function design [63] require to be addressed.

B. Background of LLMs

LLMs typically refer to the Transformer-based language models [12] containing billions of parameters and being trained on massive text data (i.e., several terabytes (TBs) scale) [64], such as GPT-3 [28] and LLaMA [65]. The extensive number of parameters and Internet-scale training data enable LLMs to master a diverse array of tasks, resulting in enhanced capabilities in language generation, knowledge representation, and logical reasoning, as well as improved generalization to novel tasks.

The development and effectiveness of LLMs are largely driven by *scaling laws*, i.e., as these models grow in size—both in terms of their parameter count, and the data they are trained on—they tend to exhibit *emergent abilities* that are not present in small models [66], [67], [68], such as in-context learning, reasoning, and generalization. Here, we briefly introduce such capabilities of LLMs in detail.

1) *In-Context Learning*: In-context learning capability eliminates the need for explicit model retraining or gradient update [28], as it can generate better responses or perform tasks by inputs cueing examples or related knowledge. Specifically, task-related texts are included in the prompts as context information, helping the LLMs to understand the situations and execute instructions.

2) *Instruction Following*: Leveraging diverse task-specific datasets formatted with natural language descriptions (also called *instruction tuning*), LLMs are shown to perform well on unseen tasks that are also described in the form of natural language [69], [70], [71]. Therefore, this capability equips LLMs with the ability to comprehend instructions for new tasks and effectively generalize across tasks not previously encountered, even in the absence of explicit examples.

3) *Step-by-Step Reasoning*: For smaller models, tackling multistep tasks, such as solving math word problems, often proves to be challenging. However, LLMs can address the complex task effectively with sophisticated prompting strategies such as chain of thought (CoT) [72], tree of thought (ToT) [73], and graph of thought (GoT) [74]. These strategies structure the problem-solving process into sequential or hierarchical steps, facilitating a more articulated and understandable reasoning pathway. In addition, prompts designed for planning enable LLMs to output sequences that reflect a progression of thoughts or actions, proving invaluable for tasks demanding logical sequence or decision-making outputs.

III. LLM-Enhanced Reinforcement Learning

A. Definition

RL agents are often tasked with making robust and deliberate decisions using multimodal information in real–real applications, whether in the MDP setting or within the context of specific task descriptions. Examples include robots designed to follow natural language instructions while navigating physical environments or visual games with tasks described in natural language [75], [76], [77]. However, it is challenging for conventional RL methods as the agent is required to simultaneously interpret complex multimodal data and optimize control policies amidst ever-changing environments [78]. Compounding these challenges are issues like sample inefficiency, the difficulty of crafting reward functions that accurately reflect multimodal inputs, and the need for robust generalization across varied tasks and settings.

The rapid advancements in LLMs present a viable solution to these challenges, thanks to their potent natural language understanding and reasoning abilities, coupled with recent progress in incorporating visual data processing [79]. This dual capability enables LLMs to interpret and act upon complex multimodal information effectively, serving as a robust helper for enhancing the RL paradigm for real-world applications.

Nevertheless, despite the powerful functionalities of LLMs, the current studies are varied and lack a standard concept correctly specifying the systematic methodology, which impedes the advancement of research in this area. Therefore, we introduce the concept called *LLM-enhanced RL* as follows.

LLM-enhanced RL refers to the methods that utilize the multimodal information processing, generating, reasoning, and other high-level cognitive capabilities of pretrained, knowledge-inherent LLM models to assist the RL paradigm.

LLM-enhanced RL differs from traditional model-based RL by leveraging knowledge-rich LLM models. This approach provides two key advantages. First, LLM equips the agent

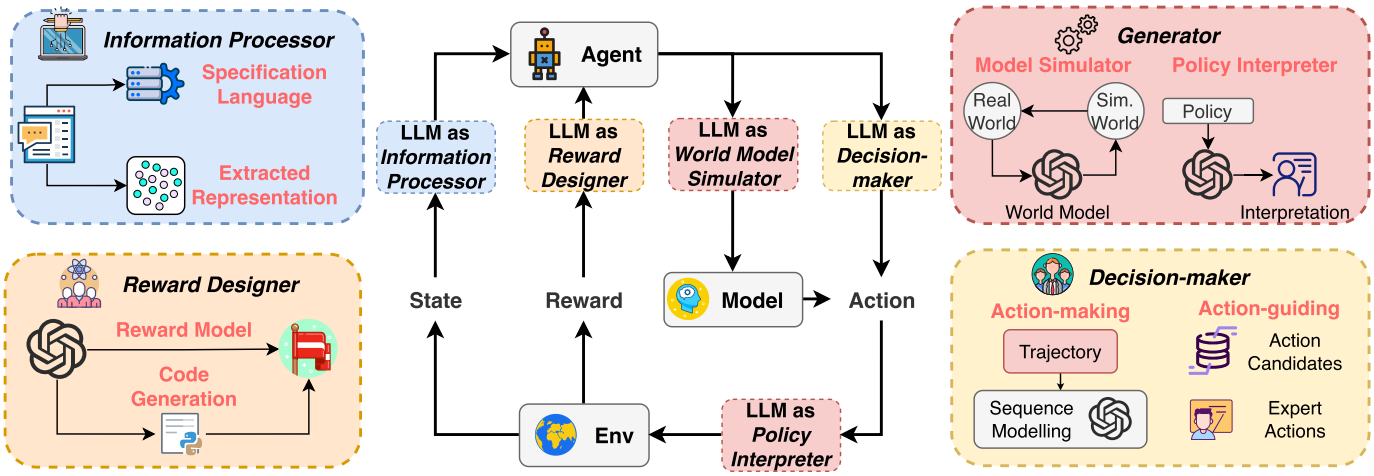

Fig. 2. Framework of LLM-enhanced RL in classical agent–environment interactions, where LLM plays different roles in enhancing RL.

with substantial pre-trained capabilities at the beginning of the learning process, such as reasoning and high-level planning, etc. Second, it offers superior generalization capabilities. Pretrained on diverse data, LLMs can effectively transfer knowledge across domains, enabling better adaptation to unseen environments than conventional data-driven models. Last, LLM-enhanced RL addresses a key limitation of pretrained models: their inability to interact with environments to expand their knowledge and ground themselves in specific domains. Through environmental interactions, this approach generates task-specific data, grounds the LLM in particular domains by in-context learning, and eventually helps them adapt to dynamic changes with continuous learning.

B. Framework

The framework of LLM-enhanced RL is illustrated in the center of Fig. 2, which is founded on the classical agent–environmental interaction paradigm. Along with the trial-and-error learning process, LLM processes the state information, redesigns the reward, assists in action selection, and interprets the policy after the action selection.

Specifically, on the one hand, when the agent receives the state and reward information from the environment, LLM is able to process or modify the information to either filter unnecessary natural language-based information or design appropriate rewards to accelerate the learning process, based on the natural language understanding and reasoning capabilities. On the other hand, when the agent is about to choose an action based on the observation, LLM can assist the action selection process by either simulating a world model or serving as the policy network to generate reasonable actions based on the modeling capability and common-sense knowledge. In addition, after the action selection process, integrating state, reward, and action information, LLM can interpret the underlying possible reasons behind the policy selection, which helps human supervisors understand the scenarios for further system optimization.

Based on the functions of LLM in the framework, we extract characteristics of LLM-enhanced RL and further divide into four different LLM roles in LLM-enhanced RL, including information processor, reward designer, generator, and decision-maker, which will be elaborated in Sections IV–VII.

C. Characteristics

The LLM-enhanced RL paradigm enhances the vanilla RL paradigm with the following characteristics.

1) *Multimodal Information Understanding*: LLMs enhance RL agents' comprehension of scenarios involving multimodal information, enabling them to learn from tasks or environments described in natural language and vision data more effectively.
2) *Multitask Learning and Generalization*: Benefiting from the multidisciplinary pretrained knowledge and powerful sequence modeling capability, LLMs empower RL agents by providing a high-capacity model capable of accommodating task variances and transferring knowledge across multiple tasks, assisting in handling multiple tasks.
3) *Improved Sample Efficiency:* Given the inherent exploratory nature, the RL paradigm demands significant samples to learn. Pretrained LLM can enhance data generation by simulation or leverage the prior knowledge to improve the sample efficiency of RL.
4) *Long-Horizon Handling*: RL becomes more challenging as the length of trajectory increases, due to the credit assignment problem. LLMs can decompose complex tasks down into subtasks to assist RL agents in planning over longer temporal horizons, aiding in the decision-making process for complex, multistep tasks such as the minecraft game.
5) *Reward Signal Generation:* Based on the context understanding and domain knowledge, LLMs contribute to the reward shaping and reward function designing, which help guide the RL toward effective policy learning in sparse-reward environments.

D. Taxonomy

In this section, we illustrate different roles of LLMs within the above framework, by detailing their functions and corresponding issues of RL they address.

1) *Information Processor*: When observation or task description involves language or visual features, it is challenging for the agent to comprehend the complex information and optimize the control policy simultaneously. To release the agent from the burden of understanding the multimodal data, LLM can serve as an information processor for environment information or task instruction information by: 1) extracting meaningful feature representations for speeding up network learning and 2) translating natural language-based environment information or task instruction information into formal specific task languages to reduce learning complexity.
Application Example: In instruction-following RL for robots, tasks can have unbounded natural language forms due to users' diverse speaking habits, which can impede RL learning performance. LLMs transform these varied natural language instructions into a unique task language, enabling more robust RL performance [48].
2) *Reward Designer:* In complex task environments where the reward is sparse or a high-performance reward function is hard to define, using the prior world knowledge, reasoning abilities, and code generation ability, LLM can serve as two roles: a) an implicit reward model to provide reward values based on the environment information, either by training or prompting and b) an explicit reward model that generates executable codes of reward functions that transparently specifies the logical calculation process of reward scalars based on the environment specifications and language-based instructions or goals.
Application Example: For complex robotic control problems, such as dexterous manipulation, reward design requires both expertise and trial-and-error. LLM designs reward functions based on knowledge and iteratively improves it based on the performance [80].
3) *Decision-Maker:* RL faces challenges such as sample inefficiency and exploration inefficiency. To address these challenges, LLMs can be leveraged as decision-makers in RL, offering promising solutions through two main approaches.
   a) *Action-Making:* LLMs treat offline RL as a sequence modeling problem, using rewards for the conditional generation of actions. Pretrained on diverse, the Internet-scale data, LLMs possess advanced semantic understanding capabilities, which can be exploited to accelerate offline RL learning.
   b) *Action-Guiding:* LLMs act as expert instructors to produce a reduced set of action candidates or expert actions.

   The action candidates constrain the original action space, thereby enhancing exploration efficiency. Expert actions encapsulate prior knowledge from LLMs. When incorporated to regularize the policy learning, this expert knowledge is distilled into an RL agent, resulting in better sample efficiency.
   Application Example: In embodied robot, given human-instruction and language-based description, LLM generates potential actions for the robot to choose from [81].
4) *Generator:* Model-based RL hinges on precise world models to learn accurate environmental dynamics and simulate high-fidelity trajectories. In addition, interpretability remains another important issue in RL. Using the multimodal information understanding capability and prior common-sense reasoning ability, LLMs can be: a) a generator to generate accurate trajectories in model-based RL and b) generate policy explanations with the prompts of related information in explainable RL (XRL). *Application Example:* In minecraft item crafting, LLMs generate abstract world models—hypothesized sequences of subgoals for a given task. These LLM-generated world models guide RL agents' exploration and learning and the RL agent verifies and corrects the world model through gameplay, combining LLM knowledge with grounded experience to achieve an order of magnitude improvement in sample efficiency over traditional methods [82].

IV. LLM as Information Processor

The normal way for deep RL with language or visual information is to jointly process the information and learn a control policy, end-to-end. However, this demands the RL agent to learn to comprehend the information and manage the task simultaneously. In addition, learning the language or visual features by simply relying on the reward function is challenging and may narrow the learned features to a narrow utility, hampering the generalization ability of the agent [16].

