

Dynamic Compressing Prompts for Efficient Inference of Large Language Models

Jinwu Hu*, Wei Zhang*, Yufeng Wang, Yu Hu, Bin Xiao, *Senior Member, IEEE*
 Mingkui Tan†, *Senior 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 Compressing Prompts (LLM-DCP). Our method reduces the number of prompt tokens while aiming to preserve the performance as much as possible. We model prompt compression as a Markov Decision Process (MDP), enabling the DCP-Agent to sequentially remove redundant tokens by adapting to dynamic contexts and retaining crucial content. We develop a reward function for training the DCP-Agent that balances the compression rate, 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 DCP-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 rates. The code for our approach will be available at <https://github.com/Fhujinwu/DCP>.

Index Terms—Large language models, Prompt Compression, Markov decision process, Curriculum Learning

I. INTRODUCTION

LARGE Language Models (LLMs) [1–6] have shown excellent performance in different tasks, including recommender systems [7] and drug design [8]. Many recently emerged prompting techniques for LLMs, such as Chain of Thought (CoT) [9], Retrieval Augmented Generation (RAG)

This work was partly supported by the National Natural Science Foundation of China under Grant 62072190.

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

Yufeng Wang is with the School of Future Technology, South China University of Technology, Guangzhou, China, and with Peng Cheng Laboratory, Shenzhen, 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, China (e-mail: jasonscut@outlook.com).

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

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

*Authors contributed equally. †Corresponding authors

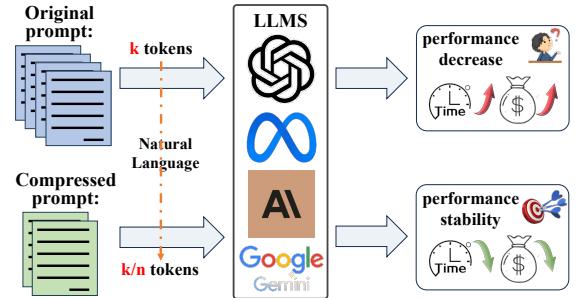


Fig. 1. Motivation for Prompt Compression of LLMs.

[10], Role-playing [11], 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 [12] (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 LLM 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 [12–20] have been explored, which can be broadly classified into white-box and black-box methods. The white-box compression method [13–16] compresses the prompt at the token-embedding level by modifying the model parameters, structure, and transformer self-attention mechanism. However, most high-performing LLMs (such as GPT-4 and Claude-3) are accessed through application programming interfaces (APIs), and the unavailability of source code severely limits the development and application of these methods. In response to the limited access to the source

code of LLMs, black-box compression methods have emerged [12, 17–20], leveraging the inherent redundancy of natural language [21]. The black-box compression method operates at the natural language level, aiming to shorten the original prompt without losing essential information. This method does not require access to the LLM source code for training or inference, reducing usage costs by directly minimizing input size. It also shortens inference time while preserving the performance of the LLM output.

Despite the recent black-box compression methods that can reduce the number of tokens in the prompt while maintaining the LLM output performance as much as possible, these methods still face certain limitations. **Firstly**, a common drawback of some existing task-aware compression methods [17, 19, 22, 23] is that they are usually fine-tuned for specific tasks, and thus often difficult to use for different downstream task. For example, LongLLMLingua [19] 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 [12, 18, 20] 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 existing methods heavily depend on black-box LLMs during training, either for providing reward signals [17, 23] or generating large-scale labeled data [12], leading to high training costs and limited practicality.

To address the above limitations, we propose a novel task-agnostic **Dynamic Compressing Prompts** method, called **LLM-DCP**, reducing the number of tokens of 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 can be viewed 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 DCP-Agent to sequentially remove redundant tokens by adapting to dynamic contexts and retaining crucial content. Furthermore, we design a reward function for training the DCP-Agent that balances the compression rate, 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 [24–26], 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-DCP achieves approximately a 3.04% improvement 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 DCP-Agent, we design a reward function that balances compression rate, output quality, and retention of key information. This reward function operates without direct supervision from the target LLM, significantly 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-DCP. Section V provides the experiments and discussions. The conclusion of this paper is in Section VI.

II. RELATED WORK

In this section, we focus on the content closely related to our work, which contains Large Language Models, Prompt Compression, and Reinforcement Learning.

A. Large Language Models

Large language models (LLMs) [1–6, 11], such as the GPT series [1, 2, 4, 11], have received significant attention for their excellent generalization and comprehension capabilities in natural language processing (NLP) such as multi-round dialogue [4, 27], document summarisation [28], and question answering [29]. A line of studies has attempted to enhance further the ability of LLMs to solve complex scenario tasks. Pan et al. [30] and Yang et al. [31] propose to use Knowledge Graph (KG) [32, 33] to enhance the reasoning power and interpretability of LLM. Wei et al. [9] 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. [10] 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, 34, 35], directly influencing the efficiency and performance

of LLMs. As prompts grow longer, prompt compression has emerged as a crucial area of research [36, 37], with the goal of reducing LLM reasoning time and computational cost while maintaining performance. Prompt compression is typically categorized into two types: the white-box method and the black-box method [38].

