Dynamic Prompt Compression for Efficient Inference of Large Language Models

Jinwu Hu, *Graduate Student Member, IEEE*, Wei Zhang, Yufeng Wang, Yu Hu, *Member, IEEE*, Bin Xiao, Mingkui Tan, *Member, IEEE*, and Qing Du

***Abstract*—Large language models (LLMs) have shown outstanding performance across a variety of tasks, partly due to advanced prompting techniques. However, these techniques often require lengthy prompts, which increase computational costs and can hinder performance because of the limited context windows of LLMs. While prompt compression is a straightforward solution, existing methods confront the challenges of retaining essential information, adapting to context changes, and remaining effective across different tasks. To tackle these issues, we propose a task-agnostic method called Dynamic Prompt Compression (LLM-DPC). Our method reduces the number of prompt tokens while minimizing any degradation in LLM performance. We model prompt compression as a Markov Decision Process (MDP), enabling the DPC-Agent to sequentially remove redundant tokens by adapting to dynamic contexts and retaining crucial content. We develop a reward function for training the DPC-Agent that balances the compression ratio, the quality of the LLM output, and the retention of key information. This allows for prompt token reduction without needing an external black-box LLM. Inspired by the progressive difficulty adjustment in curriculum learning, we introduce a Hierarchical Prompt Compression (HPC) training strategy that gradually increases the compression difficulty, enabling the DPC-Agent to learn an effective compression method that maintains information integrity. Experiments demonstrate that our method outperforms state-of-the-art techniques, especially at higher compression ratio.**

***Index Terms*—Large language models (LLMs), prompt compression, Markov decision process (MDP), curriculum learning.**

Received 3 November 2024; revised 1 February 2026; accepted 14 February 2026. Date of publication 18 March 2026; date of current version 9 April 2026. This work was supported by the Joint Funds of the National Natural Science Foundation of China under Grant U24A20327. Recommended for acceptance by L. Nie. *(Jinwu Hu and Wei Zhang contributed equally to this work.) (Corresponding authors: Mingkui Tan; Qing Du.)*

Jinwu Hu and Wei Zhang are with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China, and also with Pazhou Lab, Guangzhou 510335, China (e-mail: fhujinwu@gmail.com; zw2177738821@gmail.com).

Yufeng Wang is with the School of Future Technology, South China University of Technology, Guangzhou 510006, China, and also with Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: 202310193334@mail.scut.edu.cn).

Yu Hu is with the Department of Health Technology and Informatics, Hong Kong Polytechnic University, Hong Kong 999077, China (e-mail: jasonscut@outlook.com).

Bin Xiao is with the Department of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: xiaobin@cqupt.edu.cn).

Mingkui Tan and Qing Du are with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: mingkuitan@scut.edu.cn; duqing@scut.edu.cn).

The code for our approach will be available at https://github.com/Fhujinwu/DPC.

Digital Object Identifier 10.1109/TKDE.2026.3669558

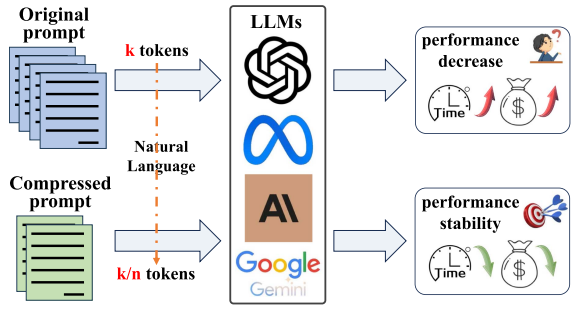

Fig. 1. Motivation for prompt compression of LLMs.

I. Introduction

LARGE language models (LLMs) [1], [2], [3], [4], [5], [6] have shown excellent performance in different tasks, including recommender systems [7], long writing [8], and drug design [9]. Many recently emerged prompting techniques for LLMs, such as Chain of Thought (CoT) [10], Retrieval Augmented Generation (RAG) [11], Role-playing [12], *etc.*, empower LLMs to handle complex and diverse tasks. However, these techniques increase the number of tokens required for the prompt, leading to additional computational and financial overhead, as well as reduced perceptual ability due to the limited context window of LLMs [13] (see Fig. 1). While model quantization and expanding the context window can partially mitigate this issue, they do not fundamentally address the cost and performance limitations caused by long prompts. Consequently, prompt compression provides a straightforward solution aimed at shortening the original prompt while preserving key information and improving the inference efficiency.

Unfortunately, prompt compression presents several challenges, partly for the following reasons. *1) Context sensitivity:* LLMs heavily rely on long prompts for context. Shortening prompts can negatively impact the ability of LLMs to generate coherent and accurate responses, requiring sophisticated compression techniques. *2) Information retention:* Compressing prompts while preserving essential information is difficult. Key details can be lost during compression, leading to degraded performance in LLM outputs. *3) Task-agnostic compression:* Developing a compression method adaptable across tasks, without being customized for specific scenarios, is particularly challenging due to the diverse nature of LLM applications.

To improve prompting efficiency, various prompt compression methods [13], [14], [15], [16], [17], [18], [19], [20], [21]

have been explored, and these can be broadly classified into white-box and black-box methods. The white-box compression methods [14], [15], [16], [17] compress the prompt at the token-embedding level by modifying the model parameters, model architecture, and self-attention mechanism. However, most high-performing LLMs (such as GPT-4 and Claude-3) are only accessible via application programming interfaces (APIs), and the lack of source code access severely limits the applicability of these approaches. In response, black-box compression methods have emerged [13], [18], [19], [20], [21], leveraging the inherent redundancy of natural language [22]. The black-box compression methods operate at the natural language level, aiming to shorten the original prompt without losing essential information. These methods do not require source code access for training or inference, thereby reducing usage costs by directly minimizing input size. They also shorten inference time while preserving LLM performance.

Despite recent advances in black-box compression methods that reduce the number of tokens in prompts while largely maintaining LLM performance, these methods still face certain limitations. **Firstly**, a common drawback of some existing task-aware compression methods [18], [20], [23], [24] is that they are usually fine-tuned for specific tasks, and thus often difficult to use for different downstream tasks. For example, LongLLMLingua [20] has to dynamically adjust the compression content according to the question, which may be difficult to use in summary tasks. **Secondly**, most task-agnostic methods [13], [19], [21] estimate token importance using information entropy from causal language models, overlooking the sequential nature of prompt compression, where each token significance depends on the evolving context. **Thirdly**, many methods heavily rely on black-box LLMs during training, either for providing reward signals [18], [24] or generating large-scale labeled data [13], leading to high training costs and limited practicality.

To address the above limitations, we propose a novel task-agnostic **D**ynamic **P**rompt **C**ompression method, called **LLM-DPC**, reducing the number of tokens of the prompt without affecting the output performance of LLMs as much as possible. Since the decision to remove or retain a token largely depends on the evolving context, we hypothesize that prompt compression is naturally formulated as a sequential decision-making process. In this process, redundancy is reduced iteratively while essential content is preserved, with each compression decision relying on the intermediate outcomes of previous iterations. Specifically, we model the prompt compression task as a Markov Decision Process (MDP), enabling the DPC-Agent to sequentially remove redundant tokens by adapting to dynamic contexts and retaining crucial content. Furthermore, we design a reward function for training the DPC-Agent that balances the compression ratio, output distribution, and retention of key information, enabling prompt token reduction without compromising the LLM understanding and output. Importantly, this reward function does not require access to a black-box LLM, significantly reducing training costs. Additionally, inspired by curriculum learning [25], [26], [27], [28], we introduce a Hierarchical Prompt Compression (HPC) training strategy that progressively increases the difficulty of compression, enabling the agent to effectively balance efficient compression with the protection of key information.

We summarize our main contributions as follows:

- We propose a task-agnostic prompt compression method that models the compression process as a sequential decision-making problem using a Markov Decision Process (MDP). This method reduces the number of prompt tokens while aiming to minimize any negative impact on the LLM output performance. Experimental results show that LLM-DPC achieves approximately a relative improvement of 3.04% in Rouge-2 score over the state-of-the-art method, along with a higher compression ratio of 12.9x on the Arxiv-March23 dataset.
- To effectively train the DPC-Agent, we design a reward function that balances compression ratio, output quality, and retention of key information. This reward function operates without direct supervision from the target LLM, reducing training costs and enhancing practicality.
- We propose a Hierarchical Prompt Compression (HPC) training strategy that introduces progressively challenging compression tasks, allowing the proposed method to balance efficient compression with the preservation of key information effectively. Experiments show that the use of HPC yields a relative improvement of 25.5% in compression ratio and 0.5 in EM metric.

The remainder of this paper is organized as follows. Related work is presented in Section II. Section III provides the problem definition and motivations. Section IV describes the proposed LLM-DPC. Section V provides the experiments and discussions. The conclusion of this paper is in Section VI.

II. Related Work

A. Large Language Models

Large language models (LLMs) [1], [2], [3], [4], [5], [6], [12], such as the GPT series [1], [2], [4], [12], have received significant attention for their excellent generalization and comprehension capabilities in natural language processing (NLP) such as multi-round dialogue [4], [29], document summarization [30], and question answering [31]. A line of studies has attempted to enhance further the ability of LLMs to solve complex scenario tasks. Pan et al. [32] and Yang et al. [33] propose to use Knowledge Graph (KG) [34], [35] to enhance the reasoning power and interpretability of LLM. Wei et al. [10] propose the Chain of Thought (CoT) to strengthen the ability of LLMs to perform complex reasoning. Brown et al. [2] propose In-Context Learning (ICL), where task-specific prompt templates are designed using a few labeled examples to guide the LLMs in generating predictions on new test data. Lewis et al. [11] explore a Retrieval Augmented Generation (RAG) fine-tuning method that combines pre-trained parametric memory with non-parametric memory. However, many existing techniques for improving LLM capabilities have dramatically increased the length of the prompt. These rich and informative prompts can contain tens of thousands of tokens, which greatly increase inference time and application costs, and lead to poor performance due to the limited size of LLMs pre-training windows.

