Intermediary Output Caching for Diffusion Model-Based Text-to-Image GenAI Services in Edge Computing Networks

YUXIN LI, Guizhou University, Guiyang, China
JIANGTIAN NIE, Nanyang Technological University, Singapore, Singapore
SHAOBO LI, Guizhou University, Guiyang, China
KEBING JIN, Guizhou University, Guiyang, China
JIANHANG TANG*, Guizhou University, Guiyang, China
YANG ZHANG, Nanjing University of Aeronautics and Astronautics, Nanjing, China
DUSIT NIYATO, College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore

The remarkable advancement of generative artificial intelligence (GenAI) has driven revolutionary applications for text-to-image generation, like Stable Diffusion and Imagen. Especially, the diffusion model can generate stunning images from natural language descriptions by using a reverse continuous denoising process. However, the computation burden of diffusion model-based GenAI services poses a significant hurdle for their practical implementation. In this work, we propose a novel edge computing-assisted GenAI framework to enable efficient GenAI service provision, where the intermediate output generated by diffusion models can be cached on edge servers and reused by various users to improve edge computing resource utilization. Assuming the existence of causally correlated auxiliary information, a long-term caching problem is formulated under intra-time-slot caching constraints by considering various maximally reusable steps of diffusion model-based GenAI tasks and the available caching capacity at edge servers. By leveraging the Lyapunov optimization framework, we transform the time-average caching problem into several deterministic problems for different time slots. We develop a deep reinforcement learning-based caching (DRC) algorithm to obtain caching decisions in each time slot, where a deep neural network (DNN)-based caching action generation module and a model-based evaluation module are designed. Finally, we conduct extensive simulation experiments by comparing the DRC algorithm with benchmark algorithms. The simulation results depict that the proposed DRC algorithm can reduce response time and improve cache hit rates significantly. The code is available at https://gitee.com/pipihinsky/DRC.

Additional Key Words and Phrases: GenAI services, diffusion model, intermediary output caching, edge computing

*Corresponding author

Authors' Contact Information: Yuxin Li, Guizhou University, Guiyang, Guizhou, China; e-mail: gs.liyx23@gzu.edu.cn; Jiangtian Nie, Nanyang Technological University, Singapore, Singapore, Singapore; e-mail: jnie001@e.ntu.edu.sg; Shaobo Li, Guizhou University, Guiyang, Guizhou, China; e-mail: lishaobo@gzu.edu.cn; Kebing Jin, Guizhou University, Guiyang, Guizhou, China; e-mail: kbjin@gzu.edu.cn; Jianhang Tang, Guizhou University, Guiyang, China; e-mail: jhtang@gzu.edu.cn; Yang Zhang, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China; e-mail: yangzhang@nuaa.edu.cn; Dusit Niyato, College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore; e-mail: DNIYATO@ntu.edu.sg.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 1558-3465/2025/4-ART
https://doi.org/10.1145/3733106

Just Accepted

1 Introduction

Generative Artificial Intelligence (GenAI) has gained significant attention due to its unprecedented ability to automate the creation of various content, for example, text, images, and videos [12]. Unlike previous technologies, the goal of GenAI is broader, aiming to create new original content that is similar to but not entirely identical to the training data. GenAI is undoubtedly the next major trend to lead the era, bringing tremendous changes to multiple fields and driving innovation and development in various industries with its immeasurable potential. Recent research has achieved remarkable progress within the realm of GenAI. For instance, ChatGPT adopts reinforcement learning from human feedback to determine the optimal responses to given instructions, whereby the reliability and accuracy of the model can be enhanced over time, enabling it to better understand human preferences in long-duration conversations [28]. Meanwhile, in the field of computer vision, the Stable Diffusion model proposed by Stability AI in 2022 has achieved remarkable success in image generation [8]. Different from previous methods, the generative diffusion model generates high-resolution images by controlling the balance between exploration and exploitation, thus achieving a harmonious unity between the diversity of generated images and the similarity to the training data.

GenAI tasks, especially those based on diffusion models, possess remarkable text-to-image generation characteristics. In such tasks, a seed is utilized to generate an original noise image, and subsequently, multiple denoising steps need to be implemented according to the provided prompt to produce a clear image [14]. In previous works, prompts were provided by users and then uploaded to the cloud or other data processing centers with powerful computing capabilities. After the results are obtained, they are transmitted back to the users' local devices [29]. However, it is extremely time-consuming to transmit the results that are finally generated between edge servers and users through wireless communication channels. Therefore, how to make full use of computation resources of both users and edge servers so that GenAI tasks can be quickly responded to becomes a crucial issue. The unloading of GenAI tasks in the edge environment is thus of great significance, as it aims to address this very problem by optimizing the utilization of available computing resources and minimizing the transmission time, thereby enhancing the efficiency and usability of GenAI applications in such settings.

Whether it is text generation or image synthesis, they may be repeatedly called by a large number of users within a specific time period. Different time requests for the same content make these applications have significant asynchronous content reuse characteristics [51]. For example, in text generation, the same paragraph or sentence may be generated and utilized multiple times due to the similar needs of multiple users. Similarly, in the field of image synthesis, images with specific styles or similar elements may also be frequently created and shared due to widespread demand. This asynchronous content reuse not only improves resource utilization through caching mechanisms, but also demonstrates the high predictability of user request distribution in GenAI applications, which is an important feature of GenAI technology in text and image processing. The advantages of deploying large-scale GenAI services in edge environments are as follows: 1) local inference at edge nodes helps to avoid transmission delays to the cloud; 2) leveraging the local data processing capabilities of edge devices enables end-device privacy protection for sensitive prompts; 3) edge computing offloading strategies dynamically allocate denoising tasks to heterogeneous devices, thereby reducing the computational load on cloud centers. These advantages highlight the irreplaceability of edge environments in terms of ensuring low latency, maintaining data sovereignty, and facilitating the collaboration of heterogeneous resources [36].

Taking GenAI tasks based on diffusion models as an example, during each denoising step, an image is inherently generated, which is termed an intermediary output [40]. These intermediary outputs possess the potential for reusability and shareability across different tasks or iterations. However, in past research endeavors, this potential has been consistently overlooked. Given that effective management and reuse of these intermediary outputs can markedly enhance the system's response speed and efficiency, and with the aim of fully capitalizing on this hitherto untapped resource to curtail the completion time of GenAI tasks, it is of utmost importance to

Just Accepted

integrate caching mechanisms within edge servers. For diffusion-based GenAI service, a novel caching framework for intermediary outputs is proposed in edge computing networks to offer efficient GenAI services. Generally speaking, a user makes multiple requests with a text prompt for different GenAI services. By caching the denoised images at intermediary stages, subsequent denoising processes can be executed by the cached intermediary images, reducing denoising time based on the user's further text prompts instead of restarting a new text-to-image generation process. Similarly, a text prompt can be reused by various users [4, 33, 53]. Previously cached intermediary denoised images can be shared and reused across multiple users, which can effectively reduce the requirements for regenerating the same content. Compared to clear final image, images with noise can have richer and more varied effects in GenAI service according to user instructions [50].

However, edge servers with limited caching capacity, cannot handle diffusion-based GenAI tasks which demand immense computation [2]. In this paper, reducing the text-to-image GenAI services response time by deploying lightweight diffusion models on edge server. Furthermore, intermediary output cached on the edge server effectively minimize redundant computations considering image transimmision and denoising computation [54]. Consequently, we focus our research on the inference and caching of intermediary output of GenAI tasks in edge environments. The contributions of this work are presented as follows.

- To effectively facilitate the efficient deployment of GenAI applications in edge computing network environments, we propose a novel edge computing-assisted GenAI framework to enable efficient GenAI service provision. In the proposed framework, multiple users request GenAI services and edge servers provide resources at the edge of the network. Each GenAI task involves a continuous generating process, where the intermediary outputs can be cached on edge servers and reused by different users.
- By assuming the presence of causally related auxiliary information, we propose an intermediary output caching problem to enhance GenAI inference performance. By considering various maximum reusable steps of GenAI tasks and available caching capacities of edge servers, we aim to minimize the long-term GenAI service provision latency while adhering to resource constraints and ensuring the stability of accumulated latency queues. Leveraging the Lyapunov optimization framework, we decouple the original time-averaged caching problem into deterministic sub-problems for different time slots, which are formulated as Mixed Integer Non-Linear Programming (MINLP) problems.
- We develop a DRC algorithm to solve the deterministic optimization problem in each time slot. A DNN-based action generation module is proposed to produce relaxed caching actions based on the observed system conditions, which can be quantized into integer caching solutions by an order-preserving quantization approach. To evaluate the generated solutions, a model-based critic module is designed using the conventional optimization model, selecting the optimal caching solutions to store in the memory pool for DNN model updating.
- Finally, the effectiveness of the DRC algorithm is evaluated through both theoretical analysis and simulation experiments. By comparing the proposed DRC algorithm with benchmark algorithms, the proposed DRC algorithm can reduce response time and improve cache hit rates significantly.

The rest of this work is shown as follows. Section 2 discusses the related works. In Section 3, we formulate the intermediary output caching problem for diffusion model-based GenAI service provision. In Section 4, we propose a DRL-based caching algorithm for intermediary output. Then, extensive simulation experiments are conducted in Section 5. Finally, Section 6 concludes the work.

2 Related Work

2.1 Application of GenAI in Edge Computing

Recently, there has been a notable surge in research efforts dedicated to exploring and optimizing GenAI services in edge computing environments. Wang et al. [35] proposed a joint optimization algorithm for service decisions,

Just Accepted

computation time, and diffusion steps of diffusion models in edge computing systems, where the interplay between GenAI model deployment, local device constraints, and real-time interactive services was considered to enhance the quality of the generated content. Xu et al. [43] proposed a novel framework for joint model caching and inference to manage pretrained foundation models at edge servers for mobile GenAI services in the Metaverse. The framework aims to optimize the allocation of resources and satisfy user requests by considering latency, energy consumption, and accuracy. Ye et al. [46] developed a two-stage, multi-dimensional resource allocation framework for GenAI, where generative diffusion models and contract theory were used to optimize the quality and reduce the latency of GenAI by adjusting prompt optimization and diffusion denoising steps, as well as CPU cycle frequency and network transmission rate. Xu et al. [42] surveyed the deployment of GenAI applications, such as ChatGPT and Dall-E, in mobile edge networks, focusing on the provision of personalized AIGC services while ensuring user privacy. Li et al. [21] proposed a generative model-driven industrial AIGC collaborative edge learning framework. The framework aims to optimize edge-based execution of GenAI tasks, utilizing a multi-task GenAI computation model and an attention-enhanced multi-agent reinforcement learning algorithm to improve system performance.

In recent years, the integration of GenAI services in wireless edge networks has garnered increasing attention for enhancing system performance and efficiency. Du et al. [11] introduced an AIGC-as-a-Service architecture for Metaverse, where the algorithm was used to optimize the selection of GenAI Service Providers for personalized user experiences in wireless edge networks. Xu et al. [41] proposed an adaptive multi-server collaborative mobile edge computing (MEC) approach for efficient GenAI, where dynamic task workload allocation across multiple edge servers was utilized to minimize task offloading make-span and improve performance. Li et al. [20] investigated a joint content caching and user association strategy to minimize content acquisition latency and handover latency in mobile edge computing (MEC) networks with high user mobility. Zeng et al. [50] proposed a delay-aware parallel offloading GenAI framework in mobile edge computing networks, where a recurrent region proposal prediction algorithm was integrated with a multi-modal parallel diffusion offloading scheme to optimize communication and computing resources while minimizing delay. Deng et al. [10] investigated the deployment and service of pre-trained foundation models in mobile edge networks, where a joint resource allocation and request routing optimization problem was formulated to balance accuracy loss and GenAI cost, and an alternating optimization algorithm was proposed to iteratively solve the problem.

The aforementioned works have explored various approaches to optimize GenAI services in wireless edge networks, focusing on task distribution, resource allocation, and performance improvement. However, these methods often rely on either centralized or overly simplified models that fail to effectively balance the diverse needs of GenAI tasks, particularly in terms of latency and content generation. In this work, we address these limitations by introducing a collaborative framework that combines small and large models to optimize both GenAI service latency and content generation capacity. We also propose a novel problem of splitting inference tasks between shared and local models, allowing for more efficient resource utilization and improved adaptability in real-world edge environments. This approach provides a more balanced and scalable solution for GenAI services, offering superior performance in both communication efficiency and content quality.