With the advances in unsupervised techniques and large-scale pretrained models for CV and NLP, the decoupled structure, where the encoders are separately trained, has gained popularity [16], [83], [84]. Utilizing the powerful representation ability and prior knowledge, the pretrained LLM or vision-language model (VLM) model can serve as an information processor for RL. They can extract observation representations for downstream networks or translate natural language into formal specifications, enabling the execution of multiple tasks. This multitask capability improves sample efficiency and zero-shot performance, allowing agents to generalize effectively across diverse and sparse-reward environments.

A. Feature Representation Extractor

Adopting the large pretrained models in CV and NLP, the learned feature representation can be a scaffold embedding for downstream network learning and increase the sample efficiency. As illustrated in Fig. 3(a), the usages can be further divided into two categories according to whether the model is trained simultaneously. One way is to directly use the frozen pretrained model to extract embeddings from the observation $\mathcal{O}_t$ and another way is to further fine-tune the pretrained model using contrastive learning with a contrastive loss $\mathcal{L}_t^c$ to achieve better adaptation in new environments.

1) Frozen Pretrained Model: Using the frozen large-scale, pretrained model is the straightforward way. Paischer et al. [85] proposed history compression via language models

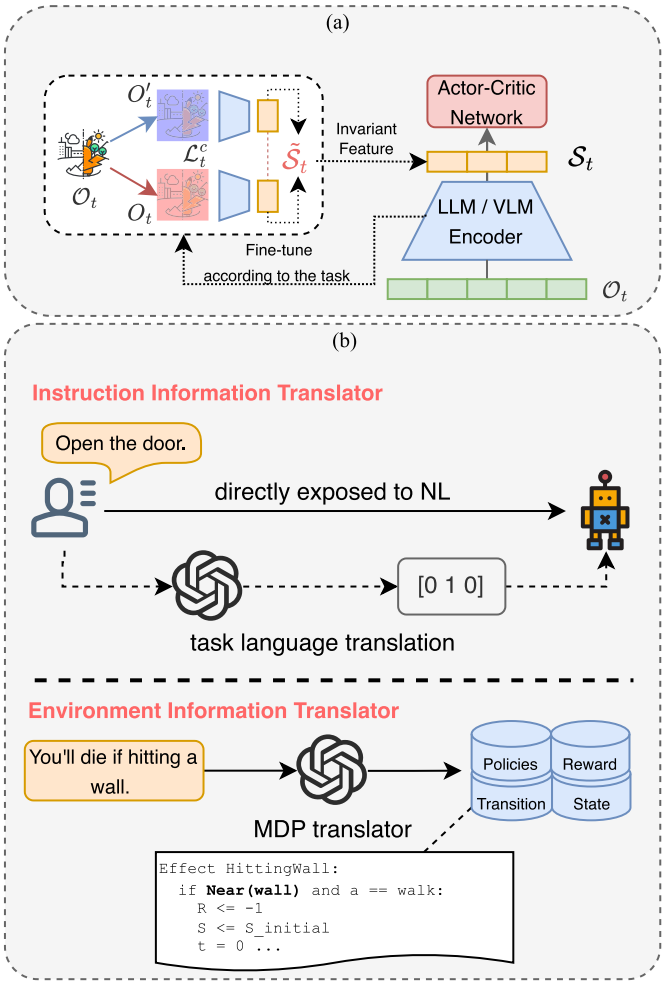

Fig. 3. LLM as an information processor. (a) Feature representation extractor: frozen/fine-tuned LLM extracts meaningful representations for downstream RL networks. In the fine-tuning process, given observation ($\mathcal{O}_t$), invariant feature abstraction ($\tilde{S}_t$) is learned with the contrastive loss ($\mathcal{L}_t^c$). Then, the invariant is fed into the actor–critic network. After fine-tuning, given different observations (O_t) and (O_t') with appearance variation, the extracted representation is invariant, leading to robust RL performance. (b) Language translator: LLM interprets diverse natural language inputs, converting them into a standardized, task-specific format that the RL agent can efficiently process and act upon.

(HELMs) to utilize a frozen pretrained language transformer to extract history representation and compression and thus addresses the problem of partially observed MDP by approximating the underlying MDP with the past representation. Specifically, the framework first uses FrozenHopfield, a frozen associative memory to map observations $[o_{t-2}, o_{t-1}, o_t]$ to a compressed representation h_t and then concatenate it with a learned encoding of the current observation via a convolutional neural network. Such a method solves the problem of how to effectively utilize the compressed history for policy optimization. After that, semantic HELM [86] proposed a human-readable memory mechanism that summarizes past visual observations in human language and uses multimodal models to associate visual inputs with language tokens. The memory is a semantic database $\mathcal{S}$ constructed by encoding prompt-augmented tokens from the vocabularies of contrastive language-image pretraining (CLIP) [87] and the pretrained language models. Given an observation o_t, the agent retrieves top-k embeddings from $\mathcal{S}$ as actor–critic input to assist the policy optimization. Such a memory mechanism provides a human-readable representation of the past and helps the agent to cope with partially observable environments. In the experiment, they used the PPO [3] algorithm and pretrained Transformer XL [88] model. When testing on the partially observable environments, they found that the extracted semantics help the memoryless agent obtain comparable scores with memory-based methods trained on long trajectories. However, one limitation of such frozen pretrained models is that the representations cannot dynamically adjust according to the task and environment.

2) Fine-Tuning Pretrained Model: When trained RL agents are deployed in real-world applications, their performance often deteriorates under significant appearance variations (out-of-distribution data) due to overfitting to training scenarios. For example, robots with vision-based navigation tasks may fail when the environment color changes. Invariant feature representations serve as a form of state abstraction that remains consistent across out-of-distribution appearance variations, such as added noise, brightness changes, or slight rotations. When encountering appearance changes or out-of-distribution data, though the observation changed, the representation (feature embedding) that fed into the policy/value network is nearly unchanged, leading to robust RL performance and increased generalization in unseen environments.

Contrastive learning is a common way to learn the invariant feature representation. It learns representations from high-dimensional data by contrasting positive examples against negatives. Given a query q, the goal is to match query q more closely to a positive key k_+ than to any negative keys $\mathbb{K} \setminus \{k_+\}$ in a set $\mathbb{K}$. This process is modeled using similarity measures, such as the dot product ($q^{\mathrm{T}}k$) or the bilinear product ($q^{\mathrm{T}}Wk$), where W is a weight matrix. To effectively learn these representations, a contrastive loss function such as InfoNCE [89] is used

$$\mathcal{L}^c = \log \frac{\exp\left(q^T W k_+\right)}{\exp\left(q^T W k_+\right) + \sum_k^{K-1} \exp\left(q^T W k_i\right)} \tag{1}$$

where exp is the exponential symbol and the whole $\mathcal{L}^c$ can be viewed as the log-loss of a softmax classifier, treating the matching of q to k_+ as a multiclass classification problem among K classes. By maximizing alignment between different changes of the same observation via the above loss, the model can learn the invariant representations.

When combining with RL, given different RL tasks, the required invariant feature representations should be adjusted accordingly. Therefore, researchers have explored different ways to improve the contrastive learning. In [90], to achieve the zero-shot capability of embodied agents, the author devised a visual prompt-based contrastive learning framework that uses a pretrained VLM to learn the visual state representations. The visual prompts are learned on expert demonstrations from domain factors, such as camera settings and stride length. By contrastively training the VLM on a pool of visual prompts along with the RL policy learning process, the learned representations are robust to the variations of environments,

leading to an increase of 18%–20% success rates on unseen scenarios and improved generalization capability. Based on the contrastive learning, another method ReCoRe [91] added an intervention-invariant regularizer in the form of an auxiliary task, such as depth prediction and image denoising to explicitly enforce invariance of learned representations to environment changes.

B. Language Translator

The unbounded and diverse representation of natural languages in both human instruction and environmental information impedes policy learning. LLM can be leveraged as a language translator to reduce the additional burden of comprehending natural language for RL agents and increase sample efficiency. As illustrated in Fig. 3(b), LLM transforms the diverse and informal natural language information into formal task-specific information, such as feature representation or task-specific languages, thus assisting the learning process of the RL agent.

1) Instruction Information Translation: One application of LLM is to translate natural language-based instructions for instruction-following applications. Pang et al. [48] investigated an *inside–out* scheme for natural language-conditioned RL by training an LLM that translates the natural language to a task-related unique language. Such an inside–out scheme prevents the policy from being directly exposed to natural language instructions and helps efficient policy learning. Another literature *STARLING* [92] used LLM to translate natural language-based instructions to game information and example game metadata. The translated information are then fed forward to *Inform7*, an interactive fiction game engine, to develop a large amount of text-based games for RL agents to master the desired skills.

2) Environmental Information Translation: On the other hand, LLM can also be used to translate the natural language environment information into formal domain-specific language that can specify MDP information, which converts natural language sentences into grounded usable knowledge for the agent. Previous works generally ground natural language into individual task components, such as task objectives description [93], rewards [94], and polices [95], [96]. To unify the information about all the components of a task, Spiegel et al. [97] introduce *RLang*, a grounded formal language capable of expressing information about every element of an MDP and solutions, such as policies, plans, reward functions, and transition functions. By using LLM to translate natural language to RLang and train RL agents upon RLang, the agents are capable of leveraging information and avoid having to learn *tabula rasa*.

C. Summarization and Outlook

For information processing, LLM is used to accelerate RL learning processing by decoupling the information processing task and the controlling task, where LLM extracts feature representations or handles the natural language-based information.

When multimodal data are involved in the environment, e.g., robot manipulation tasks, the information processing task becomes more challenging. For one thing, the misalignment or contradiction between different modalities may exist [98]. The modality weighting techniques that adaptively learn the importance of different modalities by attention mechanisms provide a potential solution for this problem [99]. In addition, CLIP-based multimodal foundation models using large-scale image–text pairs have shown outstanding zero-shot ability in various multimodal tasks [87]. By learning cross-modal and task semantics, the misaligned modality can be replaced with a virtually generated modality [100]. For another, how to effectively combine the information between different modalities into a unified representation, i.e., multimodal fusion, is also an important issue. In [101], multimodal contrastive learning with attention mechanisms, which learns the intramodal and intermodal representations, was proposed. Different attention heads align the agreement from one modality to another and vice versa, providing an informative representation for the RL agent. However, most multimodal learning methods do not consider the task information and only focus on the modality alignment, remaining an area to be explored.

In the following, we list potential directions for future research.

1) *Feature Representation Extractor:* Although the use of LLMs as feature representation extractors has shown promise in enhancing RL, several challenges persist in future research. Short-term goals include developing a computationally efficient feature extractor and improving the generalization of LLM-derived representations. In the long term, researchers should focus on exploiting task compositionality for better generalization and creating adaptive extraction methods for diverse control tasks.
2) *Language Translator:* Current existing works are still limited. Short-term objectives involve exploring LLMs' ability to handle more tasks and improving translation efficiency and accuracy in RL contexts. Long-term goals include developing multimodal translation capabilities and integrating these with RL algorithms, achieving a more general translator, and helping agent learning.

V. LLM as Reward Designer

The reward signal is the most important information to instruct agent learning in RL [1]. However, despite the fundamental importance, high-performing reward functions are known to be notoriously difficult to design [102]. First, specifying human notions of desired behavior is difficult via designed reward functions or requires huge expert demonstrations. Moreover, dense rewards that accurately provide learning signals require either manually decomposing the general goal into subgoals [103] or rewarding interesting auxiliary objectives [104]. Nevertheless, both of these methods suffer from the need for expert input and meticulous manual crafting [23].