B. Prompt Compression

Prompts have become the dominant approach in NLP tasks [2], [36], [37], directly influencing the efficiency and performance of LLMs. As prompts grow longer, prompt compression has emerged as a key research direction [38], [39], aiming to reduce LLM reasoning time and computational cost while maintaining performance. Notably, this goal differs from text dataset condensation methods such as CondenseLM [40], which construct compact synthetic datasets to improve training efficiency. Prompt compression is typically categorized into two types: white-box methods and black-box methods [41].

The **white-box compression methods** [14], [15], [16], [17] reduce the prompt at the token-embedding level by modifying model parameters, architecture, and the self-attention mechanism of the Transformer. Chevalier et al. [14] propose AutoCompressors, which adapt pre-trained language models to compress long contexts into summary vectors that serve as soft prompts, improving language model performance and reducing inference costs. Mu et al. [15] train the gist model by modifying the attention mask to compress the prompt into a smaller set of "gist" tokens to improve computational efficiency. Xiao et al. [16] propose StreamingLLM, a framework that enables large language models to handle infinite sequence lengths by utilizing attention sinks. Wingate et al. [17] propose to learn a soft prompt that compresses the context by aligning soft prompt-based model prediction with context-based predictive alignment. However, most high-performing LLMs (such as GPT-4) are accessed via APIs, and the lack of access to source code significantly restricts the development and application of these methods. The **black-box compression methods** [13], [18], [19], [20], [21] operate at the natural language level, aiming to shorten the original prompt without losing essential information. Li et al. [21] propose Selective Context, which uses self-information to identify and prune redundant input, improving LLM reasoning efficiency by reducing memory costs and generation latency while maintaining task performance on long-context tasks. Jiang et al. [19] propose LLMLingua, a coarse-to-fine prompt compression method that leverages a budget controller, a token-level iterative compression algorithm, and instruction tuning to achieve compression with minimal performance loss. Pan et al. [13] create a task-agnostic data distillation procedure for better generalizability and efficiency. Jiang et al. [20] propose LongLLMLingua to improve LLMs perception of key information for accelerated compression. Jung et al. [18] employ a computationally efficient policy network that directly edits prompts. These methods do not require access to the LLM source code for training or inference, reducing usage costs by directly minimizing input size.

C. Reinforcement Learning

Reinforcement learning (RL) [42], [43], [44], [45], [46], [47], [48] is a machine learning paradigm in which an agent interacts with its environment to achieve specific goals. In each interaction round, the agent takes an action based on the current state of the environment, receives feedback in the form of rewards or penalties, and subsequently updates its policy. The primary objective of RL is to maximize the cumulative reward. Unlike supervised learning, which aims to minimize the expected loss for a given data distribution, RL focuses on determining a strategy that maximizes the expected value of a reward function within a specified distribution. This trial-and-error approach to decision-making in uncertain environments allows RL to operate independently of labeled datasets.

To efficiently learn the optimal policy, RL has developed various algorithms. Sutton et al. [49] propose policy gradient algorithms, it puts the current state s_t into the policy network $\boldsymbol{\pi}$ and outputs the current action a_t, the policy network $\boldsymbol{\pi}_\theta(a \mid s)$ is used directly to represent and control the behavior of an agent. Schulman et al. [50] propose proximal policy optimization algorithms, where both the policy network $\boldsymbol{\pi}$ and the value evaluation model exist and introduce restrictions on the update magnitude, which is usually used for the training of LLMs. Recently, RL has been developing rapidly in the field of LLM. Ouyang et al. [12] point out that using reinforcement learning from human feedback (RLHF) [51] can enable the model to follow a broad class of written instructions. Ichter et al. [52] introduce SayCan, which uses reinforcement learning as a method to learn language-conditioned value functions that provide guidance on what can happen in the real world, extracting and leveraging the knowledge of LLM in physical tasks to complete embodied tasks. Carta et al. [53] propose GLAM, which uses the LLM as a policy that is incrementally updated as the agent interacts with the environment, and utilizes online reinforcement learning to improve the match between the LLM knowledge and the environment.

D. Curriculum Learning

Curriculum learning (CL) organizes the training process by presenting data from simple to difficult, thereby establishing a structured learning path similar to a human curriculum [25], [54]. Since its introduction [55], CL has been widely studied across different domains [56], [57], [58], [59]. Early extensions include self-paced learning [60], which adaptively selects training instances based on model competence, and difficulty-aware scheduling [56], which automatically designs curricula according to data complexity. In natural language processing, CL has been applied to a wide range of tasks, including machine translation [61] and open-domain dialogue [28], showing improvements in convergence stability and generalization. Platanios et al. [61] propose a competence-based curriculum learning framework for neural machine translation, which adaptively selects training samples according to model competence and sample difficulty, thereby reducing training time, alleviating the need for extensive heuristics, and improving overall translation performance. Wang et al. [28] propose a UPC for open-domain dialogue, which leverages iterative curriculum learning to train the chatbot from easy-to-communicate users to more challenging ones, thereby gradually enhancing user-oriented proactivity.

III. Problem Definition and Motivations

A. Problem Definition

Given original prompt $\boldsymbol{x} = \{x_i\}_{i=1}^{L}$, a prompt compression system is designed to generate a compressed prompt $\widetilde{\boldsymbol{x}} =$

$\{\widetilde{x}_i\}_{i=1}^{\widetilde{L}}$, where L and $\widetilde{L}$ represent the numbers of tokens in $\boldsymbol{x}$ and $\widetilde{\boldsymbol{x}}$, respectively. The compression rate is defined as $\rho = \widetilde{L}/L$, $\rho \in [0, 1]$, and the compression ratio is $1/\rho$. We prefer a smaller value of ρ for lower inference cost. Let $\widetilde{\boldsymbol{x}_G}$ represent the LLM-generated results derived by $\widetilde{\boldsymbol{x}}$ and $\boldsymbol{x}_G$ denotes the tokens derived by $\boldsymbol{x}$, the distribution of $\widetilde{\boldsymbol{x}_G}$ is expected to be as similar to $\boldsymbol{x}_G$ as possible. The objective of a prompt compression system can be formulated as:

$$\min_{\widetilde{\boldsymbol{x}}} KL(P(\widetilde{\boldsymbol{x}}_G|\widetilde{\boldsymbol{x}}), P(\boldsymbol{x}_G|\boldsymbol{x})) + \rho, \tag{1}$$

where $KL(\cdot, \cdot)$ denotes the Kullback–Leibler (KL) divergence, which measures the difference between two probability distributions. Specifically, $P(\tilde{\boldsymbol{x}}_G|\tilde{\boldsymbol{x}})$ and $P(\boldsymbol{x}_G|\boldsymbol{x})$ represent the output distributions of the LLM conditioned on the compressed prompt $\tilde{\boldsymbol{x}}$ and the original prompt $\boldsymbol{x}$, respectively.

B. Motivation

Many existing prompt compression methods are task-aware, which limits their generalizability across different downstream tasks. Moreover, most task-agnostic methods estimate token importance using information entropy from causal language models, neglecting the sequential nature of prompt compression, where each token significance depends on the evolving context. To address these issues, we hypothesize that prompt compression can be viewed as a dynamic, iterative decision-making process. Specifically, whether a token should be preserved or removed cannot be determined in isolation but depends on the evolving prompt state after previous compression steps. This means that prompt compression is not a one-shot pruning task, but rather a sequential decision problem in which each action influences the subsequent context and decisions. A natural idea emerges: *Could we iteratively eliminate redundancy from the prompt while preserving its critical content through a series of decisions?*

The answer is yes. Inspired by trial-and-error learning, we model prompt compression as a Markov Decision Process (MDP), where the DPC-Agent iteratively compresses the prompt by removing redundant tokens while preserving essential content, with each decision building on the outcomes of previous steps for efficient, context-aware compression. We design a reward function that balances compression ratio, output distribution, and key information retention, ensuring that the model understanding and output quality remain intact. Additionally, considering the challenges of retaining essential information while achieving a high compression ratio in the prompt compression task, we incorporate curriculum learning [25], [26], [62], progressively introducing more complex compression tasks to enhance the agent's ability to compress prompts efficiently while preserving essential content.

IV. Proposed Methods

In this paper, we propose a Dynamic Prompt Compression method, called LLM-DPC, which seeks to remove redundant content in a given input prompt, thereby reducing computational cost and better using the limited context window in LLMs. It is worth noting that compressed prompts may occasionally exhibit grammatical discontinuities, as some less informative tokens are removed during the compression process. For tasks that require more fine-grained reasoning about user preferences, such as recommendation systems [63], [64], grammatical continuity may play a more critical role. Nevertheless, the primary design objective of our method is to preserve semantic fidelity rather than grammatical fluency, ensuring that LLMs can still effectively leverage the retained key information for accurate reasoning. As shown in Fig. 2. We model the prompt compression process as a Markov Decision Process (MDP) and train a DPC-Agent to determine an optimal compression pathway. Given an input prompt, we convert it to a token sequence, which serves as the initial state in the MDP framework. At time step t, the DPC-Agent selects specific tokens to be removed, yielding a compressed token sequence that constitutes the subsequent state s_{t+1}. Then the reward is calculated according to (4). The trajectory is collected to train the DPC-Agent via our designed hierarchical prompt compression (HPC) training strategy. Additionally, the next state is input to the DPC-Agent for further iterations. This iterative process continues until the maximum trajectory length is reached. The final token sequence is decoded into compressed text, with a much lower token number, without affecting the output performance.

A. Dynamic Prompt Compression as an MDP

We seek a general DPC-Agent to remove redundant tokens for a dynamic input prompt, thereby improving the inference efficiency while maintaining the quality of the generated text as much as possible. To this end, we formulate the step-by-step removal of redundant tokens as a Markov Decision Process (MDP) [65]: $< \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{R}}, \boldsymbol{\pi} >$. The state space of the environment is $\boldsymbol{S}$ and the action space of the agent is $\boldsymbol{A}$. At time step t, the agent takes the state $s_t \in \boldsymbol{S}$ as input and performs an action $a_t \in \boldsymbol{A}$ through the policy network $\boldsymbol{\pi} : \boldsymbol{S} \times \boldsymbol{A} \rightarrow [0, 1]$. The environment changes to the next state $s_{t+1} = \boldsymbol{\mathcal{T}}(s_t, a_t)$ according to the transition function $\boldsymbol{\mathcal{T}}$ and a reward $r_t = \boldsymbol{\mathcal{R}}(s_t, a_t)$ is received with reward function $\boldsymbol{\mathcal{R}}$. In this work, the MDP is detailed as follows:

States $\boldsymbol{S}$ is the description for the environment. At time step t, the state is a compressed prompt: $s_t = \widetilde{x}_{t-1} = \{x_i\}_{i=1}^{\widetilde{L}_{t-1}}$, where $\widetilde{L}_{t-1}$ is the number of tokens after compression processing at time step $t-1$. Thus, the agent can predict which tokens need to be removed based on the current compressed prompt.

