Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Adaptive layer splitting for wireless large language model inference in edge computing: a model-based reinforcement learning approach*

Yuxuan CHEN[1], Rongpeng LI‡[1], Xiaoxue YU[1], Zhifeng ZHAO[2], Honggang ZHANG[1]

[1]*College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou 310027, China*

[2]*Zhejiang Lab, Hangzhou 310012, China*

E-mail: cyx00@zju.edu.cn; lirongpeng@zju.edu.cn; sdwhyxx@zju.edu.cn; zhaozf@zhejianglab.com; honggangzhang@zju.edu.cn

Received June 1, 2024; Revision accepted Sept. 13, 2024; Crosschecked Jan. 7, 2025

Abstract: Optimizing the deployment of large language models (LLMs) in edge computing environments is critical for enhancing privacy and computational efficiency. In the path toward efficient wireless LLM inference in edge computing, this study comprehensively analyzes the impact of different splitting points in mainstream open-source LLMs. Accordingly, this study introduces a framework taking inspiration from model-based reinforcement learning to determine the optimal splitting point across the edge and user equipment. By incorporating a reward surrogate model, our approach significantly reduces the computational cost of frequent performance evaluations. Extensive simulations demonstrate that this method effectively balances inference performance and computational load under varying network conditions, providing a robust solution for LLM deployment in decentralized settings.

Key words: Large language models (LLMs); Edge computing; Model-based reinforcement learning (MBRL); Split inference; Transformer

https://doi.org/10.1631/FITEE.2400468 **CLC number:** TP391

1 Introduction

The field of natural language processing (NLP) has recently experienced transformative changes, driven by the rapid advancement of large language models (LLMs) such as generative pre-trained Transformer 4 (GPT-4) (OpenAI, 2023) and Gemini (Gemini Team Google, 2023). These models are highly proficient at generating human-like text (Brown et al., 2020; Wei et al., 2021; Bai et al., 2022; Le Scao et al., 2022; Touvron et al., 2023), catalyzing progress across various domains (Nijkamp et al., 2022; Rozière et al., 2023; Webb et al., 2023; Jin et al., 2024). Although LLMs perform well in centralized cloud environments, they face significant scalability and privacy issues (Thirunavukarasu et al., 2023; Wu et al., 2023) in sensitive applications such as healthcare and finance (Kaddour et al., 2023), which have driven the exploration of edge computing (Mach and Becvar, 2017; Mao et al., 2017) as a complementary paradigm. Of note, edge computing processes sensitive information locally rather than

‡ Corresponding author

* Project supported by the National Key Research and Development Program of China (No. 2024YFE0200600), the National Natural Science Foundation of China (No. 62071425), the Zhejiang Key Research and Development Plan, China (No. 2022C01093), the Zhejiang Provincial Natural Science Foundation of China (No. LR23F010005), the National Key Laboratory of Wireless Communications Foundation, China (No. 2023KP01601), and the Big Data and Intelligent Computing Key Lab of CQUPT, China (No. BDIC-2023-B-001)

ORCID: Yuxuan CHEN, https://orcid.org/0009-0008-9570-2000; Rongpeng LI, https://orcid.org/0000-0003-4297-5060; Xiaoxue YU, https://orcid.org/0009-0008-1098-7589; Honggang ZHANG, https://orcid.org/0000-0003-1492-1364

transmitting it to a centralized cloud (Li E et al., 2020; Letaief et al., 2022). Consequently, it minimizes the exposure to potential privacy breaches and unauthorized access. Moreover, edge computing allows for a flexible and distributed architecture, and can accommodate allocated computational resources to specific requirements (Abbas et al., 2018; Pham et al., 2020). Therefore, the integration of edge computing and LLMs empowers LLMs with enhanced personalized and domain-specific generative capabilities (Chen YX et al., 2024). LLMs' substantial computational demands often exceed the processing capacities of communication-limited user equipment (UE) in radio access network or Internet of Things (IoT) systems (Hadi et al., 2023; Lin Z et al., 2024a; Patil and Gudivada, 2024). Correspondingly, split learning and inference (Lee et al., 2023; Qiao and Zhou, 2023; Lin Z et al., 2024b) are proposed to jointly leverage the computing capability of the UE and edge nodes (e.g., base stations (BSs)).

Though most of existing works (Chen MZ et al., 2021; Lan et al., 2021; Karjee et al., 2022; Ryu et al., 2022; Lee et al., 2023; Wang YZ et al., 2023; Lin Z et al., 2024b) focus on careful model splitting to balance the computational and communication costs, splitting different layers of LLMs is quite unique, as the intermediate outputs have consistent dimensions, leading to the same communication cost. Additionally, transmitting tensors between LLM layers over potentially noisy channels could hinder LLM inference performance. As validated in this work later, given different splitting points, the possible loss induced by unreliable wireless channels produces a significantly diverse impact on model performance. These attributes necessitate a shift in focus from optimizing transmission efficiency to managing the computational burden on the UE. In LLM deployment scenarios, wherein the scale and complexity of the models impose significant demands on the limited computational resources of the UE, the challenge lies in balancing the computational load without degrading inference performance. Thus, it is critical to identify the optimal splitting point for model inference while combating wireless channel volatility. Such a viewpoint transcends traditional single-step optimization techniques and warrants a new framework accommodating the sequential nature of decision-making.

Reinforcement learning (RL) emerges as an apt methodology, known for its proficiency in sequential decision-making tasks (Mnih et al., 2015; Li YX, 2017; Luong et al., 2019; Qian et al., 2019). RL learns optimal strategies over successive iterations, continuously adapting to the dynamic and uncertain edge environment. The iterative nature of RL, requiring multiple interactions with the LLM to ascertain the rewards of different actions, complements the continuous and unpredictable variations in wireless channels. Nevertheless, it inevitably incurs significant interaction costs before collecting sufficient records. Fortunately, with the merits in sampling efficiency, model-based RL (MBRL) is particularly suited to this context (Deisenroth and Rasmussen, 2011; Kaiser et al., 2019; Shlezinger et al., 2021; Moerland et al., 2023). In particular, MBRL capably simulates the uncertain environment and assesses the potential outcomes of actions, thus enabling a more informed and anticipatory optimization strategy for the dynamical splitting point determination process.

As illustrated in Fig. 1, we propose an adaptive layer splitting algorithm for efficient LLM inference in edge computing, where only a few selected Transformer layers are provisionally activated at the UE. Meanwhile, faced with wireless channel fluctuations, we leverage a sample-efficient MBRL-inspired approach to determine the suitable splitting point. While highlighting the key differences with existing works in Table 1, the primary contributions of this paper are summarized as follows:

1. We comprehensively evaluate the impact of LLM splitting points on LLM inference performance under varying channel conditions, and formulate the determination of appropriate LLM splitting points as a sequential decision-making process.

2. We leverage the proximal policy optimization (PPO) (Schulman et al., 2017) for adaptive splitting point determination and devise a sample-efficient reward surrogate model to facilitate the learning.

3. We conduct extensive studies to evaluate the robustness and effectiveness of the MBRL-inspired splitting point determination method.

2 Related works

2.1 Edge-enhanced LLM deployment

In the realm of LLMs, Chen L et al. (2024) and Lin B et al. (2024) provided a comprehensive