Benefiting from pretrained common-sense knowledge, code generation, and in-context learning ability, LLM has the potential to design or shape reward functions for DRL by leveraging

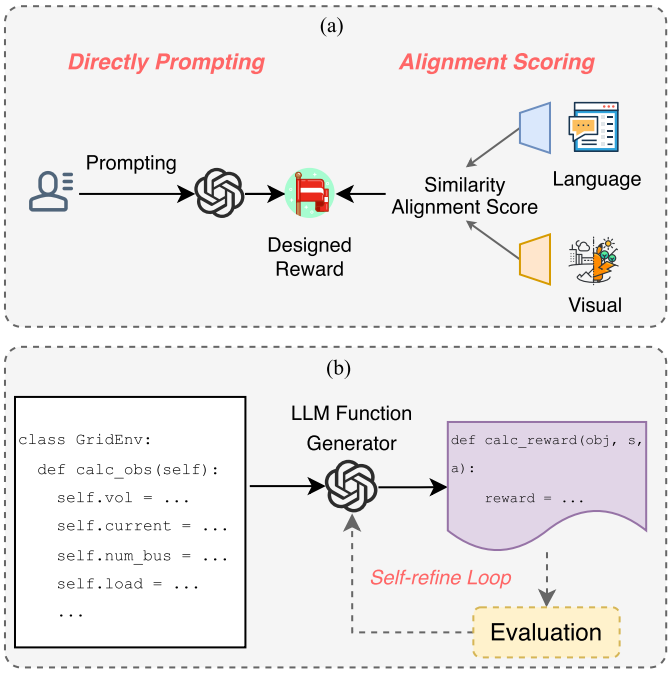

Fig. 4. LLM as a reward designer. (a) Implicit reward model: LLMs provide rewards through direct prompting or alignment scoring between language instructions and visual observations. (b) Explicit reward model: LLMs generate executable code for reward functions, with the potential for self-refinement through evaluation loops.

natural language-based instructions and environment information. In this section, we review recent literature in which LLMs act as reward models that implicitly provide reward values or explicitly write executable reward function codes detailing the calculation process of reward scalars.

A. Implicit Reward Model

A large pretrained model can be an implicit reward model that directly provides auxiliary or overall reward value based on the understanding of task objectives and observations. The methods are illustrated in Fig. 4(a). One way is by directly prompting with language descriptions and another way is by scoring the alignment between the feature representation of the visual observations and language-based instructions.

1) Direct Prompting: Kwon et al. [105] simplified the manual reward design by prompting an LLM as a proxy reward function with examples of desirable behaviors and the preferences description of the desired behaviors. Wu et al. [106] proposed a read and reward framework that utilizes LLMs to read instruction manuals to boost the learning policies of specific tasks. The framework includes a Q&A extraction module for information retrieval and summarization and a reasoning module for evaluation. Experimentally, they show RL algorithms, by their design, can obtain significant improvement in performance and training speed. Carta et al. [44] proposed an automated reward shaping method where the agent extracts auxiliary objectives from the general language goal. Using a question generation and Q&A system, the framework guides the agent in reconstructing partial information about the global goal and provides an intrinsic reward signal for the agent. This intrinsic reward incentivizes the agent to produce trajectories that help them reconstruct the partial information about the general language goal. To acquire a generalizable policy by continually learning a set of tasks is challenging for RL agents since the agent is required to retain the previous knowledge while quickly adapting to new tasks. Chu et al. [107] introduced the language agent feedback interactive RL (Lafite-RL) framework that provides interactive rewards mimicking human feedback based on LLMs' real-time understanding of the agent's behavior. By designing two prompts, one to let LLM understand the scenario and the other to instruct it about the evaluation criterion, their framework can accelerate the RL process while freeing up human effort during the interaction between agent and environment.

Algorithm 1 LAMP in [108]

1: Initialize parameters for Masked World Models (MWM)
2: Load pretrained DistilBERT [109] for language preprocessing
3: Load pretrained reusable representations for robot manipulation (R3M) visual encoder and score predictor
4: Initialize empty replay buffer
5: Populate language prompt buffer and synonym buffers with predefined samples
6: **for** each training episode **do**
7: Randomize scene textures from Ego4D and RLBench datasets
8: Sample ShapeNet objects and language prompts
9: Insert ShapeNet objects into the scene
10: Generate and process language embeddings using DistilBERT
11: Execute policy to collect transitions
12: Assign LAMP rewards using R3M score predictos
13: Update buffers and train MWM with augmented rewards
14: **end for**

2) Alignment Scoring: For visual RL, some literature utilizes VLMs as reward models to align multimodal data and calculate the similarity score using metrics, such as cosine similarity. Rocamonde et al. employ the CLIP model as a zero-shot reward model to specify tasks via natural language [63]. They first compute the probability $p_{o_t,l}$ that the agent achieves a goal given by language description l out of a set of potential goals $l' \in \mathcal{L}$ in the task set $\mathcal{L}$ using the softmax computation with temperature τ over the cosine similarity between visual state embeddings $f_\theta(o_t)$ and language description embeddings $g_\theta(l)$ across the set of potential goals l'

$$p_{o_t,l} = \frac{\exp\big(f_\theta(o_t) \cdot g_\phi(l)/\tau\big)}{\sum_{l'} \exp\big(f_\theta(o_t) \cdot g_\phi(l')/\tau\big)}. \tag{2}$$

Then, the reward is obtained by a binary reward function $r_t = r(o_{t+1}, l) = \mathbb{I}[p_{o_{t+1},l} \geq \beta]$, which thresholding the probability. Their framework only requires a single-sentence text prompt description of the desired task with minimal prompt engineering. Another work [110] constructed reward

signals based on the similarity between natural language-based description and embeddings from the pretrained VLM encoder. By labeling the expert demonstration with the reward signals, the framework effectively mitigates the problem of misgeneralization. Adeniji et al. [108] proposed the language reward modulated pretraining (LAMP) framework as a pertaining utility for RL as opposed to a downstream task reward to warm-start sample-efficient learning. The framework leverages frozen, pretrained VLMs, such as R3M [111] to generate noisy, albeit shaped exploration rewards by computing the alignment score between instructions and image observations. The algorithm is presented in Algorithm 1. They used masked world model [112], a visual model-based RL algorithm for robot manipulation based on images and instructions. The images were downloaded from Ego4D [113], and the language instructions were obtained by querying ChatGPT. The reward is then calculated from the R3M alignment score. After that, the generated rewards are optimized with standard novelty-seeking exploration rewards for language-conditioned policy. Wang et al. [114] explored preference-based RL, where the agent learns a reward function from preference labels over the behaviors. A VLM is leveraged to generate preference labels given visual observations and a text description of the task goal. The evaluation is based on a series of vision-based manipulation tasks. Results suggested that prompting VLMs to produce preference labels for reward learning leads to better performance, in contrast to treating them as reward functions to produce raw reward scores.

B. Explicit Reward Model

Another way to design reward functions is by generating executable codes that explicitly specify the details of the calculation process, as illustrated in Fig. 4(b). Compared with the implicit reward value provision, this explicit way transparently reflects the reasoning and logical process of LLMs and thus is readable for humans to further evaluate and optimize.

To help robots learn low-level actions, Yu et al. [115] harnessed the code generation ability of LLMs to define the lower level reward parameters based on high-level instructions. Using such a reward design paradigm, their work bridges the gap between high-level language instructions to low-level robot actions and can reliably tackle 90% of the designed tasks compared with the 50% of the baseline. Motivated by the capability of LLM for self-refinement [116], Song et al. [117] proposed a framework with a self-refinement mechanism for automated reward function design, including initial design, evaluation, and self-refinement loop. Their results indicate that the LLM-designed reward functions are able to rival or surpass manually designed reward functions. Similarly, *Eureka* [80] developed a reward optimization algorithm with self-reflection. The algorithm is outlined in Algorithm 2. In each iteration, it uses an environmental source code and task description to sample different reward function candidates from a coding LLM. Then, the candidates are used to instruct RL training. After training, the results are used to calculate the scores of the reward candidates. Then, the best reward function codes are selected for reflection, where LLM uses the reasoning capability to progressively improve the reward code. In the experiment, results show that their proposed method can achieve human-level performance on reward design and solve dexterous manipulation tasks that were previously infeasible by manual reward engineering. Another work Text2Reward [118] generated shaped dense reward functions as executable programs based on the environment description. Given the sensitivity of RL training and the ambiguity of language, the RL policy may fail to achieve the goal. Text2Reward addresses the problem by executing the learned policy in the environment, requesting human feedback, and refining the reward accordingly.

Algorithm 2 Eureka [80]

Require: Task description l, environment code C, coding LLM M_c, evaluation function E, initial prompt `prompt`, optimization iterations N, iteration batch size K

for $i \leftarrow 1$ **to** N **do**
  `// Sample` K `reward code candidates from coding LLM` M_c
  $R_1, \ldots, R_K \leftarrow \texttt{sample}(l, M_c, C, \texttt{prompt})$
  `// Evaluate reward candidates`
  $s_1^i = E(R_1^i),\ s_2^i = E(R_2^i),\ \ldots,\ s_K^i = E(R_K^i)$
  `// Select the best reward code`
  $\text{best} = \arg\max_k (s_1^i, \ldots, s_K^i)$
  `// Reward reflection`
  $\texttt{prompt} \leftarrow \texttt{prompt:Reflection}(R_{\text{best}}^i, s_{\text{best}}^i)$
  `// Optimize Eureka reward code`
  if $s_{\text{best}}^i > s_{\text{Eureka}}$ **then**
    $R_{\text{Eureka}}, s_{\text{Eureka}} \leftarrow (R_{\text{best}}^i, s_{\text{best}}^i)$
  end if
end for
return R_{Eureka}

C. Summarization and Outlook

The use of LLMs as reward designers in RL offers a more natural and efficient way to design complex reward functions. By leveraging NLP capabilities, LLMs simplify the traditionally challenging task of reward function design, enhancing both the efficiency and effectiveness of RL algorithms.

However, the inherent biases in LLMs may transfer to the designed reward functions, potentially resulting in suboptimal or harmful behaviors [119]. This presents a potential risk requiring careful consideration. While existing mitigation efforts primarily address biases in demographic, cultural, and political beliefs [120], task-specific biases remain understudied, thus limiting the applicability of LLM-designed reward functions. Toward this end, we list some potential solutions. First, in preference-based RL, when an RL agent optimizes against biased reward models generated by LLMs to predict human preferences, overoptimization, and overfitting may occur [121], impeding the learning of the true reward function. Reward function regularization [122] includes the agent preference generated by the value function as a regularization term to mitigate the risk, which helps recover the true underlying reward function. Second, human-in-the-loop

approaches are another direction to prevent harmful behaviors from designed reward functions. One viable solution is to design an evaluation module where humans can intervene and correct undesired behaviors and reward functions when the agent's action violates a predefined set of rules. Finally, principles from ensemble learning suggest that combining the reward functions from different LLM models mitigates bias from individual LLMs, leading to improved unbiased performance compared with a single LLM [123].

In the future, we expect advancements in the following areas.

1) *Implicit Reward Model:* An immediate focus may consider improving how well LLM-generated rewards align with human intentions. This involves refining the quality of language instructions to reduce ambiguities and inaccuracies, ensuring that the reward functions align precisely with human notions of desired behavior. In the longer term, generalization and transferability of LLM-generated rewards across different tasks and environments, especially in complex, high-dimensional visual environments, is also an important direction to explore.
2) *Explicit Reward Model:* A key limitation in reward code generation is its dependency on pretrained commonsense knowledge, which can be restrictive for highly specialized tasks not covered in training data. Therefore, short-term goals may include enhancing prompts with detailed, task-specific information and external knowledge. In addition, the reliance on manually designed templates for motion descriptions limits the adaptability. Looking further ahead, researchers might develop automated or unified processes for designing templates, moving beyond the current limitations of manual motion description templates.

VI. LLM as Decision-Maker

LLMs trained on a massive amount of data show impressive results in language understanding tasks [28], instruction-following [124], vision-language navigation [125], and tasks requiring planning and sequential reasoning [81]. Such success motivates researchers to explore the potential of LLMs for decision-making problems. In this section, we divide the role of LLM as: 1) the *action-maker* that generates actions and 2) the *action-guider* that instructs the actions.

A. Action-Making

Transformer-based models such as decision transformer (DT) [126] have shown great potential in offline RL domain. Instead of using the traditional trial-and-error way, these models treat offline RL as a sequence modeling problem, yielding promising results. As LLM itself is a large-scale Transformer-based model, a natural thought is to leverage the pretrained power of LLM within this paradigm.

We term this function of LLM as action-making, and identify two typical approaches, as illustrated in Fig. 5(a). In the first approach (left figure), pretrained LLM is fine-tuned and then employed for action generation. In the second approach (right figure), goal/instruction along with trajectory are fed to pretrained LLM. Moreover, a task-specific smaller model is appended after the fine-tuned LLM to facilitate rapid adaptation to diverse tasks.