Actions $\boldsymbol{A}$ is a discrete set of actions the agent can take. In this task, the action space is represented as a binary vector $\boldsymbol{A} = \{0, 1\}^n$, where n corresponds to the number of tokens in the current prompt state s_t. Each dimension of a_t corresponds to a specific token: 0 indicates removal, and 1 indicates preservation. At time step t, the agent gives the action $a_t \in \boldsymbol{A}$ based on the state s_t to remove redundant tokens.

Transition $\boldsymbol{\mathcal{T}}(\boldsymbol{S}, \boldsymbol{A})$ is a function $\boldsymbol{\mathcal{T}}$: $\boldsymbol{S} \times \boldsymbol{A} \rightarrow \boldsymbol{S}$ which maps a state s_t into a new state s_{t+1}. When the maximum trajectory length is reached, this episode will be terminated and s_{T+1} is $None$. Otherwise, the action (preservation/removal) at time step t for each token will result in a new prompt. It can be represented

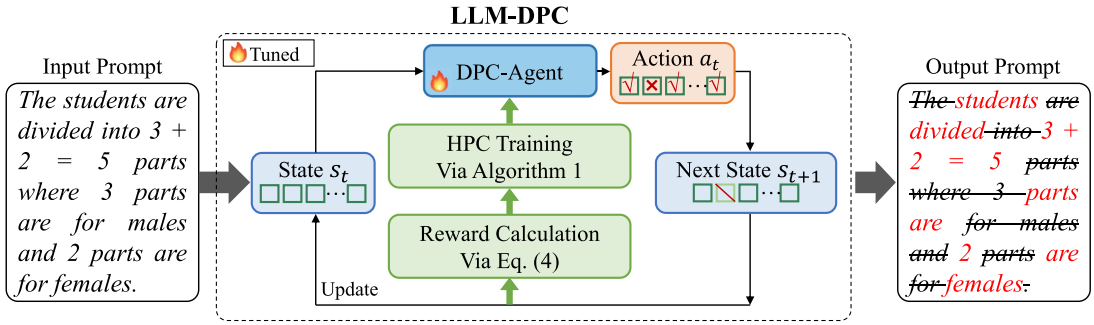

Fig. 2. General diagram of proposed LLM-DPC. We model prompt compression as a Markov Decision Process (MDP) and train a DPC-Agent to determine an optimal compression pathway. The input prompt represented as a token sequence serves as the initial state of the MDP. At time step t, the DPC-Agent performs the action to select specific tokens to retain or discard, yielding a compressed token sequence as the next state s_{t+1}. Then the reward is calculated according to (4). Our designed hierarchical prompt compression (HPC) training strategy collects the trajectory, which is applied to train the DPC-Agent. This process iterates until reaching the max trajectory length. The final token sequence is decoded into compressed text, with a much lower token number without affecting the output performance as much as possible.

as follows:

$$s_{t+1} = \boldsymbol{\mathcal{M}}_{a_t}(s_t), \tag{2}$$

where $\boldsymbol{\mathcal{M}}_{a_t}(\cdot)$ is the operation that removes redundant tokens according to action a_t.

Rewards $\boldsymbol{\mathcal{R}}(s_t, a_t)$ is the reward function. In the LLM prompt compression task, the reward can be seen as minimizing the LLM output results while reducing the length of the prompt. The details of the reward function we designed are given in the Section IV-B.

Policy $\boldsymbol{\pi}_\theta(a \mid s) : \boldsymbol{S} \times \boldsymbol{A} \to [0, 1]$ describes the behaviors of the agent. During the training process, the agent takes the current state s_t as input and outputs a probability distribution for each possible action $a_t \in \boldsymbol{A} = \{0, 1\}^n$:

$$\pi\left(a_t \mid s_t; \theta\right) = \frac{\exp\left\{f_\theta\left(s_t\right)_i\right\}}{\sum_{j=1}^{K} \exp\left\{f_\theta\left(s_t\right)_j\right\}}, \tag{3}$$

where $f_\theta(s_t)$ is the output vector of the policy network with input s_t, and i denotes the action style (0 or 1). The θ is the learnable parameter of the policy network. Here, n is the dimensionality of the action space, corresponding to the number of tokens considered for compression, and K is the total number of possible actions at each step, serving as the normalization term in the softmax.

B. Reward Function

Our goal is to reduce the number of tokens in the prompt without losing key information, not affecting LLM understanding of the prompt and the generation of results, as shown in (1). Therefore, we design a reward function that takes into account the compression ratio, the Kullback-Leibler (KL) divergence [66] of the LLM-generated result distribution, and the degree of retention of key information from the prompt. The motivation behind this design is to balance efficiency and fidelity: the compression ratio term encourages shorter prompts to reduce inference cost, the information retention term ensures that the compressed prompt preserves the critical semantic content, the KL divergence term aligns the output distributions to avoid distortions in reasoning, and the boundary constraints prevent extreme cases of over- or under-compression. The reward function is as follows:

$$\begin{aligned}\boldsymbol{\mathcal{R}}(s_t, a_t) = \alpha\frac{1}{\rho} + \beta D(s_0, s_t) \\ - \gamma KL(P(s_{tG}|s_t), P(s_{0G}|s_0)) \\ - \mathbb{I}(\rho < c_s)P_s - \mathbb{I}(\rho > c_l)P_l,\end{aligned} \tag{4}$$

where $D(\cdot, \cdot)$ is used to compute the degree of key information retention for the original prompt (*i.e.*, initial state s_0) and the compressed prompt (*i.e.*, state s_t at time step t) and here BERTScore [67] is used, we chose BERTScore because it evaluates semantic similarity using contextual embeddings from pre-trained language models, which makes it more suitable than n-gram overlap metrics for assessing whether compressed prompts retain key information [68]. c_s and c_l are hyperparameters that indicate the lower and upper bounds of the expectation compression ratio, P_s and P_l are penalties for compressing prompts that are too short (over-compressed) and too long (under-compressed), respectively. Rather than being static, the thresholds c_s and c_l are dynamically adjusted during training via (8). The $\mathbb{I}(\cdot)$ is an indicator function. The s_{0G} and s_{tG} are the outputs of the LLM according to s_0 and s_t. Here the resulting distribution $P(\cdot)$ is not obtained from the target black-box LLM, but from a distribution-aligned small model, see Section IV-D for details.

Remark: Unlike existing reinforcement learning-based summarization methods [69], [70], the reward function we designed does not consider the fluency and grammar of the compressed prompt, which is due to the fact that LLM has a good tolerance for prompts that lack fluency and grammatical errors [13], [19], [20]. Disregarding the fluency and grammar of the prompt is beneficial for obtaining a higher compression ratio. In addition, the reward function we design does not require the involvement of a black-box LLM, which is different from the existing method [18], [24].

C. Hierarchical Prompt Compression Training Strategy

Considering the challenges of retaining essential information while achieving high compression ratio in the prompt

compression task, and inspired by the progressive difficulty adjustment used in curriculum learning [25], [26], we propose Hierarchical Prompt Compression (called **HPC**) training strategy for Proximal Policy Optimization (PPO) [50] process. The HPC training strategy introduces increasingly difficult compression tasks so that the agent gradually learns to balance efficient compression and preservation of key information. The details are as follows:

Actor: The actor (also called agent) π_θ is trained in binary classification (*i.e.*, preservation or discarding of tokens) of the prompt according to the original prompt $\boldsymbol{x} = \{x_i\}_{i=1}^{L}$. To utilize the bidirectional contextual information of each token, we utilize the Transformer encoder as a feature extractor and then send the features to a linear classification layer. Specifically, at time step t, the state $s_t = \widetilde{x}_{t-1} = \{x_i\}_{i=1}^{\widetilde{L}_{t-1}}$ contains $\widetilde{L}_{t-1}$ tokens, which can be formalized as:

$$\boldsymbol{h} = f_\theta(\widetilde{x}_{t-1}), \tag{5}$$

$$p(x_i, \theta) = \text{softmax}(Wh_i + b), \tag{6}$$

where $\boldsymbol{h} = \{h_i\}_{i=1}^{\widetilde{L}_{t-1}}$ is feature vectors for all tokens, $p(x_i, \theta) \in \mathbb{R}^2$ denotes the probability distribution of label $\{0, 1\}$ for the $i-$th token x_i. Here we use *xlm-roberta-large* [71] as Transformer encoder f_θ. In the off-policy algorithm, the old policy $\pi_{\theta_{old}}$ with old parameters θ_{old} is used to collect trajectories with the environment, while the policy π_θ is updated using trajectories collected by $\pi_{\theta_{old}}$.

Critic: The critic $V_\phi(s)$ denotes the state-value function, which estimates the expected long-term return starting from state s. It is used to calculate the advantage, which can guide the actor in learning more efficiently and stably. Similar to the actor, the critic is composed of a pre-trained *xlm-roberta-large* [71] as an encoder, and with two linear layers. Besides, the old critic $V_{\phi_{old}}(s)$ is used to collect trajectories, and the new critic $V_{\phi_{new}}(s)$ is updated using the collected trajectories.