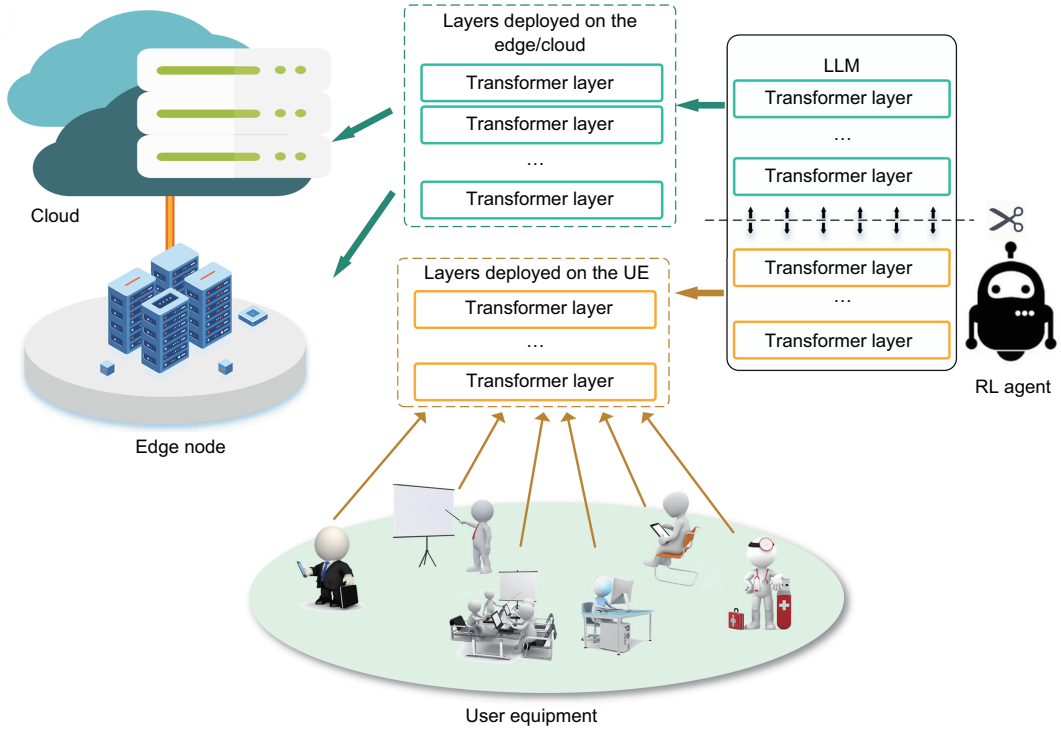

Fig. 1 A high-level architecture of the framework, depicting the distribution of the large language model (LLM) across the edge and the user equipment (UE), highlighting the role of the reinforcement learning (RL) agent in managing interactions between the LLM and wireless networks

Table 1 Summary and comparison of related works

Reference	Brief description	Limitation
Lin Z et al. (2024a)	Discusses the deployment of LLMs at 6G edges, advocating edge computing to optimize LLM deployment	Lacks detailed methodologies for handling computational constraints at edge nodes
Gupta O and Raskar (2018)	Introduces split learning, which splits models between clients and servers to avoid transferring raw data	Focuses on DNNs without considering the unique challenges of wireless channels and high deployment costs associated with LLMs
Lee et al. (2023)	Explores split inference in wireless networks, distributing DNNs for collaborative inference	Does not account for the specific challenges of LLMs, such as high computational requirements and the impact of volatility of wireless channels on performance
Shlezinger et al. (2021)	Investigates model-based machine learning for communication systems, emphasizing the advantages of predictive modeling in dynamic environments	Limited focus on LLMs and their unique characteristics, including high processing demands and sensitivity to channel noise

LLMs: large language models; 6G: sixth-generation; DNNs: deep neural networks

overview of the challenges faced by LLMs in cloud-based settings, particularly emphasizing the constraints related to data privacy and processing efficiency. Satyanarayanan et al. (2009) proposed the concept of edge computing as a viable solution. Chen YX et al. (2024) proposed an LLM cloud–edge collaboration framework, which uses small-scale models deployed at edge nodes to enhance the generative capabilities of cloud-based LLMs. Lin Z et al. (2024a) emphasized the importance of optimizing

LLM deployment at the sixth-generation (6G) mobile communication edges but did not provide specific methodologies for leveraging edge computing capabilities. Dong et al. (2024) proposed to use LLMs as offline compilers to meet low-latency requirements in edge computing. However, these approaches do not fully address the computational constraints of edge nodes. Our research bridges this gap by introducing a dynamic layer-splitting framework to alleviate these limitations.

2.2 Split inference in distributed computing

Gupta O and Raskar (2018) introduced the concept of split models between clients and servers to avoid transferring raw data, offering new perspectives on distributed computing and privacy-preserving artificial intelligence (AI). Chen MZ et al. (2021), Ryu et al. (2022), and Lin Z et al. (2024b) broadened the application of split learning to encompass the distributed deployment of deep neural networks (DNNs) in wireless networks. Lan et al. (2021), Karjee et al. (2022), Lee et al. (2023), and Wang YZ et al. (2023) distributed different portions of DNNs between edge nodes and cloud for collaborative inference task execution, thus reducing response latency and improving the scalability.

Different from these existing works (Lan et al., 2021; Karjee et al., 2022; Lee et al., 2023; Wang YZ et al., 2023), LLM splitting presents unique challenges. Due to the large computational demands of LLMs, deploying different layers across the UE and edge nodes requires careful consideration of the computational load on the UE. The significant disparity in computational capabilities between heterogeneous devices further complicates the deployment process (Ong, 2024; Zhang MJ et al., 2024). Moreover, splitting at different points within the LLM significantly affects the overall inference performance, as demonstrated by the simulation results in Section 3. To our best knowledge, this belongs to the first efforts to address this important issue, and lays the foundation for further adaptive layer splitting to balance LLM inference performance and UE computation cost.

2.3 RL in network optimization

RL emerges as a pivotal tool for optimizing decision-making processes in dynamic and uncertain environments, as highlighted by previous works (Zhu et al., 2022; Icarte et al., 2023). Ke and Astuti (2023) and Yang et al. (2023) demonstrated RL's efficacy in enhancing performance optimization within distributed networks, highlighting its potential to adapt and respond to evolving environmental conditions. In the context of wireless network environments, Liu et al. (2020) and Li X et al. (2022) leveraged RL to address the challenges of resource allocation and network traffic management, showcasing its capability to optimize system performance amidst the fluctuating nature of wireless communications. With its predictive modeling capabilities and remarkable sampling efficiency, MBRL is adept at navigating environments with variable factors (Kaiser et al., 2019; Egorov and Shpilman, 2022). To address the issues of computational constraints and wireless channel volatility, our research draws inspiration from MBRL and develops a computation-efficient reward surrogate model to optimize LLM deployment at the edge.

3 System model and problem formulation

In this section, we begin with a comprehensive description of the system model, which highlights the deployment of a splitting LLM across the wireless network. Afterward, we discuss the impact of the layer splitting point on LLM performance under various channel conditions. Finally, we formulate the channel-aware splitting point optimization problem to balance the UE computational load and LLM inference performance.

Beforehand, we summarize the main notations used in Table 2.

3.1 System model

To characterize the LLM provisioning with model splitting, we consider primarily a system model illustrated in Fig. 2. Without loss of generality, we assume that for an L-layer LLM, the first p layers $L_{\mathrm{U}}(\cdot)$ are deployed in the UE, and the remaining $L-p$ layers $L_{\mathrm{C}}(\cdot)$ are in the edge, where the splitting point p is adjustable. Based on this, for an input $\boldsymbol{x} \in \mathbb{R}^{d_{\mathrm{in}}}$, typically a sequence of text tokens, the UE transforms $\boldsymbol{x}$ into a higher-dimensional intermediate tensor $\boldsymbol{y} \in \mathbb{R}^{d_{\mathrm{mid}}}$, that is,

$$\boldsymbol{y} = L_{\mathrm{U}}(\boldsymbol{x}; \theta_{\mathrm{UE}}), \tag{1}$$

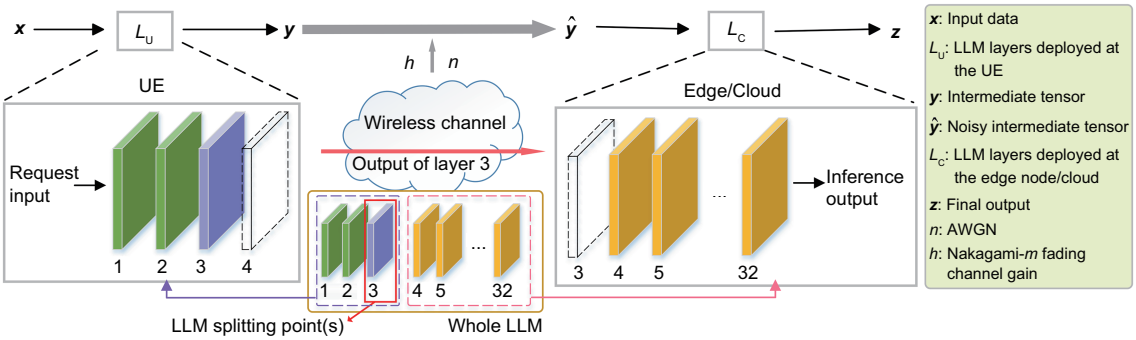

Fig. 2 Overview of the split model architecture in wireless channel, with layer 3 designated as the example splitting point. We use the 32-layer LLAMA2-7B model as an example