2.2 Content Caching in Edge Computing

With the deployment of edge servers, existing content caching methods have been phased out by new features in edge environments, and therefore cannot be directly applied to edge environments. Therefore, researchers are proposing and studying new ideas and technologies for data caching in edge environments. Xia et al. [39] proposed a novel latency-aware online approach to address the online edge data distribution problem, aiming to minimize transmission cost and ensure low latency in dynamic edge caching systems. They combine Lyapunov optimization and game theory to provide a distributed solution with provable performance guarantees. Zyrianoff

Just Accepted

et al. [55] proposed a proactive edge caching framework for the Cloud-to-Things continuum, which decouples the caching strategy algorithm from the underlying architecture to enable customization based on application-specific requirements. Hui et al. [19] investigated a joint content caching and user association strategy to minimize content acquisition latency and handover latency in MEC networks with high user mobility. Wu et al. [37] developed a knowledge cache-driven Personalized Federated Learning architecture to enhance communication efficiency and improve model performance by transferring personalized knowledge from a server-side knowledge cache during training. Zhang et al. [51] studied an edge caching approach for the Internet of Vehicles that uses multi-agent deep reinforcement learning to minimize content distribution delay and improve content hit and success rates in dynamic environments.

Recent studies have focused extensively on edge-based inference, exploring various strategies to optimize performance and reduce latency in resource-constrained environments. Shao et al. [31] investigated task-oriented communication for multi-device cooperative edge inference, proposing a learning-based scheme that optimizes local feature extraction and distributed encoding to reduce communication overhead and latency, while incorporating a selective retransmission mechanism to further enhance performance. Yan et al. [45] studied the joint optimization of model placement and online model splitting decisions to minimize the energy-and-time cost of device-edge co-inference in the presence of wireless channel fading, formulating an optimal stopping problem and deriving an analytical model splitting decision to efficiently optimize the model placement. Long et al. [24] introduced a collaborative learning framework leveraging a cloud-edge-device architecture for next POI recommendations, where the framework efficiently handles inference tasks by reducing on-device computational burdens while offering region-specific and personalized recommendations. Wang et al. [35] introduced a framework designed to optimize resource allocation, decrease inference latency, and maintain result accuracy on CPU-only edge devices, demonstrating a reduction in multi-DNN inference latency. Bajpai and Hanawal [18] developed a distributed inference method for deep neural networks, where Early Exit strategies were utilized to minimize inference latency, aiming to improve accuracy while reducing the response cost from mobile to edge platforms.

Recent research in edge caching and resource allocation has made significant strides in addressing the challenges posed by dynamic and resource-constrained edge environments. However, many existing solutions are limited in their ability to effectively handle the increasing complexity of GenAI task inference, especially when it comes to balancing computational load across edge servers and local devices. While previous works focus on optimizing data distribution, energy consumption, and latency in edge caching, they do not fully investigate the potential of result reuse and sharing from intermediate inference stages. To overcome these limitations, we propose a novel algorithm for GenAI task inference in edge environments. Our approach not only supports inference at both edge servers and local devices but also introduces a caching mechanism that stores intermediate inference results, enabling efficient result reuse and sharing across devices. This innovative strategy enhances computational efficiency, reduces redundant processing, and significantly improves overall system performance in edge-based GenAI applications.

Just Accepted

3 System Model and Problem Formulation

In this section, we first introduce the system architecture of the proposed model, followed by an in-depth discussion of the diffusion model-based GenAI task model and its processes. Next, we delve into the computational model, defining the resources necessary for efficient data processing. We then discuss the communication model, highlighting interactions among various system components. Finally, we outline the caching model designed to optimize data storage. The notations adopted in this section are summarized in Table 1.

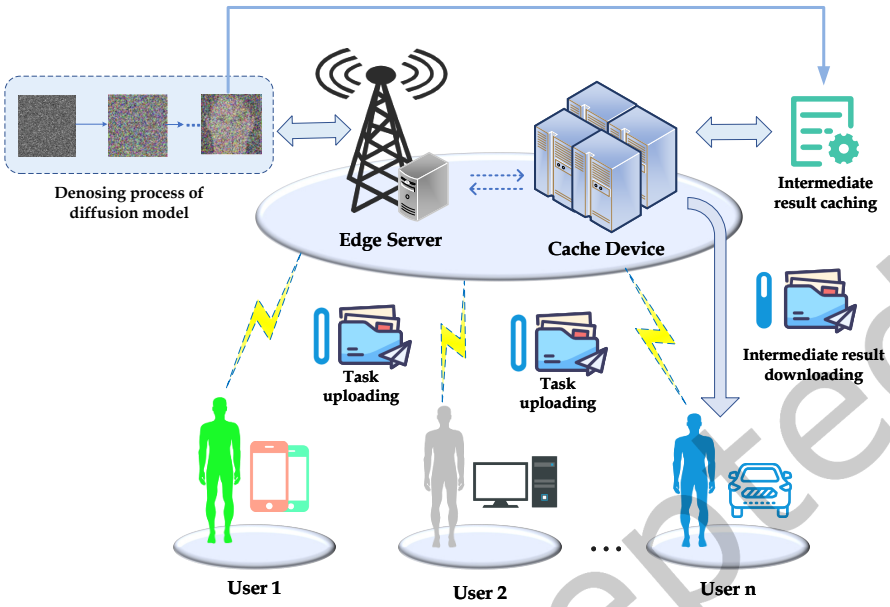

Fig. 1. Intermediary output caching for GenAI tasks in edge computing.

Just Accepted

3.1 System Architecture

In this work, an intermediary output caching framework for diffusion model-based GenAI applications is proposed to offer customized content generation services for multiple users in edge computing systems, as shown in Fig. 1. The proposed framework comprises a powerful edge server serving as the edge computing node and multiple users. Each user with an onboard computer and a limited battery generates a request for GenAI services, allowing them to execute computational tasks. Due to the continuous denoising process of the diffusion model-based GenAI task, a collaborative inference procedure is achieved, where shared inference workloads are completed on the edge server, and intermediate outputs are transmitted to users for final results with remaining denoising steps. The sets of users and GenAI service requests are denoted by $\mathcal{U} = \{1, 2, \ldots, U\}$ and $\mathcal{R} = \{1, 2, \ldots, R\}$, respectively.

The high performance and superior generative quality of diffusion models always come at the expense of larger model sizes and increased computational overheads. These diffusion models impose immense computational demands, often necessitating cloud-based inference implementations with high-end GPUs [44]. Lightweight diffusion models, like MobileDiT and BK-SDM, can be deployed on edge servers, offering a more efficient solution for real-time image generation [32]. A similar denoising process for GenAI tasks results in computational redundancy, imposing an additional burden on cloud resources. When such denoising processes are centralized within cloud servers, they consume substantial computational resources and introduce latency due to the necessity of transmitting large volumes of data back and forth between the user and the cloud. We can significantly reduce the latency associated with images transmission by deploying lightweight diffusion models on edge servers. Furthermore, intermediary output cached on the edge servers effectively minimize redundant computations. For diffusion models, initial denoising can be done at the edge, and intermediary output can be cached for later steps instead of slowly re-denoising from scratch or waiting for denoising results from the core cloud [3]. Edge server

Table 1. Key notations in System Model

Notation	Description
n	User index
$\mathcal{U}$	Set of users
r	User's request
$\mathcal{R}$	Set of users' requests
t	Time slot index
τ	Length of time slot
T	Set of time slots
m	Currently executed denoising step
S_r	Maximum denoising step for user request r
S_r^*	Maximum denoising step that is reusable and shareable
$\mathrm{MSE}_m(t)$	Mean square error at denoising step m
$\mathrm{PSNR}_m(t)$	Peak signal-to-noise ratio at denoising step m
$D_{r,m}$	Size of the result generated during the denoising process
$I_{r,m}(t)$	Location decision for performing denoising tasks in time slot t
$J_{r,m}(t)$	Caching decision for storing the intermediary output generated by denoising tasks in time slot t
$O_{r,m}(t)$	Caching decision for storing the intermediary output generated by denoising tasks in the previous time slot
R_n	Transmission rate between the user and edge server
W_n	Bandwidth assigned to user n from edge server
p	Transmission power of edge server
δ^2	Gaussian noise power
f_{es}	Computation capacity of edge server
f_n	Local computing capabilities of users
$\mathbb{L}_{n,r,m}^{ES}$	Denoising time consumed at the edge server
$\mathbb{L}_{n,r,m}^{local}$	Denoising time consumed at the local device
$T_{n,r,m}$	Time consumption for downloading results
$C_e(t)$	Time consumption for caching intermediary output
D_e	Maximum storage space of the edge server
V	Lyapunov control parameter
$\bar{\mathcal{L}}$	Long-term average system latency constraint
$\mathcal{Q}(t)$	Cumulative delay of GenAI service provision

Just Accepted

and users execute the GenAI service process within the architecture depicted in Fig. 2. We use r to represent an GenAI service request. The procedure of diffusion model-based GenAI service provision for multiple users with the aid of an edge server is shown as follows, which can be illustrated in Fig. 2. First of all, since GenAI service request r requires diverse content generation capabilities, the user's local device employs semantic extraction to convert the raw data into text prompts. This process reduces the amount of data transmitted using logical instructions to represent GenAI service requests. Subsequently, these text prompts are sent to the edge server for processing. Secondly, the edge server utilizes a pre-trained text-to-image diffusion model-based GenAI task model to enable versatile and controllable image generation based on these text prompts as conditional information. A

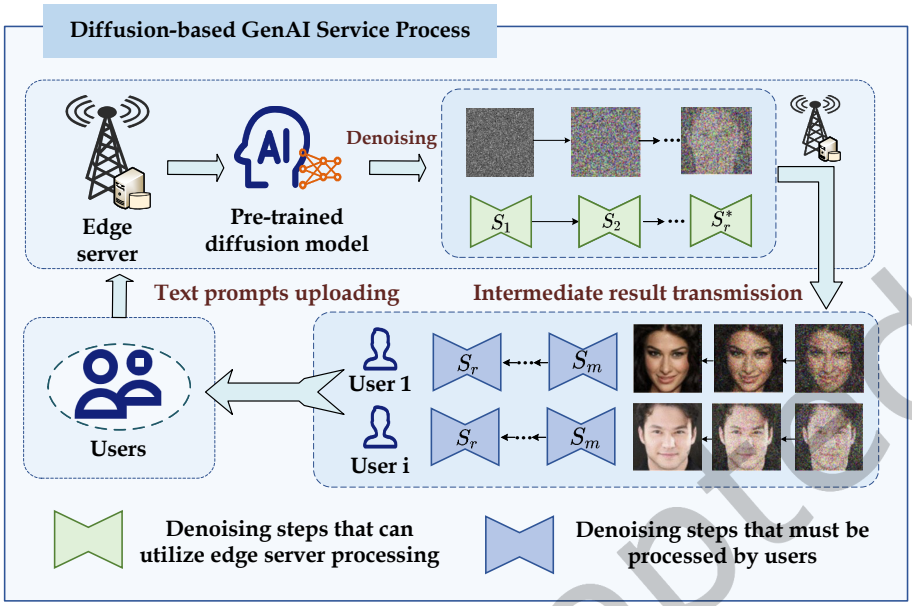

Fig. 2. The process of providing diffusion model-based GenAI services in edge computing network.

preliminary image is generated by the generator based on the input text vector, and this initially generated image often contains a significant amount of noise. To produce an image that is clearer and more aligned with the input text description, the diffusion model typically incorporates a denoising process. Thirdly, to ensure that the image can be smoothly and rapidly restored to a clear state from a noisy one, the early denoising steps are processed using edge server with powerful computing capabilities. We use r to represent a diffusion model-based GenAI service request. For request r, we denote the denoising step as m, which should not exceed the total denoising step S_r. It is noteworthy that, after each denoising operation, a clearer intermediate image with a lower noise level is generated, compared to the previous one. Specifically, the denoising model on the edge server is sampled, and for each denoising step m , an output is generated, which is denoted as $D_{r,m}$. Thereafter, the edge server dispatches the intermediate output to all users to finalize the remaining denoising steps through local computation. In contrast to conventional caching techniques, intermediary output caching in diffusion model-based denoising, as outlined in [40], is specifically designed for applications that necessitate multiple iterations of noise reduction, such as high-quality image generation tasks. This approach substantially reduces redundant computations by facilitating the reuse of intermediary output, thereby optimizing both real-time performance and computational efficiency. By mitigating the need for repetitive processing at each iteration, the efficiency of the entire denoising process is markedly improved, enabling faster convergence and more resource-efficient execution.