Learning Objectives: The goal of the learning is to maximize the expected long-term return $\mathcal{J}(\theta)$:

$$\begin{aligned}\mathcal{J}(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[G(\tau)] \\ &= \mathbb{E}_{\tau\sim\pi_{\theta_{old}}(\tau)}[\min(\delta A^{\pi_{\theta_{old}}}(s_t, a_t), \\ &\quad \text{clip}\left(\delta, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_{old}}}(s_t, a_t))],\end{aligned} \tag{7}$$

where $G(\tau)$ is the total return of the trajectory $\tau = \{s_t, a_t, r_t, v_t, A^{\pi_{\theta_{old}}}(s_t, a_t)\}$ obtained by $\pi_{\theta_{old}}$ and $V_{\phi_{old}}$, $\delta = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the ratio of the probability of action a_t given by π_θ and $\pi_{\theta_{old}}$ for state s_t, and ϵ is a hyperparameter, we set to 0.15 in this paper. The operation clip $(\delta, 1-\epsilon, 1+\epsilon)$ constrains δ to the range $[1-\epsilon, 1+\epsilon]$, and $A^{\pi_{\theta_{old}}}(s_t, a_t) = r_t - V_{\phi_{old}}(s_t)$ is the advantage at t.

HPC Training: The overview of the optimization process of the HPC training strategy is presented in Algorithm 1. Specifically, given a prompt for compression dataset $\mathcal{D}$, we use $\pi_{\theta_{old}}$ and $V_{\phi_{old}}(s)$ to interact with the environment to collect the trajectory $\tau = \{s_t, a_t, r_t, v_t\}$, and compute the advantage $A^{\pi_{\theta_{old}}}(s_t, a_t)$. During the collection of trajectories, the HPC training strategy increases the compression difficulty and guides the learning of the DPC-Agent incrementally by gradually decreasing the compression ratio range $[c_s, c_l]$ (see (4)) and the maximum trajectory length $T_{\max}$ in stage (P_i). The c_s and c_l are adjusted as follows:

$$\begin{cases} c_s = 0.6 - (P_i + \frac{t}{T_{\max}})\psi \\ c_l = 1.0 - (P_i + \frac{t}{T_{\max}})\psi \end{cases}, \tag{8}$$

where ψ set to 0.1, P_i denotes the i^{th} stage, with i starting at 1 and $P_1 = 1$. Notably the learning stage size is set to 3, and $T_{\max} = 2$ except for the third stage where $T_{\max} = 1$. This easy to difficult curriculum learning strategy effectively improves the performance of prompt compression. In this way, the thresholds (c_s and c_l) are not fixed but progressively adjusted across curriculum stages: each stage provides a stable target interval, while the feasible region gradually tightens as P_i increases. We then put τ into the replay buffer $\mathcal{B}$. When a certain number of trajectories (such as M) have been collected, they are used to train the actor and critic. In particular, we begin by uniformly sampling sequences from the replay buffer $\mathcal{B}$, then compute the expected long-term return $\mathcal{J}(\theta)$ to optimize the policy parameters π_θ. Additionally, the Temporal Difference (TD) error δ_t is calculated to refine the critic parameters V_ϕ:

$$\delta_t = G_t - V_\phi(s_t), \tag{9}$$

Algorithm 1: The HPC Training for LLM-DPC.

Input: The prompt for compression dataset $\mathcal{D}$, the DPC-Agent $\pi_{\boldsymbol{\theta}}$, the critic V_ϕ the replay buffer $\mathcal{B}$, the maximum trajectory number of the buffer M, the iteration number of training m, the number of curriculum learning stages P and the coefficients c_s and c_l.

Initialize buffer $\mathcal{B}$, actor parameters θ and critic parameters ϕ.
while Not convergence **do**
  for P_i in P **do**
    Calculate c_s and c_l via (8).
    for x_i in $\mathcal{D}$ **do**
      Collect a trajectory $\tau = \{s_t, a_t, r_t, v_t, A^{\pi_{\theta_{old}}}(s_t, a_t)\}$ with old $\pi_{\theta_{old}}$ and $V_{\phi_{old}}$.
      Put τ into $\mathcal{B}$.
      if $length(\mathcal{B}) == M$ **then**
        for $iteration = 1, 2, \ldots, M$ **do**
          Uniformly sample $\tau \in \mathcal{B}$.
          Calculate $\mathcal{J}(\theta)$ via (7).
          Update θ to maximize $\mathcal{J}(\theta)$.
          Calculate TD error via (9).
          Update ϕ to minimize TD error δ_t.
        end for
        Empty the replay buffer $\mathcal{B}$.
        Update $\theta_{old} \leftarrow \theta$.
        Update $\phi_{old} \leftarrow \phi$.
        end if
      end for
  end for
end while

where G_t represents the total expected return starting from time step t. After conducting a certain number of training iterations using the samples from the existing replay buffer $\mathcal{B}$, we clear the buffer and update the parameters of the old policy $\pi_{\theta_{old}}$ and critic $V_{\phi_{old}}$. This process is then repeated until convergence is achieved.

D. Distribution Alignment

The core challenge of our reward function design lies in computing the KL divergence term based on the output distribution of the target black-box LLM (*e.g.*, GPT-4o-mini). Directly querying such models for every training step would be financially expensive and computationally infeasible. To address this, we adopt a distribution alignment strategy by training a smaller language model M_s to approximate the distribution of the target LLM. Concretely, we use instruction-tuning to align M_s with the behavior of the target LLM. Given a dataset of paired instructions and responses generated by a black-box LLM of the target family. The optimization process of M_s can be formulated as follows:

$$\min_{\theta_{\mathcal{M}_s}} \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(\boldsymbol{x}_i, \boldsymbol{y}_{i,\text{LLM}}; \theta_{\mathcal{M}_s}\right)\right], \tag{10}$$

where $\theta_{\mathcal{M}_s}$ is the parameters of M_s, $(\boldsymbol{x}_i, \boldsymbol{y}_{i,\text{LLM}})$ is the pair of instruction $\boldsymbol{x}_i$ and the black-box LLM generated texts $\boldsymbol{y}_{i,\text{LLM}}$, and N is the number of all examples used for instruction tuning. Notably, the Llama3-8B [72] is selected for M_s. This strategy significantly improves efficiency. Once the small language model is aligned, the DPC-Agent can be trained entirely offline, avoiding repeated and costly API calls to the target black-box LLM. The alignment step is a one-time cost, after which the small language model can be reused throughout training, reducing overhead by orders of magnitude and making the overall solution practical.

V. Experiment

In this section, we introduce the experimental settings in Section V-A. Our LLM-DPC is compared against the state-of-the-art (SOTA) prompt compression methods in Section V-B. We show relevant examples of the proposed LLM-DPC in Section V-C. We also performed numerous ablation experiments to validate the effectiveness of LLM-DPC and to gain a deeper understanding of the proposed method in Section V-D. Additionally, we further discuss the effect of different hyperparameters on the proposed method in Section V-E.

A. Experimental Settings

1) Compared Methods: Following the previous working setup [13], we compare the proposed LLM-DPC with three SOTA task-agnostic prompt compression methods.

- *Selective-Context [21]:* Use a small model to compute the self-information of each token and fuse it into a lexical unit *u* (each lexical unit consists of multiple tokens $(x_t, \ldots, x_{t+\alpha})$), retaining lexical unit self-information over a threshold value.
- *LLMLingua [19]:* It dynamically assigns different compression ratios $(\tau, \tau_{que}, \tau_{ins}, \tau_{dems})$ to the various parts of the prompt, divide the prompt into multiple segments $S = \{s_1, s_2, \ldots, s_m\}$, where tokens greater than threshold in each segment are retained.
- *LLMLingua-2 [13]:* It treats prompt compression as a token classification task, and it is available in LLMLingua-2-small and LLMLingua-2 versions.

2) Datasets: To comprehensively evaluate the effectiveness of the proposed LLM-DPC, we conduct experiments on four different datasets on summarization, conversation, reasoning, and In-context learning (ICL) tasks.

- *Arxiv-March23:* It is a dataset consisting of the latest academic papers from the arXiv preprint repository, collected since March 2023. For our experimental evaluation, we employ a subset of 500 entries sourced from the dataset created by Li et al. [21], which includes only the first two sections of each article to avoid excessive length.
- *ShareGPT:* A dataset of 90 k conversations collected from *sharegpt.com*[1], involving multiple rounds of dialogue between users and ChatGPT in multiple languages and scenarios. We test the conversation task using sharegpt575 [21], which contains 575 multi-round dialogue examples.
- *GSM8K [73]:* A widely used dataset for testing logic and mathematics in language modeling, containing 8.5 k high-quality linguistically diverse mathematical problems.
- *BBH [74]:* A subset of the BIG-Bench dataset [75], it focuses on a set of 23 challenging tasks covering multi-step arithmetic, algorithmic reasoning, language comprehension, and world knowledge. It is specifically designed to assess CoT prompting. For our experiments, we chose six tasks to test, including *Boolean Expressions*, *Causal Judgement*, *Date Understanding*, *Disambiguation QA*, *Dyck Languages*, and *Formal Fallacies*.

3) Evaluation Metrics: Following the settings of LLMLingua [19], we use BLEU [76], BLEURT [77], ROUGE [78] and BERTScore (BS F1) [67] as evaluation metrics for Arxiv-March and ShareGPT. We use Exact Match (EM) [19] as a metric for GSM8K and BBH. In addition, the compression ratio $(1/\rho)$ is also included in the assessment metrics to ensure fairness. Note that we use the tokenizer of Llama3[2] to calculate the number of tokens.

4) Implementation Details: Our proposed LLM-DPC is implemented using PyTorch framework with PyTorch version 2.1.2 and runs on the 80 GB NVIDIA A800 GPU with CUDA version 12.1. We use Adam as our optimizer to update the parameters of neural networks. The learning rate is set to 10^{-5} for the actor model and 10^{-6} for the critic model. The batch size is set to 4 and a total of 4 epochs are trained. The first and second stages are trained for 1 epoch respectively, and the third stage is trained for 2 epochs. The P_s and P_l in (4) are set to 200 and 100, respectively.