Table 2 Notations used in the paper

Notation	Definition
L	Total number of layers in the LLM
p	Adjustable splitting point, indicating the number of layers deployed at the UE
$L_{\mathrm{U}}(\cdot), L_{\mathrm{C}}(\cdot)$	Layers deployed on the UE and the edge, respectively
$\boldsymbol{x} \in \mathbb{R}^{d_{\mathrm{in}}}$	Input data
$\boldsymbol{y}, \hat{\boldsymbol{y}} \in \mathbb{R}^{d_{\mathrm{mid}}}$	Intermediate tensor before and after the channel, respectively
$\theta_{\mathrm{UE}}, \theta_{\mathrm{edge}}$	Parameters of the LLM layers deployed on the UE and the edge, respectively
h	Nakagami-m fading channel gain
m	Nakagami-m fading channel shape parameter
Ω	Nakagami-m fading channel spread parameter
$n \sim \mathcal{N}(0, \sigma^2)$	Gaussian-distributed noise with mean 0 and variance σ^2
h_{th}	Threshold for channel gain below which packet loss occurs
P	Probability of packet loss
$\boldsymbol{z} \in \mathbb{R}^{d_{\mathrm{out}}}$	Inference output provided by the edge
θ_{surr}	Parameter of the reward surrogate model

where θ_{UE} denotes the parameters of layers in $L_{\mathrm{U}}(\cdot)$. This procedure inevitably incurs a certain computational load on the UE, typically measured in floating point operations per second (FLOPs). Typically, for layer 1 with an input sequence of length d_{in} with hidden dimension d_{mid}, using a multi-head self-attention mechanism with κ heads, the computations for the involved multi-head attention mechanism and feed-forward network components can be obtained as follows: $\mathrm{FLOPs}(L_1) = \frac{3d_{\mathrm{in}}d_{\mathrm{mid}}^2}{\kappa} + \frac{2d_{\mathrm{in}}^2 d_{\mathrm{mid}}}{\kappa} + 9d_{\mathrm{in}}d_{\mathrm{mid}}^2$. Thus, the computational load on the UE can be approximated as

$$C_{\mathrm{UE}}(p) = \sum_{i=1}^{p} \mathrm{FLOPs}(L_i) = p\left(\frac{3d_{\mathrm{in}}d_{\mathrm{mid}}^2}{\kappa} + \frac{2d_{\mathrm{in}}^2 d_{\mathrm{mid}}}{\kappa} + 9d_{\mathrm{in}}d_{\mathrm{mid}}^2\right). \tag{2}$$

The intermediate tensor $\boldsymbol{y}$ is transmitted from the UE to the edge over a wireless communication channel. Mathematically, the received signal $\hat{\boldsymbol{y}} \in \mathbb{R}^{d_{\mathrm{mid}}}$ after a Nakagami-m fading channel (Nakagami et al., 1960; Beaulieu and Cheng, 2005) can be represented as

$$\hat{\boldsymbol{y}} = h\boldsymbol{y} + n, \tag{3}$$

where $h \sim \mathrm{Nakagami}(m, \Omega)$ represents the Nakagami-m fading with shape parameter m and spread parameter Ω, and $n \sim \mathcal{N}(0, \sigma^2)$ represents the noise following a normal distribution with zero mean and variance σ^2. In our simulation, each element of the intermediate tensor is independently subjected to a packet loss probability, abstracting each element as a separate packet to capture the impact of noise at a granular level. When $h = 1$, it degenerates to an additive white Gaussian noise (AWGN) channel. The probability density function (PDF) of the Nakagami-m distribution for the channel gain h is given by the following expression:

$$f(h) = \frac{2m^m h^{2m-1}}{\Gamma(m)\Omega^m} \mathrm{e}^{-\frac{mh^2}{\Omega}}, \tag{4}$$

where $\Gamma(m)$ denotes the gamma function. The shape parameter m controls the severity of fading. When $m = 1$, the Nakagami-m distribution degenerates to a Rayleigh fading channel; at lower values of m, the channel experiences more severe fading, leading

to greater variability in noise levels. In practical scenarios, particularly when splitting LLMs between the UE and base stations, the mobile nature of devices often leads to rapidly changing channel conditions. For instance, as a user moves from an open outdoor area into a building or dense urban environment, the shape parameter m in the Nakagami-m distribution would decrease, reflecting more severe multipath fading and greater noise intensity fluctuations. This dynamic environment necessitates continuous adaptation of the splitting strategy to maintain inference performance under varying noise conditions.

Moreover, when the channel gain h falls below a certain threshold h_{th}, the lower signal-to-noise ratio (SNR) and relatively high bit error rate (BER) imply retransmission, thus exceeding the latency requirement in quality of service (QoS) with a rather high probability. Hence, such a case can be regarded as a packet loss. Accordingly, recalling the formula of Nakagami-m distribution, the probability of packet loss can be expressed as

$$\begin{aligned} P(\text{packet loss}) &= P(h < h_{\text{th}}) \\ &= \int_0^{h_{\text{th}}} \frac{2m^m h^{2m-1}}{\Gamma(m)\Omega^m} \mathrm{e}^{-\frac{mh^2}{\Omega}} \mathrm{d}h. \end{aligned} \tag{5}$$

Subsequently, the edge delivers an inference output $\boldsymbol{z} \in \mathbb{R}^{d_{\text{out}}}$ as

$$\boldsymbol{z} = L_{\text{C}}(\hat{\boldsymbol{y}}; \theta_{\text{edge}}). \tag{6}$$

The performance of the LLMs is commonly quantified using the perplexity (PPL) metric, a standard means in NLP to evaluate how well a probability model predicts a sample. Given an LLM and a sequence of N tokens (i.e., $w_1, w_2, \cdots, w_N$), PPL is defined as

$$\begin{aligned} &\text{PPL} \\ &= \exp\left(-\frac{1}{N}\sum_{k=1}^{N} \log P_{\text{LLM}}(w_k | w_1, w_2, \ldots, w_{k-1})\right), \end{aligned} \tag{7}$$

where $P_{\text{LLM}}(w_k | w_1, w_2, \ldots, w_{k-1})$ denotes LLM's prediction probability from the previous $k-1$ tokens. A lower PPL signifies superior model performance, demonstrating the model's proficiency in accurately predicting the subsequent word in a sequence. Hence, in this context, PPL can serve as a unified metric to assess the impact of channel impairments on the LLM's ability.

3.2 Problem formulation

Beforehand, we investigate the impact of splitting points on the inference performance under different channel conditions and, in Fig. 3, we present the corresponding simulation results considering several mainstream open-source LLMs, including LLAMA2-7B, LLAMA2-13B (Touvron et al., 2023), Mistral-7B (Jiang et al., 2023), Aya-23-8B (Üstün et al., 2024), OpenChat-8B (Wang G et al., 2023), and Prem-1B (Gupta R and Sosio, 2024). Consistent with our intuition, it can be observed from Fig. 3 that for the same setting, a lower SNR or larger packet loss rate generally yields inferior performance (i.e., larger PPL). More interestingly, an earlier model splitting (i.e., a smaller p) could worsen the inference performance while the channel conditions significantly affect the performance of a given splitting point. Such an observation is also consistent with the widely recognized fact that earlier layers in LLMs are responsible for learning the general or basic features of the training dataset (Zhang XH et al., 2023). These findings underscore the importance of strategically selecting the splitting point in the LLM architecture to maintain the desired performance.

On the other hand, high computational load on the UE would lead to increased latency and energy consumption, further undermining the benefits of deploying LLMs in edge environments. As indicated in Eq. (2), the computational load on the UE is approximately proportional to the number of layers processed locally. This proportionality yields a contradicting phenomenon that an earlier model splitting worsens the inference performance but ameliorates the computational cost at the computation-limited UE. In other words, to minimize the overall system PPL while effectively balancing the computational load on the UE, the problem turns to identifying the optimal splitting point p under volatile channel conditions, that is,

$$p^* = \arg\min_{p} \left(\text{PPL}(p; \sigma, m) + \lambda C_{\text{UE}}(p)\right), \tag{8}$$

where $\text{PPL}(p; \sigma, m)$ quantifies the inference performance of the LLM, taking into account the splitting point p, the noise intensity σ, and the Nakagami-m fading shape parameter m which directly affects the packet loss probability. Moreover, the weight λ balances the trade-off between the inference performance and the computational load.