The edge server deploys an efficient caching model that caches the intermediary outputs generated during the denoising process, achieving reuse and sharing of intermediary outputs for diffusion model-based GenAI tasks. In image processing, overlapping regions often exist between image blocks. To significantly accelerate the denoising process and optimize computational resources, an efficient strategy is adopted: after the initial denoising stage is completed, the processed intermediary output, along with text prompts, are utilized as the starting point for subsequent steps. By adopting this approach, we not only achieve the reuse of intermediary

Just Accepted

output but also facilitate their sharing through a caching model implemented on edge server. Denote the maximum reusable and shareable denoising steps as S_r^*, with values ranging from 1 to S_r. Specifically, when step m satisfies $m \leq S_r^*$, the intermediate image result generated by step m is reusable and shareable; And when $m > S_r^*$, the intermediate image result generated by step m is not reusable. Through this caching mechanism, edge server not only effectively reduces unnecessary redundant calculations, but also significantly accelerates the processing speed of similar tasks in the future, thereby improving the overall efficiency and performance of the system.

It is assumed that a time slot-based system is considered, where the length of a time slot is set to τ. The set of time slots is denoted by $T = \{0, 1, 2, \ldots\}$. In each time slot, the edge server is responsible for continuously monitoring the denoising process and determining the optimal stopping point to end the denoising process on edge server. Next, a binary variable $I_{r,m}(t)$ is introduced to indicate the execution position of the denoising step m for each request r within each time slot t. Specifically, when $I_{r,m}(t) = 1$, it indicates that the denoising step m for request r will be executed in the edge server, utilizing edge server's computing resources and optimization algorithms for processing. When $I_{r,m}(t) = 0$, it indicates that denoising step m will be performed locally by the user. Once the final step of edge server denoising is determined, edge server will securely transmit the intermediary output generated in the final step to the user's local device, allowing the user to continue with subsequent calculations or processing tasks locally. This design can efficiently and flexibly handle GenAI tasks, giving users great flexibility and autonomy.

3.2 Diffusion Model-based GenAI Task Model

The diffusion model-based GenAI task, encompasses two crucial steps within its core process: (i) semantic extraction and text prompt generation, and (ii) diffusion model-based GenAI task generation. For example, the diffusion model-based GenAI tasks are applied in diverse scenarios such as simulated vehicle visual environment, bespoke advertising visuals, virtual tourism experiences, and realistic product visualizations, so these scenarios all necessitate the generation of a large number of images. These scenarios can be transformed into text-to-image GenAI services and can use text prompts to represent image-generation requirements through semantic extraction. Regarding prompts, transformer-based prompt extraction techniques such as GPT and BERT directly utilize the encoding ability of pre-trained transformer models to identify key semantic units in prompts through self-attention weight visualization. In text-to-image generation, the prompt acts as the guiding instruction that conveys the user's intention and desired visual elements to the generation system. The quality and specificity of the prompt directly influence the final appearance and details of the generated image [49]. Specifically, we extract the *nouns* and their associated key *verbs* from audio or text to designate as the entities. When a driver requests a navigation service through a language command, like "guide the vehicle to an intersection.", the $\{[Guide], [to], [intersection]\}$ will be extracted from this sentence. It is assumed that text prompts are extremely small, such that these transmission time can be disregarded.

The pre-trained diffusion model-based GenAI task model is utilized to generate a noisy initial image from a text prompt, which then undergoes a series of continuous denoising steps, totaling S_r denoising iterations. In the detailed collaborative GenAI task procedure illustrated in Fig. 2, the edge server initiates a shared denoising process for the connected users. Specifically, at each inference step $m \sim \mathcal{S} = \{1, 2, 3, \ldots, S_r\}$, the system samples from the diffusion model and applies a denoising algorithm to refine the image quality progressively. This iterative process results in a series of intermediate denoised outputs, denoted as $\{D_{r,1}, D_{r,2}, \ldots, D_{r,m-1}, D_{r,m}, D_{r,m+1}, \ldots, D_{r,S_r}\}$, where each subsequent output represents a further enhancement in image clarity and detail.

1) **Quantifying Denoising Effectiveness in Diffusion Model:** To quantify the effectiveness of this denoising process, we use two key metrics, mean square error (MSE) and peak signal-to-noise ratio (PSNR), between the original image and the denoised output image to monitor the effectiveness of each round of denoising. Both PSNR and MSE are widely used metrics for evaluating the effectiveness of image-denoising

Just Accepted

algorithms, which can provide quantitative measures of the pixel-level differences between the original and processed images, directly assessing the denoising performance. Specifically, MSE calculates the average squared difference between the corresponding pixel values of the original and processed images [23]. MSE can serve as a measurement of the distortion degree introduced by noise, where higher values of MSE indicate greater deviation from the original image. PSNR (in dB) is derived from MSE and incorporates the maximum possible signal power to provide a more interpretable evaluation of image quality, making it easier to assess the relative quality of the processed image compared to the original. A higher PSNR value indicates that the image has less noise and retains higher fidelity to the original. Thus, PSNR offers an intuitive and standardized measure that complements the numerical output of MSE, allowing for a more comprehensive understanding of image restoration performance [27]. As a result, both PSNR and MSE are proposed for pixel-wise image quality evaluations, which can be used to evaluate the mid-denoising results. When the essential details of an image generation request are thoroughly grasped—including the total number of denoising steps, the precise dimensions of the initial noisy image, and the desired dimensions of the final result—it becomes possible to ascertain the PSNR for each image output during the denoising process. To ensure the effectiveness of the denoising and maintain control over image quality, a critical threshold, designated as P_{max}, is established as the benchmark for evaluation. In practical applications, if the PSNR of an image produced by a denoising step falls below this preset threshold P_{max}, the image is typically considered to contain substantial residual noise and exhibit a lack of sharpness in its contours. Based on this observation, a reasonable hypothesis is formulated: images that score low on PSNR have the potential for diverse generation. In simpler terms, these images can be utilized as raw data to create a multitude of varied images that meet various text description requirements, through additional denoising and the integration of supplementary information, such as text prompts. The MSE and PSNR between the original image and the output image after one round of denoising within each time slot are given by

$$\mathrm{MSE}_m(t) = \frac{1}{v \cdot w} \sum_{i=0}^{v-1} \sum_{j=0}^{w-1} [D_{r,m}^{i,j}(t) - D_{r,m-1}^{i,j}(t)]^2, \forall m \in \{1, \ldots, S_r\}, \tag{1}$$

where $D_{r,m}^{i,j}(t)$ is the pixel value of the original image at coordinates (i, j), $D_{r,m-1}^{i,j}(t)$ is the pixel value of the denoised image at the same coordinates, and v and w are the height and width of the image, respectively.

$$\mathrm{PSNR}_m(t) = 20 \cdot \log_{10} \left(\frac{MAX}{\sqrt{MSE_m(t)}}\right), \forall m \in \{1, \ldots, S_r\}, \tag{2}$$

where MAX is the maximum possible value of image pixel values (for 8-bit grayscale images, MAX is usually set to 255).

It is worth noting that we set the maximum denoising step with $PSNR_m$ less than P_{max} as a key threshold, represented by S_r^*, to indicate the maximum denoising step where the output result is reusable and can be shared by multiple users. Once S_r^* is reached, the subsequent denoising steps are independently completed by each user on their local devices. This flexible partitioning strategy not only alleviates the server's burden but also fully leverages the advantages of distributed computing, thereby accelerating the overall inference process. Users continue to execute the remaining denoising steps based on the received intermediate image results until a high-quality, low-noise image output is ultimately generated.

2) **GenAI Task Execution Model:** Based on the comprehensive consideration of system running time cost and user running time cost, the computational model decides whether the denoising step m of user request r should be executed on the edge server or the local device within each time slot. The binary variable $I_{r,m}(t)$ is utilized to represent whether the denoising step m for user request r is performed at the edge server or locally at each time instant t. Specifically, it is expressed as,

Just Accepted

$$I_{r,m}(t) = \begin{cases} 1, & \text{if the denoising step } m \text{ is executed on the edge server,} \\ 0, & \text{if the denoising step } m \text{ is executed on local.} \end{cases} \tag{3}$$

For the entire system, the operational cost associated with processing all user requests is meticulously divided into the subsequent three pivotal stages: firstly, the time cost incurred by performing denoising operations on edge servers; secondly, the transmission time cost necessary for relaying intermediary output, which have undergone initial denoising, from the edge server to the users' local devices; and finally, the time cost involved in users continuing the denoising process on their local devices utilizing the received intermediary output. Collectively, these three stages constitute the overall operational expenditure required by the entire system to address user requests.

It is presumed that the edge server possesses adequate computational resources, including high-speed multi-core CPUs. As a result, the execution process of text-to-image generation is overlooked, with the primary emphasis placed on the denoising of images. The execution time of the denoising steps on the ES device and on the user's local device are given by the following two formulas, respectively

$$\mathbb{L}^{ES}_{n,r,m}(t) = \frac{\sum_{m=1}^{S_r} I_{r,m}(t) \cdot D_{r,m}}{f_{es}}, \forall n \in \mathcal{U}, r \in \mathcal{R}, \tag{4}$$

where f_{es} represents the computation capacity of the edge server in terms of CPU cycles.

$$\mathbb{L}^{local}_{n,r,m}(t) = \frac{\sum_{m=1}^{S_r} (1 - I_{r,m}(t)) D_{r,m}}{f_n}, \forall n \in \mathcal{U}, r \in \mathcal{R}, \tag{5}$$

where f_n denotes the executable computation capacity of local devices $n \in \{1, \ldots, U\}$, which is posited to exhibit dynamic random fluctuations within the total computation capacity $f_{\max}$.

3.3 Wireless Communication Model

The wireless channels between the edge server and users are assumed to be independently and identically distributed block fading, remaining static within each time slot. Here, we consider the application of Frequency Division Multiple Access (FDMA) in this context. With $g(t)$ representing the fading power gain in different time slots, the available frequency spectrum can be divided into multiple sub-bands using FDMA, and each user or a group of users can be assigned to a specific sub-band for communication. The downlink transmission rate is given by:

$$R_n(t) = W_n \cdot \log_2 \left(1 + \frac{p \cdot g(t)}{\delta^2}\right), \forall n \in \mathcal{U}, \tag{6}$$

where p denotes the transmit power from edge server to users, δ^2 represents the gaussian noise power, and W_n signifies the downlink bandwidth of edge server.