[1]https://sharegpt.com/

[2]https://huggingface.co/meta-llama/Meta-Llama-3-8B

TABLE I
PERFORMANCE OF DIFFERENT METHODS ON THE CONVERSATION (SHAREGPT) AND SUMMARIZATION (ARXIV-MARCH23) TASKS

Method	Pub.'Year	BLEU ↑	BLEURT ↑	Rouge-1 ↑	Rouge-2 ↑	Rouge-L ↑	BS F1 ↑	Tokens ↓	$1/\rho$ ↑
ShareGPT									
Selective-Context [21]	EMNLP'2023	38.53	-0.21	51.27	38.35	43.51	78.30	183	3.3x
LLMLingua [19]	EMNLP'2023	38.71	-0.21	51.43	38.62	43.57	78.27	186	3.2x
LLMLingua-2-small [13]	ACL'2024	56.79	0.37	76.09	58.47	63.56	89.54	191	3.1x
LLMLingua-2 [13]	ACL'2024	61.97	0.47	78.64	63.07	67.50	90.87	184	3.3x
LLM-DPC (Ours)	-	**64.93**	**0.54**	**80.24**	**65.54**	**69.89**	**91.80**	**175**	**3.4x**
Arxiv-March23									
Selective-Context [21]	EMNLP'2023	8.83	-0.61	43.43	13.46	18.92	73.75	933	11.8x
LLMLingua [19]	EMNLP'2023	5.70	-0.74	32.29	8.78	15.17	69.60	1276	8.7x
LLMLingua-2-small [13]	ACL'2024	8.56	**-0.45**	45.52	15.47	21.09	75.49	1017	10.9x
LLMLingua-2 [13]	ACL'2024	**10.84**	-0.57	48.49	14.62	19.95	75.15	920	12.0x
LLM-DPC (Ours)	-	10.10	-0.55	**48.81**	**15.94**	**21.63**	**75.91**	**855**	**12.9x**

TABLE II
PERFORMANCE OF DIFFERENT METHODS ON THE REASONING (GSM8K), AND IN-CONTEXT LEARNING (BBH) TASKS

Method	Pub.'Year	*1-shot constraint*			*half-shot constraint*		
		EM ↑	Tokens ↓	$1/\rho$ ↑	*EM* ↑	Tokens ↓	$1/\rho$ ↑
GSM8K							
Selective-Context [21]	EMNLP'2023	76.57	436	5.4x	76.15	182	13.0x
LLMLingua [19]	EMNLP'2023	76.72	462	5.1x	77.02	174	13.6x
LLMLingua-2-small [13]	ACL'2024	75.66	425	5.6x	76.80	151	15.7x
LLMLingua-2 [13]	ACL'2024	76.87	415	5.7x	76.80	**140**	**16.9x**
LLM-DPC (Ours)	-	**77.03**	**343**	**6.9x**	**77.03**	153	15.5x
BBH							
Selective-Context [21]	EMNLP'2023	82.81	278	2.8x	81.91	152	5.1x
LLMLingua [19]	EMNLP'2023	81.68	271	2.9x	**84.72**	162	4.8x
LLMLingua-2-small [13]	ACL'2024	82.73	274	2.8x	82.12	155	5.0x
LLMLingua-2 [13]	ACL'2024	82.41	255	3.0x	82.64	145	5.3x
LLM-DPC (Ours)	-	**83.16**	**251**	**3.1x**	83.98	**145**	**5.3x**

For the training of model M_s in Section IV-D, we use the alpaca-gpt4-data[3] dataset (randomly selected 80% for the training set and 20% for the test set) to fine-tune Llama3-8B, and the training framework used is LLaMA-Factory[4]. Notably, the training hyperparameters use the default settings for full fine-tuning provided by LLaMA-Factory. We randomly selected 2048 prompt samples from the alpaca-gpt4-data dataset as training data to train the DPC-Agent. During the testing phase, we control the compression ratio (*e.g.*, 3x or 10x) by controlling the maximum step size. We employ the GPT-4o-mini-2024-07-18[5] as the target LLMs, with greedy decoding at a temperature of 0 for enhanced stability across experiments.

B. Comparison Experiments

We compare the proposed LLM-DPC and three SOTA prompt compression methods to demonstrate the superior performance of our proposed method. We conduct experiments on a variety of downstream tasks, including conversation task (see Table I), summarization task (see Table I), reasoning task (see Table II), and In-context learning task (see Table II).

Excellent performance of the LLM-DPC in the conversation task: As shown in Table I, LLM-DPC outperforms other SOTA methods in the conversation task. Specifically, compared to LLMLingua-2, the LLM-DPC achieves about a 4.8% (61.97→64.93) relative improvement on BLEU and about a 1.0% (90.87→91.80) relative improvement on BS F1 at higher compression ratio (3.3x →3.4x). The proposed LLM-DPC achieves a 17.0% relative improvement over the classical method, Selective-Context, on the BLEU metric. We conclude that LLM-DPC removes redundant tokens according to prompt dynamic inputs, which allows outperforming SOTA methods in all metrics while maintaining a high compression ratio.

Excellent performance of the LLM-DPC in the summarization task: As shown in Table I, our proposed LLM-DPC outperforms SOTA methods in the summarization task. Specifically, the proposed LLM-DPC achieves a relative improvement of 9.0% (14.62→15.94) on Rouge-2 metric compared to LLMLingua-2, while having a higher compression ratio (12.0x→12.9x). Our proposed LLM-DPC is not optimal in BLEU metric compared to LLMLingua-2, the main reason is that our DPC-Agent is trained on conversation data, while LLMLingua-2 is trained on the summarization task dataset, MeetingBank [79]. Meanwhile, it exactly proves that the proposed LLM-DPC still achieves better prompt compression performance in the cross-task situation.

[3]https://huggingface.co/datasets/llm-wizard/alpaca-gpt4-data
[4]https://github.com/hiyouga/LLaMA-Factory
[5]https://platform.openai.com/

LLM-DPC trade-off between performance and compression ratio: As shown in Table II, the proposed LLM-DPC outperforms the SOTA methods in the reasoning task. Specifically, with the *1-shot constraint*, our proposed LLM-DPC has a relative improvement of 21.1% (5.7x→6.9x) in compression ratio and 0.2% (76.87→77.03) in EM metric compared to LLMLingua-2. With *half-shot constraint*, our proposed LLM-DPC has a relative improvement of 0.3% in EM metric over LLMLingua-2, with a compression ratio of 15.5x. We conclude that the proposed LLM-DPC trades off between performance and compression ratio. In the reasoning task, the performance of the EM metric is not significantly improved between our proposed method and the existing prompt compression methods with approximately the same compression ratio, a possible factor is that the target black-box model, GPT-4o-mini, already performs well on this task, even though the prompt of the CoT is not complete.

Excellent performance of LLM-DPC in In-context learning task: As shown in Table II, the proposed LLM-DPC outperforms the SOTA methods in the EM metric at a higher compression ratio. Specifically, with the *1-shot constraint*, the proposed LLM-DPC achieves a relative improvement of about 1.0% (82.41→83.16) in EM metric compared to LLMLingua-2, along with a relative improvement of 3.3% (3.0x→3.1x) in compression ratio. With *half-shot constraint*, the LLM-DPC improves the EM metric by a relative 1.6% compared to LLMLingua-2 while maintaining the same compression ratio.

Overall, our proposed LLM-DPC is a task-agnostic prompt compression method that achieves to outperform the SOTA methods on four challenging tasks, such as the summarization task and reasoning task, by training only on the QA type dataset. On the one hand, it is because we model the prompt compression task as an MDP, and the DPC-Agent is able to remove redundant tokens according to the dynamic prompt inputs. On the other hand, it is because the reward function we designed balances the compression ratio, the output distribution of LLM, and the key information retention.

C. Examples of LLM-DPC

We show an example of LLM-DPC and LLMLingua-2 on a reasoning task to demonstrate the effect of prompt compression, as shown in Fig. 3. The LLM-DPC and LLMLingua-2 are both token-level prompt compression methods, and although the compressed prompts are poorly readable, this does not have a significant impact on the understanding of the prompts by the LLM. In addition, under the 1-shot constraint setting, our LLM-DPC achieves a compression ratio of **6.9**×, which is higher than that of LLMLingua-2 (**5.7**×). Our proposed LLM-DPC retains more key information, which makes the prompts obtained after LLM-DPC compression allow LLM to output more accurate answers compared to LLMLingua-2.

D. Ablation Studies

We follow the experimental setup of Section V-A and conduct a variety of ablation experiments to validate the effectiveness of modeling prompt compression as an MDP and the proposed HPC training strategy. Here, we experiment with the reasoning task in the GSM8K dataset.

TABLE III
ABLATION STUDY ON THE GSM8K DATASET WITH *1-SHOT CONSTRAINT*

Version	*1-shot constraint*		
	EM ↑	Tokens ↓	$1/\rho$ ↑
Random	76.04	428	5.5x
LLM-DPC (w/o Training)	76.19	<u>422</u>	<u>5.6x</u>
LLM-DPC (w/o HPC)	<u>76.57</u>	431	5.5x
LLM-DPC (Ours)	**77.03**	**343**	**6.9x**

Effectiveness of prompt compression with MDP: We compare LLM-DPC and random deletion tokens to demonstrate the effectiveness of modeling prompt compression as an MDP, as shown in Table III. Compared to the random deletion tokens, our method achieves a relative improvement of 1.3% (76.04→77.03) in EM metric and a relative improvement of 25.5% (5.5x→6.9x) in compression ratio. A primary reason is the modeling of prompt compression as an MDP, where the trained DPC-Agent is able to iteratively refine the prompt by removing redundant tokens while preserving essential content, with each decision building on the outcomes of previous steps for efficient, context-aware compression. These results empirically confirm that modeling prompt compression as a sequential decision-making process is beneficial.

Effectiveness of HPC training strategy: We compare LLM-DPC and LLM-DPC (w/o HPC) to verify the effectiveness of the proposed Hierarchical Prompt Compression training strategy, as shown in Table III. Compared to LLM-DPC (w/o HPC), LLM-DPC has a relative improvement of 0.6% (76.57→77.03) in EM metrics and a relative improvement of 25.5% (5.5x→6.9x) in the compression ratio. An important reason is that the HPC training strategy setting makes the training difficulty incremental step by step, which helps the DPC-Agent to better learn how to remove the redundant tokens in the dynamic prompt input.