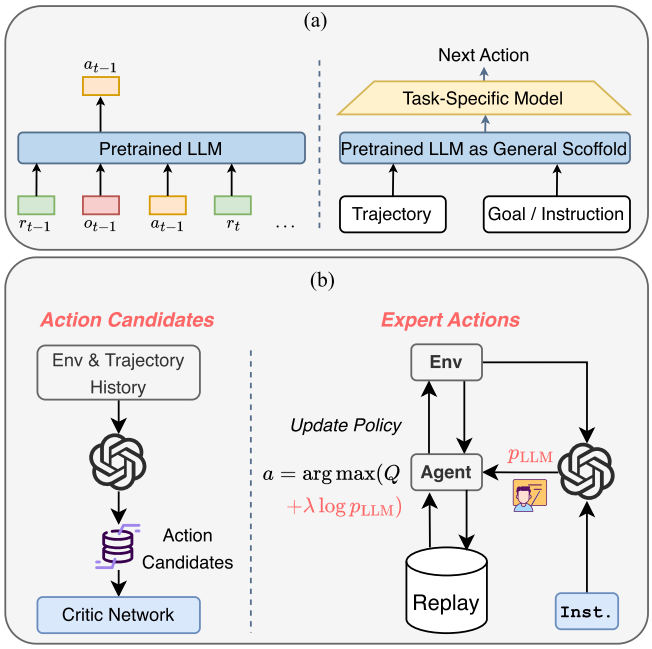

Fig. 5. LLM as a decision-maker. (a) Action-making: given a T-length trajectory $\tau = (\hat{R}_1, s_1, a_1, \ldots, \hat{R}_T, s_T, a_T)$ as a sequence of ordered return-to-go $\hat{R}$, action a, and states s, LLM learns to predict future action a'_t by minimizing the mean squared error loss $\mathcal{L} = \sum_t \|a_t - a'_t\|_2^2$. (b) Action-guiding: LLM generates a reduced set of action candidates for agents or generates expert actions to regularize RL learning.

Pretrained LLMs outperform basic DT in generalization and sample efficiency, especially for sparse-reward and long-horizon tasks. LLMs' latent representations, learned from diverse linguistic data, provide valuable prior knowledge for new tasks. This knowledge enables LLMs to solve unseen tasks with less training data, by transferring knowledge from similar tasks and predicting high-reward actions even with sparse feedback. For instance, comparing pretrained LLM with basic DT, Li et al. [127] reported a 43.6% improvement in out-of-distribution (novel) task completion rates while requiring less training data (e.g., 500 versus 10 K). In addition, Shi et al. [128] demonstrated a 50% performance gain in sparse-reward environments like Kitchen and Reacher2d. For long-horizon tasks, pretrained representations encode future information, guiding decision-making overextended sequences. For instance, in AntMaze, a long-horizon navigation environment, pretrained representations yield five times higher scores compared with non-pre-trained counterparts [129].

Several studies have further explored the application of LLMs in offline RL, demonstrating their versatility and effectiveness across various tasks and benchmarks. Reid et al. [130] investigated the transferability of general language models on specific RL tasks. Fined-tuned on offline RL tasks (control and games), these general language models outperform DT and reduce training time by 3×–6× on D4RL benchmark [131]. Li et al. [127] used pretrained LLM as a general scaffold for

task-specific model learning, where goals were added along with observations as the input for LLM. Results on embodied decision-making tasks demonstrate that their proposed method outperforms others with less training data, especially when generalizing to novel tasks. In addition, they found representations in pretrained language models can aid learning and generalization even outside of language. To unify language reasoning with actions in a single policy, Mezghani et al. [132] generated textual captions interleaved with actions when training the Transformer-based policy. Results show that by using captions describing the next subgoals, the reasoning policy can consistently outperform the caption-free baseline. For scenarios where data collection is costly and risky, Shi et al. [128] proposed a general framework to effectively use pretrained LLM for offline RL. The pretrained LLM is based on DT. To combine the pretrained knowledge and task-related domain knowledge, they fine-tuned pretrained LLM with the low-rank adaptation (LoRA) method. The architecture of Transformer is based on GPT-2 model with 12 layers and 12 attention heads. To fine-tune the LLM, they obtained the trajectory data from the D4RL dataset. Results show their method achieves state-of-the-art performance in sparse-reward tasks with limited data samples. To integrate multimodal data, e.g., vision and language, into the offline RL, Zitkovich et al. [133] co-fine-tuned VLMs on both robotic trajectory data and Internet-scale vision-language tasks, e.g., visual Q&A. In the framework, they incorporate the actions as natural language tokens and co-fine-tune models on both vision and language datasets. Results show that such co-fine-tune methods can increase generalization performance and the COT reasoning can help the agent perform multistage semantic reasoning and solve complex tasks.

B. Action-Guiding

The action-guider role is illustrated in Fig. 5(b). As an action-guider, LLM guides the action selection by either generating reasonable action candidates or expert actions. By instructing the action selection, LLM improves the sample efficiency and exploration efficiency posed by enormous action spaces and natural language.

1) Action Candidates: In environments such as text-based games, action spaces are large and only a tiny fraction of actions are accessible. Although RL agents can learn through extensive trials, they often face exploration efficiency issues, especially in multitask settings, where agents must manage various tasks simultaneously. LLMs address this challenge by generating a reduced set of action candidates based on task understanding. These candidates are likely to yield high rewards and are applicable across multiple tasks, enhancing exploration efficiency and reducing the need for ineffective exploration. Yao et al. [134] trained a GPT-2 model to generate the candidates. To maximize long-term rewards, another neural network is used to calculate the Q-values of these candidates. When tested in 28 man-made text games from Jericho framework [135], they found that the method excludes nonuseful actions, speeds up the exploration, and consistently achieves higher scores by more than 20%. Following this work, another study [81] proposed the SayCan framework, where instruction-following robots are integrated with an embodied LLM to understand tasks. When receiving instructions, the embodied LLM generates a high-level step-by-step plan. When acting, LLM produces action candidates based on task prompts and then the candidate with the largest critic value is executed. Being evaluated in an office kitchen with real-world robotic tasks from 101 instructions, their method can complete long-horizon, abstract, and natural language instructions on a mobile manipulator.

2) Expert Actions: Traditional RL agents cannot converge to desirable equilibrium in human–AI collaboration or learn efficiently for complex tasks, due to the lack of expert demonstrations. With the understanding of human behavior and general knowledge, LLM solves the issues by producing high-quality expert actions to regularize RL agents. In human–AI collaboration, *instructRL* [136] used LLM to generate prior policy based on human instructions and uses this prior to regularizing the RL objective. Specifically, *instructRL* augments the policy update function with an auxiliary term $p_{\text{LLM}}[\text{lang}(a_t)|\text{lang}(\tau_t^i), \text{inst}]$, the probability of choosing an action based on the trajectory and instructions. Experiments show *instructRL* converges to policies aligned with human preferences. Similarly, to address the sample inefficiency issue of RL, Zhou et al. [137] included the policy difference between the student model and LLM-based teacher into the RL learning loss. Their experiments on simulation platforms demonstrate that the method reduced training iterations by a factor of 1–9. In [138], LLM was leveraged to generate a high-level expert motion plan of robotics tasks, guiding RL policies to efficiently solve robotics control tasks. The high-level language plan breaks long-horizon tasks into stages to execute. Then, a single RL policy was trained across all states and stepped through the language plan. Their results show the proposed method solves long-horizon tasks from raw visual input spanning different benchmarks at success rates of over 85%, out-performing classical, language-based, and end-to-end approaches.

C. Summarization and Outlook

The sample inefficiency and exploration inefficiency remain long-standing challenges for deep RL, particularly in environments with sparse rewards or where data collection is expensive or risky. LLM provides three ways to solve the problems. First, LLM as action-makers treat RL as a conditional sequence modeling problem. The supervised way fine-tunes pretrained LLMs to predict future actions. Benefiting from learned prior knowledge, LLM can perform well even in out-of-distribution, sparse-reward, and long-horizon tasks. Second, LLM as action-guiders generates potential action candidates for RL. With task comprehension, these action candidates promote agents to explore potentially high-task-value states, thus increasing exploration efficiency. Last, LLM generates expert actions to help RL learn from demonstrations. By incorporating demonstrations from LLM, and RL learns specific prior knowledge and improves sample efficiency.

Safety issues are another important topic when using LLMs as decision-makers in RL, particularly for costly and risky

tasks [139]. For action making, DT-based offline RL learns the optimal policy from precollected datasets. Recently, integrating safety constraints has been explored by some works using methods such as pessimistic estimations [140] and stationary distribution correction [141]. These methods set a constant constraint threshold before training. To dynamically adjust the threshold during deployment, a constrained DT that dynamically relabels the reward based on safety-reward trade-offs was proposed [142]. For action-guiding, LLMs are viewed as instructors to provide expert actions. Ensuring the safety of actions generated by LLMs differs from that in action-making. Leveraging ideas from LLM-based agent research, we propose several potential solutions. First, LLMs can be equipped with testing modules that execute code to evaluate action safety and feasibility [143]. Second, implementing a carefully designed human-in-the-loop framework enables safety intervention through human oversight [144]. Finally, developing memory modules with reasoning mechanisms that adaptively learn safety boundaries from past experiences provides a continual learning-based approach to ensuring long-term safety [145].

In the following, we list the challenges and future directions of the two roles as follows.

1) *Action-Making*: Directly employing pretrained large-scale LLM to generate actions demands huge computational resources and huge data for fine-tuning it in specific tasks. In the short term, future work may consider more cost-effective methods such as the LoRA [146] to exploit the power of LLM in direct decision-making. Long-term goals involve developing innovative techniques to efficiently exploit the power of LLMs in direct decision-making, potentially creating hybrid models that combine the strengths of LLMs with more lightweight, task-specific architectures.
2) *Action-Guiding*: Since the LLM acts as an instructor to provide expert actions, the bias and limitations are also inherited by the agent. In addition, LLM itself cannot inherently interrogate or intervene in the environment, which limits LLMs' capabilities to intentionally aggregate information. The short-term goal is to address the inherited biases and limitations when LLMs act as instructors providing expert actions, focusing on methods to filter or correct biased information. In the long term, it is crucial to develop mechanisms for LLMs to actively interrogate and intervene in the environment, enabling them to intentionally aggregate information and improve their capabilities through real-world interactions. Therefore, how to use the information gained from real-world interactions to improve the LLM itself in terms of actuality and reasoning is another important problem.

VII. LLM as Generator

The generative capability of LLMs can be applied to environmental simulation and behavior explanation. On the one hand, growing interests in applying RL to the real world and risky data-collection process suggests a need for imaginary rollouts [147], [148]. Possessed with a powerful modeling capability and world knowledge, LLMs can serve as a *world model simulator* to learn complex environmental dynamics with high fidelity by iteratively predicting the next state and reward, thus increasing the sample efficiency in model-based RL [147], [149]. On the other hand, in RL, interpretability remains an important security issue in current black-box AI systems as they are increasingly being deployed to help end-users with everyday tasks. Explanations of policy can improve the end-user understanding of the agent's decision-making and inform the reward function design for agent learning. In such aspects, LLM can act as a policy interpreter based on their knowledge and reasoning ability. In this section, we classify the above two roles of LLM as a generator and review recent related works.

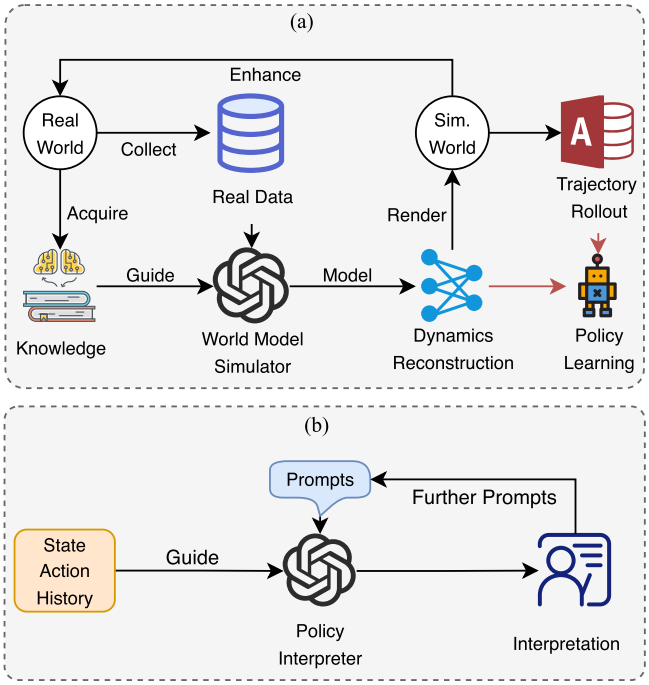

Fig. 6. LLM as a generator. (a) World model simulator: LLM uses real-world data and knowledge to model dynamics, generate simulated worlds, and assist policy learning. (b) Policy interpreter: LLM generates interpretations of agent behavior based on state–action history and prompts, potentially leading to XRL.