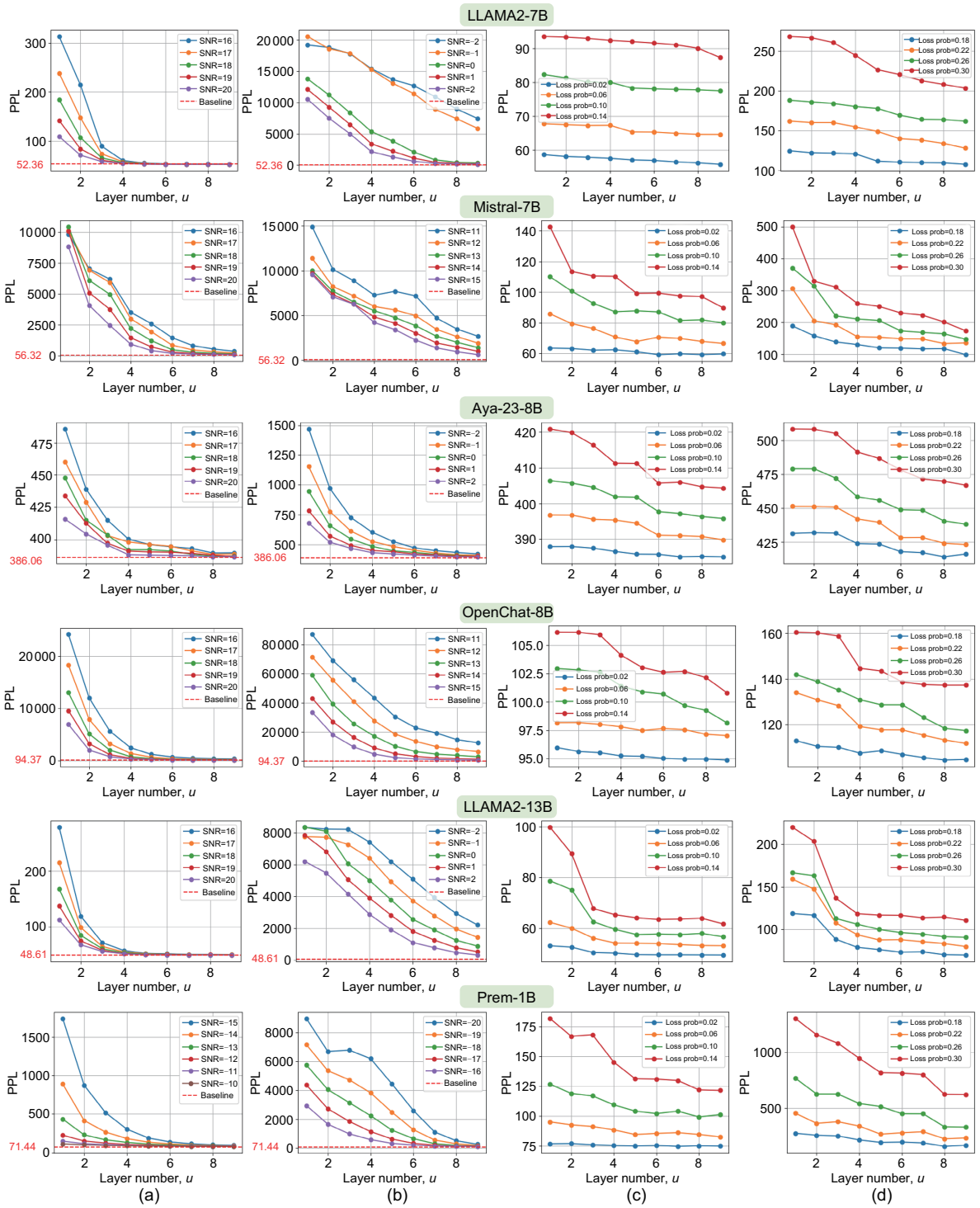

Fig. 3 Illustrations of the impact on perplexity (PPL) across different layers for various large language models (LLMs): (a) high signal-to-noise ratio (SNR); (b) low SNR in additive white Gaussian noise (AWGN); (c) low packet loss probability; (d) high packet loss probability under Nakagami-m fading

Considering the variability of network conditions and the complexities of real-time decision-making in distributed systems, we reformulate this optimization problem as a sequential decision-making task. In this context, RL is particularly well-suited for this scenario, due to its capability to adapt

to the evolving environment and optimize decisions accordingly.

4 RL for splitting point optimization

In this section, we investigate the application of RL to dynamically optimize the splitting point of LLMs across the UE and edge computing resources, thus adaptively responding to channel variations.

4.1 Markov decision process (MDP)

Splitting point adjustment under volatile channels can be formalized as an MDP, consisting of a tuple $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$. In particular, as illustrated in Fig. 4, the state space $\mathcal{S}$ encompasses the key factors such as the noise intensity σ, the Nakagami-m fading shape parameter m, and the current splitting point p, namely, $s = \{\sigma, m, p\} \in \mathcal{S}$. The action space $\mathcal{A}$ is designed to accommodate a range of possible adjustments to the splitting point, allowing for both finer-grained and more substantial modifications. Specifically, $\mathcal{A}$ includes actions such as moving the splitting point upward or downward by u layers (where u can take values such as 1, 2, 3, and so forth) or maintaining the current position. Mathematically, this can be expressed as $a \in \mathcal{A} = \{-u, \ldots, -1, 0, +1, \ldots, +u\}$. This generalized action space provides the RL agent with the flexibility to optimize the splitting strategy dynamically in response to varying channel environments. For a time step t, given an action a_t under state s_t, the environment state will transit to the next state s_{t+1} following the transition probability P, which is contingent on the selected action and real-time channel conditions. Meanwhile, a reward following the reward function R can be obtained as

$$r_t = R(s_t, a_t) = -\big(\mathrm{PPL}(s_{t+1}) + \lambda C_{\mathrm{UE}}(s_{t+1})\big). \quad (9)$$

Finally, the long-term overall objective can be formalized as

$$\begin{aligned} &J(\theta) \\ =&\mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T}\gamma^t r_t\right] = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T}\gamma^t R(s_t, a_t)\right] \\ =&\mathbb{E}_{\pi_\theta}\left[-\sum_{t=1}^{T}\gamma^t\big(\mathrm{PPL}(s_{t+1}) + \lambda C_{\mathrm{UE}}(s_{t+1})\big)\right], \end{aligned} \quad (10)$$

where the discount factor γ involves the significance of future rewards. Correspondingly, it requires learning a policy π_θ parameterized by θ to attain the maximum of Eq. (10).

4.2 PPO approach

For dynamically adjusting the splitting point of LLMs within cloud–edge–UE networks, we apply the PPO algorithm (Schulman et al., 2017) to iteratively learn the policy π_θ. Specifically, PPO uses two neural networks, namely, the policy network $\pi_\theta(a|s)$ and the value function network $V_\phi(s)$, which are parameterized by θ and ϕ, respectively, to dictate the action a given the state s and estimate the expected discounted return from state s. Notably, PPO leverages a clipped surrogate objective function $L^{\mathrm{CLIP}}(\theta)$,

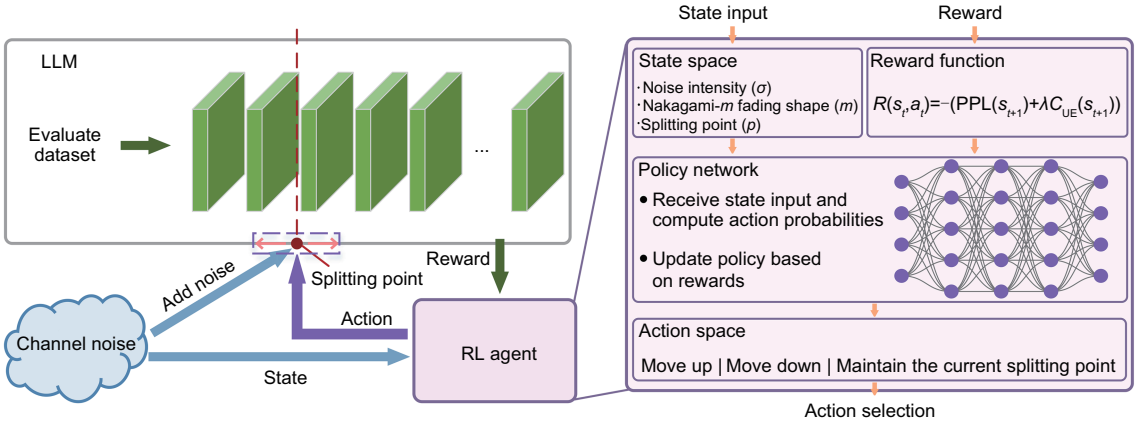

Fig. 4 Illustrations of the reinforcement learning (RL) setup, including the large language model (LLM), RL agent, and channel noise modules. The RL agent optimizes the splitting point of the LLM by receiving state inputs (noise intensity, Nakagami-m fading shape, and splitting point), computing action probabilities via the policy network, and updating the policy based on the reward function

which approximates the true objective $J(\theta)$ but introduces a clipping mechanism to limit the magnitude of policy updates. Mathematically, the clipped objective can be written as

$$L^{\mathrm{CLIP}}(\theta)=\mathbb{E}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}\hat{A}_t,\right.\right.\\ \left.\left.\mathrm{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)},1-\epsilon,1+\epsilon\right)\hat{A}_t\right)\right], \tag{11}$$

where θ_{old} represents the policy parameter for sampling and ϵ represents the clipping threshold. The clipping mechanism $\mathrm{clip}(\cdot)$ ensures that the policy ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}$ does not deviate significantly from 1, thus preventing large, destabilizing updates. This mechanism is critical in RL scenarios in which stability and reliability are paramount, especially in dynamically changing environments such as cloud–edge–UE networks. The advantage estimator $\hat{A}_t$, which can be computed using the generalized advantage estimation (GAE), is formulated as

$$\hat{A}_t=\sum_{l=0}^{\infty}(\gamma\xi)^l\delta_{t+l}, \tag{12}$$

where $\delta_t=r_t+\gamma V_\phi(s_{t+1})-V_\phi(s_t)$ denotes the temporal difference error, γ is the discount factor, and ξ is the GAE parameter controlling the bias-variance trade-off.