Overall, these ablation studies highlight that the DPC-Agent architecture and the HPC training strategy are two integral and complementary components of our method. Neither alone can achieve SOTA performance: the DPC-Agent provides the capacity for dynamic, token-level decision-making, while the HPC strategy enables stable and progressive learning of a high-performing policy. The superior results of the full LLM-DPC system ultimately stem from the synergy of these two components working together.

E. Discussion

We conduct extensive experiments on the reasoning task to discuss the effect of more details on LLM-DPC, such as the effect of each part of the reward function on the LLM-DPC.

Effect of the reward function $\mathcal{R}$: We study the effects of compression ratio, LLM output distribution, and information retention in our proposed reward function on the performance of the proposed LLM-DPC by adjusting $\alpha = 0$, $\beta = 0$ or $\gamma = 0$ in (4). As shown in Table IV, when $\alpha = 0$, the lack of compression ratio of the reward signal may cause the DPC-Agent to delete some key information, leading to a decrease in the EM metric. When $\beta = 0$, the reward signal lacks the reward of key information retention, which may cause the DPC-Agent not to pay attention to the key information retention of the

Compressed Prompt by LLM-DPC:

……(omitted). ~~Question: In a certain school, 2/3 of the male students like to play basketball, but only 1/5 of the female students like to play basketball. What percent of the~~ population ~~of the school~~ do ~~not like~~ to play basketball if ~~the~~ ratio ~~of~~ the ~~male~~ to ~~female~~ students is 3:2 and ~~there are~~ 1000 students?~~\n~~Let's ~~think step~~ by step~~\nThe~~ students ~~are~~ divided ~~into~~ 3 + 2 = 5 ~~parts where 3~~ parts are ~~for males~~ ~~and~~ 2 ~~parts~~ are ~~for~~ females.~~\n~~Each part represent~~s~~ 1000/5 = 200 students.~~\nSo~~, there are ~~3 x~~ 200 ~~= 600~~ male~~s.\nAnd~~ there ~~are 2 x~~ 200 ~~= 400~~ females.~~\nHence,~~ 600 x 2/3 = 400 male~~s~~ ~~play basketball.\nAnd~~ 400 ~~x 1/5 = 80 female~~s play ~~basketball.\nA~~ total ~~of 400 + 80~~ = 480 ~~students~~ play ~~basketball.\nTherefore,~~ ~~1000 - 480 = 520 do not like~~ to play ~~basketball.\nThe~~ percentage of ~~the school that do not like to play~~ basketball ~~is 520~~/1000 * 100 ~~= 52\nThe~~ answer is 52~~\n~~

Compressed Prompt by LLMLingua-2:

……(omitted). ~~Question: In a certain school,~~ 2/3 ~~of the~~ male ~~students like to play basketball, but only~~ 1/5 ~~of the~~ female ~~students like to play basketball. What percent of the population of the school do not like to play basketball if the~~ ratio ~~of the male to female students is~~ 3:2 ~~and there are~~ 1000 ~~students?\nLet's think step by step\nThe students are~~ divided ~~into~~ 3 + 2 = 5 parts ~~where~~ 3 ~~parts are for~~ males ~~and~~ 2 ~~parts are for~~ females.~~\nEach part represents 1000/5 =~~ 200 ~~students.\nSo, there are~~ 3 ~~x 200 =~~ 600 males.~~\nAnd there are~~ 2 ~~x 200 =~~ 400 females.~~\nHence, 600 x 2/3 =~~ 400 ~~males play basketball.\nAnd 400 x~~ 1/5 = 80 females ~~play basketball.\nA total of 400 + 80 =~~ 480 ~~students play basketball.\nTherefore,~~ 1000 - 480 =520 ~~do not like to play basketball.\nThe~~ percentage ~~of the school that do not like to play basketball is~~ 520/1000 * 100 = 52~~\nThe answer is~~ 52\n

Question: Twenty dozen cups cost $1200 less than the total cost of half a dozen plates sold at $6000 each. Calculate the total cost of buying each cup.

The Generation of LLM-DPC:

To solve the problem, we need to break it down step by step.\n\n1. **Calculate the total cost of half a dozen plates:**\n - Half a dozen plates = 6 plates.\n - Cost of each plate = $6000.\n - Total cost of half a dozen plates = 6 plates \u00d7 $6000/plate = $36,000.\n\n2. **Determine the cost of twenty dozen cups:**\n - According to the problem, twenty dozen cups cost $1200 less than the total cost of half a dozen plates.\n - Therefore, the cost of twenty dozen cups = Total cost of half a dozen plates - $1200.\n - Cost of twenty dozen cups = $36,000 - $1200 = $34,800.\n\n3. **Calculate the number of cups:**\n - Twenty dozen cups = 20 \u00d7 12 = 240 cups.\n\n4. **Calculate the total cost of buying each cup:**\n - Total cost of twenty dozen cups = $34,800.\n - Cost per cup = Total cost / Number of cups = $34,800 / 240 cups.\n\nNow, let's perform the division:\n\n\\[\n\\text{Cost per cup} = \\frac{34,800}{240} = 145.\n\\]\n\nThus, the total cost of buying each cup is **$145**

The Generation of LLMLingua-2:

To solve the problem, we need to break it down step by step.\n\n1. **Calculate the total cost of half a dozen plates:**\n - Half a dozen plates = 6 plates / 2 = 3 plates.\n - Cost of each plate = $6000.\n - Total cost of 3 plates = 3 plates \u00d7 $6000/plate = $18000.\n\n2. **Determine the cost of twenty dozen cups:**\n - According to the problem, twenty dozen cups cost $1200 less than the total cost of half a dozen plates.\n - Therefore, the cost of twenty dozen cups = Total cost of half a dozen plates - $1200.\n - Cost of twenty dozen cups = $18000 - $1200 = $16800.\n\n3. **Calculate the total number of cups:**\n - Twenty dozen cups = 20 \u00d7 12 = 240 cups.\n\n4. **Calculate the total cost of buying each cup:**\n - Total cost of twenty dozen cups = $16800.\n - Cost per cup = Total cost / Total number of cups = $16800 / 240 cups.\n\n5. **Perform the division:**\n - Cost per cup = $16800 / 240 = $70.\n\nThus, the total cost of buying each cup is **$70**

Fig. 3. Cases study on GSM8K dataset in *1-shot constraint*. The red highlights the words that are preserved. The ~~strikethrough~~ highlights the words that are removed.

TABLE IV
EXPERIMENTAL RESULTS FOR THE COMPONENT OF THE REWARD FUNCTION ON THE GSM8K DATASET WITH *1-SHOT CONSTRAINT*

α	β	γ	*1-shot constraint* $EM\uparrow$	Tokens $\downarrow$	$1/\rho\uparrow$
	✓	✓	76.57	339	7.0x
✓		✓	76.70	396	6.0x
✓	✓		76.72	**323**	**7.3x**
✓	✓	✓	**77.03**	343	6.9x

prompt before and after compression. In addition, when $\gamma = 0$, the reward signal lacks attention to the effect of the prompt on the LLM output distribution before and after compression, leading to a decrease in the EM metric. In summary, the reward function we designed takes into account the compression ratio, the KL distribution of the LLM output, and the retention of key information, so that the trained DPC-Agent balances the compression ratio and the performance.

Effect of the ψ in (8): To study the effect of ψ in (8) on the performance of LLM-DPC in the HPC training strategy, we take different values of ψ and conduct experiments in the reasoning task. As shown in Fig. 4, the performance of LLM-DPC is optimal when $\psi = 0.10$. With $\psi = 0.05$, the range of compression ratios is smaller, resulting in a compression ratio of only 4.8x. When $\psi \geq 0.10$, the compression ratios are all in a more appropriate range, but when $\psi = 0.15$ and $\psi = 0.20$, the compression ratios vary slightly more during the training process of HPC, which leads to the difficulty of learning the best compression strategy for the DPC-Agent.

End-to-end efficiency analysis: One of the key objectives of prompt compression is to reduce the overall system inference time. To this end, we evaluate the end-to-end efficiency of different methods on the GSM8K dataset under the 1-shot constraint. Specifically, we report the latency of prompt compression (P. Lat.), the inference latency of the target LLM (I. Lat.), and the overall system latency (S. Lat. = P. Lat. + I. Lat.). As shown in Table V, the results demonstrate that the compression

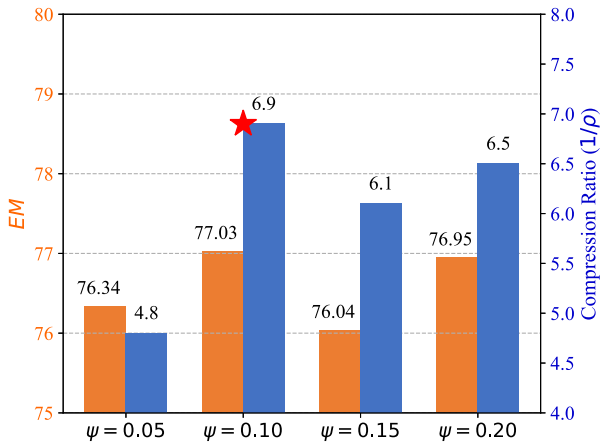

Fig. 4. Experimental results for different values of ψ on the GSM8K dataset with *1-shot constraint*.

TABLE V
END-TO-END EFFICIENCY ANALYSIS ON THE GSM8K DATASET UNDER THE 1-SHOT CONSTRAINT. P. LAT. DENOTES THE LATENCY OF PROMPT COMPRESSION, I. LAT. DENOTES THE INFERENCE LATENCY OF THE TARGET LLM, AND S. LAT. REPRESENTS THE OVERALL SYSTEM LATENCY (P. LAT. + I. LAT.). ALL LATENCY VALUES ARE MEASURED IN SECONDS.