The **white-box compression method** [13–16] reduces the prompt at the token-embedding level by modifying model parameters, architecture, and the self-attention mechanism of the Transformer. Chevalier et al. [13] 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. [14] 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. [15] propose StreamingLLM, a framework that enables large language models to handle infinite sequence lengths by utilizing attention sinks. Wingate et al. [16] proposed to learn a soft prompt compressed the context by aligning soft prompt-based model prediction with context-based predictive alignment. However, most high-performing LLMs (such as GPT-4 and Claude-3) 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 method** [12, 17–20] operates at the natural language level, aiming to shorten the original prompt without losing essential information. Li et al. [20] 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. [18] 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. [12] create a task-agnostic data distillation procedure for better generalizability and efficiency. Jiang et al. [19] propose LongLLMLingua to improve LLMs perception of key information for accelerated compression. Jung et al. [17] employ a computationally efficient policy network that directly edits prompts. This type of method does 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) [39–45] 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. proposed policy gradient [46] algorithms, it puts the current state s_t into the policy network π and outputs the current action a_t , the policy network $\pi_\theta(a | s)$ is used directly to represent and control the behavior of an agent. Schulman et al. proposed proximal policy optimization [47] algorithms, where both the policy network π 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. [11] point out that using reinforcement learning from human feedback (RLHF) [48] can enable the model to follow a broad class of written instructions. Brohan et al. [49] 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. [50] 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.

III. PROBLEM DEFINITION AND MOTIVATIONS

A. Problem Definition

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

$$\min_{\tilde{\mathbf{x}}} KL(P(\tilde{\mathbf{x}}_G|\tilde{\mathbf{x}}), P(\mathbf{x}_G|\mathbf{x})) + \rho, \quad (1)$$

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. Each compression step should reduce redundant information while leveraging the outcomes of previous steps to achieve efficient compression progressively. 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 DCP-Agent iteratively compresses

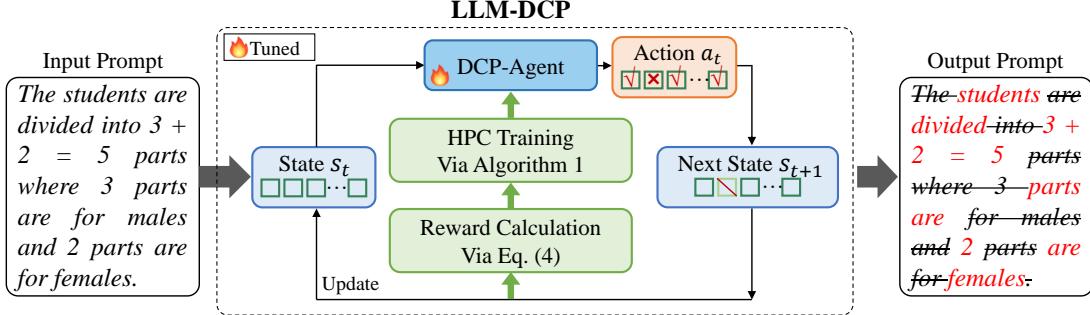


Fig. 2. General diagram of proposed LLM-DCP. We model prompt compression as a Markov Decision Process (MDP) and train a DCP-Agent to determine an optimal compression pathway. The input prompt represented as a token sequence serves as the initial state of the MDP. At time step t , the DCP-Agent performs 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 Eq. (4). Our designed hierarchical prompt compression (HPC) training strategy collects the trajectory, which is applied to train the DCP-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.

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 rate, 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 high compression rates in the prompt compression task, we incorporate curriculum learning [24, 25, 51], 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 compressing prompts method, called LLM-DCP, which seeks to remove redundant content in a given input prompt, thereby reducing computational cost and better using the limited context window in LLMs. As shown in Fig. 2. We model the prompt compression process as a Markov Decision Process (MDP) and train a DCP-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 DCP-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 Eq. (4). The trajectory is collected to train the DCP-Agent via our designed hierarchical prompt compression (HPC) training strategy. Additionally, the next state is input to the DCP-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 Compressing Prompts as an MDP

We seek a general DCP-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 Markov Decision Process

(MDP) [52]: $\langle S, A, \mathcal{T}, \mathcal{R}, \pi \rangle$. The state space of the environment is S and the action space of the agent is A . At time step t , the agent takes the state $s_t \in S$ as input and performs an action $a_t \in A$ through the policy network $\pi : S \times A \rightarrow [0, 1]$. The environment changes to the next state $s_{t+1} = \mathcal{T}(s_t, a_t)$ according to the transition function \mathcal{T} and a reward $r_t = \mathcal{R}(s_t, a_t)$ is received with reward function \mathcal{R} . In this work, the MDP is detailed as follows:

States S is the description for the environment. At time step t , the state is a compressed prompt: $s_t = \tilde{x}_{t-1} = \{x_i\}_{i=1}^{\tilde{L}_{t-1}}$, where \tilde{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 A is a discrete set of actions the agent can take. In this task, the action space $A = \{0, 1\}^n$ is labeled for each token, with 0 indicating removal and 1 indicating preservation. At time step t , the agent gives the action $a_t \in A$ based on the state s_t to remove redundant tokens.

