Article

Not peer-reviewed version

The Sleep Mechanism of LLMs

Zipeng Ye *

Posted Date: 5 August 2025

doi: 10.20944/preprints202508.0071.v2

Keywords: LLMs; post-training; prompts engineering

 doi:10.20944/preprints202508.0071.v2

Article

The Sleep Mechanism of LLMs

Zipeng Ye

Independent Researcher, China; yezp13@gmail.com

Abstract

In this paper, we introduce an innovative perspective, proposing that prompts in large language models (LLMs) can be viewed as hypernetworks. From this viewpoint, we further suggest that prompt engineering acts as a form of post-training for LLMs. Building upon this foundation, we present a novel training-free approach to transform system prompts into model parameters, serving as a sleep mechanism within LLMs. Our method effectively enables the conversion of knowledge and memory contained in system prompts into model parameters through the sleep mechanism, enhancing the adaptability and efficiency of language models without traditional training processes.

Keywords: LLMs; post-training; prompts engineering

---

1. Introduction

In recent years, the field of natural language processing has seen significant breakthroughs, largely driven by the advent of large language models (LLMs) such as GPT [6] and BERT [2]. These models have demonstrated remarkable capabilities in understanding, generating, and interacting with human language. Although pre-training serves as a foundational stage where models learn from a large corpus, it is the subsequent processes of post-training and prompt engineering that refine these models, enhancing their applicability to specific tasks. Post-training, commonly referred to as fine-tuning, involves adjusting the pre-trained models using additional task-specific data to achieve higher accuracy and relevance in designated applications. This step enables models to specialize, essentially bridging the gap between general understanding and task-oriented proficiency.

The integration of reinforcement learning (RL) with LLMs is transforming the landscape of artificial intelligence by allowing models to adaptively learn from interactions, thus refining decision-making processes. Reinforcement learning provides a mechanism for models to optimize their behavior based on feedback, making sequential decisions through trial-and-error in dynamic environments. Notable RL techniques, such as proximal policy optimization (PPO) [8] and group relative policy optimization (GRPO) [4] have proven effective in diverse settings, offering strategies for reward maximization and policy improvement. When applied to LLMs, these methods enable models to enhance their language understanding and generation capabilities, fostering intelligent and contextually aware responses that improve over time.

Prompt engineering, on the other hand, has emerged as a powerful technique for guiding LLMs without the need for exhaustive retraining [10]. By intelligently crafting input prompts, users can elicit desired responses from models, effectively manipulating the generation process. Prompts act as dynamic modifiers, shaping the output through strategic cues embedded within the input. This technique offers a versatile alternative to fine-tuning, leveraging the inherent capabilities of pre-trained models to respond to nuanced instructions. In particular, prompt engineering can be seen as a form of on-the-fly adjustment, providing real-time control over model behavior while maintaining computational efficiency.

Phenomenologically speaking, the effects of prompt engineering exhibit striking similarities to those of post-training, both aiming to refine and tailor model behavior. Prompt engineering involves crafting precise and strategic prompts that guide large language models in generating desired outputs,

 doi:10.20944/preprints202508.0071.v2

analogous to how fine-tuning uses task-specific data and rules to adjust model parameters for improved performance. This meticulous design process shares the meticulous nature of the fine-tuning phase, where careful modifications are made to achieve optimal results. In essence, from the perspective of hypernetworks, prompt engineering can be viewed as a form of post-training, providing real-time adaptation without necessitating extensive retraining. By influencing model responses through contextual input, both approaches leverage existing knowledge within the model to enhance its capacity to specialize in varied linguistic tasks.

An ideal and efficient post-training process aims for the model to distill patterns and learn specific principles, storing them as system-prompt-like representations in its memory. However, because system prompts cannot be excessively long, the model must employ a sleep mechanism to gradually convert these system prompts into model parameters. This paper delves into the exploration and research of the sleep mechanism in LLMs. Key contributions of this paper include the following:

- We propose a novel perspective where prompts in LLMs can be viewed as hypernetworks, referred to as "prompts as hypernetworks" for short.
- From the perspective of "prompts as hypernetworks", we propose a novel perspective in which prompt engineering is essentially a form of post-training for LLM.
- We propose a novel training-free approach to transform system prompts into model parameters, which is a sleep mechanism of LLM.

2. Prompt As Hypernetworks

2.1. Hypernetworks

Hypernetworks [5] is an approach to using a single network to generate weights for another network. It uses a cluster of networks rather than a single network, which possesses greater capabilities. Hypernetworks can be expressed in the following form:

$$f_{regular}(I_{regular}) = f_{hyper}(I_{hyper})(I_{regular}), \tag{1}$$

where f_{hyper} is the hypernetworks and f_{hyper} is the regular networks. Upon receiving hyper-input I_{hyper}, a hypernetwork outputs regular networks.