	EM ↑	$1/\rho$ ↑	P. Lat. ↓	I. Lat. ↓	S. Lat. ↓
Selective-Context [21]	76.57	5.4x	0.92	4.77	5.69
LLMLingua [19]	76.72	5.1x	0.96	4.84	5.80
LLMLingua-2-small [13]	75.66	5.6x	**0.43**	4.76	5.19
LLMLingua-2 [13]	76.87	5.7x	1.60	4.52	6.12
LLM-DPC (Ours)	**77.03**	**6.9x**	1.20	**3.89**	**5.09**

TABLE VI
EVALUATION OF INFORMATION RETENTION USING BERTSCORE (BS F1) BETWEEN THE ORIGINAL AND COMPRESSED PROMPTS ON THE ARXIV-MARCH23 AND GSM8K DATASETS

Method	**Arxiv-March23**		**GSM8K**	
	BS F1 ↑	$1/\rho$ ↑	BS F1 ↑	$1/\rho$ ↑
Selective-Context [21]	59.06	11.8x	57.23	13.0x
LLMLingua [19]	59.45	8.7x	**62.63**	13.6x
LLMLingua-2-small [13]	59.61	10.9x	56.99	15.7x
LLMLingua-2 [13]	59.02	12.0x	56.56	**16.9x**
LLM-DPC (Ours)	**70.23**	**12.9x**	60.24	15.5x

overhead (P. Lat.) is modest compared to the savings in LLM inference time. For example, LLM-DPC achieves an overall system latency of 5.09 s, which is substantially lower than Selective-Context (5.69 s) and LLMLingua (5.80 s), while also attaining the best accuracy ($EM = 77.03$) and compression ratio (6.9x). These results confirm that our method indeed reduces the end-to-end system inference time, validating its efficiency in practice.

Evaluation of information retention: One of the key objectives of prompt compression is to minimize information loss while reducing token usage. To this end, we evaluate the information retention capability of different methods using BERTScore (BS F1) [67], which measures semantic similarity between the original and compressed prompts. As shown in Table VI, LLM-DPC attains the highest BS F1 score on Arxiv-March23 and achieves comparable performance on GSM8K, while offering stronger compression ratios than existing baselines. These results confirm that the reward-driven design of LLM-DPC enables it to selectively retain semantically important tokens, thereby effectively reducing information loss even under strong compression.

TABLE VII
EXPERIMENTAL RESULTS FOR DIFFERENT BLACK-BOX LLMS ON THE GSM8K DATASET WITH *HALF-SHOT CONSTRAINT*. CLAUDE IS CLAUDE-3-HAIKU-20240307, AND GLM IS GLM-4-PLUS.

Method	*half-shot constraint*			
	Claude ↑	GLM ↑	Tokens ↓	$1/\rho$ ↑
Selective-Context [21]	69.37	77.79	182	13.0x
LLMLingua [19]	69.22	79.07	174	13.6x
LLMLingua-2-small [13]	68.01	77.63	151	15.7x
LLMLingua-2 [13]	67.85	76.19	**140**	**16.9x**
LLM-DPC (Ours)	**69.44**	**79.76**	153	15.5x

Performance on different target LLMs: To evaluate the generalization ability of LLM-DPC across different target black-box LLMs, we conduct experiments under the half-shot constraint on Claude-3-Haiku-20240307 and GLM-4-Plus. Following the implementation details described in Section V-A4, the DPC-Agent is first trained with a reward function and then directly applied to these target LLMs without additional adaptation. As shown in Table VII, LLM-DPC achieves competitive or superior performance compared with existing baselines, while consistently maintaining higher compression ratios. These results demonstrate that our method can effectively generalize to different target LLMs, confirming its robustness in practical black-box scenarios.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present LLM-DPC, a novel task-agnostic approach for prompt compression in LLMs, aimed at reducing the number of tokens while maintaining output quality. We model the prompt compression task as a Markov Decision Process (MDP), enabling the DPC-Agent to iteratively compress the prompt by removing redundant tokens while preserving essential content, with each decision building on the outcomes of previous steps for efficient, context-aware compression. A carefully designed reward function is introduced to balance compression ratio, output distribution, and key information retention, ensuring effective compression without compromising LLM performance. Furthermore, we propose the Hierarchical Prompt Compression (HPC) training strategy, which employs a progressive training scheme to gradually increase compression difficulty, allowing the agent to learn an efficient compression strategy. We conduct experiments on a variety of downstream tasks, including the conversation task, the summarization task, the reasoning task, and the In-context learning task. Experiments demonstrate that our method performs better at higher compression ratio than state-of-the-art methods.

While the LLM-DPC achieves strong results across multiple tasks and LLMs, there remain several promising directions for further research. First, exploring alternative smoother penalty functions (*e.g.*, linear or sigmoid) in (4) and their interaction with curriculum-based training may provide finer control over

learning dynamics. Second, although we primarily evaluated our method on four benchmarks, extending it to other application domains, such as recommendation or dialogue personalization, would further demonstrate its generality.

REFERENCES

[1] A. Radford et al., "Improving language understanding by generative pre-training," San Francisco, CA, USA, 2018.

[2] T. B. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst. 33: Annu. Conf. Neural Inf. Process. Syst. 2020*, Dec. 2020, pp. 1877–1901.

[3] J. Hu et al., "Test-time learning for large language models," in *Proc. 42nd Int. Conf. Mach. Learn.*, Vancouver, BC, Canada, OpenReview.net, 2025, pp. 24823–24849.

[4] OpenAI, "GPT-4 technical report," 2023, *arXiv:2303.08774*.

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

[6] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," 2023, *arXiv:2307.09288*.

[7] Z. Zhao et al., "Recommender systems in the era of large language models (LLMs)," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 6889–6907, Nov. 2024.

[8] Q. Wang et al., "Generating long-form story using dynamic hierarchical outlining with memory-enhancement," in *Proc. Conf. Nations Americas Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., NAACL 2025 - Vol. 1: Long Papers*, Albuquerque, New Mexico, USA, Association for Computational Linguistics, 2025, pp. 1352–1391.

[9] J. Li et al., "Empowering molecule discovery for molecule-caption translation with large language models: A chatGPT perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 6071–6083, Nov. 2024.

[10] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. Adv. Neural Inf. Process. Syst. 35: Annu. Conf. Neural Inf. Process. Syst.*, New Orleans, LA, USA, 2022, pp. 24824–24837.

[11] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. Adv. Neural Inf. Process. Syst. 33: Annu. Conf. Neural Inf. Process. Syst.*, 2020, pp. 9459–9474.

[12] L. Ouyang et al., "Training language models to follow instructions with human feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 27730–27744.

[13] Z. Pan et al., "Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression," in *Proc. Findings Assoc. Comput. Linguistics*, Bangkok, Thailand, Association for Computational Linguistics, 2024, pp. 963–981.

[14] A. Chevalier, A. Wettig, A. Ajith, and D. Chen, "Adapting language models to compress contexts," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Singapore, Association for Computational Linguistics, 2023, pp. 3829–3846.

[15] J. Mu, X. Li, and N. D. Goodman, "Learning to compress prompts with gist tokens," in *Proc. Adv. Neural Inf. Process. Syst. 36: Annu. Conf. Neural Inf. Process. Syst.*, New Orleans, LA, USA, 2023, pp. 19327–19352.

[16] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis, "Efficient streaming language models with attention sinks," in *Proc. 12th Int. Conf. Learn. Representations*, Vienna, Austria, OpenReview.net, 2024, pp. 1–21.

[17] D. Wingate, M. Shoeybi, and T. Sorensen, "Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models," in *Proc. Findings Assoc. Comput. Linguistics: EMNLP 2022*, 2022, pp. 5621–5634.

[18] H. Jung and K.-J. Kim, "Discrete prompt compression with reinforcement learning," *IEEE Access*, vol. 12, pp. 72578–72587, 2024.

[19] H. Jiang, Q. Wu, C. Lin, Y. Yang, and L. Qiu, "LLMLingua: Compressing prompts for accelerated inference of large language models," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Singapore, Association for Computational Linguistics, 2023, pp. 13358–13376.

[20] H. Jiang et al., "LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression," in *Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics (Vol. 1: Long Papers)*, 2024, pp. 1658–1677.

[21] Y. Li, B. Dong, F. Guerin, and C. Lin, "Compressing context to enhance inference efficiency of large language models," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Singapore, Association for Computational Linguistics, 2023, pp. 6342–6353.

[22] C. E. Shannon, "Prediction and entropy of printed english," *Bell System Tech. J.*, vol. 30, no. 1, pp. 50–64, 1951.

[23] F. Xu, W. Shi, and E. Choi, "RECOMP: Improving retrieval-augmented lms with context compression and selective augmentation," in *Proc. 12th Int. Conf. Learn. Representations*, Vienna, Austria, OpenReview.net, 2024, pp. 1–25.

[24] S. Shandilya et al., "TACO-RL: Task aware prompt compression optimization with reinforcement learning," in *Proc. Findings Assoc. Comput. Linguistics*, 2025, pp. 1582–1597.

[25] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 4555–4576, Sep. 2022.

[26] H. Huang, D. Ye, L. Shen, and W. Liu, "Curriculum-based asymmetric multi-task reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 6, pp. 7258–7269, Jun. 2023.

[27] X. Li et al., "A category-aware curriculum learning for data-free knowledge distillation," *IEEE Trans. Multimedia*, vol. 26, pp. 9603–9618, 2024.

[28] Y. Wang et al., "Enhancing user-oriented proactivity in open-domain dialogues with critic guidance," in *Proc. 34th Int. Joint Conf. Artif. Intell.*, 2025, pp. 8268–8276.

[29] W. Nie, Y. Bao, Y. Zhao, and A. Liu, "Long dialogue emotion detection based on commonsense knowledge graph guidance," *IEEE Trans. Multimedia*, vol. 26, pp. 514–528, 2024.

[30] X. Cai, S. Liu, J. Han, L. Yang, Z. Liu, and T. Liu, "ChestXRayBERT: A pretrained language model for chest radiology report summarization," *IEEE Trans. Multimedia*, vol. 25, pp. 845–855, 2023.

[31] S. Rezayi et al., "Exploring new frontiers in agricultural NLP: Investigating the potential of large language models for food applications," *IEEE Trans. Big Data*, vol. 11, no. 3, pp. 1235–1246, Jun. 2025.

[32] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3580–3599, Jul. 2024.

[33] L. Yang, H. Chen, Z. Li, X. Ding, and X. Wu, "Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3091–3110, Jul. 2024.

[34] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 494–514, Feb. 2022.