Transition $\mathcal{T}(S, A)$ is a function $\mathcal{T} : S \times A \rightarrow S$ which maps a state s_t into a new state s_{t+1} . When the maximum trajectory length is reached, this episode will terminate 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 as:

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

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

Rewards $\mathcal{R}(s_t, a_t)$ 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 subsection IV-B.

Policy $\pi_\theta(a | s) : S \times A \rightarrow [0, 1]$ 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 $a_t \in A = \{0, 1\}^n$:

$$\pi(a_t | s_t; \theta) = \frac{\exp \{f_\theta(s_t)_i\}}{\sum_{j=1}^N \exp \{f_\theta(s_t)_j\}}, \quad (3)$$

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.

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 Eq. (1). Therefore, we design a reward function that takes into account the compression ratio, the Kullback-Leibler (KL) divergence [53] of the LLM-generated result distribution, and the degree of retention of key information from the prompt. The reward function is as follows:

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

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 [54] is used, 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. 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 subsection IV-D for details.

Remark: Unlike existing reinforcement learning-based summarization methods [55, 56], the reward function we designed without considering 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 [12, 18, 19]. Disregarding the fluency and grammar of the prompt is beneficial for obtaining a higher compression rate. In addition, the reward function we design does not require the involvement of a black-box LLM, which is different from the existing method [17, 23].

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 [24, 25], we propose Hierarchical Prompt Compression (called **HPC**) training strategy for Proximal Policy Optimization (PPO) [47] 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 $x = \{x_i\}_{i=1}^L$. To utilize the bidirectional contextual information of each token, we utilize the Transformer encoder as a feature extractor and then

Algorithm 1 The HPC Training for LLM-DCP

```

Input: The prompt for compression dataset  $\mathcal{D}$ , the DCP-Agent  $\pi_\theta$ , the critic  $V_\phi$  the reply 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  $c_l$ .
1: Initialize buffer  $\mathcal{B}$ , actor parameters  $\theta$  and critic parameters  $\phi$ .
2: while Not convergence do
3:   for  $P_i$  in  $P$  do
4:     Calculate  $c_s$  and  $c_l$  via Eq.(8).
5:     for  $x_i$  in  $\mathcal{D}$  do
6:       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  $V_{\phi_{old}}$ .
7:       Put  $\tau$  into  $\mathcal{B}$ .
8:     if  $length(\mathcal{B}) == M$  then
9:       for  $iteration = 1, 2, \dots, M$  do
10:        Uniformly sample  $\tau \in \mathcal{B}$ .
11:        Calculate  $\mathcal{J}(\theta)$  via Eq.(7).
12:        Update  $\theta$  to maximize  $\mathcal{J}(\theta)$ .
13:        Calculate TD error via Eq. (9).
14:        Update  $\phi$  to minimize TD error  $\delta_t$ .
15:      end for
16:      Empty the replay buffer  $\mathcal{B}$ .
17:      Update  $\theta_{old} \leftarrow \theta$ .
18:      Update  $\phi_{old} \leftarrow \phi$ .
19:    end if
20:  end for
21: end for
22: end while

```

send the features to a linear classification layer. Specifically, at time step t , the state $s_t = \tilde{x}_{t-1} = \{x_i\}_{i=1}^{\tilde{L}_{t-1}}$ contains \tilde{L}_{t-1} tokens, which can be formalized as:

$$\mathbf{h} = f_\theta(\tilde{x}_{t-1}), \quad (5)$$

$$p(x_i, \theta) = \text{softmax}(W\mathbf{h}_i + b), \quad (6)$$

where $\mathbf{h} = \{h_i\}_{i=1}^{\tilde{L}_{t-1}}$ 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* [57] 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 $V_\phi(s)$ is used to estimate the expected return of the state s_t and calculate the advantage, which can aid the actor in learning more efficiently and stably. Similar to the actor, the critic is composed of a pre-trained *xlm-roberta-large* [57] 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 $\mathcal{J}(\theta)$:

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

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

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/ρ ↑
ShareGPT									
Selective-Context [20]	EMNLP'2023	38.53	-0.21	51.27	38.35	43.51	78.30	183	3.3x
LLMLingua[18]	EMNLP'2023	38.71	-0.21	51.43	38.62	43.57	78.27	186	3.2x
LLMLingua-2-small [12]	ACL'2024	56.79	0.37	76.09	58.47	63.56	89.54	191	3.1x
LLMLingua-2 [12]	ACL'2024	61.97	0.47	78.64	63.07	67.50	90.87	184	3.3x
LLM-DCP (Ours)	-	64.93	0.54	80.24	65.54	69.89	91.80	175	3.4x
Arxiv-March23									
Selective-Context [20]	EMNLP'2023	8.83	-0.61	43.43	13.46	18.92	73.75	933	11.8x
LLMLingua[18]	EMNLP'2023	5.70	-0.74	32.29	8.78	15.17	69.60	1276	8.7x
LLMLingua-2-small [12]	ACL'2024	8.56	-0.45	45.52	<u>15.47</u>	<u>21.09</u>	<u>75.49</u>	1017	10.9x
LLMLingua-2 [12]	ACL'2024	10.84	-0.57	48.49	14.62	19.95	75.15	920	12.0x
LLM-DCP (Ours)	-	<u>10.10</u>	<u>-0.55</u>	48.81	15.94	21.63	75.91	855	12.9x