The denoising steps of the image are carried out on the edge server in the early stage. At this time, $I_{r,m}(t)$ is set to 1, indicating that the image is being denoised by the edge server. Assuming that the image ceases denoising on the edge server at step m and transitions to local denoising, at this juncture, $I_{r,m}(t)$ is set to 0, signaling that the denoising task has been shifted from edge server to the local device. It is noteworthy that once the image terminates denoising on the edge server and moves to local denoising, this shift is irreversible, implying that all subsequent denoising steps will be executed locally. The aforementioned procedure is constrained by the following inequality.

$$I_{r,m-1}(t) \geq I_{r,m}(t), \forall r \in \mathcal{R}, \forall m \in \{1, \ldots, S_r\}. \tag{7}$$

Just Accepted

Given a user's request for a denoising position variable set of $\{1, 1, 1, 1, 0, 0\}$, this signifies that during the first four denoising steps, the image is processed by the edge server. From the fifth step onward, the image transitions to local denoising. The transition from $I_{r,m}(t) = 1$ to $I_{r,m}(t) = 0$ marks a pivotal step, necessitating the transfer of intermediate variables outputted from this step from the edge server to the local device. This transmission process will incur a specific delay in time slot t, which we can ascertain as follows:

$$T_{n,r,m}(t) = \frac{(I_{r,m-1}(t) - I_{r,m+1}(t))I_{r,m}(t) \cdot D_{r,m}}{R_n(t)}, \forall n \in \mathcal{U}, r \in \mathcal{R}. \tag{8}$$

3.4 Intermediary output Caching Model

Given a set of users' requests R in time slot t, a data caching strategy to cover those data requests can be presented as $J_r(t) = \{J_1(t), J_2(t), \ldots, J_R(t)\}$, where $J_r(t) = \left\{J_{r,m}(t), \forall m \in S_r\right\}$. The symbol $J_{r,m}(t)$ denotes whether the intermediary output of the denoising step m requested by the user r is cached on edge server:

$$J_{r,m}(t) = \begin{cases} 1, & \text{if the intermediary output of the denoising step } m \text{ is cached on the edge server,} \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The storage resources on an edge server are usually limited. Thus, the competition between users usually makes it impossible for a request to cache all its intermediate image data on every edge server. Thus, the number of intermediate images cached in any time slot on an edge server cannot violate the available server capacity constraint:

$$\sum_{r=1}^{R} \sum_{m=1}^{S_r} J_{r,m}(t) \cdot D_{r,m} \leq D_e, \forall t \in T, \tag{10}$$

where D_e is the maximum storage space of the edge server e. Moreover, for each request r, it can only be cached with at most one intermediary output generated by the denoising step m, with no situation arising where a single request caches multiple intermediary outputs from various denoising steps. Thus we set the following restrictions:

$$\sum_{m=1}^{S_r} J_{r,m}(t) \leq 1, \forall t \in T, r \in \mathcal{R}. \tag{11}$$

As new data is transferred from edge server to users for caching, additional network delay is incurred, resulting in data migration costs that are calculated based on the number of newly cached data items. Here, we use $O_{r,m}(t)$ to denote the caching situation of the output result of the denoising step m in the previous time slot:

$$O_{r,m}(t) = 1 - J_{r,m}(t-1), \forall r \in \mathcal{R}, m \in [1, S_r]\,. \tag{12}$$

We denote $O_{r,m}(t) = 0$ if the output result of the denoising step m is already cached on edge server at the start of time slot t, otherwise $O_{r,m}(t) = 1$.

The data caching cost generated by the caching strategy is a key element of the system cost model, where the data caching cost is measured based on the storage resources used by users in each time slot. Therefore, the data caching cost in time slot t can be calculated as follows:

$$C_e(t) = \sum_{r=1}^{R} \sum_{m=1}^{S_r} \frac{c \cdot I_{r,m}(t) \cdot O_{r,m}(t) \cdot D_{r,m}}{f_e(t)}, \forall t \in T. \tag{13}$$

Just Accepted

where c is the unit cost of using data resources on the edge server for data caching and $f_e(t)$ is the caching capability of edge server to store intermediary outputs within each time slot t, i.e., $0 \leq f_e(t) \leq f_e^{max}$.

The proposed intermediary output caching model can be used for various GenAI services, e.g. image-to-image and text-to-text generation tasks. For example, for the image-to-image GenAI task, the proposed caching model can be used to store the intermediary output generated in denoising process, and can reuse the cached images when processing different tasks [9]. The storage capacity limitation of edge servers (10) reflects the inherent resource constraints in practical deployments. In response to the storage capacity constraints of edge servers, the current strategy prioritizes utilizing high-speed memory (such as DRAM or PCM arrays) to cache frequently accessed intermediary outputs to minimize computational latency. However, memory capacity is limited, making it difficult to support long-term storage of large-scale task data. To address the bottleneck, future work will extend the hierarchical caching architecture, combining the complementary advantages of memory and disk storage: memory will handle high-frequency "hot data" while disk storage will accommodate low-frequency or historical intermediate results [48]. A dynamic migration strategy will be employed to balance speed and capacity requirements.

3.5 GenAI Task Exection Cost Problem Formulation

For the entire system, the cost of all user requests during runtime can be clearly divided into three stages: first, the time cost required for initial denoising on the edge; Second, the transmission time cost for transferring intermediary outputs that have undergone initial processing from the server to the user's local area; And finally, the time cost for further denoising using these intermediary outputs locally. If reusable intermediary outputs are cached in the previous round, the preliminary denoising step on the edge server can be avoided, thereby directly saving the time cost required for this process, i.e. $\mathbb{L}_{n,r,m}^{ES}(t) = 0$. At this point, the overall time consumption only includes two parts: one is the transmission time cost incurred by transferring these cached intermediary outputs from the server to the user's local location. The second is the time cost of using these intermediary outputs locally by the user to continue performing subsequent denoising operations. Considering the diffusion model-based GenAI intermediary output caching system architecture and different types of costs, such as transmission time cost and denoising computation cost, the total system cost is calculated by adding all of the above costs:

$$\mathbb{L}_{n,r,m}(t) = T_{n,r,m}(t) + O_{r,m}(t) \cdot \mathbb{L}_{n,r,m}^{ES}(t) + \mathbb{L}_{n,r,m}^{local}(t), \forall n \in \mathcal{U}, r \in \mathcal{R}. \tag{14}$$

When the last user's request in the system is successfully completed, we can accumulate the total time cost $\mathbb{L}_{1,r,m}(t)$ of the entire system, which comprehensively reflects the time consumption from initial denoising to local processing by the end user:

$$\mathbb{L}_{r,m}(t) = \max\{\mathbb{L}_{n,r,m}(t), n \in U\}, \forall r \in R, m \in [1, S_r]\,. \tag{15}$$

From a user perspective, it is expected to shorten the completion time of GenAI tasks and ensure that task content can be flexibly adjusted according to individual needs; From a system perspective, it is necessary to efficiently execute GenAI tasks and cache and utilize the intermediary output generated by the tasks to maximize their value, promote the reuse and sharing of intermediary outputs with each user. While pursuing this optimization objective, it is also necessary that the time-averaged system latency perceived by the users in the long term be stabilized. Thus, system typically has a long-term average system latency constraint, denoted by $\bar{\mathcal{L}}$, for requests served by users. Therefore, the following inequality must be fulfilled:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \frac{\sum_{r\in\mathcal{R}} \sum_{m\in S} O_{r,m}(t) \cdot \mathbb{L}_{r,m}(t)}{\sum_{r\in\mathcal{R}} \sum_{m\in S} O_{r,m}(t)} \leq \bar{\mathcal{L}}. \tag{16}$$

Our goal is to mininize GenAI service provision time, which can be formulated as a long-term optimization problem.

Just Accepted

$$\mathcal{P}_1 : \min \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{r=1}^{R} \sum_{m=1}^{S_r} \mathbf{E}\left[\mathbb{L}_{r,m}(t)\right], \tag{17}$$

$$\text{s.t.} \quad (3), (7), (9), (10), (11)$$

$$0 \le f_{es} \le f_{es}^{\max}, \tag{18}$$

$$0 \le f_n \le f_n^{\max}, \forall n \in \mathcal{U}, \tag{19}$$

$$W_n \ge 0, \forall n \in \mathcal{U}, \tag{20}$$

$$0 \le \sum_{n=1}^{U} W_n \le W_{max}, \tag{21}$$

$$0 \le \sum_{t=0}^{T} C_e(t) \le C_e^{max}. \tag{22}$$

Constraint (3) is defined as the determination of the location for the denoising steps in GenAI tasks. Constraint (7) indicates that the transition of the denoising position of the task from the edge server to the local is irreversible. Constraint (9) represents the caching status of the intermediary outputs that are output by the denoising step of the task. The maximum storage value of the edge server is imposed by constraint (10). Constraint (11) is stated to indicate that each user request r can have at most one cached intermediary output that is output by the denoising step m. The maximum CPU cycle frequency for the edge server and all local users is constrained by Constraints (18) and (19), respectively. In addition, constraint (20) explicitly stipulates that the transmission power from the edge server to the user must be nonnegative, while Constraint (21) further limits the sum of all transmission powers to not exceed W_{max}. To avoid unlimited caching, we also set a caching time constraint for intermediary output in Constraint (22).

4 Deep Reinforcement Learning-based Caching Algorithm

This section utilizes the DRC algorithm to explore intermediary output caching decisions associated with GenAI tasks.

4.1 Problem Transformation

For the caching problem, it is crucial to adopt algorithms capable of handling stochastic processes due to the uncertainty in user request and the dynamic changes in edge computing network. Lyapunov optimization, by defining a scalar function, can effectively assess system stability and guide optimal caching decision, ensuring good performance even under fluctuating network conditions [1]. Through the application of Lyapunov optimization, the problem $\mathcal{P}_1$ can be structured to optimize the time averages of particular objectives while adhering to specific time average constraints [17]. Meanwhile, Lyapunov optimization has no prior information requirement on the channel statistics or the computation task request process [25]. Specifically, the proposed DRC algorithm applies Lyapunov optimization to decouple the multi-stage stochastic MINLP into deterministic per-slot MINLP subproblems. It guarantees to satisfy all the long-term constraints by solving the per-timeslot subproblems that are much smaller in size. Then, the DRC algorithm solve the per-timeslot MINLP caching problems with very low computational complexity.

In order to optimally solve the constrained optimization problem that involves minimizing costs in GenAI tasks, it is theoretically necessary for detailed and comprehensive information about the system to be available across all time slots. However, this idealized requirement is often challenging to fulfill in the complex and ever-changing scenarios of the real world, where obtaining comprehensive information is not only costly but also technically

Just Accepted

demanding. Given this context, a more practical and efficient strategy must be adopted to effectively meet the long-term delay constraints of user requests. This involves ingeniously transforming the original non-convex problem P1 into an easily solvable linear convex problem through mathematical transformations and algorithm design. Such a conversion not only significantly reduces computational complexity but also allows for better adaptation to dynamic changes and uncertainties in practical applications, while ensuring that solution efficiency is maintained.

The indicator of cumulative waiting time, defined to stabilize and optimize the system's waiting time performance over time through quantitative analysis, is established. Specifically, the cumulative delay $Q(t)$, which represents the total amount of overdue delays that have been accumulated in the first t time slots, is calculated as follows [38]:

$$Q(t+1) = \max\{Q(t) + \mathcal{L}_{avg}(t) - \bar{\mathcal{L}}, 0\}, \forall t \in T, \tag{23}$$

where $\mathcal{L}_{avg}(t) = \frac{\sum_{r\in\mathcal{R}}\sum_{m\in S} O_{r,m}(t)\cdot\mathbb{L}_{r,m}(t)}{\sum_{r\in\mathcal{R}}\sum_{m\in S} O_{r,m}(t)}$, and $Q(0) = 0$ because there is no latency at the very beginning.