[35] L. Hu, Z. Liu, Z. Zhao, L. Hou, L. Nie, and J. Li, "A survey of knowledge enhanced pre-trained language models," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 4, pp. 1413–1430, Apr. 2024.

[36] T. Schick and H. Schütze, "Exploiting Cloze-questions for few-shot text classification and natural language inference," in *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguistics: Main Vol., EACL 2021*, 2021, pp. 255–269.

[37] V. Sanh et al., "Multitask prompted training enables zero-shot task generalization," in *Proc. 10th Int. Conf. Learn. Representations, ICLR 2022, Virtual Event*, 2022, pp. 1–216.

[38] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proc. Conf. Empirical Methods Natural Lang. Process., EMNLP 2021, Virtual Event / Punta Cana*, Dominican Republic, Association for Computational Linguistics, 2021, pp. 3045–3059.

[39] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1951–19535, 2023.

[40] C. Shen, Y.-S. Ong, and J. T. Zhou, "CondenseLM: LLMs-driven text dataset condensation via reward matching," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2025, pp. 1237–1252.

[41] Z. Wan et al., "Efficient large language models: A survey," *Trans. Mach. Learn. Res.*, vol. 2024, pp. 1–67, 2024.

[42] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[43] Z. Yang, H. Qu, M. Fu, W. Hu, and Y. Zhao, "A maximum divergence approach to optimal policy in deep reinforcement learning," *IEEE Trans. Cybern.*, vol. 53, no. 3, pp. 1499–1510, Mar. 2023.

[44] N. Xu et al., "Multi-level policy and reward-based deep reinforcement learning framework for image captioning," *IEEE Trans. Multimedia*, vol. 22, no. 5, pp. 1372–1383, May 2020.

[45] P. Lv et al., "User-guided personalized image aesthetic assessment based on deep reinforcement learning," *IEEE Trans. Multimedia*, vol. 25, pp. 736–749, 2023.

[46] S. Guo et al., "Sample efficient offline-to-online reinforcement learning," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 3, pp. 1299–1310, Mar. 2024.

[47] Y. Zhang, P. Zhao, Q. Wu, B. Li, J. Huang, and M. Tan, "Cost-sensitive portfolio selection via deep reinforcement learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 236–248, Jan. 2022.

[48] J. Hu et al., "Efficient dynamic ensembling for multiple LLM experts," in *Proc. 34th Int. Joint Conf. Artif. Intell.*, 2025, pp. 8095–8103.

[49] R. S. Sutton, D. A. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst. 12*, 1999, pp. 1057–1063.
[50] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017,*arXiv:1707.06347*.
[51] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Proc. Adv. Neural Inf. Process. Syst. 30: Annu. Conf. Neural Inf. Process. Syst. 2017*, Long Beach, CA, USA, 2017, pp. 4299–4307.
[52] B. Ichter et al., "Do as I can, not as I say: Grounding language in robotic affordances," in *Proc. Conf. Robot Learn.*, 2022, pp. 287–318.
[53] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P. Oudeyer, "Grounding large language models in interactive environments with online reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*2023, vol. 202, pp. 3676–3713.
[54] E. Abbe, E. Cornacchia, and A. Lotfi, "Provable advantage of curriculum learning on parity targets with mixed inputs," in *Proc. Adv. Neural Inf. Process. Syst. 36: Annu. Conf. Neural Inf. Process. Syst. 2023*, New Orleans, LA, USA, 2023, pp. 24291–24321.
[55] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.
[56] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," in *Proc. 34th Int. Conf. Mach. Learn.*2017, vol. 70, pp. 1311–1320.
[57] S. Gao et al., "Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, no. 16, pp. 18030–18038.
[58] E. Sayar, G. Iacca, O. S. Oguz, and A. Knoll, "Diffusion-based curriculum reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. 38: Annu. Conf. Neural Inf. Process. Syst.*, 2024, pp. 97587–97617.
[59] J. Zhao, J. Xu, Y. Xu, J. Fang, P. Chao, and X. Zhou, "CCML: Curriculum and contrastive learning enhanced Meta-learner for personalized spatial trajectory prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 9, pp. 4499–4514, Sep. 2024.
[60] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *Proc. Adv. Neural Inf. Process. Syst. 23: 24th Annu. Conf. Neural Inf. Process. Syst. 2010. Proc. a meeting held 6-9 Dec. 2010*, Vancouver, British Columbia, Canada, Curran Associates, Inc., 2010, pp. 1189–1197.
[61] E. A. Platanios, O. Stretcu, G. Neubig, B. Póczos, and T. M. Mitchell, "Competence-based curriculum learning for neural machine translation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, pp. 1162–1172.
[62] T. Xu et al., "Evidence reasoning and curriculum learning for document-level relation extraction," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 2, pp. 594–607, Feb. 2024.
[63] F. Liu, Z. Cheng, L. Zhu, Z. Gao, and L. Nie, "Interest-aware message-passing GCN for recommendation," in *Proc. WWW '21: Web Conf. 2021, Virtual Event / Ljubljana*, 2021, pp. 1296–1305.
[64] F. Liu and L. Nie, *Advancing Recommender Systems With Graph Convolutional Networks*. Berlin, Germany:Springer, 2025.
[65] M. van Otterlo and M. Wiering, *Reinforcement Learning and Markov Decision Processes*. Berlin, Heidelberg:Springer Berlin Heidelberg, 2012, pp. 3–42.
[66] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.
[67] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with BERT," in *Proc. 8th Int. Conf. Learn. Representations*, 2020, pp. 1–43.
[68] T. Sellam, D. Das, and A. P. Parikh, "BLEURT: Learning robust metrics for text generation," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 7881–7892.
[69] P. Laban, A. Hsi, J. F. Canny, and M. A. Hearst, "The summary loop: Learning to write abstractive summaries without examples," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 5135–5150.
[70] D. G. Ghalandari, C. Hokamp, and G. Ifrim, "Efficient unsupervised sentence compression by fine-tuning transformers with reinforcement learning," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics (Vol. 1: Long Papers), ACL 2022*, 2022, pp. 1267–1280.
[71] A. Conneau et al., "Unsupervised cross-lingual representation learning at scale," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, Jul. 2020, pp. 8440–8451.
[72] A. Dubey et al., "The Llama 3 herd of models," 2024, *arXiv:2407.21783*.
[73] K. Cobbe et al., "Training verifiers to solve math word problems," 2021, *arXiv:2110.14168*.
[74] M. Suzgun et al., "Challenging big-bench tasks and whether chain-of-thought can solve them," in *Proc. Findings Assoc. Comput. Linguistics: ACL 2023*, 2023, pp. 13003–13051.
[75] A. Srivastava et al., "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," *Trans. Mach. Learn. Res.*, vol. 2023, 2023. [Online]. Available: https://openreview.net/forum?id=uyTL5Bvosj
[76] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, Philadelphia, PA, USA, ACL, 2002, pp. 311–318.
[77] T. Sellam, D. Das, and A. Parikh, "BLEURT: Learning robust metrics for text generation," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, Jul. 2020, pp. 7881–7892.
[78] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Proc. Text Summarization Branches Out*, 2004, pp. 74–81.
[79] Y. Hu, T. Ganter, H. Deilamsalehy, F. Dernoncourt, H. Foroosh, and F. Liu, "MeetingBank: A benchmark dataset for meeting summarization," in *Proc. 61st Annu. Meeting Assoc. Comput. Linguistics (Vol. 1: Long Papers)*, 2023, pp. 16409–16423.

Jinwu Hu (Graduate Student Member, IEEE) received the MS degree from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2023. He is currently working toward the PhD degree with the South China University of Technology, Guangzhou, China. He has authored or coauthored papers in top venues, including ICML, NeurIPS, CVPR, IJCAI, ACM MM, *IEEE Transactions on Image Processing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Cybernetics*, *IEEE Transactions on Medical Imaging*. His research interests include large language models, computer vision, and reinforcement learning. He was a reviewer for many journals/conferences, including *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ICML, ICLR, CVPR, ECCV.

Wei Zhang received the BS degree from Hubei Polytechnic University, Huangshi, China, in 2022, the MS degree from Hubei University, Wuhan, China. He is currently working toward the PhD degree with the South China University of Technology, Guangzhou, China. His research interests include computer vision, large language models, and reinforcement learning.

Yufeng Wang was born in Hubei, China, in 1999. He received the BEng degree in electronic and information engineering from the Wuhan University of Technology, in 2020, the master's degree in information and communication engineering with the University of Electronic Science and Technology of China, in 2023. He is currently working toward the PhD degree in software engineering with the South China University of Technology. His research interests include large language models and reinforcement learning.

Yu Hu (Member, IEEE) received the Bachelor of Science degree in electrical engineering and automation from the School of Information and Electric Engineering, China University of Mining and Technology, Xuzhou, China, in 2017, and the PhD degree in computer science from the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China, in 2023. He is currently a postdoctoral with the the Hong Kong Polytechnic University (PolyU). His current research endeavors revolve around the domains of high-dimensional data clustering, tensor-based machine learning, graph learning, domain adaptation, and their applications for bioinformatics.

Bin Xiao received the BS and MS degrees in electrical engineering from Shanxi Normal University, Xian, China, in 2004 and 2007, and the PhD degree in computer science from Xidian University, Xian, China. He is currently a professor with the Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include image processing and pattern recognition.

Mingkui Tan (Member, IEEE) received the Bachelor Degree in environmental science and engineering and master's degree in control science and engineering from Hunan University, Changsha, China, in 2006 and 2009, respectively, and the PhD degree in computer science from Nanyang Technological University, Singapore, in 2014. He is currently a professor with the School of Software Engineering, South China University of Technology. From 2014 to 2016, he was a senior research associate on computer vision with the School of Computer Science, University of Adelaide, Australia. He was an associate editor for *IEEE Transactions on Pattern Analysis and Machine Intelligence*. His research interests include machine learning, sparse analysis, deep learning, and large-scale optimization.

Qing Du received the BS degree in computer science and technology, the master's degree in computer application, and the PhD degree in computer application from the South China University of Technology, China, in 2002, 2005, and 2014, respectively, where she is currently an associate professor with the School of Software Engineering. Her research interests include information retrieval, recommendation systems, natural language processing, and deep learning.