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 DCP-Agent incrementally by gradually decreasing the compression rate range $[c_s, c_l]$ (see Eq. 4) and the maximum trajectory length T_{max} in stage (P_i). The c_s and c_l are adjusted as follows:

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

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. We then put τ into the reply 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's parameters V_ϕ :

$$\delta_t = G_t - V_\phi(s_t), \quad (9)$$

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

Due to the fact that the target black-box LLMs (e.g., GPT-4o-mini) are not available for the resulting distribution $P(\tilde{x}_G)$

generated by the compressed prompt \tilde{x} , we align a small model with the distribution of the target LLMs by instruction fine-tuning. Specifically, we use a pre-trained small language model M_s for instruction tuning using the data pairs generated by black-box LLM of the target family. The optimization process of M_s can be formulated as follows:

$$\min_{\theta_{M_s}} \mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N \mathcal{L}(\mathbf{x}_i, \mathbf{y}_{i,LLM}; \theta_{M_s}) \right], \quad (10)$$

where θ_{M_s} is the parameters of M_s , $(\mathbf{x}_i, \mathbf{y}_{i,LLM})$ is the pair of instruction \mathbf{x}_i and the black-box LLM generated texts $\mathbf{y}_{i,LLM}$, and N is the number of all examples used for instruction tuning. Notably, the Llama 3-8B [58] is selected for M_s .

V. EXPERIMENT

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

A. Experimental Settings

1) *Compared methods:* Following the previous working setup [12], we compare the proposed LLM-DCP with only three SOTA task-agnostic prompt compression methods.

- Selective-Context [20]: 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 $(x_t, \dots, x_{t+\alpha})$), retaining lexical unit self-information over a threshold value.
- LLMLingua [18]: 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 $S = \{s_1, s_2, \dots, s_m\}$, where tokens greater than threshold in each segment are retained.

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

Method	Pub.'Year	$EM \uparrow$	1-shot constraint		half-shot constraint		
			Tokens \downarrow	$1/\rho \uparrow$	$EM \uparrow$	Tokens \downarrow	$1/\rho \uparrow$
GSM8K							
Selective-Context [20]	EMNLP'2023	76.57	436	5.4x	76.15	182	13.0x
LLMLingua[18]	EMNLP'2023	76.72	462	5.1x	<u>77.02</u>	174	13.6x
LLMLingua-2-small [12]	ACL'2024	75.66	425	5.6x	<u>76.80</u>	<u>151</u>	<u>15.7x</u>
LLMLingua-2 [12]	ACL'2024	<u>76.87</u>	<u>415</u>	<u>5.7x</u>	76.80	140	16.9x
LLM-DCP (Ours)	-	77.03	343	6.9x	77.03	153	15.5x
BBH							
Selective-Context [20]	EMNLP'2023	<u>82.81</u>	278	2.8x	81.91	152	5.1x
LLMLingua[18]	EMNLP'2023	81.68	271	2.9x	84.72	162	4.8x
LLMLingua-2-small [12]	ACL'2024	82.73	274	2.8x	82.12	155	5.0x
LLMLingua-2 [12]	ACL'2024	82.41	<u>255</u>	<u>3.0x</u>	82.64	145	5.3x
LLM-DCP (Ours)	-	83.16	251	3.1x	<u>83.98</u>	145	5.3x

- LLMLingua-2 [12]: 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-DCP, 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. [20], which includes only the first two sections of each article to avoid excessive length.
- ShareGPT: A dataset of 90k conversations collected from sharegpt.com¹, involving multiple rounds of dialogue between users and ChatGPT in multiple languages and scenarios. We test the conversation task using sharegpt575 [20], which contains 575 multi-round dialogue examples.
- GSM8K [59]: A widely used dataset for testing logic and mathematics in language modeling, containing 8.5k high-quality linguistically diverse mathematical problems.
- BBH [60]: A subset of the BIG-Bench dataset [61], 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 [18], we use BLEU [62], BLEURT [63], ROUGE [64] and BERTScore (BS F1) [54] as evaluation metrics for Arxiv-March and ShareGPT. We use Exact Match (EM) [18] 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 Llama3² to calculate the number of tokens.

4) *Implementation Details*: Our proposed LLM-DCP is implemented using PyTorch framework with Pytorch version 2.1.2 and runs on the 80G memory-sized 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 Eq. 4 are set to 200 and 100, respectively.