In dynamic environments, $Q(t)$ adapts to fluctuating workloads by dynamically adjusting caching strategies. For instance, if network conditions degrade (e.g., sudden spikes in requests), $Q(t)$ increases, triggering stricter penalties in subsequent slots to prioritize latency reduction. Based on Equation (23), it is noted that if the delay in the previous time slot exceeds $\bar{\mathcal{L}}$, the cumulative delay is increased. This increase can be utilized as a penalty, which serves to adjust data caching strategies in order to stabilize system latency over time, as specified in Equation (16). Consequently, the long-term delay constraint outlined in Equation (16) can be transformed into a new constraint that is based on cumulative delay as follows:

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[Q(t)] \leq 0. \tag{24}$$

Motivated by Equation (23), we utilize a drift-based algorithm that both stabilizes the queues and optimizes a performance objective. To achieve this, a Lyapunov function can be defined as

$$\mathcal{L}(Q(t)) \triangleq \frac{1}{2}Q^2(t), \forall t \in T. \tag{25}$$

Define $\Delta(Q(t))$ as the change in $\mathcal{L}(Q(t))$ from slot t to slot $t+1$, which is referred to as the Lyapunov drift. To quantifying the dynamic changes in the caching queue, we introduce Lyapunov drift [7]. By minimizing the drift, the system is able to suppress the unbounded growth of $Q(t)$, thereby satisfying long-term constraints $\mathcal{L}_{avg}(t) \leq \bar{\mathcal{L}}$. Here, the Lyapunov drift $\Delta(Q(t))$ is applied in each time slot to enhance the system stability:

$$\Delta(Q(t)) = \mathbb{E}[\mathcal{L}(Q(t+1)) - \mathcal{L}(Q(t))|Q(t)], \forall t \in T. \tag{26}$$

We define the following drift-plus-penalty function to minimize the total cost objective function while stabilizing the caching queues [22]. Instead of taking a control action to minimize a bound on $\Delta(Q(t))$, we minimize a bound on the following drift-plus-penalty function. Here, the Lyapunov drift-plus-penalty function $\mathcal{DP}(t)$, which is defined as follow:

$$\mathcal{DP}(t) = \Delta(Q(t)) + V\cdot\mathbb{E}[\mathbf{L}(t)|Q(t)], \forall t \in T, \tag{27}$$

where $V \geq 0$ is a parameter that is used for adjusting the trade-off between the system cost $\mathbf{L}(t)$ and the number of time slots that are needed to converge the time-averaged latency back to $\bar{\mathcal{L}}$ when Equation (16) is violated. And $\mathbf{L}(t) = \sum_{r=1}^{R}\sum_{m=1}^{S_r}\mathbb{L}_{r,m}(t)$.

The cumulative delay $Q(t)$ enforces long-term stability by accumulating violations and steering optimization toward latency reduction. The drift-plus-penalty framework integrates penalties for excess delays while balancing immediate costs, and the tunable parameter V provides flexibility to handle dynamic environments.

Next, lemma 1 establishes an upper bound for the average loss $\mathcal{DP}(t)$, which plays a crucial part in the transformation of $\mathcal{P}_1$.

Just Accepted

Lemma 1. *The upper bond of $\mathcal{DP}(t)$ can be defined as*

$$\mathcal{DP}(t) \leq \varpi + Q(t)\mathbb{E}[\mathcal{L}_{avg}(t) - \bar{\mathcal{L}}|Q(t)] + V \cdot \mathbb{E}[\mathbf{L}(t)|Q(t)], \forall t \in T, \tag{28}$$

where ϖ is a constant term, which can be expressed as $\varpi = \frac{1}{2}\bar{\mathcal{L}}^2$.

Proof. Firstly, we use Equation (25) to compute a bound on the change in the Lyapunov function from one slot to the next,

$$\begin{aligned}
&\mathcal{L}\left(Q(t+1)\right) - \mathcal{L}(Q(t)) \\
&\overset{(25)}{=} \frac{1}{2}Q^2(t+1) - \frac{1}{2}Q^2(t) \\
&= \frac{1}{2}\left[Q^2(t+1) - Q^2(t)\right] \\
&\overset{(23)}{\leq} \frac{1}{2}\left[(Q(t) + \mathcal{L}_{avg}(t) - \bar{\mathcal{L}})^2 - Q^2(t)\right] \\
&= \frac{1}{2}\left[\mathcal{L}_{avg}^2(t) + \bar{\mathcal{L}}^2 + 2Q(t)\cdot\mathcal{L}_{avg}(t) - 2Q(t)\cdot\bar{\mathcal{L}} - 2\mathcal{L}_{avg}(t)\cdot\bar{\mathcal{L}}\right] \\
&= \frac{1}{2}\left(\mathcal{L}_{avg}(t) - \bar{\mathcal{L}}\right)^2 + Q(t)\cdot\mathcal{L}_{avg}(t) - Q(t)\cdot\bar{\mathcal{L}} \\
&= \frac{1}{2}\left(\mathcal{L}_{avg}(t) - \bar{\mathcal{L}}\right)^2 + Q(t)\cdot\left(\mathcal{L}_{avg}(t) - \bar{\mathcal{L}}\right),
\end{aligned} \tag{29}$$

where $\bar{\mathcal{L}}$ represents the long-term average delay of the system, and $\mathcal{L}_{avg}(t)$ represents the average delay when the user request fails to find the target content in the cache during the $t-1$ time slot (i.e. the average delay when the cache misses). Because of $\mathcal{L}_{avg}(t) \geq 0$, we can derive $\frac{1}{2}\left(\mathcal{L}_{avg}(t) - \bar{\mathcal{L}}\right)^2 \leq \frac{1}{2}\bar{\mathcal{L}}^2$. Then, we introduce the following constant

$$\varpi = \frac{1}{2}\bar{\mathcal{L}}^2 \tag{30}$$

Based on Equation (29) and Equation (30), we obtain

$$\Delta(Q(t)) \leq \varpi + Q(t)\cdot\left(\mathcal{L}_{avg}(t) - \bar{\mathcal{L}}\right). \tag{31}$$

By taking the conditional exception on both sides of Equation (31) and adding the penalty term, we can obtain

$$\mathcal{DP}(t) \leq \varpi + Q(t)\mathbb{E}[\mathcal{L}_{avg}(t) - \bar{\mathcal{L}}|Q(t)] + V \cdot \mathbb{E}[\mathbf{L}(t)|Q(t)]. \tag{32}$$

The proof is completed.

By leveraging the opportunistic expectation minimization technique, we decide on the caching action to minimize the right-hand side of lemma 1. It is worth noting that the first term is constant, which can be determined at the beginning of each time slot [22]. Then, in each time slot t, the data caching strategy is being formulated by finding the optimal solution to problem $\mathcal{P}_2$, which is defined as follows,

$$\mathcal{P}_2 : \min\left[Q(t)(\mathcal{L}_{avg}(t) - \bar{\mathcal{L}}) + V \cdot \mathbf{L}(t)\right] \tag{33}$$

$$\text{s.t.} \quad (3), (7), (10), (11), (16), (19).$$

The parameter V governs the balance between immediate system cost minimization and long-term queue stability. A larger V emphasizes reducing the instantaneous latency $\mathbf{L}(t)$ at the cost of increased cumulative delay $Q(t)$, while a smaller V prioritizes suppressing queue growth to ensure stability at the cost of higher short-term latency penalties. As observed from $\mathcal{P}_2$, after implementing the drift plus penalty function, when the average delay of the user's task exceeds $\bar{\mathcal{L}}$, a penalty is applied to $\mathcal{P}_2$ to reduce the system delay. Additionally, as $Q(t)$ increases,

Just Accepted

it becomes crucial to minimize $\mathcal{L}_{avg}(t)$ in order to stabilize the system and ensure that $\mathcal{L}_{avg}(t)$ converges to the long-term budget.

In the optimization problem $\mathcal{P}_2$ that is involved, the cache decision variable $J_{r,m}$ and the denoising position decision variable $I_{r,m}$ are represented in binary variable form, and their impact on the objective function is characterized by significant nonlinear characteristics. To solve the $\mathcal{P}_2$ problem more efficiently and achieve the goal of simplifying the issue, we analyze the specific relationship between the denoising position decision variable $I_{r,m}$ and the caching decision variable $J_{r,m}$. According to Equation (11), it can be clarified that for each request, a maximum of intermediary output generated by only one denoising step m can be cached throughout the entire processing flow. Given that the J vector satisfies that, except for one element that is 1, all other elements are 0, and $J_{r,1} = 0$. Assuming the existence of $n(1 < n < m)$ such that $J_{r,n} = 1$, then for the element $I_{r,m}$, we have the following condition,

$$I_{r,m} = \begin{cases} 1, & \sum_{i=1}^{m} J_{r,i} = 0 \text{ and } m < n \\ 1, & \sum_{i=1}^{m} J_{r,i} = 1 \text{ and } m = n \\ 0, & \sum_{i=1}^{m} J_{r,i} = 1 \text{ and } m > n. \end{cases} \tag{34}$$

Specifically, when $\sum_{i=1}^{m} J_{r,i} = 0$ and $m < n$, this means that the GenAI task request is still in the stage of denoising operation on the edge server, and no intermediary output that can be cached have been generated during this process. And when $\sum_{i=1}^{m} J_{r,i} = 1$ and $m = n$, it indicates that the edge server has successfully cached the output result of step n of the GenAI task request, which can be shared and reused in subsequent related processing steps. Furthermore, when $\sum_{i=1}^{m} J_{r,i} = 1$ and $m > n$, it means that the GenAI task will perform corresponding operations on the local device from the $n + 1$ step to the final denoising step to complete the denoising process of the entire GenAI task.

Building upon this foundation, to learn a better strategy to reduce the time it taken for the system to complete GenAI tasks, we propose a new caching strategy DRC based on reinforcement models to solve the optimization problem $\mathcal{P}_2$ proposed in the previous section.

4.2 Algorithm Overview

We propose a DRL-based algorithm for making caching decisions to obtain the optimal solution for Problem $\mathcal{P}_2$ with low time complexity. In contrast to traditional DRL approaches [26, 30, 52], which utilize a Markov decision process for formulating optimization problems, the DRC algorithm applies an order-preserving quantization technique to generate intermediary output caching decisions based on the methods. A novel actor-critic structure is developed for the proposed DRC algorithm. A DNN model learns from previous caching solutions and generates multiple candidate decisions in the actor module. In contrast to the conventional critic module that typically relies on a DNN model, the DRC algorithm uses a model-based optimization scheme to evaluate the best service response solutions after generating caching decisions. In complex and dynamically changing computing environments, the DRC algorithm aims to calculate the cache decision function F and formulate decisions $J_{r,m} \in \{0, 1\}$ based on the channel status $g(t)$ of different time slots. This strategy can be represented as follows,

$$F : g(t) \longrightarrow J_{r,m}(t), \tag{35}$$

where $g(t)$ affects the transmission time of intermediary output for GenAI tasks at different time intervals. The DRC framework, outlined in Fig. 3, consists of three main components,

1) **Input Processing:** We input the channel status $g(t)$ and cache decision $J_{r,m}(t)$ into the DNN network, and output a continuous decision between 0 and 1. It should be noted that the initial caching decision $J_{r,m}(t)$ is 0, and the algorithm will continuously learn to obtain the optimal caching decision to reduce the completion time of user GenAI tasks[16].

Just Accepted

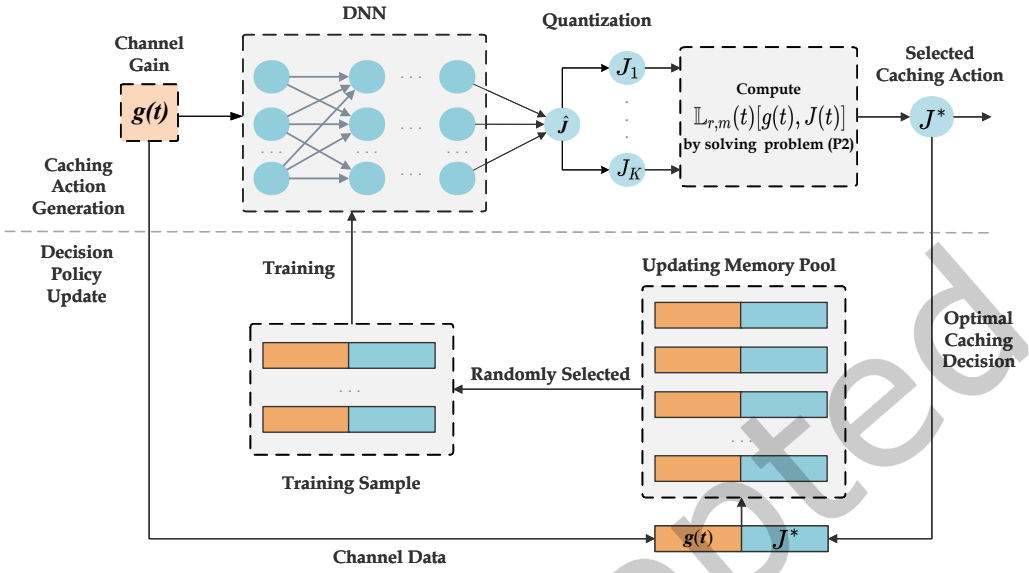

Fig. 3. An overview of the DRC algorithm.