A. World Model Simulator

Serving as a world model simulator, LLM is trained as a: 1) *trajectory rolloutor*, which autoregressively generates accurate trajectories for the agent to learn and plan and 2) *dynamics representation learner*, which predicts the latent representation of the world using representation learning. A flowchart of the world model is illustrated in Fig. 6(a). From the real world, knowledge and real data can be collected to construct a world model simulator, which further models the dynamics representation of the world, generates trajectories, and helps the policy learning of the agent in the real world.

1) Trajectory Rolloutor: Similar to the DT [150] in offline RL, pretrained large-scale models were used in model-based to synthesize trajectories. In 2022, Micheli et al. [151] proposed

IRIS, an agent that employs a discrete autoencoder and an autoregressive Transformer to learn the world model for Atari games. With the equivalent of two hours of gameplay in the Atari 100k benchmark, the proposed method outperforms humans in ten out of 26 games. Similarly, Robine et al. [152] applied a transformer to build a sample-efficient world model for Atari games. Utilizing such a Transformer-based world model (TWM), the RL agent can solve the long-term dependency and train a state-of-the-art policy on the Atari 100k benchmark based on generated meaningful experiences from TWM. Visual RL enables the RL agent to learn from visual observations effectively. Chen et al. [153] proposed TransDreamer, a Transformer-based model-based RL agent that leverages a Transformer for dynamics predictions in 2-D and 3-D visual RL tasks. Experiments showed the TransDreamer agent can outperform Dreamer with long-range memory access and memory-based reasoning. Based on the development of supervised pretraining methods, Seo et al. [154] proposed a framework that learns world model dynamics from action-free video representations. The framework added a video-based intrinsic bonus for better-guiding exploration to effectively encourage agents to learn diverse behaviors. The experimental results demonstrated their proposed method can improve the performances of vision-based RL on manipulation and locomotion tasks by transferring the pretrained representations from unseen domains.

2) Dynamics Representation Learner: Using representation learning techniques, the latent representation of the future can be learned to assist decision-making. Seo et al. [112] introduced a visual model-based RL framework that decouples visual representation learning and dynamic learning by training an autoencoder with a vision Transformer to reconstruct pixel-given masked observations and learn the dynamics from the latent space. By such a decoupling approach, their proposed method achieved state-of-the-art performance on visual robotic tasks from metaworld and RLBench. Utilizing the fact that language contains rich information signals and can help agents to predict the future, Lin et al. [45] proposed Dynalang, where an agent that learns a multimodal world model to predict future text and image representations and thereby instruct the decision-making process. The algorithm is presented in Algorithm 3. In the training part, an LLM-based world model implemented with recurrent state space model (RSSM) [155] computes the state representation z_t based on the collected transitions. After that, the representation z_t and the action a_t are fed into the RSSM to predict the future representations z_{t+1}. In addition, the language $\hat{l}_t$ and images $\hat{x}_t$ are predicted to reconstruct the original state. Then, the world model is trained to learn the next representation and reconstruct observations from the representations. After updating, the world model imagined (sampled) rollouts, and the policy is trained to maximize the imagined rewards. Compared with other works that use language only for predicting actions, multimodal information enables Dynalang to handle tasks that require grounded language generation, obtaining higher scores than methods provided with only task descriptions. To solve the out-of-distribution generalization problem in visual control tasks of RL, Poudel et al. [156] proposed the language grounded world model (LanGWM), which focuses on learning language-grounded visual features to enhance the world model learning. To improve the generalization of the learned visual features, they masked the bounding boxes and predicted them with given language descriptions. Utilizing the expressing ability of language in higher level concepts and global contexts, the proposed LanGWM method yields state-of-the-art results on out-of-distribution tests.

Algorithm 3 Training Part of Dynalang [45]

Require: Rewards r_t, episode continue flag c_t, images x_t, language tokens l_t, actions a_t, model state (h_t, z_t).

1: **while** training **do**
2: Draw batch of transitions $\{(r_t, c_t, x_t, l_t, a_t)\}$ from replay buffer.
3: Use world model to compute multimodal representations z_t, future predictions $\hat{z}_{t+1}$, and decode $\hat{x}_t$, $\hat{l}_t$, $\hat{r}_t$, $\hat{c}_t$.
4: Update world model to minimize $\mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{repr}}$.
5: Imagine rollouts from all z_t using π.
6: Update actor to minimize $\mathcal{L}_\pi$.
7: Update critic to minimize $\mathcal{L}_V$.
8: **end while**

B. Policy Interpreter

XRL is an emerging subfield of both explainable machine learning and RL that has attracted considerable attention recently. XRL aims to elucidate the decision-making process of learning agents. According to a survey in XRL [157], the categories of XRL include the feature importance, learning process and MDP, and policy level. Currently, the usage of LLMs in XRL has only been limited to the policy level, i.e., as the policy interpreter. Therefore, though there is limited literature related to this area, we think this field is important and requires more attention in the future. The following will introduce the role of LLMs as policy interpreters and an outlook regarding other categories of XRL will be provided in the last subsection.

As the policy interpreter, LLM generates explanations with the prompts of state and action description or trajectory information. An illustration is depicted in Fig. 6. Using the trajectory history of states and actions as context information, LLMs can be prompts to generate readable interpretations of current policies or situations for humans.

Das et al. [158] proposed a unified framework called State2Explanation (S2E) that learns a joint embedding model between state–action pairs and concept-based explanation. Based on the learned models, the explanation can help inform reward shaping during an agent's training and provide insights to end-users at deployment. Another work [45] first distilled the policy into a decision tree, derived the decision path, and then prompts an LLM to generate a natural language explanation based on the decision path. In addition, Lu et al. [159] introduced a framework that decomposes the overall reward into multiple subrewards based on specific object properties, defines actions as high-level motion primitives executed

at precise 3-D positions to simplify decision-making and integrates LLMs to enable interactive and flexible querying of explanations.

C. Summarization and Outlook

As a generator, LLMs can be integrated into model-based RL or XRL, i.e., serving as world model simulators or policy interpreters, respectively. As world model simulators, LLMs enhance model-based RL by autoregressively generating accurate trajectories (trajectory rollout) and by predicting latent world representations (world representation learners), significantly improving sample efficiency and decision-making accuracy. In the realm of XRL, LLMs provide valuable insights for both end-user understanding and reward shaping by generating explanations based on trajectory information. However, the current usages of LLMs in XRL are rather limited and have great potential for future work. Below are discussions on the key limitations and future work directions of the two roles.

1) *World Model Simulator*: LLMs encounter challenges in aligning their abstract knowledge with the specific requirements of different environments, leading to limitations in functional competence and grounding. This misalignment affects their effectiveness in generating trajectories and interacting with the environment. In addition, the current model-based agents, which rely on purely observational world models, present difficulties for human adaptation. These models are typically modifiable only through observational data, which is not an effective means for humans to communicate complex intentions or adjustments. Furthermore, while LLMs hold promise for enhancing multitask learning by generating trajectories and dynamic representations applicable to multiple tasks, there remains a scarcity of research exploring this potential. Looking ahead, in the short term, researchers might prioritize improving LLMs' alignment with specific environment requirements. For the long term, a promising direction would be to integrate language instructions or an adapter into the world model. This approach could lead to a more flexible and adaptive world model that can better accommodate human intentions and adjustments, as well as support more effective multitask learning through enhanced trajectory generation and dynamic representations.
2) *Policy Interpreter*: As policy interpreters, the quality of explanations depends on the LLM's understanding of feature representations and implicit logic of policy. How to utilize domain knowledge or examples to improve the understanding of a complex policy is still a major issue. In the short term, researchers could focus on enhancing LLMs' ability to interpret the correlation between observations and policy selection, providing insights into the decision-making process of RL agents. This could involve developing techniques to better leverage domain knowledge and examples for improving LLMs' understanding of complex policies. In the long-term, the field could explore advanced applications of LLMs in XRL, such as analyzing the learning process and MDP to reveal influences on agent behavior. Researchers could develop sophisticated prompting techniques for LLMs to answer nuanced "why" and "why-not" questions with MDP context, providing deeper explanations of agent decision-making.

VIII. Discussion

In Sections III–VII, we introduced the concept "LLM-enhanced RL" and developed a corresponding framework, which we extended to the integration of multimodal AI models such as visual-language models. We then discussed different LLM-enhanced RL approaches. This section provides a comprehensive analysis of the LLM-enhanced RL approach. We begin with a comparative analysis of different LLM roles, highlighting their advantages and limitations. Following this, we explore potential real-world applications of the LLM-enhanced RL paradigm. Finally, we discuss future opportunities and challenges, taking into account both the untapped capabilities and inherent limitations of LLMs, with a particular focus on their application in multimodal information environments.

A. Comparison of Different LLM-Enhanced RL Approaches

This section provides a comparative analysis of the four LLM-enhanced RL approaches, helping researchers understand the strengths and limitations of each approach.

1) *Information Processor*: As an information processor, LLM excels in handling complex, multimodal inputs, particularly in translating natural language instructions or environment information into a format more readily usable by RL agents. This role significantly enhances the agent's ability to understand and interact with complex environments. However, it faces challenges in computational efficiency and may struggle with highly specialized domain knowledge not covered in its pretraining.
2) *Reward Designer*: LLMs serving as reward designers offer a more intuitive and flexible approach to defining reward functions, especially in complex or sparse-reward environments. This role can significantly improve the alignment of RL objectives with human intentions. The main limitation lies in ensuring that generated rewards accurately reflect task-specific nuances and long-term goals, particularly in highly specialized domains.
3) *Decision-Maker*: As decision-makers, LLMs can either directly generate actions or guide action selection, leveraging their vast knowledge base to improve sample efficiency and exploration in RL. This role is particularly effective in tasks requiring complex reasoning or long-term planning. However, it may face challenges in real-time decision-making scenarios due to computational overhead and may inherit biases present in the LLM's training data.
4) *Generator*: In the generator role, LLMs can simulate complex environments for model-based RL and provide

interpretable explanations of RL policies. This capability is invaluable for improving sample efficiency and making RL more transparent and understandable. The main challenges include aligning generated simulations with real-world dynamics and ensuring the relevance and accuracy of policy explanations.

B. Applications of LLM-Enhanced RL

Based on the characteristics of LLM-enhanced RL, such as multimodal information understanding and multitask learning and generation, we believe that LLM-enhanced RL opens up a wide array of potential applications. Here, we list several applications to inspire researchers.

1) *Robotics*: RL is widely used in robots to learn how to make decisions and execute actions to achieve goals. Utilizing natural language understanding and general logical reasoning abilities, LLM-enhanced RL can: a) improve the efficiency of human–robot interaction; b) help robots understand human needs and behavioral logic; and c) enhance decision-making and planning capabilities.
2) *Autonomous Driving*: Autonomous driving uses RL to make decisions in complex, ever-changing environments that involve understanding both sensor data (visual, lidar, and radar) and contextual information (traffic laws and human behavior). LLM-enhanced RL could employ LLMs to: a) process this multimodal information and natural language instruction or b) design comprehensive rewards based on multidisciplinary metrics, such as safety, efficiency, and passenger comfort.
3) *Energy Management*: In the energy system, operators or users apply RL to efficiently manage the usage, transportation, conversion, and storage of multiple energy with high uncertainty brought by renewable resources. LLM-enhanced RL in such cases can: a) improve the RL agents' ability to handle multiobjective tasks, such as economy, safety, and low carbon, by reward function designing and b) increase the sample efficiency for the new energy system.
4) *Healthcare Recommendation*: RL is used to learn recommendations or suggestions in healthcare [160]. LLMs can utilize the domain knowledge to analyze the vast amount of patient data and medical histories: a) accelerating the learning process of RL recommendation and b) providing more accurate diagnostic and treatment recommendations in healthcare.