The update to the policy parameter θ is performed using a gradient ascent step on the clipped objective function $L^{\mathrm{CLIP}}(\theta)$. Mathematically, we have

$$\theta\leftarrow\theta+\alpha\nabla_\theta L^{\mathrm{CLIP}}(\theta). \tag{13}$$

This gradient ascent step ensures that the policy is updated iteratively to maximize the expected reward while maintaining stability through the clipping mechanism.

During the training process, we use an experience replay mechanism specifically adapted to the dynamic nature of the channel conditions in our scenario. Notably, state–action pairs, including the current splitting point p, noise intensity σ, and Nakagami-m fading channel shape parameter m, are stored in a replay buffer. At each training step, a mini-batch of these pairs is sampled from the replay buffer to update the policy network, ensuring that the model learns from the past experience under varying network conditions. This technique helps break the temporal correlations inherent in sequential channel variations, leading to more stable learning.

4.3 Reward surrogate model for faster RL

The integration of MBRL into our LLM split optimization scenario significantly enhances the efficiency and effectiveness of our RL approach. Nevertheless, the limited reasoning capability of LLM makes the learning process sluggish. Therefore, inspired by the classical MBRL, which uses a predictive model to simulate the environment, we adopt a surrogate model to approximate the reward function, thereby boosting the learning efficiency. Specifically, we approximate the $\mathrm{PPL}(p;\sigma,m)$ that needs to be computed by running the LLM and compute a DNN-based surrogate model $\widetilde{\mathrm{PPL}}(p;\sigma,m,\theta_{\mathrm{surr}})$ parameterized by θ_{surr} to minimize the mean squared error (MSE) as follows:

$$\mathrm{MSE}=\mathbb{E}\left[\left(\mathrm{PPL}(p;\sigma,m)-\widetilde{\mathrm{PPL}}(p;\sigma,m,\theta_{\mathrm{surr}})\right)^2\right]. \tag{14}$$

Notably, we use cross-validation (Stone, 1974) to prevent overfitting and ensure the generalizability of the surrogate model.

Incorporating the surrogate model directly into the reward calculation, the reward at time step t can be redefined as

$$\tilde{R}(s_t,a_t)=-\left(\widetilde{\mathrm{PPL}}(p_t;\sigma_t,m_t,\theta_{\mathrm{surr}})+\lambda C_{\mathrm{UE}}(p_t)\right), \tag{15}$$

where $\widetilde{\mathrm{PPL}}(p_t;\sigma_t,m_t,\theta_{\mathrm{surr}})$ represents the estimated PPL provided by the surrogate model. This reformulation significantly reduces the training burden by replacing the direct LLM inference with an efficient approximation, enabling more rapid policy evaluation and iteration of an RL policy. From a theoretical standpoint (Romoff et al., 2018), the surrogate model effectively reduces the variance in reward estimation by providing a smoothed approximation of the true reward landscape. This smoothing is particularly advantageous in high-dimensional action spaces, where small perturbations in actions could lead to large fluctuations in PPL if calculated directly.

Correspondingly, during the training process of RL, the term $\hat{A}_t$ in Eq. (12) can be re-written as

follows:

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\xi)^l \Big[-\big(\widetilde{\mathrm{PPL}}(s_{t+1+l}) + \lambda C_{\mathrm{UE}}(s_{t+1+l})\big) + \gamma V_\phi(s_{t+2+l}) - V_\phi(s_{t+1+l}) \Big]. \tag{16}$$

Based on this, we can compute $L^{\mathrm{CLIP}}(\theta)$, which consequently facilitates the update of θ.

By incorporating this model-based approach, we achieve substantial gains in computational efficiency in Table 3, enabling the RL agent to flexibly accommodate changing deployment environments.

Finally, we summarize the algorithm in Algorithm 1.

5 Simulation settings and results

5.1 Simulation setup

To validate the effectiveness of this RL-based adaptive LLM splitting point determination method, we use a 32-layer LLM, the LLAMA2-7B model (Touvron et al., 2023), and the WikiText-2 dataset (Merity et al., 2016) which contains 4355 sentences with an average length of 20 words, to evaluate the PPL under varied network conditions. Particularly, we simulate a changing, fading-induced packet loss probability in the range between 0 and 0.3. Moreover, we primarily consider three representative cases:

1. Case L: a low packet loss probability 0–0.1 and an initial splitting point p_{init_L} near the input (layers 1–5);

2. Case H: a high packet loss probability 0.1–0.3 and an initial splitting point p_{init_H} far from the input (layers 6–10);

3. Case A: complete range of packet loss probability 0–0.3 and the initial splitting point p_{init_A} (layers 1–10).

In addition, the default hyperparameters for the PPO (Schulman et al., 2017) algorithm and the channels are given in Table 4.

Table 3 Performance comparison of proximal policy optimization (PPO) with or without reward surrogate model under case *A*

Metric	Without surrogate	With surrogate
Reward at 24 000 steps	2.9736	2.9663
Training duration	>24 d	7.7 min
Computational resource consumption	16.3 GB	<1 GB

Algorithm 1 PPO with reward surrogate model for adaptive splitting point determination in wireless LLM inference

Initialize the policy network parameter θ and value function network parameter ϕ
Initialize the learning rate α, discount factor γ, GAE parameter ξ, and clipping threshold ϵ
Initialize the surrogate model parameter θ_{surr}
Initialize the replay buffer $\mathcal{D}$
Initialize the flag USE_SURROGATE $\leftarrow$ False
Initialize the EPOCH_COUNTER $\leftarrow 0$
Set threshold T for starting surrogate model training
for each training epoch **do**
  EPOCH_COUNTER $\leftarrow$ EPOCH_COUNTER $+ 1$
  for each interaction step t **do**
    Observe state s_t
    Select action a_t according to $\pi_\theta(\cdot|s_t)$
    Execute action a_t and obtain reward r_t. The environment transits to state s_{t+1}
    if USE_SURROGATE **then**
      Store transition (s_t, a_t, s_{t+1}) in replay buffer $\mathcal{D}$
    else
      Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$
    end if
  end for
  Sample a mini-batch of transitions $\Phi \sim \mathcal{D}$
  for each transition in mini-batch **do**
    if USE_SURROGATE **then**
      Compute surrogate reward $\widetilde{\mathrm{PPL}}(s_{t+1})$
      Update the advantage estimator $\hat{A}_t$ using surrogate reward with Eq. (16)
    else
      Compute the advantage estimator $\hat{A}_t$ using GAE from Eq. (12)
    end if
    Compute the clipped surrogate objective $L^{\mathrm{CLIP}}(\theta)$ with Eq. (11)
    Perform gradient ascent on $L^{\mathrm{CLIP}}(\theta)$ with function (13)
  end for
  if not USE_SURROGATE and EPOCH_COUNTER $\geq T$ **then**
    Train surrogate model $\widetilde{\mathrm{PPL}}(p; \sigma, m, \theta_{\mathrm{surr}})$ to minimize MSE with Eq. (14)
    Set USE_SURROGATE $\leftarrow$ True
  end if
end for

Table 4 PPO algorithm hyperparameters

Hyperparameter	Value
Learning rate α	0.0003
Discount factor γ	0.99
Clipping parameter ϵ	0.2
Update frequency n_{step}	400
Batch size B	100
Number of steps per episode	5
GAE parameter ξ	0.95

To reduce the computational cost of evaluating the LLM performance at each step, by collecting 9718 pieces of practical records, we derive a reward surrogate model as in Section 4.3. Our results show that a multi-layer perceptron (MLP) yields a test loss of 0.00 548 in MSE and 0.050 in mean absolute error (MAE), thus providing sufficient accuracy. Therefore, we use this MLP-based reward surrogate model to accelerate the evaluation process.