For the training of model M_s in subsection IV-D, we use the alpaca-gpt4-data³ 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-Factory⁴. 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 DCP-Agent. During the testing phase, we control the compression rate (e.g. 3x or 10x) by controlling the maximum step size. We employ the GPT-4o-mini-2024-07-18⁵ 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-DCP 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-DCP in the conversation task. As shown in Table I, LLM-DCP outperforms other SOTA methods in the conversation task. Specifically, compared to LLMLingua-2, the proposed LLM-DCP improves about 4.8% (61.97→64.93) on BLEU and about 1.0% (90.87→91.80) on BS F1 at higher compression ratio (3.3x

¹<https://sharegpt.com/>

²<https://huggingface.co/meta-llama/Meta-Llama-3-8B>

³<https://huggingface.co/datasets/llm-wizard/alpaca-gpt4-data>

⁴<https://github.com/hiyouga/LLaMA-Factory>

⁵<https://platform.openai.com/>

Compressed Prompt by LLM-DCP:

.....(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?~~ Let's think step by step. The students are divided into $3 + 2 = 5$ parts where ~~3 parts are for males and 2 parts are for females~~. Each part represents $1000/5 = 200$ students. So, there are $3 \times 200 = 600$ males. And there are $2 \times 200 = 400$ females. Hence, $600 \times 2/3 = 400$ males play basketball. And $400 \times 1/5 = 80$ females play basketball. A total of $400 + 80 = 480$ students play basketball. Therefore, $1000 - 480 = 520$ do not like to play basketball. The percentage of the school that do not like to play basketball is $520/1000 * 100 = 52$. The answer is 52.

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?~~ Let's think step by step. The students are divided into $3 + 2 = 5$ parts where ~~3 parts are for males and 2 parts are for females~~. Each part represents $1000/5 = 200$ students. So, there are $3 \times 200 = 600$ males. And there are $2 \times 200 = 400$ females. Hence, $600 \times 2/3 = 400$ males play basketball. And $400 \times 1/5 = 80$ females play basketball. A total of $400 + 80 = 480$ students play basketball. Therefore, $1000 - 480 = 520$ do not like to play basketball. The percentage of the school that do not like to play basketball is $520/1000 * 100 = 52$. The answer is 52.

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-DCP:

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



The Generation of LLMLingua-2:

To solve the problem, we need to break it down step by step.
 n1. **Calculate the total cost of half a dozen plates:
 - Half a dozen plates = 6 plates / 2 = 3 plates.
 - Cost of each plate = \$6000.
 - Total cost of 3 plates = $3 \times 6000 = \$18,000$.
 n2. **Determine the cost of twenty dozen cups:
 - According to the problem, twenty dozen cups cost \$1200 less than the total cost of half a dozen plates.
 - Therefore, the cost of twenty dozen cups = Total cost of half a dozen plates - \$1200.
 - Cost of twenty dozen cups = $\$18,000 - \$1200 = \$16,800$.
 n3. **Calculate the total number of cups:
 - Twenty dozen cups = $20 \times 12 = 240$ cups.
 n4. **Calculate the total cost of buying each cup:
 - Total cost of twenty dozen cups = $\$16,800 / 240 = \70 .
 Now, let's perform the division:
 $\text{Cost per cup} = \frac{\$16,800}{240} = \$70$.
 Thus, the total cost of buying each cup is $\boxed{\$70}$.



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.

→3.4x). The proposed LLM-DCP achieves a 17.0% relative improvement over the classical method, Selective-Context, on the BLEU metric. We conclude that LLM-DCP 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-DCP in the summarization task. As shown in Table I, our proposed LLM-DCP outperforms SOTA methods in the summarization task. Specifically, the proposed LLM-DCP achieves a relative improvement of 9.03% (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-DCP is not optimal in BLEU metric compared to LLMLingua-2, the main reason is that our DCP-Agent is trained on conversation data, while LLMLingua-2 is trained on the summarization task dataset, MeetingBank [65]. Meanwhile, it exactly proves that the

proposed LLM-DCP still achieves better prompt compression performance in the cross-task situation.

LLM-DCP trade-off between performance and compression ratio. As shown in Table II, the proposed LLM-DCP outperforms the SOTA method in the reasoning task. Specifically, with the *1-shot constraint*, our proposed LLM-DCP 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-DCP 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-DCP 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 rate, a possible factor is that the target black-

TABLE III

ABLATION STUDY ON THE GSM8K DATASET WITH *1-shot constraint*.

Version	<i>1-shot constraint</i>		
	<i>EM</i> \uparrow	Tokens \downarrow	$1/\rho \uparrow$
Random	76.04	428	5.5x
LLM-DCP (w/o Training)	76.19	422	5.6x
LLM-DCP (w/o HPC)	76.57	431	5.5x
LLM-DCP (Ours)	77.03	343	6.9x

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-DCP in In-context learning task. As shown in Table II, the proposed LLM-DCP outperforms the SOTA method in the *EM* metric at a higher compression ratio. Specifically, with the *1-shot constraint*, the proposed LLM-DCP achieves a relative improvement of about 1.0% (82.41 \rightarrow 83.16) in *EM* metric compared to LLMLingua-2, along with a relative improvement of 3.3% (3.0x \rightarrow 3.1x) in compression ratio. With *half-shot constraint*, the proposed LLM-DCP improves the *EM* metric by a relative 1.6% (82.64 \rightarrow 83.98) compared to LLMLingua-2 while maintaining the same compression ratio.