C. Opportunities for LLM-Enhanced RL

Although current works in LLM-enhanced RL have already shown better performance in several aspects, more unexplored areas remain to be explored and may lead to significant improvement. Below, we summarize the potential opportunities of LLM-enhanced RL from both the perspectives of RL and LLM capabilities, respectively.

1) *RL*: Existing work such as [127], [132], and [136] mainly focus on the general RL while various specialized branches of RL are still under-exploited, e.g., multiagent RL, safe RL, transfer RL, XRL, multitask RL, in-context RL and human-centric RL. In the multiagent area, compared with various works about multi-LLM-agent collaboration [161], [162], LLM-based multiagent RL remains largely unexplored [163]. A recent survey discussed some open research problems within this field [164]. LLM-enhanced strategies could be employed to facilitate communication and collaboration among RL agents. The natural language understanding capabilities of LLMs can be used to interpret and generate instructions or strategies among agents, enhancing cooperative behaviors or competition strategies; safe RL could benefit from the reasoning and predictive capabilities of LLMs to design cost functions that encourage safety criteria compliance, reducing the risks of dangerous exploration. In transfer RL, LLMs can assist in identifying similarities between tasks or environments, leveraging their vast knowledge base to facilitate knowledge transfer and thus improve learning efficiency and adaptability across different tasks; for multitask RL, LLMs enhance agents by processing diverse entity representations and aligning instructions (information processor), generating reduced action sets and expert actions for various tasks (decision-maker), and creating task-specific trajectories and dynamic representations (generator). Recently, in-context RL has emerged as a promising field by leveraging the in-context learning capability of LLMs to solve decision-making problems [165]. Given a query state and an in-context offline dataset, a trained LLM exhibits both online exploration and offline conservation while avoiding extensive trial-and-error and is sample-efficient. In addition, human-centric RL is another prominent field in healthcare and robotics, where AI and humans learn and communicate with each other for collaboration [166]. In this setting, an LLM naturally serves as a perfect mediator to facilitate bidirectional communication through natural language interactions.
2) *LLM*: While LLMs have been integrated with RL in various methods, several promising directions remain to be explored to further enhance LLM capabilities for RL. It can be divided into the language model view and the language agent view as follows.
    a) *From the model perspective*: LLM could be enhanced by an external knowledge base and continual learning. Retrieval-augmented generation (RAG) techniques help LLM to retrieve the most relevant data in an external database, which could be used to attach knowledge in the task domains [167]. Continual learning techniques are also a hot field that enables models to continuously acquire new knowledge while retaining previously learned capabilities [168]. This technique allows both LLMs and RL agents to continuously evolve and adapt to new tasks or environments while maintaining their fundamental pre-trained abilities, leading to more flexible and robust learning systems.

b) *From the Agent Perspective*: Equipping LLM-based agents with specialized modules—namely, planning modules, memory modules, and action modules—can significantly enhance the capabilities of LLMs [169], [170]. In the planning module, multistep reasoning techniques, such as CoT-based reasoning [72] and Monte Carlo tree search (MCTS) [171] can be used to enhance LLM's long-term planning ability, improving the long-term task decomposition ability of RL agents; for the memory module, long- and short-term human-memories and corresponding memory retrieval mechanisms provide a way to store previous experiences and learn adaptively along with the RL agents [172]; for the action module, tool integration presents another promising direction to unlock new possibilities for LLMs. External tools such as mathematical solvers [173] and Internet browsers [174] can augment LLMs' capabilities in RL tasks requiring complex mathematical computations and real-time information processing, respectively. Furthermore, collaboration of multiple LLM agents [175] is another promising direction. With each LLM playing different roles and acquiring different information, multiple distinctive LLM agents solve complex problems by communicating and exchanging information. In the context of LLM-enhanced RL, different LLMs could potentially serve different roles and work together to guide RL. For example, in a group of three LLM agents, one LLM focuses on guiding short-term or immediate problem-solving for RL; another LLM is responsible for long-term planning for RL; the last LLM serves as a coordinator responsible for overseeing the group and adjusting horizon consideration accordingly.

D. Challenges of LLM-Enhanced RL

While LLM improves different issues in the RL paradigm, the success of LLM-enhanced RL is inherently tied to the capabilities and limitations of the underlying LLM. The primary concerns revolve around the inherent limitations of LLMs, the adaptability of LLMs in RL environments, computational demands, and the broader implications of deploying such systems in real-world scenarios.

1) *Inherent Limitations of LLMs*: The effectiveness of LLM-enhanced RL systems is fundamentally constrained by the inherent capabilities and limitations of the underlying language models. The presence of systematic biases and potential hallucinations in pretrained LLMs can significantly impact the reliability of multimodal input interpretation, potentially compromising the overall performance of the RL agent. This necessitates the development of robust evaluation frameworks to systematically characterize and delineate the capability boundaries of LLMs given specified RL contexts. Furthermore, incorporating uncertainty quantification methods aids in identifying unreliable answers, increasing the trustworthiness of LLMs' responses [176].
2) *Adaptability of LLMs in RL Environments*: Despite the vast knowledge base of LLMs, they may struggle to adapt to specific or novel RL task environments that are not well-represented in their training data. This calls for methods to expand the task-related knowledge for LLM and ground LLMs' inherent knowledge in specific domains. In terms of expanding the task-related knowledge, RAG-related techniques attach LLMs with the external domain-knowledge without further training [167]. When fine-tuning LLMs, employing data augmentation techniques such as generating synthetic data is another way to expand the diversity of training scenarios and improve generalization to novel environments [177]. In addition, continual learning mechanisms help preserve previously acquired knowledge while adapting to new information [178]. For domain-specific knowledge grounding, approaches include training value functions to evaluate the long-term utility of LLM instructions [81]. Expert trajectories provide another valuable source of information, allowing LLMs to learn optimal decision-making patterns through in-context learning.
3) *Computational Demands*: The integration of LLMs into the RL learning process introduces complexities in terms of computational overhead and inference times, which slow down the learning process of RL. To address this challenge, several potential solutions have been proposed across different levels. At the data level, input compression techniques such as prompt pruning [179] can be employed to directly shorten the model input, significantly reducing inference time without substantial loss in performance. At the model level, efficient architecture designs like mixture-of-experts (MoE) [180] enable conditional computation, activating only relevant parts of the model for each input, thus reducing computational costs. Similarly, structured state-space models (SSMs) [181] offer linear-time complexity for sequence modeling, providing a more efficient alternative to traditional attention mechanisms. At the system level, advanced caching strategies [182] and asynchronous processing techniques [183] can be implemented to reuse intermediate computations and parallelize the execution, respectively.
4) *Ethical, Legal, and Safety Concerns*: In practical usages, the use of LLMs involves complex ethical, legal, and safety concerns. Data privacy, intellectual property, and accountability for AI decisions should be carefully discussed. To address these issues, researchers are developing robust frameworks for responsible AI deployment. At the privacy level, differential privacy techniques [184] are being implemented to protect individual data during training and inference. For transparency, efforts focus on developing interpretable AI systems, allowing stakeholders to audit AI-driven decisions [185]. To enhance system robustness, adversarial training methods are being explored to strengthen LLM-RL systems against

potential attacks [186]. Regarding ethical and legal issues, a recent survey also analyzed potential solutions to integrate ethical standards and societal values into LLM [187].

IX. Conclusion

LLMs, with their pre-trained knowledge bases and powerful capabilities such as reasoning and in-context learning, present as a viable solution to enhance RL in terms of natural language understanding, multitask generalization, task planning, and sample efficiency. In this survey, we have defined this paradigm as *LLM-enhanced RL* and summarized its characteristics, together with opportunities and challenges. To formalize the research scope and methodology of LLM-enhanced RL, we propose a structured framework to systematically categorize the roles of LLM based on the functionalities within the classical agent–environment interaction paradigm. According to functionalities, we categorize the roles of LLMs into information processor, reward designer, decision-maker, and generator. For each category, we review current literature based on their methods and applications, outline the methodologies, discuss the addressed issues, and provide insights into future directions. These are listed as follows.

1) *Information Processor*: LLMs extract observational representations and formal specification languages for RL agents, thus increasing the sample efficiency. Future directions include incorporating goal-based information and integrating multimodal environmental information to obtain stronger information extraction ability.
2) *Reward Designer*: For intricate or unquantifiable tasks, LLMs can leverage pretrained knowledge to generate high-performing rewards that are notoriously difficult for humans to design. Current methods still require consistently modifying the prompts and instructions of LLMs, calling for future work on automated self-evolving frameworks without human intervention.
3) *Decision-Maker*: LLMs generate direct actions or indirect advice for the agent to improve exploration efficiency. Huge computational overhead is a major issue in online interaction. Cost-effective methods to reduce the huge computational overhead of LLM in online RL is an important direction.
4) *Generator*: With the generative capability and world knowledge, LLMs are used as: 1) a high-fidelity world model to reduce real-world learning cost and 2) a language-based policy interpreter to explain the agent policy. Leveraging human instructions to improve the accuracy and generalizability of the world model and policy interpreter would be a crucial direction.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[2] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[4] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[5] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[6] C. Berner et al., "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.

[7] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[8] J. Schrittwieser et al., "Mastering Atari, go, chess and shogi by planning with a learned model," 2019, *arXiv:1911.08265*.

[9] H. Zhao et al., "Mobile battery energy storage system control with knowledge-assisted deep reinforcement learning," *Energy Conver. Econ.*, vol. 3, no. 6, pp. 381–391, Dec. 2022.

[10] N. B. Schmid et al., "Rebel: A general game playing AI," *Science*, vol. 373, no. 6556, pp. 664–670, 2021.

[11] N. Brown and T. Sandholm, "Superhuman AI for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, Aug. 2019.

[12] A. Vaswani, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–12.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.

[14] Y. Jiang, S. S. Gu, K. P. Murphy, and C. Finn, "Language as an abstraction for hierarchical deep reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–13.

[15] P. Anderson et al., "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proc. CVPR*, Jun. 2018, pp. 3674–3683.

[16] A. Stooke, K. Lee, P. Abbeel, and M. Laskin, "Decoupling representation learning from reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 9870–9879.

[17] J. Luketina et al., "A survey of reinforcement learning informed by natural language," 2019, *arXiv:1906.03926*.

[18] A. Mandlekar et al., "What matters in learning from offline human demonstrations for robot manipulation," 2021, *arXiv:2108.03298*.

[19] C. Lynch et al., "Interactive language: Talking to robots in real time," *IEEE Robot. Autom. Lett.*, early access, Jul. 13, 2023, doi: 10.1109/LRA.2023.3295255.

[20] Z. Yang et al., "Towards applicable reinforcement learning: Improving the generalization and sample efficiency with policy ensemble," in *Proc. Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 3659–3665. [Online]. Available: https://api.semanticscholar.org/CorpusID:248887230

[21] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (Mis)design for autonomous driving," *Artif. Intell.*, vol. 316, Mar. 2023, Art. no. 103829.

[22] F. Dworschak, S. Dietze, M. Wittmann, B. Schleich, and S. Wartzack, "Reinforcement learning for engineering design automation," *Adv. Eng. Informat.*, vol. 52, Apr. 2022, Art. no. 101612.

[23] S. Booth, W. B. Knox, J. Shah, S. Niekum, P. Stone, and A. Allievi, "The perils of trial-and-error reward design: Misdesign through overfitting and invalid task specifications," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2023, vol. 37, no. 5, pp. 5920–5929.

[24] L. L. Di Langosco, J. Koch, L. D. Sharkey, J. Pfau, and D. Krueger, "Goal misgeneralization in deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 12004–12019.

[25] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[27] A. Radford and K. Narasimhan. (2018). *Improving Language Understanding By Generative Pre-Training*. [Online]. Available: https://api.semanticscholar.org/CorpusID

[28] T. B. Brown et al., "Language models are few-shot learners," in *Proc. NeurIPS*, vol. 33, 2020, pp. 1877–1901.

[29] A. Chowdhery et al., "PaLM: Scaling language modeling with pathways," *J. Mach. Learn. Res.*, vol. 24, no. 240, pp. 1–113, 2023.