5.2 Simulation results

We first present the performance of PPO with or without the reward surrogate model and compare the two circumstances with baseline RL schemes (i.e., advantage actor–critic (A2C) (Mnih et al., 2016) and deep Q-network (DQN) (Mnih et al., 2015)). Fig. 5 presents the corresponding results. It can be observed from Fig. 5 that for case H and case A, PPO yields significantly superior performance than A2C and DQN, while for case L, most of the RL approaches lead to similar performance. Moreover, PPO trained with reward surrogate model closely resembles that with actual rewards. Furthermore, Table 3 compares the PPO process with or without reward surrogate model under case A in terms of reward, training duration, and computational resource consumption. The results indicate that the reward surrogate model significantly reduces the training time and computational resource consumption while achieving comparable rewards.

Fig. 6 presents a violin plot comparing the reward distributions of four different strategies: the trained PPO agent with true reward training, the trained PPO agent using the reward surrogate model, a random policy, and an untrained PPO agent. The plot shows that the trained agents, both standard and MBRL-enhanced ones, lead to higher average rewards and tighter reward distribution, indicating more consistent and superior performance compared to the random and untrained agents.

Fig. 7 illustrates 500 splitting points determined by the trained PPO agent under case A, and complements a locally estimated scatterplot smoothing (LOESS) (Cleveland, 1979) trend line, whose slope and R-squared value provide quantitative insights into the relationship between channel conditions and

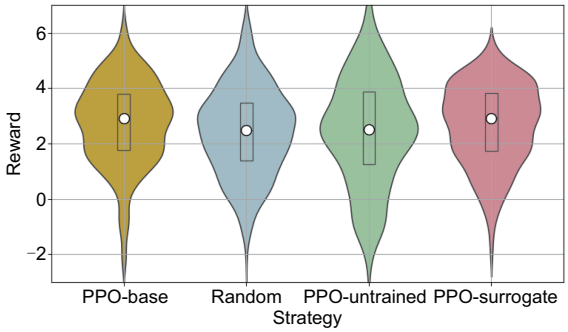

Fig. 6 Reward distribution for different strategies. This violin plot compares the reward distributions across four different strategies. The width of each violin represents the density of rewards at different values, with wider sections indicating a higher probability of observing rewards in that range. The central white dot represents the median reward, while the thick black bar in the center denotes the interquartile range (IQR)

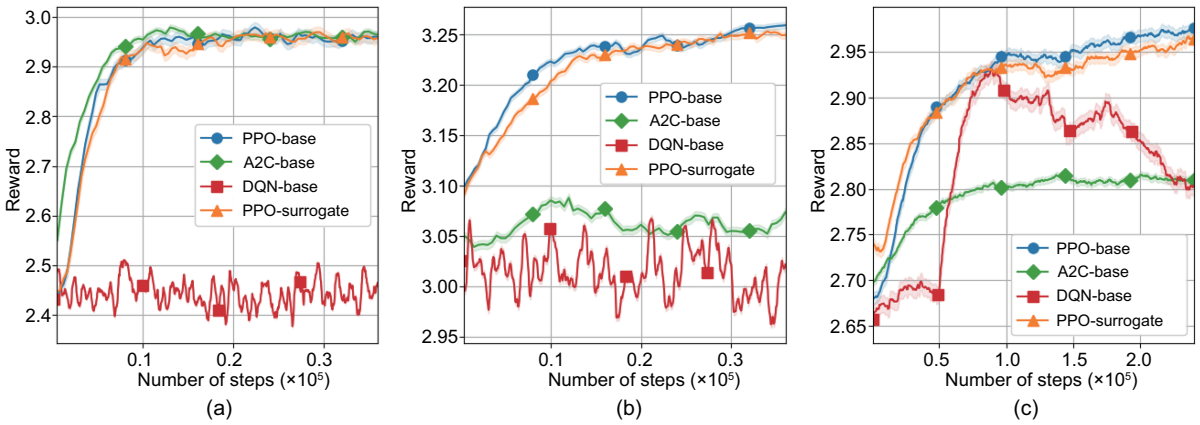

Fig. 5 Comparison of training performances for different reinforcement learning (RL) approaches under case L (a), case H (b), and case A (c)

splitting point decisions. It can be observed from Fig. 7 that as noise intensity σ increases, the agent prefers to place the splitting point farther from the input layers. This strategic adjustment helps mitigate the adverse effects of noise on model performance by leveraging cloud's more robust processing capabilities.

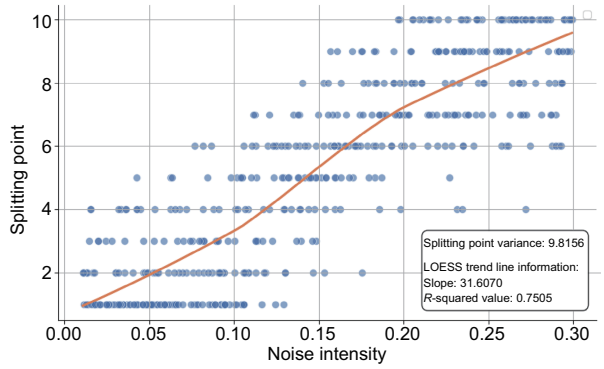

Fig. 7 Splitting points determined by the trained PPO agent across different noise intensities, along with a LOESS trend line

In addition to the baseline simulation where the action space is limited to single-layer adjustments, we conduct further simulations with enlarged action spaces to evaluate the impact of larger adjustments on the training process and final rewards. As shown in Fig. 8, allowing larger adjustments (e.g., moving by two or three layers) leads to slower convergence but ultimately achieves higher rewards. This trade-off suggests that while a larger action space can explore a wider range of configurations, it may require more training steps to stabilize. However, in both simulation and practical scenarios, single-layer adjustments offer notable advantages. They provide fine-grained control over the splitting point, allowing the model to quickly adapt to changes in channel conditions. This is particularly beneficial in dynamic environments where frequent and subtle adjustments are necessary to maintain the optimal performance.

The performance impact of various hyperparameter settings on PPO training is analyzed in Fig. 9. Fig. 9a indicates that higher learning rates ($\alpha = 0.0005$ and 0.0007) lead to faster initial learning but may introduce higher variance in the rewards. Fig. 9b demonstrates that larger batch sizes ($B = 150$ and 200) generally result in smoother and more stable reward curves, yielding better gradient estimates. Fig. 9c reveals that moderate clip ranges

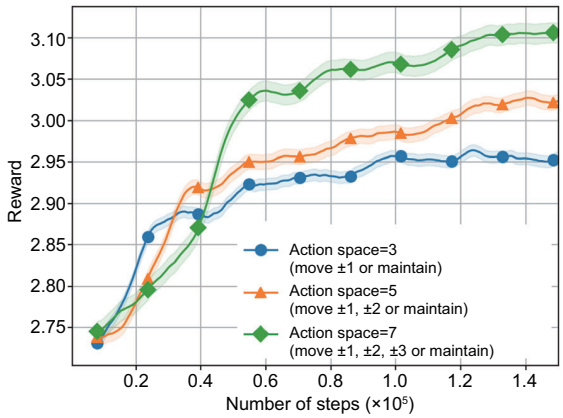

Fig. 8 Comparison of training performances across different action spaces (action space=3, 5, or 7) for varying movement ranges

(ϵ=0.2 and 0.3) strike a balance between stability and performance, whereas too small or too large clip ranges can degrade the performance. Fig. 9d shows that a larger GAE parameter (ξ=0.99) produces more impressive long-term reward accumulation, emphasizing the importance of temporal smoothing in advantage estimation.

6 Conclusions and future works

In this paper, we presented an MBRL framework for dynamically optimizing the splitting point of LLMs deployed across the UE and the edge, so as to enhance the efficiency and performance of LLMs under wireless network conditions. In particular, we formulated the problem as an MDP, and introduced a reward surrogate model to significantly shorten the overall training time. The simulation results demonstrated the framework efficacy in managing the trade-off between inference performance and computational load at the UE. Meanwhile, comprehensive validations in mainstream open-source LLMs clearly demonstrated that an earlier model splitting could worsen the point inference performance, which might provide an independent interest to the community. Our proposed framework offers a structured approach to dynamically deploying LLMs across heterogeneous devices. In practical applications, such as in smart cities and industrial IoT, this framework can enhance the flexibility of LLM deployment while alleviating the computational constraints associated with running LLMs on edge devices.