2) **Decision Optimization:** The order-preserving quantization method is adopted to perform reasonable and accurate quantization processing on the continuous cache decision values that are output by DNN, thereby transforming them into several sets of binary decisions characterized by clear discrete features. Once the quantitative transformation is completed, each binary decision obtained is further analyzed and evaluated. The specific operation involves substituting each binary decision into the previously defined optimization problem $\mathcal{P}_2$, from which the optimal decision is selected. Finally, the $[g, J_{r,m}]$ tuple corresponding to this optimal decision is extracted and stored in the experience pool.
3) **Training Iteration:** When we perform random sampling operations from the experience pool, we obtain new training samples. These new training samples are not randomly selected, but are based on a set of relevant data such as $[g, J_{r,m}]$ tuples accumulated in the experience pool, filtered and validated in different contexts. After successfully obtaining these new training samples, we immediately initiate a periodic training process to conduct targeted training on the DNN. By finely adjusting and optimizing its own network parameters, DNN continuously improves its output caching strategy until convergence.

4.3 Caching Decision Generation Module

For each independent time slot, external environmental factors such as channel gain are assumed to be relatively stable internally. However, when focusing on different time slots t, where $t = 1, 2, \ldots, T$ it is found that the external environment exhibits significant fluctuation characteristics.

A fully DNN architecture is chosen to effectively approximate the highly complex mapping relationship between channel states and caching decisions. The DNN used consists of four layers: an input layer, two hidden layers, and an output layer. In order to output relaxed cache decisions, different activation functions are selected for different layers. Among these, the ReLU activation function is applied to the hidden layers, while the sigmoid

Just Accepted

activation function is chosen for the output layer. The sigmoid function is characterized by its property of mapping the output value to the interval (0, 1). Specifically, when $t = 1$, which is in the initial stage of the first time slot, the parameter o of DNN follows the rule of zero mean normal distribution for random initialization operation. In this initial state, DNN outputs a relaxed caching action first [13].

Subsequently, we quantified the relaxed cache decision $\hat{J} \in (0, 1)$ into K sets of binary cache decisions. For each quantization decision J_k, the condition $J_{k,i} \geq J_{k,j}$ must be satisfied, and this condition should hold for all $i, j \in \{1, \ldots, N\}$ while satisfying $J_{t,i} \geq J_{t,j}$. More specifically, for a given range of $1 \leq K \leq N + 1$, the set J_k consisting of K quantized actions generated by the relaxation action J_t is as follows,

1) The first binary caching decision J_1 is obtained as

$$J_{1,i} = \begin{cases} 1, \hat{J}_i > 0.5 \\ 0, \hat{J}_i \leq 0.5 \end{cases} \tag{36}$$

2) To generate the remaining binary caching decision

$$J_{k,i} = \begin{cases} 1, \hat{J}_i > \hat{J}_{(k-1)}, \\ 1, \hat{J}_i = \hat{J}_{(k-1)} \text{ and } \hat{J}_{(k-1)} \leq 0.5, \\ 0, \hat{J}_i = \hat{J}_{(k-1)} \text{ and } \hat{J}_{(k-1)} > 0.5, \\ 0, \hat{J}_i < \hat{J}_{(k-1)}. \end{cases} \tag{37}$$

The specific order-preserving quantization process is shown in Fig. 4. In the quantization process, the size of K is directly related to the diversity of candidate decisions. When the value of K is larger, the number of binary cache decision groups generated increases, resulting in a richer diversity of candidate decisions. Consequently, the likelihood of finding the global optimal cache decision among numerous candidate schemes is greater. However, although an increase in the K value introduces more possibilities for discovering the optimal solution, it inevitably leads to higher computational complexity. The major computational complexity of the DRC algorithm comes from solving $\mathcal{P}_2$ K times in each time slot to select the best caching action. Evidently, a larger K in general leads to a better caching decision in each time slot and accordingly a better caching policy in the long term. Therefore, there exists a fundamental performance-complexity tradeoff in setting the value of K. In the initial stage, a strategy of fixing the value of K is adopted to ensure the relative stability and operability of the experimental process. As the number of experiments gradually increases, rich past experience and relevant data on system performance under different K values are accumulated. At this point, flexible and reasonable adjustments to the value of K can be made based on these past experiences and observed performance indicators, thereby achieving an effective balance between efficiency and complexity [16, 47]. Moreover, we can gradually reduce K during the learning process to speed up the algorithm without compromising the performance. In contrast to Reference [16], addressing the low-dimensional channel allocation problem with a limited action space, the proposed order-preserving quantization method is designed for intermediary outputs caching decisions in diffusion model, requiring the handling of high-dimensional search spaces. Building upon the theoretical foundation of Reference [16], we extend dynamic optimization and stability management mechanisms to meet the requirements of large-scale caching decision scenarios [15].

The candidate caching actions $J(t)$ generated by ordered quantization strictly maintain the ordering relationship of the original continuous solution $\hat{J}$. This ensures that the cache actions corresponding to smaller system latency $\mathbf{L}(t)[g(t), J(t)]$ remain at the forefront of the candidate set after quantization, thereby ensuring that the optimization direction aligns with the problem $\mathcal{P}_2$. Objective function $\mathcal{P}_2$ and its constraints are continuous and differentiable twice within their respective domains, so $\mathcal{P}_2$ is about the convex problem on $J_{r,m}$. We obtain the optimal caching decision J^* by using tools such as CVXPY to solve convex problem $\mathcal{P}_2$.

Just Accepted

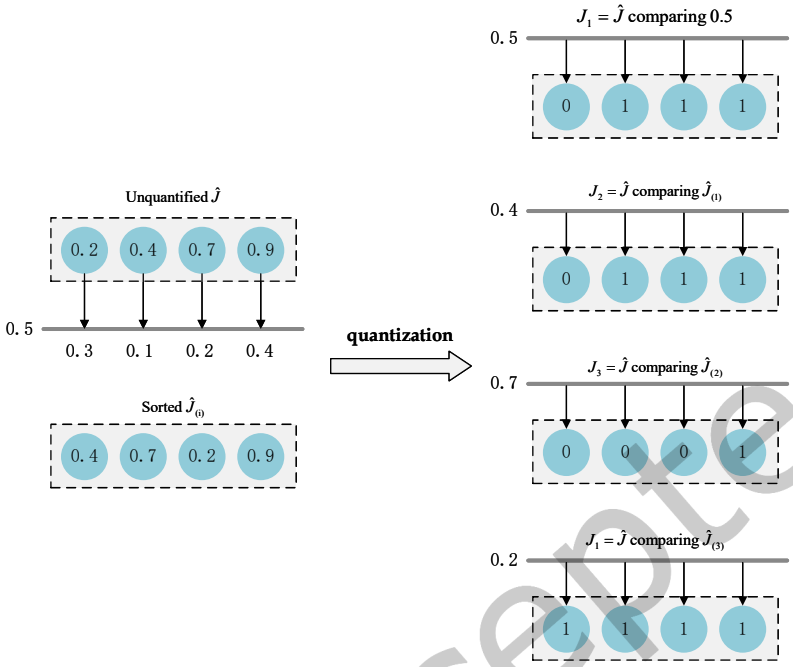

Fig. 4. The specific order-preserving quantization process.

4.4 DNN Model Updating Module

When processing the continuous cache decision values output by DNN, we use the order preserving quantization method to transform them into several sets of binary decisions with clear discrete features. Subsequently, the best decision among those is selected after each binary decision obtained is substituted into the optimization problem $\mathcal{P}_2$ that is previously defined. The optimal caching solution obtained through this operation will be used to update the caching strategy of DNN. Specifically, we first set and maintain an initially empty and limited capacity memory. At the t-th time slot, new training data samples, namely the pair of $[\mathbf{g(t)}, J^*]$, will be added to the memory. When the storage space of the memory reaches saturation. That is, the newly generated data samples will replace the oldest data sample in the memory according to the replacement rule, in order to ensure that the memory can continuously and effectively store data information related to DNN cache policy updates.

Consequently, the experience replay mechanism is utilized to append the newly acquired channel state and the binary $[\mathbf{g(t)}, J^*]$ of the unloading position to the replay memory. When the memory reaches its full capacity, the oldest data sample is replaced. Subsequently, a random batch of sample data is retrieved from the memory to enhance the DNN network. At the same time, by applying the Adam algorithm to update the parameter α_t of DNN, it is to reduce in reducing the average cross entropy loss [19],

$$L(\alpha_t) = -\frac{1}{|\mathbf{T}|}\sum_{t\in\mathbf{T}}\Big((J_t^*)^{\mathsf{T}}\log f_{\alpha_t}(g(t)) + (1-J_t^*)^{\mathsf{T}}\log\big(1-f_{\alpha_t}(g(t))\big)\Big), \tag{38}$$

where $|\mathbf{T}|$ represents the size of $\mathbf{T}$. The superscript T is used to signify the transpose operator. Meanwhile, the log function refers to the element-wise logarithm operation performed on a vector.

Just Accepted

Thanks to the random sampling, the correlation between training samples diminishes, thereby accelerating the convergence speed. Due to the restricted memory space, the DNN only updates based on recent experience, ensuring that the caching strategy remains adaptable to recent channel changes.

4.5 Algorithm Description and Analysis

Based on the above modules, the pseudocode of DRC algorithm can be summarized as Algorithm 1. In line with Algorithm 1, the present channel state $g(t)$ is fed into the network to derive a relaxed caching decision J^*. Subsequently, the optimal caching decision is quantified and then stored in the form of a binary $[g(t), J^*]$ within the experience pool. The DNN undergoes training during each time slot. Overall, the DNN networks engage in repeated learning from the prime state-action pairs $[g(t), J^*]$. As the network model is being trained, it generates more favorable caching decision outputs. Meanwhile, considering the limitations of memory space, only the most recent data samples that have been learned, which pertain to the up-to-date and more refined unloading strategies, are taken into account. This closed-loop reinforcement learning mechanism enhances its caching strategy until it converges and attains an efficient intermediary output caching strategy.

Algorithm 1: DRC model-based GenAI task intermediary output caching decision

Input: Current channel gain $g(t)$, the number of quantized decisions K
Output: Intermediary output caching decision J^*, total costa time $\mathbf{L}(t)[g(t), J(t)]$

Initialize the DNN with random parameters θ ;
Set iteration number T and the training interval;
for $t = 1, 2, \ldots, T$ **do**
    DNN generates a relaxed caching decision $\hat{J}$;
    Quantize $\hat{J}$ to generate K_t binary caching decision $J_k, k \in 1, 2, \ldots, K$;
    for $k = 1, 2, \ldots, K$ **do**
        Given the caching decision J_k, calculate the optimization objective $\mathbf{L}(t)[g(t), J(t)]$;
    end
    Obtain the best intermediary output caching decision J^*;
    Add the newly obtained channel state and caching binary $[g(t), J^*]$ to the experience pool;
    if *The experience pool is full* **then**
        Randomly sample a batch of the dataset $\{(g(\varpi), J^*_\varpi) \mid \varpi \in \mathbf{T}\}$ in the experience pool;
        Train DNN and Update parameters using $\{(g(\varpi), J^*_\varpi) \mid \varpi \in \mathbf{T}\}$ and Adam algorithm;
    end
end

Next, we analyze the time complexity and optimality of the DRC algorithm,

1) **Time Complexity Analysis:** The procedure of the DRC algorithm includes caching action generation, order-preserving quantization and decision policy updating. First, the time complexity for generating and quantifying caching decisions and adding new states to the experience pool is $\mathbf{O}(1)$. The time complexity for selecting the optimal caching middle-denoising results decision is $\mathbf{O}(K)$, where K denotes the number of quantified decisions [16]. The time complexity of the DRC algorithm is simultaneously influenced by the intricacy involved in solving caching decisions sub-problem $\mathcal{P}_2$. Given the aforementioned caching actions, the optimal solutions of downloading the middle-denoising results for N users must be decided, with a time complexity of $\mathbf{O}(N)$. Therefore, the total time complexity of each iteration is $\mathbf{O}(K + N)$. Given that the algorithm involves γ iterations, the overall time complexity is $\mathbf{O}(\gamma \cdot (K + N))$ [34].

Just Accepted

2) **Optimality Analysis:** Theorems are presented to illustrate the optimality analysis of the proposed DRC algorithm.