[30] A. J. Thirunavukarasu et al., "Large language models in medicine," *Nature Med.*, vol. 29, no. 8, pp. 1930–1940, 2023.

[31] D. A. Boiko, R. MacKnight, B. Kline, and G. Gomes, "Autonomous chemical research with large language models," *Nature*, vol. 624, no. 7992, pp. 570–578, Dec. 2023.

[32] G. Jiang, Z. Ma, L. Zhang, and J. Chen, "EPlus-LLM: A large language model-based computing platform for automated building energy modeling," *Appl. Energy*, vol. 367, Aug. 2024, Art. no. 123431.

[33] H. Tan et al., "General generative AI-based image augmentation method for robust rooftop PV segmentation," *Appl. Energy*, vol. 368, Aug. 2024, Art. no. 123554.

[34] X. Zhou et al., "ElecBench: A power dispatch evaluation benchmark for large language models," 2024, *arXiv:2407.05365*.

[35] J. Liang et al., "Code as policies: Language model programs for embodied control," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2023, pp. 9493–9500.

[36] T. Webb, K. J. Holyoak, and H. Lu, "Emergent analogical reasoning in large language models," *Nature Human Behaviour*, vol. 7, no. 9, pp. 1526–1541, Jul. 2023.

[37] J. Wei et al., "Larger language models do in-context learning differently," 2023, *arXiv:2303.03846*.

[38] J. Huang and K. Chen-Chuan Chang, "Towards reasoning in large language models: A survey," 2022, *arXiv:2212.10403*.

[39] J. Yu et al., "KoLA: Carefully benchmarking world knowledge of large language models," 2023, *arXiv:2306.09296*.

[40] I. Singh et al., "ProgPrompt: Program generation for situated robot task planning using large language models," *Auto. Robots*, vol. 47, no. 8, pp. 999–1012, Dec. 2023.

[41] N. Stiennon et al., "Learning to summarize from human feedback," 2022, *arXiv:2009.01325*.

[42] E. Akyürek et al., "What learning algorithm is in-context learning? Investigations with linear models," 2023, *arXiv:2211.15661*.

[43] Y. Du et al., "Guiding pretraining in reinforcement learning with large language models," 2023, *arXiv:2302.06692*.

[44] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer, "Grounding large language models in interactive environments with online reinforcement learning," 2023, *arXiv:2302.02662*.

[45] J. Lin et al., "Learning to model the world with language," 2023, *arXiv:2308.01399*.

[46] H. Li et al., "Auto MC-reward: Automated dense reward design with large language models for minecraft," 2023, *arXiv:2312.09238*.

[47] S. Chakraborty et al., "RE-MOVE: An adaptive policy design for robotic navigation tasks in dynamic environments via language-based feedback," 2023, *arXiv:2303.07622*.

[48] J.-C. Pang, X.-Y. Yang, S.-H. Yang, and Y. Yu, "Natural language-conditioned reinforcement learning with inside-out task language development and translation," 2023, *arXiv:2302.09368*.

[49] D. Kalashnikov et al., "QT-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," 2018, *arXiv:1806.10293*.

[50] K. Wang, B. Kang, J. Shao, and J. Feng, "Improving generalization in reinforcement learning with mixture regularization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 7968–7978.

[51] R. Devidze, P. Kamalaruban, and A. Singla, "Exploration-guided reward shaping for reinforcement learning under sparse rewards," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 5829–5842.

[52] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," 2019, *arXiv:1904.07854*.

[53] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, and A. Dragan, "Inverse reward design," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–9.

[54] C. Xiao, Y. Wu, C. Ma, D. Schuurmans, and M. Müller, "Learning to combat compounding-error in model-based reinforcement learning," 2019, *arXiv:1912.11206*.

[55] T. M. Moerland et al., "Model-based reinforcement learning: A survey," *Found. Trends Mach. Learn.*, vol. 16, no. 1, pp. 1–118, Jan. 2023.

[56] M. Cho, J. Park, S. Lee, and Y. Sung, "Hard tasks first: Multi-task reinforcement learning through task scheduling," in *Proc. 41st Int. Conf. Mach. Learn.*, 2024, pp. 1–22. [Online]. Available: https://openreview.net/forum?id=haUOhXo70o

[57] J. Feng, M. Chen, Z. Pu, T. Qiu, and J. Yi, "Efficient multi-task reinforcement learning via task-specific action correction," 2024, *arXiv:2404.05950*.

[58] L. Sun, H. Zhang, W. Xu, and M. Tomizuka, "PaCo: Parameter-compositional multi-task reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 21495–21507.

[59] G. Zhang, A. Jain, I. Hwang, S.-H. Sun, and J. J. Lim, "Efficient multi-task reinforcement learning via selective behavior sharing," in *Proc. ICLR*, 2024, pp. 1–14. [Online]. Available: https://openreview.net/forum?id=LYGHdwyXUb

[60] N. V. Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics*, vol. 9, no. 9, p. 1363, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/9/1363

[61] T. Xiao et al., "Robotic skill acquisition via instruction augmentation with vision-language models," 2023, *arXiv:2211.11736*.

[62] H. Yuan et al., "Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks," 2023, *arXiv:2303.16563*.

[63] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, and D. Lindner, "Vision-language models are zero-shot reward models for reinforcement learning," 2023, *arXiv:2310.12921*.

[64] M. Shanahan, "Talking about large language models," 2022, *arXiv:2212.03551*.

[65] H. Touvron et al., "LLaMA: Open and efficient foundation language models," 2023, *arXiv:2302.13971*.

[66] J. Kaplan et al., "Scaling laws for neural language models," 2020, *arXiv:2001.08361*.

[67] J. Hoffmann et al., "Training compute-optimal large language models," 2022, *arXiv:2203.15556*.

[68] J. Wei et al., "Emergent abilities of large language models," *Trans. Mach. Learn. Res.*, vol. 2022, Jun. 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:249674500

[69] V. Sanh et al., "Multitask prompted training enables zero-shot task generalization," 2021, *arXiv:2110.08207*.

[70] L. Ouyang et al., "Training language models to follow instructions with human feedback," 2022, *arXiv:2203.02155*.

[71] Y. Cheng, H. Zhao, X. Zhou, J. Zhao, Y. Cao, and C. Yang, "GAIA—A large language model for advanced power dispatch," 2024, *arXiv:2408.03847*.

[72] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," 2023, *arXiv:2201.11903*.

[73] S. Yao et al., "Tree of thoughts: Deliberate problem solving with large language models," 2023, *arXiv:2305.10601*.

[74] M. Besta et al., "Graph of thoughts: Solving elaborate problems with large language models," in *Proc. AAAI Conf. Artif. Intell.*, Mar. 2024, vol. 38, no. 16, pp. 17682–17690.

[75] Z. Rao et al., "Visual navigation with multiple goals based on deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5445–5455, Dec. 2021.

[76] C. Huang et al., "Deductive reinforcement learning for visual autonomous urban driving navigation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5379–5391, Dec. 2021.

[77] M. Yang, W. Huang, W. Tu, Q. Qu, Y. Shen, and K. Lei, "Multitask learning and reinforcement learning for personalized dialog generation: An empirical study," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 49–62, Jan. 2021.

[78] Z. He, J. Li, F. Wu, H. Shi, and K.-S. Hwang, "DeRL: Coupling decomposition in action space for reinforcement learning task," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 8, no. 1, pp. 1030–1043, Feb. 2024.

[79] Y. Liu et al., "A survey of visual transformers," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–21, 2023.

[80] Y. J. Ma et al., "Eureka: human-level reward design via coding large language models," Apr. 2023, *arXiv:2310.12931*.

[81] M. Ahn et al., "Do as i can, not as i say: Grounding language in robotic affordances," 2022, *arXiv:2204.01691*.

[82] K. Nottingham et al., "Do embodied agents dream of pixelated sheep: Embodied decision making using language guided world modelling," 2023, *arXiv:2301.12050*.

[83] A. Srinivas, M. Laskin, and P. Abbeel, "CURL: Contrastive unsupervised representations for reinforcement learning," 2020, *arXiv:2004.04136*.

[84] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman, "Data-efficient reinforcement learning with self-predictive representations," 2020, *arXiv:2007.05929*.

[85] F. Paischer et al., "History compression via language models in reinforcement learning," 2023, *arXiv:2205.12258*.

[86] F. Paischer, T. Adler, M. Hofmarcher, and S. Hochreiter, "Semantic helm: A human-readable memory for reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2024, pp. 1–12.

[87] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. ICML*, 2021, pp. 8748–8763.

[88] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," 2019, *arXiv:1901.02860*.

[89] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.

[90] W. K. Kim et al., "Efficient policy adaptation with contrastive prompt ensemble for embodied agents," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2024, pp. 1–12.

[91] R. P. K. Poudel, H. Pandya, S. Liwicki, and R. Cipolla, "ReCoRe: Regularized contrastive representation learning of world model," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 22904–22913.

[92] S. Basavatia, K. Murugesan, and S. Ratnakar, "STARLING: Self-supervised training of text-based reinforcement learning agent with large language models," 2024, *arXiv:2406.05872*.

[93] R. Patel, E. Pavlick, and S. Tellex, "Grounding language to non-markovian tasks with no supervision of task specifications," in *Proc. Robot., Sci. Syst.*, 2020, pp. 1–12.

[94] T. R. Sumers, M. K. Ho, R. D. Hawkins, K. Narasimhan, and T. L. Griffiths, "Learning rewards from linguistic feedback," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 7, pp. 6002–6010.

[95] J. Liang et al., "Code as policies: Language model programs for embodied control," 2023, *arXiv:2209.07753*.

[96] C. H. Song et al., "LLM-planner: Few-shot grounded planning for embodied agents with large language models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 2998–3009.

[97] B. A. Spiegel et al., "Informing reinforcement learning agents by grounding language to Markov decision processes," in *Proc. Workshop Training Agents Found. Models RLC*, 2024, pp. 1–33. [Online]. Available: https://openreview.net/forum?id=uFm9e4Ly26

[98] B. Gordon et al., "Mismatch quest: Visual and textual feedback for image-text misalignment," in *Proc. Eur. Conf. Comput. Vis.* London, U.K.: Springer, 2025, pp. 310–328.

[99] Y. Wang, J. He, D. Wang, Q. Wang, B. Wan, and X. Luo, "Multimodal transformer with adaptive modality weighting for multimodal sentiment analysis," *Neurocomputing*, vol. 572, Mar. 2024, Art. no. 127181.

[100] X. Zhao, S. Poria, X. Li, Y. Chen, and B. Tang, "Toward robust multimodal learning using multimodal foundational models," 2024, *arXiv:2401.13697*.

[101] F. Lygerakis, V. Dave, and E. Rueckert, "M2CURL: Sample-efficient multimodal reinforcement learning via self-supervised representation learning for robotic manipulation," 2024, *arXiv:2401.17032*.

[102] J. Eschmann, "Reward function design in reinforcement learning," in *Proc. Reinforcement Learn. Algorithms, Anal. Appl.*, 2021, pp. 25–33.

[103] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 166–175.

[104] S. Mirchandani, S. Karamcheti, and D. Sadigh, "ELLA: Exploration through learned language abstraction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 29529–29540.

[105] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh, "Reward design with language models," in *Proc. 11th Int. Conf. Learning Represent.*, Sep. 2022, pp. 1–12.

[106] Y. Wu et al., "Read and reap the rewards: Learning to play atari with the help of instruction manuals," 2023, *arXiv:2302.04449*.

[107] K. Chu, X. Zhao, C. Weber, M. Li, and S. Wermter, "Accelerating reinforcement learning of robotic manipulations via feedback from large language models," 2023, *arXiv:2311.02379*.

[108] A. Adeniji, A. Xie, C. Sferrazza, Y. Seo, S. James, and P. Abbeel, "Language reward modulation for pretraining reinforcement learning," 2023, *arXiv:2308.12270*.

[109] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," 2019, *arXiv:1910.01108*.

[110] C. Kim et al., "Guide your agent with adaptive multimodal rewards," 2023, *arXiv:2309.10790*.

[111] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3M: A universal visual representation for robot manipulation," 2022, *arXiv:2203.12601*.

[112] Y. Seo et al., "Masked world models for visual control," 2023, *arXiv:2206.14244*.