Despite these achievements, several limitations and challenges remain. Though the validation of the

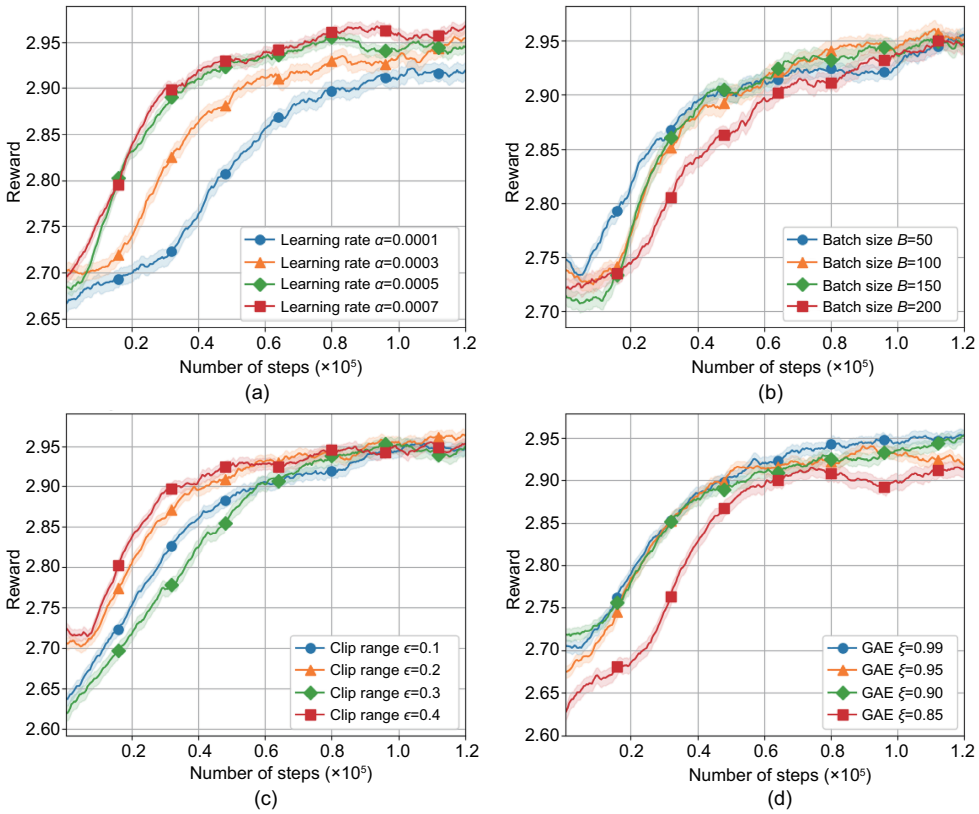

Fig. 9 Impact of various hyperparameters with respect to different numbers of training steps in the proximal policy optimization (PPO) algorithm: (a) learning rate α; (b) batch size B; (c) clip range ϵ; (d) GAE parameter ξ

impact of splitting points on the performance of some widely adopted LLMs has been provided, given the versatility of LLMs, the generality issue still needs further attention. The lack of a more accurate channel model and the absence of communication-efficient distribution learning approaches (e.g., quantization) in data transmission also demand future research. Furthermore, future research could explore adaptive mechanisms that dynamically adjust the action space based on the current learning phase or environmental conditions, thereby balancing the need for quick convergence and high reward attainment. Additionally, further investigation is required into the scalability of our framework in larger, more complex network environments and its generalization across different LLM architectures. We will explore these important directions in the future.

Contributors

Yuxuan CHEN and Rongpeng LI designed the research. Yuxuan CHEN performed the simulation, processed the data, and drafted the paper. Rongpeng LI and Honggang ZHANG helped organize the paper. Honggang ZHANG, Zhifeng ZHAO, and Xiaoxue YU revised the paper. Yuxuan CHEN and Rongpeng LI finalized the paper.

Conflict of interest

Honggang ZHANG is a guest editor of this special issue; he was not involved with the peer review process of this paper. All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available in zjunice github at https://github.com/zjunice.

References

Abbas N, Zhang Y, Taherkordi A, et al., 2018. Mobile edge computing: a survey. *IEEE Int Things J*, 5(1):450-465. https://doi.org/10.1109/JIOT.2017.2750180

Bai YT, Jones A, Ndousse K, et al., 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. https://arxiv.org/abs/2204.05862

Beaulieu NC, Cheng C, 2005. Efficient Nakagami-m fading

channel simulation. *IEEE Trans Veh Technol*, 54(2):413-424. https://doi.org/10.1109/TVT.2004.841555

Brown TB, Mann B, Ryder N, et al., 2020. Language models are few-shot learners. Proc 34th Int Conf on Neural Information Processing Systems, Article 159. https://doi.org/10.5555/3495724.3495883

Chen L, Ahmed NK, Dutta A, et al., 2024. The landscape and challenges of HPC research and LLMs. https://arxiv.org/abs/2402.02018

Chen MZ, Gündüz D, Huang KB, et al., 2021. Distributed learning in wireless networks: recent progress and future challenges. *IEEE J Sel Areas Commun*, 39(12):3579-3605. https://doi.org/10.1109/JSAC.2021.3118346

Chen YX, Li RP, Zhao ZF, et al., 2024. NetGPT: an AI-native network architecture for provisioning beyond personalized generative services. *IEEE Netw*, 38(6):404-413. https://doi.org/10.1109/MNET.2024.3376419

Cleveland WS, 1979. Robust locally weighted regression and smoothing scatterplots. *J Am Stat Assoc*, 74(368):829-836. https://doi.org/10.1080/01621459.1979.10481038

Deisenroth MP, Rasmussen CE, 2011. PILCO: a model-based and data-efficient approach to policy search. Proc 28th Int Conf on Machine Learning, p.465-472. https://doi.org/10.5555/3104482.3104541

Dong QF, Chen XL, Satyanarayanan M, 2024. Creating edge AI from cloud-based LLMs. Proc 25th Int Workshop on Mobile Computing Systems and Applications, p.8-13. https://doi.org/10.1145/3638550.3641126

Egorov V, Shpilman A, 2022. Scalable multi-agent model-based reinforcement learning. Proc 21st Int Conf on Autonomous Agents and Multiagent Systems, p.381-390.

Gemini Team Google, 2023. Gemini: a family of highly capable multimodal models. https://arxiv.org/abs/2312.11805

Gupta O, Raskar R, 2018. Distributed learning of deep neural network over multiple agents. *J Netw Comput Appl*, 116:1-8. https://doi.org/10.1016/j.jnca.2018.05.003

Gupta R, Sosio N, 2024. Introducing Prem-1B. https://blog.premai.io/introducing-prem-1b/

Hadi MU, Tashi QA, Qureshi R, et al., 2023. A survey on large language models: applications, challenges, limitations, and practical usage. https://www.techrxiv.org/doi/full/10.36227/techrxiv.23589741.v1

Icarte RT, Klassen TQ, Valenzano R, et al., 2023. Learning reward machines: a study in partially observable reinforcement learning. *Artif Intell*, 323:103989. https://doi.org/10.1016/j.artint.2023.103989

Jiang AQ, Sablayrolles A, Mensch A, et al., 2023. Mistral 7B. https://arxiv.org/abs/2310.06825

Jin MY, Yu QK, Shu D, et al., 2024. Health-LLM: personalized retrieval-augmented disease prediction system. https://arxiv.org/abs/2402.00746

Kaddour J, Harris J, Mozes M, et al., 2023. Challenges and applications of large language models. https://arxiv.org/abs/2307.10169

Kaiser L, Babaeizadeh M, Milos P, et al., 2019. Model-based reinforcement learning for Atari. https://arxiv.org/abs/1903.00374

Karjee J, Naik SP, Anand K, et al., 2022. Split computing: DNN inference partition with load balancing in IoT-edge platform for beyond 5G. *Meas Sens*, 23:100409. https://doi.org/10.1016/j.measen.2022.100409

Ke CH, Astuti L, 2023. Applying multi-agent deep reinforcement learning for contention window optimization to enhance wireless network performance. *ICT Express*, 9(5):776-782. https://doi.org/10.1016/j.icte.2022.07.009

Lan Q, Zeng QS, Popovski P, et al., 2021. Progressive feature transmission for split inference at the wireless edge. https://arxiv.org/abs/2112.07244

Le Scao T, Fan A, Akiki C, et al., 2022. BLOOM: a 176B-parameter open-access multilingual language model. https://arxiv.org/abs/2211.05100