Theorem 1. *By using the DRC algorithm to solve $\mathcal{P}_2$, the optimal completion time for GenAI tasks is* $\mathbf{O}(\frac{1}{V})$.

Proof. In time slot t, $J_{opt}(t)$ is defined as the optimal synchronization solution for $\mathcal{P}_2$, associated with the minimum averaged synchronization delay $\mathbf{L}_{opt}$,

$$\mathbf{L}_{opt} = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\mathbf{L}(J_{opt}(t))|\boldsymbol{Q}(t)]. \tag{39}$$

In time slot t, let $J^{\S}(t)$ be the optimal decision synchronization solution with a synchronization delay of $\mathbf{L}^{\S}(t)$. Based on Equation (16), we can derive an inequality

$$\exists J^{\S}(t), \mathbb{E}[\mathcal{L}_{avg}(J^{\S}(t)) - \bar{\mathcal{L}}|\boldsymbol{Q}(t)] \leq \theta, \theta \to 0^{+}. \tag{40}$$

As a result, we can derive the following statement,

$$E[\mathbf{L}(J^{\S}(t))|\boldsymbol{Q}(t)] \leq E[\mathbf{L}(J_{opt}(t))|\boldsymbol{Q}(t)]. \tag{41}$$

Based on Equations (27), (28), (39) and (40), we can obtain

$$\begin{aligned}
&\Delta(\boldsymbol{Q}(t)) + V \cdot \mathbb{E}[\mathbf{L}(J^{\S}(t))|\boldsymbol{Q}(t)] \\
\leq &\boldsymbol{Q}(t)\mathbb{E}[(\mathcal{L}_{avg}(J^{\S}(t)) - \bar{\mathcal{L}})|\boldsymbol{Q}(t)] + V \cdot \mathbb{E}[\mathbf{L}(J^{\S}(t))|\boldsymbol{Q}(t)] + \varpi \\
\leq &\boldsymbol{Q}(t)\mathbb{E}[(\mathcal{L}_{avg}(J^{\S}(t)) - \bar{\mathcal{L}})|\boldsymbol{Q}(t)] + V \cdot \mathbb{E}[\mathbf{L}(J_{opt}(t))|\boldsymbol{Q}(t)] + \varpi \\
\leq &\theta \cdot \boldsymbol{Q}(t) + V \cdot \mathbb{E}[\mathbf{L}(J_{opt}(t))|\boldsymbol{Q}(t)] + \varpi \\
= &V \cdot \mathbb{E}[\mathbf{L}(J_{opt}(t))|\boldsymbol{Q}(t)] + \varpi.
\end{aligned} \tag{42}$$

Based on Equation (41), we aggregate the optimal caching decisions for all time slots and obtain

$$\begin{aligned}
&\mathbb{E}[\mathcal{L}(\boldsymbol{Q}(T)) - \mathcal{L}(\boldsymbol{Q}(0))] + V \cdot \sum_{t=0}^{T-1}\mathbb{E}[\mathbf{L}(J^{\S}(t))|\boldsymbol{Q}(t)] \\
\leq &V \cdot \sum_{t=0}^{T-1}\mathbb{E}[\mathbf{L}(J_{opt}(t))|\boldsymbol{Q}(t)] + T \cdot \varpi \\
= &V \cdot T \cdot \mathbf{L}_{opt} + T \cdot \varpi.
\end{aligned} \tag{43}$$

Considering the facts of $\mathcal{L}(\boldsymbol{Q}(T)) \geq 0$ and $\mathcal{L}(\boldsymbol{Q}(0)) = 0$, we can obtain

$$\begin{aligned}
\mathbf{L}^{\S} &\leq \frac{1}{T}(\mathbb{E}[\mathcal{L}(\boldsymbol{Q}(T)) - \mathcal{L}(\boldsymbol{Q}(0))] + \lim_{T\to\infty}\sum_{t=0}^{T-1}\mathbb{E}[\mathbf{L}(J^{\S}(t))|\boldsymbol{Q}(t)]) \\
&\leq \mathbf{L}_{opt} + \frac{\varpi}{V}.
\end{aligned} \tag{44}$$

Based on Equation (31) and $\bar{\mathcal{L}}$ representing the long-term average delay of the system, we can know that $\mathbf{O}(\frac{\varpi}{V}) = \mathbf{O}(\frac{1}{V})$. Thus, the time-average system cost of our DRC algorithm is bounded by $\mathbf{O}(\frac{1}{V})$. We complete the proof.

Theorem 2. *By applying the DRC algorithm, the time-average accumulated latency is bounded by* $\mathbf{O}(V)$.

Just Accepted

Proof. Based on Equations (23) and (24), we assume the existence of an $J^*(t)$ and a positive number ψ to satisfy

$$\mathbb{E}[\mathcal{L}_{avg}(J^*(t)) - \bar{\mathcal{L}}|\boldsymbol{Q}(t)] \leq -\psi. \tag{45}$$

From Equations (27) and (28), we can obtain

$$\Delta(\boldsymbol{Q}(t)) \leq \varpi + \boldsymbol{Q}(t)\mathbb{E}[\mathcal{L}_{avg}(J^*(t)) - \bar{\mathcal{L}}|\boldsymbol{Q}(t)]. \tag{46}$$

Denote $\mathbf{L}_{max}$ and $\mathbf{L}_{min}$ as the largest and smallest system cost respectively. According to Equation (39) , we have

$$\Delta(\boldsymbol{Q}(t)) + V \cdot \mathbf{L}_{min} \leq \varpi + \boldsymbol{Q}(t)\mathbb{E}[\mathcal{L}_{avg}(J^*(t)) - \bar{\mathcal{L}}|\boldsymbol{Q}(t)] + V \cdot \mathbf{L}_{max}. \tag{47}$$

Defining $\varpi^\dagger = \varpi + V \cdot (\mathbf{L}_{max} - \mathbf{L}_{min})$, we can derive the following inequality:

$$\begin{aligned} \Delta(\boldsymbol{Q}(t)) &\leq \varpi^\dagger + \boldsymbol{Q}(t)\mathbb{E}[\mathcal{L}_{avg}(J^*(t)) - \bar{\mathcal{L}}|\boldsymbol{Q}(t)] \\ &\overset{(38)}{\leq} \varpi^\dagger - \psi \cdot \boldsymbol{Q}(t). \end{aligned} \tag{48}$$

By adding expectation to both sides of Equation (41), we obtain

$$\mathbb{E}[\Delta(\boldsymbol{Q}(t))] = \mathbb{E}[\mathcal{L}(\boldsymbol{Q}(t+1)) - \mathcal{L}(\boldsymbol{Q}(t))] \leq \varpi^\dagger - \psi \cdot \mathbb{E}[\boldsymbol{Q}(t)]. \tag{49}$$

We accumulate the task completion time from time slot $t = 0$ to $t = T - 1$,

$$\begin{aligned} \mathbb{E}[\mathcal{L}(\boldsymbol{Q}(1)) - \mathcal{L}(\boldsymbol{Q}(0))] &\leq \varpi^\dagger - \psi \cdot \mathbb{E}[\boldsymbol{Q}(0)], \\ \mathbb{E}[\mathcal{L}(\boldsymbol{Q}(2)) - \mathcal{L}(\boldsymbol{Q}(1))] &\leq \varpi^\dagger - \psi \cdot \mathbb{E}[\boldsymbol{Q}(1)], \\ &\cdots \\ \mathbb{E}[\mathcal{L}(\boldsymbol{Q}(T)) - \mathcal{L}(\boldsymbol{Q}(T-1))] &\leq \varpi^\dagger - \psi \cdot \mathbb{E}[\boldsymbol{Q}(T-1)]. \end{aligned} \tag{50}$$

The time-average accumulated latency can be obtained by the sum of Equation (43) of each time slot divided by the number of total time slots T,

$$\frac{1}{T}\lim_{T\to\infty}\sum_{t=0}^{T-1}\mathbb{E}[\boldsymbol{Q}(t)] \leq \frac{\varpi^\dagger - \frac{1}{T}\mathbb{E}[\mathcal{L}(\boldsymbol{Q}(T))]}{\psi} \leq \frac{\varpi^\dagger}{\psi}. \tag{51}$$

Considering the fact that $\mathbf{O}(\frac{\varpi^\dagger}{\psi}) = \mathbf{O}(V)$, the time-average accumulated latency of DRC algorithm is bounded by $\mathbf{O}(V)$. The proof is completed.

Theorems 1 and 2 indicate that increasing the Lyapunov parameter V allows for the upper bound of synchronization time to be made arbitrarily tight. As parameter V approaches $+\infty$, the DRC algorithm minimizes task completion time associated with the time-averaged optimization problem in $\mathcal{P}_2$, while increasing overdue delay. Consequently, it is necessary to adjust the value of the parameter V to achieve a desired balance between the time-average task completion time and overdue delay.

5 Simulation Experiments

In this section, we conduct simulation experiments. Firstly, the simulation parameters are introduced. Next, we discuss the benchmark strategies based on intermediary output caching is discussed, and it is compared with the previously proposed DRC algorithm. Finally, the simulation results are presented, followed by a discussion of the findings.

Just Accepted

5.1 Parameter Settings

The Lyapunov control parameter V is fixed at 1800, and the number of time slots is fixed at $10,000$ [34]. To minimize the effect of random variations in the simulation experiments, each experimental group is executed 12 times, with the average value of the evaluation indicators calculated for each group [2].

In an edge network environment, there is an edge server and the number of users is $U = 10$, where each user has 50 requests, i.e. $R = 50$. The maximum total denoising step for each GenAI task is set to $S_r = 20$, and the maximum shared step is set to $S_r^* = 6$. In the wireless transmission model of user n and edge server, assuming the simulation environment is in a 5G environment, we have the following settings: the transmission power from the edge server to the user's local $p = 50$ W, the Gaussian noise power $Q^2 = 10^{-13}$ W, and the downlink bandwidth W_n of the edge server is within the range of [80,100]. The maximum CPU frequency for user n and the maximum CPU frequency for edge server are set to $f_n^{max} = 1.5$ GHz and $f_{max} = 20$ GHz, respectively. We set each GenAI generated image to 500 KB and set the maximum cache space of the edge server to C_e^{max} is 3 GB. In order to compare the performance of the DRC algorithm, we compare it with the other three representative caching strategies to demonstrate its superiority.

1) Online-Optimal Data Caching Methodology (OO): This particular methodology aims to identify the optimal solution for the $\mathcal{P}_2$ problem within each separate time slot. Given that Equation (16) represents a long-term latency constraint, we utilize Equation (52) as a constraint to propel the operation of OO during individual time slots .
$$\frac{\sum_{r\in\mathcal{R}}\sum_{m\in S} O_{r,m}(t)\cdot \mathbb{L}_{r,m}(t)}{\sum_{r\in\mathcal{R}}\sum_{m\in S} O_{r,m}(t)} \leq \bar{\mathcal{L}}. \tag{52}$$
2) Deep Neural Networks Driven Caching Approach (DNN): DNNs are multilayer neural networks with a structure of input, hidden, and output layers, trained via forward and backward propagation using optimization algorithms, and are widely applied in various fields like image recognition, natural language processing, etc. for handling complex data and learning patterns.
3) No-Cache Local Denoising Computation (NCLDC) [6]: All denoising inference steps are executed entirely on the local device, eliminating the need for any caching to remote servers. This approach ensures minimal data tranmission volume and maximizes privacy, as no intermediate data is transmitted over the network. By performing all computations locally, NCLDC provides a robust baseline for evaluating the performance and efficiency of distributed inference strategies in edge computing networks.

The OO algorithm is a timeslot-based algorithm designed to identify the optimal caching solution for the $\mathcal{P}_2$ problem while adhering to Equation (52) latency constraints. The DNN algorithm leverages multilayer neural networks to process complex tasks efficiently using deep learning techniques. The NCLDC algorithm, by performing denoising pocess on the local device, makes it a suitable benchmark for evaluating distributed inference strategies. All benchmark algorithms inverigated caching problems with same objectives in edge computing networks. Consequently, the comparisions of these algorithms show the effectiveness of caching intermediary outputs for GenAI services.