Overall, our proposed LLM-DCP is a task-agnostic prompt compression method that achieves to outperform the SOTA method 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 DCP-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-DCP

We show an example of LLM-DCP and LLMLingua-2 on a reasoning task to demonstrate the effect of prompt compression, as shown in Fig.3. The LLM-DCP and LLMLingua-2 are both tokens-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, our proposed LLM-DCP retains more key information, which makes the prompts obtained after LLM-DCP 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 GSM8K dataset.

Effectiveness of prompt compression with MDP. We compare LLM-DCP 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, the proposed LLM-DCP achieves a relative

TABLE IV

EXPERIMENTAL RESULTS FOR THE COMPONENT OF THE REWARD FUNCTION ON THE GSM8K DATASET WITH *1-shot constraint*.

α	β	γ	<i>1-shot constraint</i>		
			<i>EM</i> \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

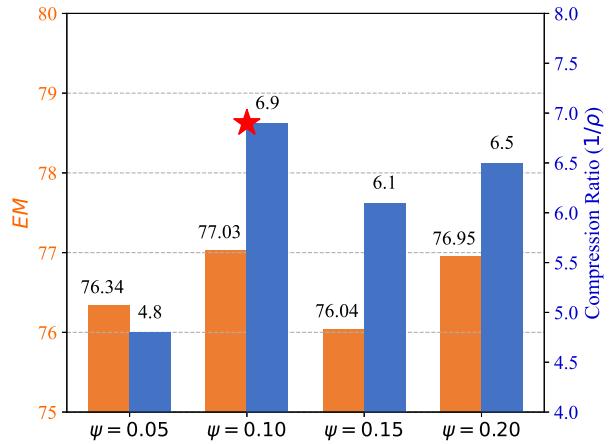


Fig. 4. Experimental results for different values of ψ on the GSM8K dataset with *1-shot constraint*.

improvement of 1.3% (76.04 \rightarrow 77.03) in *EM* metric and a relative improvement of 25.5% (5.5x \rightarrow 6.9x) in compression ratio. A primary reason is the modeling of prompt compression as MDP, the trained DCP-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.

Effectiveness of HPC training strategy. We compare LLM-DCP and LLM-DCP (w/o HPC) to verify the effectiveness of the proposed Hierarchical Prompt Compression training strategy, as shown in Table III. Compared to LLM-DCP (w/o HPC), LLM-DCP has a relative improvement of 0.6% (76.57 \rightarrow 77.03) in *EM* metrics and a relative improvement of 25.5% (5.5x \rightarrow 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 DCP-Agent to better learn how to remove the redundant tokens in the dynamic prompt input.

E. Discussion

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

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-DCP by adjusting $\alpha = 0$, $\beta = 0$ or $\gamma = 0$

TABLE V
EXPERIMENTAL RESULTS FOR GLM-4-PLUS ON THE GSM8K DATASET
WITH *half-shot constraint*.

Method	Pub.'Year	<i>half-shot constraint</i>		
		<i>EM</i> ↑	Tokens ↓	$1/\rho$ ↑
Selective-Context [20]	EMNLP'2023	77.79	182	13.0x
LLMLingua[18]	EMNLP'2023	79.07	174	13.6x
LLMLingua-2-small [12]	ACL'2024	77.63	151	15.7x
LLMLingua-2 [12]	ACL'2024	76.19	140	16.9x
LLM-DCP (Ours)	-	79.76	153	15.5x

in Eq. 4. As shown in Table IV, when $\alpha = 0$, the lack of compression ratio of the reward signal may cause the DCP-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 DCP-Agent not to pay attention to the key information retention of the 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 DCP-Agent balances the compression ratio and the performance.

Effect of the ψ in Eq. 8. To study the effect of ψ in Eq. 8 on the performance of LLM-DCP in the proposed 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-DCP 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 DCP-Agent.

Performance on different target LLMs. We study the performance of the proposed LLM-DCP and SOTA methods on GLM-4-Plus⁶ with the GSM8K dataset. As shown in Table V, the comparison results of our proposed LLM-DCP and SOTA methods on *EM* metric under *half-shot constraint* present consistency with the target LLM as GPT-4o-min. Therefore, LLM-DCP is applicable to different black-box LLMs.

VI. CONCLUSION

In this paper, we present LLM-DCP, a novel task-agnostic approach for prompt compression in Large Language Models (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 DCP-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 design

reward function is introduced to balance compression rate, 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 rates than state-of-the-art methods.