Lee J, Lee H, Choi W, 2023. Wireless channel adaptive DNN split inference for resource-constrained edge devices. *IEEE Commun Lett*, 27(6):1520-1524. https://doi.org/10.1109/LCOMM.2023.3269769

Letaief KB, Shi YM, Lu JM, et al., 2022. Edge artificial intelligence for 6G: vision, enabling technologies, and applications. *IEEE J Sel Areas Commun*, 40(1):5-36. https://doi.org/10.1109/JSAC.2021.3126076

Li E, Zeng LK, Zhou Z, et al., 2020. Edge AI: on-demand accelerating deep neural network inference via edge computing. *IEEE Trans Wirel Commun*, 19(1):447-457. https://doi.org/10.1109/TWC.2019.2946140

Li X, Lu LY, Ni W, et al., 2022. Federated multi-agent deep reinforcement learning for resource allocation of vehicle-to-vehicle communications. *IEEE Trans Veh Technol*, 71(8):8810-8824. https://doi.org/10.1109/TVT.2022.3173057

Li YX, 2017. Deep reinforcement learning: an overview. https://arxiv.org/abs/1701.07274

Lin B, Zhang C, Peng T, et al., 2024. Infinite-LLM: efficient LLM service for long context with DistAttention and distributed KVCache. https://arxiv.org/abs/2401.02669

Lin Z, Qu GQ, Chen QY, et al., 2024a. Pushing large language models to the 6G edge: vision, challenges, and opportunities. https://arxiv.org/abs/2309.16739

Lin Z, Qu GQ, Chen XH, et al., 2024b. Split learning in 6G edge networks. *IEEE Wirel Commun*, 31(4):170-176. https://doi.org/10.1109/MWC.014.2300319

Liu D, Sun CJ, Yang CY, et al., 2020. Optimizing wireless systems using unsupervised and reinforced-unsupervised deep learning. *IEEE Netw*, 34(4):270-277. https://doi.org/10.1109/MNET.001.1900517

Luong NC, Hoang DT, Gong SM, et al., 2019. Applications of deep reinforcement learning in communications and networking: a survey. *IEEE Commun Surv Tutor*, 21(4):3133-3174. https://doi.org/10.1109/COMST.2019.2916583

Mach P, Becvar Z, 2017. Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commun Surv Tutor*, 19(3):1628-1656. https://doi.org/10.1109/COMST.2017.2682318

Mao YY, You CS, Zhang J, et al., 2017. A survey on mobile edge computing: the communication perspective. *IEEE Commun Surv Tutor*, 19(4):2322-2358. https://doi.org/10.1109/COMST.2017.2745201

Merity S, Xiong CM, Bradbury J, et al., 2016. Pointer sentinel mixture models. https://arxiv.org/abs/1609.07843

Mnih V, Kavukcuoglu K, Silver D, et al., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529-533. https://doi.org/10.1038/nature14236

Mnih V, Badia AP, Mirza M, et al., 2016. Asynchronous methods for deep reinforcement learning. Proc 33rd Int Conf on Machine Learning, p.1928-1937. https://doi.org/10.5555/3045390.3045594

Moerland TM, Broekens J, Plaat A, et al., 2023. Model-based reinforcement learning: a survey. *Foundat Trends® Mach Learn*, 16(1):1-118. https://doi.org/10.1561/2200000086

Nakagami M, 1960. The *m*-distribution—a general formula of intensity distribution of rapid fading. In: Hoffman WC (Ed.), Statistical Methods in Radio Wave Propagation. Pergamon, UK, p.3-36. https://doi.org/10.1016/B978-0-08-009306-2.50005-4

Nijkamp E, Pang B, Hayashi H, et al., 2022. CodeGen: an open large language model for code with multi-turn program synthesis. https://arxiv.org/abs/2203.13474

Ong I, 2024. Efficient Distributed LLM Inference with Dynamic Partitioning. Technical Report UCB/EECS-2024-108, California, USA.

OpenAI, 2023. GPT-4 Technical Report, San Francisco, USA.

Patil R, Gudivada V, 2024. A review of current trends, techniques, and challenges in large language models (LLMs). *Appl Sci*, 14(5):2074. https://doi.org/10.3390/app14052074

Pham QV, Fang F, Ha VN, et al., 2020. A survey of multi-access edge computing in 5G and beyond: fundamentals, technology integration, and state-of-the-art. *IEEE Access*, 8:116974-117017. https://doi.org/10.1109/ACCESS.2020.3001277

Qian YC, Wu J, Wang R, et al., 2019. Survey on reinforcement learning applications in communication networks. *J Commun Inform Netw*, 4(2):30-39. https://doi.org/10.23919/JCIN.2019.8917870

Qiao LT, Zhou Y, 2023. Timely split inference in wireless networks: an accuracy-freshness tradeoff. *IEEE Trans Veh Technol*, 72(12):16817-16822. https://doi.org/10.1109/TVT.2023.3294494

Romoff J, Henderson P, Piché A, et al., 2018. Reward estimation for variance reduction in deep reinforcement learning. Proc 2nd Conf on Robot Learning, p.674-699.

Rozière B, Gehring J, Gloeckle F, et al., 2023. Code LLAMA: open foundation models for code. https://arxiv.org/abs/2308.12950

Ryu J, Won D, Lee Y, 2022. A study of split learning model. 16th Int Conf on Ubiquitous Information Management and Communication, p.1-4. https://doi.org/10.1109/IMCOM53663.2022.9721798

Satyanarayanan M, Bahl P, Caceres R, et al., 2009. The case for VM-based cloudlets in mobile computing. *IEEE Pervas Comput*, 8(4):14-23. https://doi.org/10.1109/MPRV.2009.82

Schulman J, Wolski F, Dhariwal P, et al., 2017. Proximal policy optimization algorithms. https://arxiv.org/abs/1707.06347

Shlezinger N, Farsad N, Eldar YC, et al., 2021. Model-based machine learning for communications. https://arxiv.org/abs/2101.04726

Stone M, 1974. Cross-validatory choice and assessment of statistical predictions. *J Roy Stat Soc Ser B*, 36(2):111-133. https://doi.org/10.1111/j.2517-6161.1974.tb00994.x

Thirunavukarasu AJ, Ting DSJ, Elangovan K, et al., 2023. Large language models in medicine. *Nat Med*, 29(8):1930-1940. https://doi.org/10.1038/s41591-023-02448-8

Touvron H, Martin L, Stone K, et al., 2023. LLAMA 2: open foundation and fine-tuned chat models. https://arxiv.org/abs/2307.09288

Üstün A, Aryabumi V, Yong ZX, et al., 2024. Aya model: an instruction finetuned open-access multilingual language model. Proc 62nd Annual Meeting of the Association for Computational Linguistics, p.15894-15939. https://doi.org/10.18653/v1/2024.acl-long.845

Wang G, Cheng SJ, Zhan XY, et al., 2023. OpenChat: advancing open-source language models with mixed-quality data. https://arxiv.org/abs/2309.11235

Wang YZ, Guo K, Hong W, et al., 2023. Split learning in wireless networks: a communication and computation adaptive scheme. IEEE/CIC Int Conf on Communications in China, p.1-6. https://doi.org/10.1109/ICCC57788.2023.10233330

Webb T, Holyoak KJ, Lu HJ, 2023. Emergent analogical reasoning in large language models. *Nat Hum Behav*, 7(9):1526-1541. https://doi.org/10.1038/s41562-023-01659-w

Wei J, Bosma M, Zhao VY, et al., 2021. Finetuned language models are zero-shot learners. https://arxiv.org/abs/2109.01652

Wu SJ, Irsoy O, Lu S, et al., 2023. BloombergGPT: a large language model for finance. https://arxiv.org/abs/2303.17564

Yang K, Shi CS, Shen C, et al., 2023. Offline reinforcement learning for wireless network optimization with mixture datasets. *IEEE Trans Wirel Commun*, 23(10):12703-12716. https://doi.org/10.1109/TWC.2024.3395624

Zhang MJ, Cao JN, Shen XM, et al., 2024. EdgeShard: efficient LLM inference via collaborative edge computing. https://arxiv.org/abs/2405.14371

Zhang XH, Yu BW, Yu HY, et al., 2023. Wider and deeper LLM networks are fairer LLM evaluators. https://arxiv.org/abs/2308.01862

Zhu LW, Takami G, Kawahara M, et al., 2022. Alleviating parameter-tuning burden in reinforcement learning for large-scale process control. *Comput Chem Eng*, 158:107658. https://doi.org/10.1016/j.compchemeng.2022.107658