[113] K. Grauman et al., "Ego4D: Around the world in 3,000 hours of egocentric video," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 18995–19012.

[114] Y. Wang et al., "RL-VLM-F: Reinforcement learning from vision language foundation model feedback," 2024, *arXiv:2402.03681*.

[115] W. Yu et al., "Language to rewards for robotic skill synthesis," 2023, *arXiv:2306.08647*.

[116] A. Madaan et al., "Self-refine: Iterative refinement with self-feedback," 2023, *arXiv:2303.17651*.

[117] J. Song, Z. Zhou, J. Liu, C. Fang, Z. Shu, and L. Ma, "Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics," 2023, *arXiv:2309.06687*.

[118] T. Xie et al., "Text2reward: Reward shaping with language models for reinforcement learning," in *Proc. 12th Int. Conf. Learn. Representations*, 2024, pp. 1–37.

[119] E. Ferrara, "Should ChatGPT be biased? Challenges and risks of bias in large language models," 2023, *arXiv:2304.03738*.

[120] I. O. Gallegos et al., "Bias and fairness in large language models: A survey," *Comput. Linguistics*, vol. 50, no. 3, pp. 1097–1179, Sep. 2024.

[121] L. Gao, J. Schulman, and J. Hilton, "Scaling laws for reward model overoptimization," in *Proc. 40th Int. Conf. Mach. Learn.*, vol. 202, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., Jun. 2023, pp. 10835–10866.

[122] S. Chakraborty, A. Singh, A. Bhaskar, P. Tokekar, D. Manocha, and A. Singh Bedi, "REBEL: A regularization-based solution for reward overoptimization in robotic reinforcement learning from human feedback," 2023, *arXiv:2312.14436*.

[123] A. Radwan, L. Zaafarani, J. Abudawood, F. AlZahrani, and F. Fourati, "Addressing bias through ensemble learning and regularized fine-tuning," 2024, *arXiv:2402.00910*.

[124] Y. Inoue and H. Ohashi, "Prompter: Utilizing large language model prompting for a data efficient embodied instruction following," 2022, *arXiv:2211.03267*.

[125] A. Majumdar, A. Shrivastava, S. Lee, P. Anderson, D. Parikh, and D. Batra, "Improving vision-and-language navigation with image-text pairs from the web," in *Proc. Eur. Conf. Comput. Vis.*, Glasgow, U.K. Cham, Switzerland: Springer, 2020, pp. 259–274.

[126] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 1273–1286.

[127] S. Li et al., "Pre-trained language models for interactive decision-making," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 31199–31212.

[128] R. Shi, Y. Liu, Y. Ze, S. S. Du, and H. Xu, "Unleashing the power of pre-trained language models for offline reinforcement learning," 2023, *arXiv:2310.20587*.

[129] Z. Zeng, C. Zhang, S. Wang, and C. Sun, "Goal-conditioned predictive coding for offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2024, pp. 1–21.

[130] M. Reid, Y. Yamada, and S. S. Gu, "Can Wikipedia help offline reinforcement learning?" 2022, *arXiv:2201.12122*.

[131] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4RL: Datasets for deep data-driven reinforcement learning," 2020, *arXiv:2004.07219*.

[132] L. Mezghani, P. Bojanowski, K. Alahari, and S. Sukhbaatar, "Think before you act: Unified policy for interleaving language reasoning with actions," 2023, *arXiv:2304.11063*.

[133] B. Zitkovich et al., "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Proc. Conf. Robot Learn.*, 2023, pp. 2165–2183.

[134] S. Yao, R. Rao, M. Hausknecht, and K. Narasimhan, "Keep CALM and explore: Language models for action generation in text-based games," 2020, *arXiv:2010.02903*.

[135] M. Hausknecht, P. Ammanabrolu, M.-A. Côté, and X. Yuan, "Interactive fiction games: A colossal adventure," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 7903–7910.

[136] H. Hu and D. Sadigh, "Language instructed reinforcement learning for human-AI coordination," 2023, *arXiv:2304.07297*.

[137] Z. Zhou, B. Hu, C. Zhao, P. Zhang, and B. Liu, "Large language model as a policy teacher for training reinforcement learning agents," 2023, *arXiv:2311.13373*.

[138] M. Dalal, T. Chiruvolu, D. S. Chaplot, and R. Salakhutdinov, "Plan-seq-learn: Language model guided RL for solving long horizon robotics tasks," in *Proc. 12th Int. Conf. Learn. Represent.*, 2024, pp. 1–29. [Online]. Available: https://openreview.net/forum?id=hQVCCxQrYN

[139] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, 2015.

[140] H. Xu, X. Zhan, and X. Zhu, "Constraints penalized Q-learning for safe offline reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 8, pp. 8753–8760.

[141] J. Lee et al., "COptiDICE: Offline constrained reinforcement learning via stationary distribution correction estimation," 2022, *arXiv:2204.08957*.

[142] Z. Liu et al., "Constrained decision transformer for offline safe reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 21611–21630.

[143] S. Naihin et al., "Testing language model agents safely in the wild," 2023, *arXiv:2311.10538*.

[144] W. Huang, H. Liu, Z. Huang, and C. Lv, "Safety-aware human-in-the-loop reinforcement learning with shared control for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 11, pp. 16181–16192, Nov. 2024.

[145] N. Shinn et al., "Reflexion: Language agents with verbal reinforcement learning," 2023, *arXiv:2303.11366*.

[146] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," 2021, *arXiv:2106.09685*.

[147] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," 2020, *arXiv:1912.01603*.

[148] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, "Mastering atari with discrete world models," 2022, *arXiv:2010.02193*.

[149] Y. Matsuo, Y. LeCun, M. Sahani, D. Precup, D. Silver, M. Sugiyama, E. Uchibe, and J. Morimoto, "Deep learning, reinforcement learning, and world models," *Neural Netw.*, vol. 152, pp. 267–275, Aug. 2022.

[150] L. Chen et al., "Decision transformer: Reinforcement learning via sequence modeling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 15084–15097.

[151] V. Micheli, E. Alonso, and F. Fleuret, "Transformers are sample-efficient world models," in *Proc. 11th Int. Conf. Learn. Represent.*, Sep. 2022, pp. 1–21.

[152] J. Robine, M. Höftmann, T. Uelwer, and S. Harmeling, "Transformer-based world models are happy with 100k interactions," 2023, *arXiv:2303.07109*.

[153] C. Chen, Y.-F. Wu, J. Yoon, and S. Ahn, "TransDreamer: Reinforcement learning with transformer world models," 2022, *arXiv:2202.09481*.

[154] Y. Seo, K. Lee, S. James, and P. Abbeel, "Reinforcement learning with action-free pre-training from videos," 2022, *arXiv:2203.13880*.

[155] D. Hafner et al., "Learning latent dynamics for planning from pixels," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2555–2565.

[156] R. P. K. Poudel, H. Pandya, C. Zhang, and R. Cipolla, "LanGWM: Language grounded world model," 2023, *arXiv:2311.17593*.

[157] S. Milani, N. Topin, M. Veloso, and F. Fang, "A survey of explainable reinforcement learning," 2022, *arXiv:2202.08434*.

[158] D. Das, S. Chernova, and B. Kim, "State2explanation: Concept-based explanations to benefit agent learning and user understanding," in *Proc. 37th Conf. Neural Inf. Process. Syst.*, 2023, pp. 1–27. [Online]. Available: https://openreview.net/forum?id=xGz0wAIJrS

[159] W. Lu, X. Zhao, S. Magg, M. Gromniak, M. Li, and S. Wermter, "A closer look at reward decomposition for high-level robotic explanations," in *Proc. IEEE Int. Conf. Develop. Learn. (ICDL)*, Nov. 2023, pp. 429–436.

[160] C. Yu, J. Liu, S. Nemati, and G. Yin, "Reinforcement learning in healthcare: A survey," *ACM Comput. Surveys*, vol. 55, no. 1, pp. 1–36, Jan. 2023.

[161] S. Agashe, Y. Fan, A. Reyna, and X. E. Wang, "LLM-coordination: Evaluating and analyzing multi-agent coordination abilities in large language models," 2023, *arXiv:2310.03903*.

[162] S. S. Kannan, V. L. N. Venkatesh, and B.-C. Min, "SMART-LLM: Smart multi-agent robot task planning using large language models," 2023, *arXiv:2309.10062*.

[163] O. Slumbers, D. H. Mguni, K. Shao, and J. Wang, "Leveraging large language models for optimised coordination in textual multi-agent reinforcement learning," in *Proc. ICLR*, 2023, pp. 1–14.

[164] C. Sun, S. Huang, and D. Pompili, "LLM-based multi-agent reinforcement learning: Current and future directions," 2024, *arXiv:2405.11106*.

[165] J. Lee et al., "Supervised pretraining can learn in-context reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2024, pp. 1–27.

[166] Z. Buçinca, S. Swaroop, A. E. Paluch, S. A. Murphy, and K. Z. Gajos, "Towards optimizing human-centric objectives in AI-assisted decision-making with offline reinforcement learning," 2024, *arXiv:2403.05911*.

[167] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," 2020, *arXiv:2005.11401*.

[168] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 8, pp. 5362–5383, Aug. 2024.

[169] L. Wang et al., "A survey on large language model based autonomous agents," *Frontiers Comput. Sci.*, vol. 18, no. 6, Dec. 2024, Art. no. 186345.

[170] Y. Cheng et al., "Exploring large language model based intelligent agents: Definitions, methods, and prospects," 2024, *arXiv:2401.03428*.

[171] C. B. Browne et al., "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[172] Z. Zhang et al., "A survey on the memory mechanism of large language model based agents," 2024, *arXiv:2404.13501*.

[173] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large language model connected with massive Apis," 2023, *arXiv:2305.15334*.

[174] R. Nakano et al., "WebGPT: Browser-assisted question-answering with human feedback," 2021, *arXiv:2112.09332*.

[175] Y. Talebirad and A. Nadiri, "Multi-agent collaboration: Harnessing the power of intelligent LLM agents," 2023, *arXiv:2306.03314*.

[176] Z. Lin, S. Trivedi, and J. Sun, "Generating with confidence: Uncertainty quantification for black-box large language models," 2023, *arXiv:2305.19187*.

[177] R. S. Y. C. Tan et al., "Inferring cancer disease response from radiology reports using large language models with data augmentation and prompting," *J. Amer. Med. Inform. Assoc.*, vol. 30, no. 10, pp. 1657–1664, Sep. 2023.

[178] Q. Gao et al., "A unified continual learning framework with general parameter-efficient tuning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 11483–11493.

[179] W. Zhou, Y. E. Jiang, R. Cotterell, and M. Sachan, "Efficient prompting via dynamic in-context learning," 2023, *arXiv:2305.11170*.

[180] Z.-F. Gao, P. Liu, W. Xin Zhao, Z.-Y. Lu, and J.-R. Wen, "Parameter-efficient mixture-of-experts architecture for pre-trained language models," 2022, *arXiv:2203.01104*.

[181] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," 2024, *arXiv:2312.00752*.

[182] L. Del Corro, A. Del Giorno, S. Agarwal, B. Yu, A. Awadallah, and S. Mukherjee, "SkipDecode: Autoregressive skip decoding with batching and caching for efficient LLM inference," 2023, *arXiv:2307.02628*.

[183] Y. Chen, Z.-h. Ding, Z. Wang, Y. Wang, L. Zhang, and S. Liu, "Asynchronous large language model enhanced planner for autonomous driving," 2024, *arXiv:2406.14556*.

[184] Z. Charles et al., "Fine-tuning large language models with user-level differential privacy," 2024, *arXiv:2407.07737*.

[185] E. Cambria, L. Malandri, F. Mercorio, N. Nobani, and A. Seveso, "XAI meets LLMs: A survey of the relation between explainable AI and large language models," 2024, *arXiv:2407.15248*.

[186] S. Xhonneux, A. Sordoni, S. Günnemann, G. Gidel, and L. Schwinn, "Efficient adversarial training in LLMs with continuous attacks," 2024, *arXiv:2405.15589*.

[187] C. Deng et al., "Deconstructing the ethics of large language models from long-standing issues to new-emerging dilemmas: A survey," 2024, *arXiv:2406.05392*.