In the practice of convolutional neural networks, there have already been numerous methods and applications [1,11,12] of hypernetworks. In recent years, as the powerful capabilities of attention mechanisms [9] have been recognized, transformers have gradually replaced convolutional neural networks. Hypernetworks is also a perspective for viewing the relationship between networks and different inputs. The attention mechanism is crucial to the transformer architecture and LLMs. Studies indicate that attention mechanisms can be viewed as hypernetworks [7]. Furthermore, as a key element of attention mechanisms, context plays a central role from the perspective of hypernetworks. Moreover, as part of the context, prompts, especially system prompts, play a significant role from the perspective of hypernetworks.

2.2. Partial Specialization

Hypernetworks receive one portion of the input (hyper-input) to generate a specific neural network, which then processes another portion of the input (regular-input) to produce the output. Let us consider this fact from a reversed perspective. A neural network that receives multiple inputs is essentially a hypernetwork. Fixing a portion of the input results in a form of partial specialization, thereby generating a neural network that processes the remaining input. In LLM, context serves as input, with system prompts being a part of the input that is completely fixed. Therefore, prompts, particularly system prompts, within LLMs can be viewed as the hyper-input to hypernetworks.

The analytical process described above can be represented by the following formula:

$$O = f(I_{context}) = f([I_{hyper}, I_{regular}]) = f(I_{hyper})(I_{regular}), \tag{2}$$

 doi:10.20944/preprints202508.0071.v2

where O is the output, f is the LLM, $I_{context}$ is the context, I_{hyper} is prompts (system prompts) and $I_{regular}$ is the remaining context. $f(I_{hyper})$ is the partial specialized LLMs based on system prompts.

3. Prompt Engineering As Post-training

Prompt engineering is the process of designing and refining prompts to enhance the performance of LLM. It involves crafting specific input prompts that effectively guide the model in generating desired outputs. This technique is critical in natural language processing tasks where the quality and relevance of the generated text are paramount. By experimenting with various prompt formulations, prompt engineering seeks to improve the model's ability to understand context and deliver accurate and coherent responses across a range of applications, including content creation, question answering, information retrieval, and interactive dialogue systems.

Phenomenologically speaking, the effects of prompt engineering are similar to those of post-training. Moreover, the meticulous process by which humans design prompts is quite similar to the careful crafting of fine-tuning data or rules. In fact, from the viewpoint of hypernetworks, prompt engineering is essentially a form of post-training. In Eq. 2, $f(I_{hyper})$ is a network determined by system prompts I_{hyper}, which could be regarded as a new network that has undergone post-training.

Recent studies have shown that LLMs can achieve the same results as training using the mechanism known as "notions of contextual blocks" without updating their weights [3]. This perspective supports our point of view. However, this approach is akin to a person who never sleeps and constantly recalls all the experiences they've lived through, as it depends on maintaining an exceptionally long context.

4. Sleep Mechanism

4.1. Analysis

The ideal sleep mechanism involves converting the system prompts into model parameters. In Section 3, we reveal that prompt engineering is essentially a form of post-training. It offers insights into the implementation of our sleep mechanism. The most straightforward approach is to fine-tune the model using outputs with system prompts, enabling the model to operate without requiring system prompts. Although this method is bound to be effective, we aim to discover a more elegant approach that does not require training. Using specific assumptions and approximations, we convert the fine-tuning process into a closed-form solution.

The transformer architecture primarily consists of two main modules: the attention layer and the feedforward layer. Since different tokens do not interact within the feedforward layer, our focus is primarily on the attention layer. The attention mechanism is the following:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{3}$$

During inference, we utilize K and V from the context, together with Q from the last token. We decompose the context into system prompts and other tokens and rewrite the attention mechanism:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q[K_{sys}, K_{others}]^\top}{\sqrt{d_k}}\right)[V_{sys}, V_{others}], \tag{4}$$

where the subscript sys denotes system prompts, while $others$ refers to the remaining context. We attempt to separate these two components in the formula. Although most of them are linear, the softmax function is non-linear. We can address this by introducing an undetermined coefficient.

$$\alpha \cdot \text{softmax}\left(\frac{QK_{sys}^\top}{\sqrt{d_k}}\right)V_{sys} + (1-\alpha)\cdot \text{softmax}\left(\frac{QK_{others}^\top}{\sqrt{d_k}}\right)V_{others}, \tag{5}$$

 doi:10.20944/preprints202508.0071.v2

where α is the ratio of system prompts. The reason why this can be rewritten in such a manner is due to the normalization property of the softmax function. This expression includes two terms, both of which structurally resemble the attention mechanism. The second term essentially represents the computation process after removing the system prompts, while the first term needs to be eliminated. The final form should exclude the system prompt components, yet permit some adjustments to the network parameters.

$$\text{softmax}\left(\frac{Q'K'^{\top}_{others}}{\sqrt{d_k}}\right)V'_{others}, \tag{6}$$

where Q', K'_{others} and V'_{others} are calculated using the updated parameters.