REFERENCES

- [1] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, “Improving language understanding by generative pre-training,” 2018.
- [2] T. B. Brown, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.
- [3] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin *et al.*, “Opt: Open pre-trained transformer language models,” *arXiv preprint arXiv:2205.01068*, 2022.
- [4] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, “Gpt-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.
- [5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.
- [6] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahaire, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, “Llama 2: Open foundation and fine-tuned chat models,” *arXiv preprint arXiv:2307.09288*, 2023.
- [7] Z. Zhao, W. Fan, J. Li, Y. Liu, X. Mei, Y. Wang, Z. Wen, F. Wang, X. Zhao, J. Tang, and Q. Li, “Recommender systems in the era of large language models (llms),” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 11, pp. 6889–6907, 2024.
- [8] J. Li, Y. Liu, W. Fan, X.-Y. Wei, H. Liu, J. Tang, and Q. Li, “Empowering molecule discovery for molecule-caption translation with large language models: A chatgpt perspective,” *IEEE Transactions on Knowledge and Data Engineering*, 2024.
- [9] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.
- [10] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
- [11] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, “Training language models to follow instructions with human feedback,” *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.
- [12] Z. Pan, Q. Wu, H. Jiang, M. Xia, X. Luo, J. Zhang, Q. Lin, V. Rühle, Y. Yang, C.-Y. Lin *et al.*, “Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression,” *Proceedings of the 62th Annual Meeting of the Association for Computational Linguistics*, 2024.
- [13] A. Chevalier, A. Wettig, A. Ajith, and D. Chen, “Adapting language models to compress contexts,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 3829–3846.
- [14] J. Mu, X. Li, and N. Goodman, “Learning to compress prompts with gist tokens,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [15] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis, “Efficient streaming language models with attention sinks,” *The Twelfth International Conference on Learning Representations*, 2023.
- [16] D. Wingate, M. Shoeybi, and T. Sorensen, “Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models,” in *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022, pp. 5621–5634.
- [17] H. Jung and K.-J. Kim, “Discrete prompt compression with reinforcement learning,” *IEEE Access*, 2024.