Given that the role of K and Q represents the correlation between tokens, we believe that this should be preserved, while V can be influenced by system prompts. Thus, our target form is as follows:

$$\text{softmax}\left(\frac{QK^{\top}_{others}}{\sqrt{d_k}}\right)(V_{others} + V_{delta}), \tag{7}$$

where V_{delta} is the effect of system prompts on the model. We have the following equation:

$$\text{softmax}\left(\frac{QK^{\top}_{others}}{\sqrt{d_k}}\right)V_{delta} = \alpha \cdot \left[\text{softmax}\left(\frac{QK^{\top}_{sys}}{\sqrt{d_k}}\right)V_{sys} - \text{softmax}\left(\frac{QK^{\top}_{others}}{\sqrt{d_k}}\right)V_{others}\right]. \tag{8}$$

Here, V_{others} and K_{others} are related to the training corpus, which can be approximately replaced by statistical metrics.

$$V_{delta} = \alpha \cdot \left[\frac{\text{softmax}\left(\frac{\bar{Q}K^{\top}_{sys}}{\sqrt{d_k}}\right)}{\text{softmax}\left(\frac{\bar{Q}\bar{K}^{\top}}{\sqrt{d_k}}\right)}V_{sys} - \bar{V}\right], \tag{9}$$

where $\bar{K}$, $\bar{Q}$ and $\bar{V}$ represent the mean of the variables corresponding to the attention layer. Due to the complexity of the coefficients, we can further approximate and simplify the above expression to the following form:

$$V_{delta} = \alpha \cdot (\beta \cdot V_{sys} - \bar{V}), \tag{10}$$

where α and β are parameters of the algorithm or a statistical measure of the model. Since V is computed via a linear layer, we can update the parameters according to this formula.

4.2. Training-free Approach

Based on the above analysis, we can naturally derive a training-free algorithm for updating the model parameters. First, we compute V_{sys} based on the system prompts, which essentially constitutes a pre-filling process. Next, we need to compute $\bar{V}$, which can be directly replaced by the bias of the linear layer or calculated as the mean using the historical records from the KV cache. α can be considered as the learning rate, while β can be considered as the influence strength of the system prompts.

5. Conclusions

In conclusion, this research presents a novel conceptual framework that reinterprets prompts in LLMs as hypernetworks, thereby highlighting their crucial role in shaping and refining model behavior. By redefining prompt engineering as a form of post-training, we have developed a pioneering training-free approach that utilizes system prompts as a sleep mechanism, converting their embedded knowledge and memory directly into model parameters. This mechanism significantly advances the adaptability and efficiency of LLMs, offering a compelling alternative to conventional training

 doi:10.20944/preprints202508.0071.v2

methodologies. Our findings underscore the potential of prompt-based adaptations to enhance the performance and flexibility of artificial intelligence systems, paving the way for further innovation in the development of more intelligent and responsive language models.

References

1. Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11030–11039, 2020.
2. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.
3. Benoit Dherin, Michael Munn, Hanna Mazzawi, Michael Wunder, and Javier Gonzalvo. Learning without training: The implicit dynamics of in-context learning. *arXiv preprint arXiv:2507.16003*, 2025.
4. Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
5. David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, pp. 1–18, 2017.
6. Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.
7. Simon Schug, Seijin Kobayashi, Yassir Akram, João Sacramento, and Razvan Pascanu. Attention as a hypernetwork. *arXiv preprint arXiv:2406.05816*, 2024.
8. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
9. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
10. Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023.
11. Zipeng Ye, Zhiyao Sun, Yu-Hui Wen, Yanan Sun, Tian Lv, Ran Yi, and Yong-Jin Liu. Dynamic neural textures: Generating talking-face videos with continuously controllable expressions. *arXiv preprint arXiv:2204.06180*, 2022.
12. Zipeng Ye, Mengfei Xia, Ran Yi, Juyong Zhang, Yu-Kun Lai, Xuwei Huang, Guoxin Zhang, and Yong-jin Liu. Audio-driven talking face video generation with dynamic convolution kernels. *IEEE Transactions on Multimedia*, 25:2033–2046, 2022.