⁶<https://bigmodel.cn>

- [18] H. Jiang, Q. Wu, C.-Y. Lin, Y. Yang, and L. Qiu, “Llmlingua: Compressing prompts for accelerated inference of large language models,” *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- [19] H. Jiang, Q. Wu, X. Luo, D. Li, C.-Y. Lin, Y. Yang, and L. Qiu, “Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression,” *Proceedings of the 62th Annual Meeting of the Association for Computational Linguistics*, 2024.
- [20] Y. Li, B. Dong, F. Guerin, and C. Lin, “Compressing context to enhance inference efficiency of large language models,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 6342–6353.
- [21] C. E. Shannon, “Prediction and entropy of printed english,” *Bell system technical journal*, vol. 30, no. 1, pp. 50–64, 1951.
- [22] F. Xu, W. Shi, and E. Choi, “Recomp: Improving retrieval-augmented lms with context compression and selective augmentation,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [23] S. Shandilya, M. Xia, S. Ghosh, H. Jiang, J. Zhang, Q. Wu, and V. Rühle, “Taco-rl: Task aware prompt compression optimization with reinforcement learning,” *arXiv preprint arXiv:2409.13035*, 2024.
- [24] X. Wang, Y. Chen, and W. Zhu, “A survey on curriculum learning,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 4555–4576, 2021.
- [25] H. Huang, D. Ye, L. Shen, and W. Liu, “Curriculum-based asymmetric multi-task reinforcement learning,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 6, pp. 7258–7269, 2022.
- [26] X. Li, L. Jiao, Q. Sun, F. Liu, X. Liu, L. Li, P. Chen, and S. Yang, “A category-aware curriculum learning for data-free knowledge distillation,” *IEEE Transactions on Multimedia*, vol. 26, pp. 9603–9618, 2024.
- [27] W. Nie, Y. Bao, Y. Zhao, and A. Liu, “Long dialogue emotion detection based on commonsense knowledge graph guidance,” *IEEE Transactions on Multimedia*, vol. 26, pp. 514–528, 2024.
- [28] X. Cai, S. Liu, J. Han, L. Yang, Z. Liu, and T. Liu, “Chestxraybert: A pretrained language model for chest radiology report summarization,” *IEEE Transactions on Multimedia*, vol. 25, pp. 845–855, 2023.
- [29] S. Rezayi, Z. Liu, Z. Wu, C. Dhakal, B. Ge, H. Dai, G. Mai, N. Liu, C. Zhen, T. Liu, and S. Li, “Exploring new frontiers in agricultural nlp: Investigating the potential of large language models for food applications,” *IEEE Transactions on Big Data*, pp. 1–12, 2024.
- [30] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, “Unifying large language models and knowledge graphs: A roadmap,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3580–3599, 2024.
- [31] 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 Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3091–3110, 2024.
- [32] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, “A survey on knowledge graphs: Representation, acquisition, and applications,” *IEEE transactions on neural networks and learning systems*, vol. 33, no. 2, pp. 494–514, 2021.
- [33] L. Hu, Z. Liu, Z. Zhao, L. Hou, L. Nie, and J. Li, “A survey of knowledge enhanced pre-trained language models,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 4, pp. 1413–1430, 2024.
- [34] T. Schick and H. Schütze, “Exploiting cloze-questions for few-shot text classification and natural language inference,” in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 255–269.
- [35] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. Le Scao, A. Raja *et al.*, “Multitask prompted training enables zero-shot task generalization,” in *International Conference on Learning Representations*, 2022.
- [36] B. Lester, R. Al-Rfou, and N. Constant, “The power of scale for parameter-efficient prompt tuning,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 3045–3059.
- [37] 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 Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.
- [38] Z. Wan, X. Wang, C. Liu, S. Alam, Y. Zheng, J. Liu, Z. Qu, S. Yan, Y. Zhu, Q. Zhang, M. Chowdhury, and M. Zhang, “Efficient large language models: A survey,” *Transactions on Machine Learning Research*, 2024, survey Certification. [Online]. Available: <https://openreview.net/forum?id=bsCCJHbO8A>
- [39] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.
- [40] Z. Yang, H. Qu, M. Fu, W. Hu, and Y. Zhao, “A maximum divergence approach to optimal policy in deep reinforcement learning,” *IEEE Transactions on Cybernetics*, vol. 53, no. 3, pp. 1499–1510, 2023.
- [41] N. Xu, H. Zhang, A.-A. Liu, W. Nie, Y. Su, J. Nie, and Y. Zhang, “Multi-level policy and reward-based deep reinforcement learning framework for image captioning,” *IEEE Transactions on Multimedia*, vol. 22, no. 5, pp. 1372–1383, 2020.
- [42] P. Lv, J. Fan, X. Nie, W. Dong, X. Jiang, B. Zhou, M. Xu, and C. Xu, “User-guided personalized image aesthetic assessment based on deep reinforcement learning,” *IEEE Transactions on Multimedia*, vol. 25, pp. 736–749, 2023.
- [43] S. Guo, L. Zou, H. Chen, B. Qu, H. Chi, P. S. Yu, and Y. Chang, “Sample efficient offline-to-online reinforcement learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 3, pp. 1299–1310, 2024.
- [44] Y. Zhang, P. Zhao, Q. Wu, B. Li, J. Huang, and M. Tan, “Cost-sensitive portfolio selection via deep reinforcement learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 236–248, 2022.
- [45] J. Hu, Y. Wang, S. Zhang, K. Zhou, G. Chen, Y. Hu, B. Xiao, and M. Tan, “Dynamic ensemble reasoning for llm experts,” *arXiv preprint arXiv:2412.07448*, 2024.
- [46] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Advances in neural information processing systems*, vol. 12, 1999.
- [47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [48] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” *Advances in neural information processing systems*, vol. 30, 2017.
- [49] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian *et al.*, “Do as i can, not as i say: Grounding language in robotic affordances,” in *Conference on robot learning*. PMLR, 2023, pp. 287–318.
- [50] 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,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 3676–3713.
- [51] T. Xu, J. Qu, W. Hua, Z. Li, J. Xu, A. Liu, L. Zhao, and X. Zhou, “Evidence reasoning and curriculum learning for document-level relation extraction,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 2, pp. 594–607, 2024.
- [52] M. Van Otterlo and M. Wiering, “Reinforcement learning and markov decision processes,” in *Reinforcement learning: State-of-the-art*. Springer, 2012, pp. 3–42.
- [53] S. Kullback and R. A. Leibler, “On information and sufficiency,” *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
- [54] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “Bertscore: Evaluating text generation with bert,” *International Conference on Learning Representations*, 2020.
- [55] P. Laban, A. Hsi, J. Canny, and M. A. Hearst, “The summary loop: Learning to write abstractive summaries without examples,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 5135–5150.
- [56] D. Ghalandari, C. Hokamp, and G. Ifrim, “Efficient unsupervised sentence compression by fine-tuning transformers with reinforcement learning,” in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 1267–1280.
- [57] A. Conneau, K. Khadwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, “Unsupervised cross-lingual representation learning at scale,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 8440–8451. [Online]. Available: <https://aclanthology.org/2020.acl-main.747>
- [58] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, “The llama 3 herd of models,” *arXiv preprint arXiv:2407.21783*, 2024.
- [59] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, “Training verifiers to solve math word problems,” *arXiv preprint arXiv:2110.14168*, 2021.
- [60] M. Suzgun, N. Scales, N. Schärli, S. Gehrman, Y. Tay, H. W. Chung, A. Chowdhery, Q. Le, E. Chi, D. Zhou *et al.*, “Challenging big-bench

- tasks and whether chain-of-thought can solve them,” in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 13 003–13 051.
- [61] A. Srivastava, A. Rastogi, A. Rao, A. A. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso *et al.*, “Beyond the imitation game: Quantifying and extrapolating the capabilities of language models,” *Transactions on machine learning research*, 2023.
- [62] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.
- [63] T. Sellam, D. Das, and A. Parikh, “BLEURT: Learning robust metrics for text generation,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 7881–7892. [Online]. Available: <https://aclanthology.org/2020.acl-main.704>
- [64] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in *Text summarization branches out*, 2004, pp. 74–81.
- [65] Y. Hu, T. J. Ganter, H. Deilamsalehy, F. Dermontcourt, H. Foroosh, and F. Liu, “Meetingbank: A benchmark dataset for meeting summarization,” in *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.