5.2 Performance Evaluation

This section first validates the convergence behavior of the proposed DRC algorithm over 3000 iterations. The corresponding training loss curve is presented in Fig. 5. From the curve, it is evident that with an increase in the number of training iterations, the model's training loss steadily decreases and eventually stabilizes around 0.03. This indicates that the DRC algorithm can converge and reach a stable state within a finite number of iterations. Although the convergence curve shows minor fluctuations, these are primarily attributed to the inherent randomness in the sampling of the training data.

Just Accepted

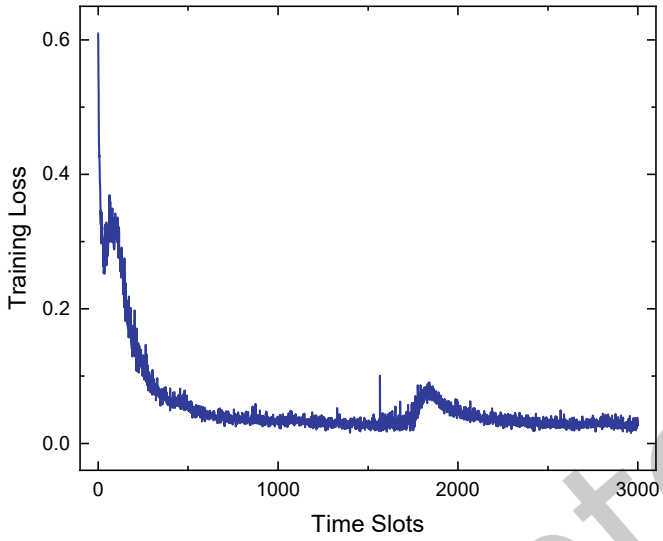

Fig. 5. Training loss curve of the DRC Algorithm.

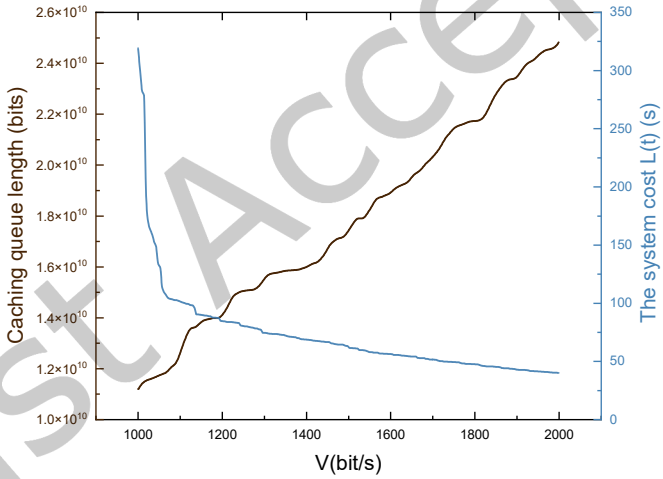

Fig. 6. Impacts of Lyapunov V on caching queue length and the system cost

In Fig. 6, we further show the impact of the Lyapunov control parameter V on the performance of the DRC method, where $V \in [1000, 2000]$. All the points in Fig. 6 are the average performance after convergence. In practice, Lyapunov parameter V should be adjusted to a reasonable value through multiple sets of experiments to achieve a balance between the capacity constraint of the caching queue and the end-to-end delay target [5]. Fig. 2 verifies the analytical results related to Lyapunov optimization established in Theorems 1 and 2, which shows that the system cost is inversely proportional and the sum of caching queue backlogs of queues is proportional to V [1].

Just Accepted

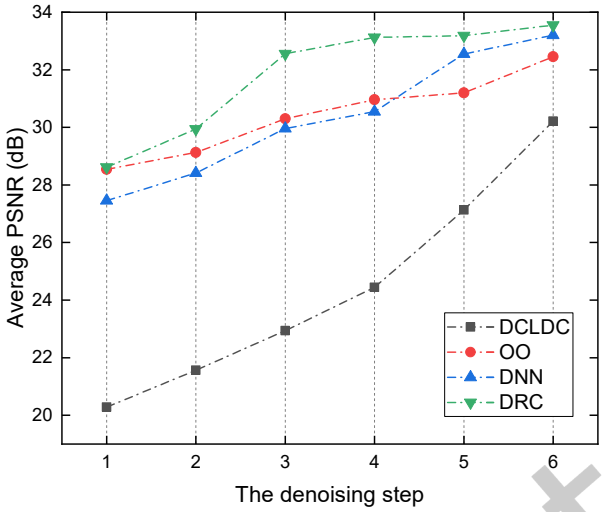

Fig. 7. Denoising outputs evaluation of different caching algorithms.

Fig. 7 illustrates the impact of different caching algorithms on the image denoising process. The PSNR (measured in dB) is used as the evaluation metric to evaluate the quality of the denoised outputs. As the denoising steps increase from 1 to 6, the PSNR achieved by the DCLDC algorithm increases sharply from 20.285 dB to 30.216 dB. Unlike the DRC algorithm, which reuses previously denoised intermediary outputs to accelerate the denoising procedure, DCLDC must reconstruct image details progressively from the original noisy distribution. As a result, the proposed DRC algorithm can obtain the best generation outputs among all algorithms.

Fig. 8 illustrates various experimental results obtained by setting different numbers of users. The request of each user is set to $r = 10$, the maximum denoising step S_r is set to 20, the maximum reusable denoising step S_r^* is set to 6, and the maximum cache capacity D_e is set to 3G. As can be seen from Fig. 8 (a), the response time gradually increases with the increase in the number of users, which is in line with our intuition. In addition, compared with other benchmark algorithms, the proposed DRC algorithm significantly reduces the response time. Fig. 8 (b) shows the cache hit rates achieved by different cache algorithms. As the number of users increases from 5 to 25, the cache hit rate increases rapidly. The reason is that with the increase in the number of users, a large number of concurrent access requests can quickly fill the cache with various commonly used data, enabling the cache to quickly accumulate more valuable data that may be accessed again in the future. To evaluate the effectiveness of our intermediary output caching, we examine the time saved compared to the case without caching, namely the "reduced time cost of all requests". As shown in Fig. 8 (c), with the increase in the number of users, the time cost saved per request also increases, and among them, our proposed DRC algorithm performs the best.

In this experimental group, the number of users is set to 20. Then, the impacts of different maximum reusable steps are studied in Fig. 9. As shown in Fig. 9 (a), when the maximum reusable step increases, the response time gradually decreases. The reason is that when we set a larger maximum reusable step, although the transmission cost will increase, the execution time on the edge server can be saved by using the cached intermediary outputs. When the maximum reusable denoising step increases from 5 to 9, the cache hit rate will slowly decrease, as shown in Fig. 9 (b). This is because smaller denoising steps will generate intermediary outputs with more noise and more blurred outlines. However, such results can produce multiple possible picture effects and will be reused

Just Accepted

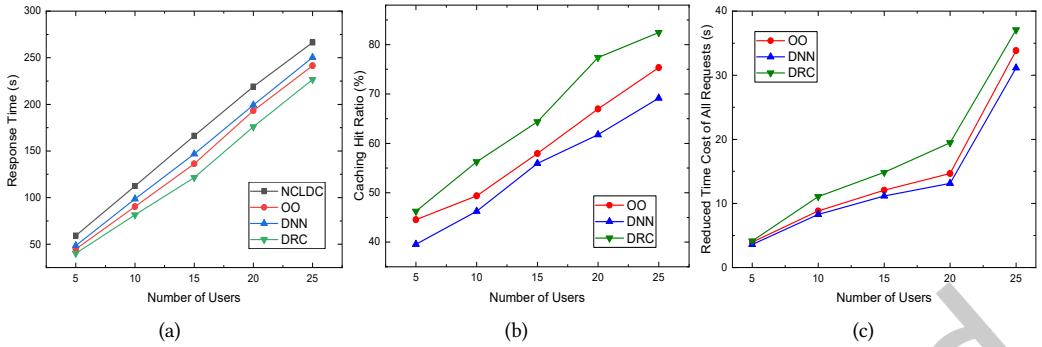

Fig. 8. Impacts of different number of users on (a) response time, (b) caching hit ratio , and (c) reduced time cost of all requests.

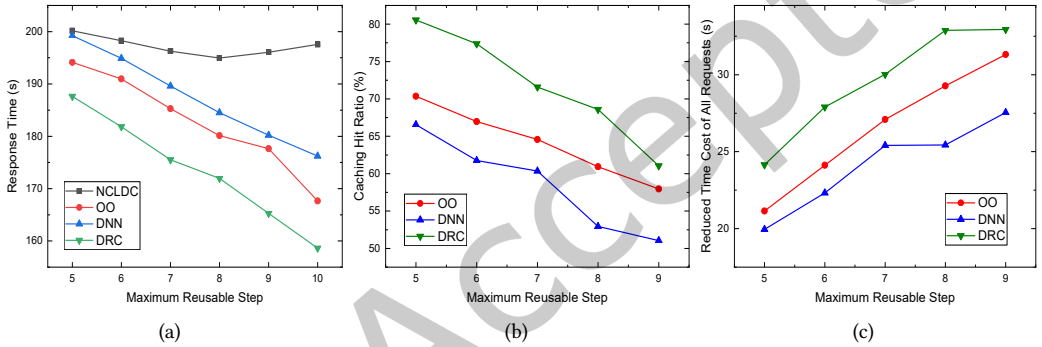

Fig. 9. Impacts of differnet maximum reusable step on (a) response time, (b) caching hit ratio , and (c) reduced time cost of all requests.

by more users, thus leading to a higher cache hit rate. In Fig. 9 (c), although the cache hit rate will decrease as the maximum reusable step increases, in the actual experiment, we find that the DRC algorithm proposed in this paper significantly outperforms other algorithms in terms of the performance of reduced time cost of all requests, which proves the necessity of caching intermediary output. In this experimental group, the number of users is set to 20. Then, the impacts of different maximum reusable steps are studied in Fig. 9. As shown in Fig. 9 (a), when the maximum reusable step increases, the response time gradually decreases. The reason is that when we set a larger maximum reusable step, although the transmission cost will increase, the execution time on the edge server can be saved by using the cached intermediary outputs. When the maximum reusable denoising step increases from 5 to 9, the cache hit rate will slowly decrease, as shown in Fig. 9 (b). This is because smaller denoising steps will generate intermediary outputs with more noise and more blurred outlines. However, such results can produce multiple possible picture effects and will be reused by more users, thus leading to a higher cache hit rate. In Fig. 9 (c), although the cache hit rate will decrease as the maximum reusable step increases, in the actual experiment, we find that the DRC algorithm proposed in this paper significantly outperforms other algorithms in terms of the performance of reduced time cost of all requests, which proves the necessity of caching intermediary output.

Just Accepted

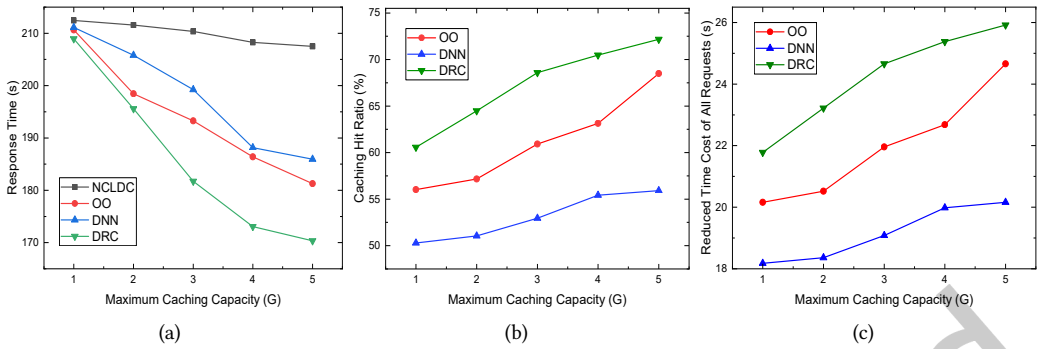

Fig. 10. Impacts of differnet maximum caching capacity on (a) response time, (b) caching hit ratio , and (c) reduced time cost of all requests.

The Fig. 10 illustrates the impact of the maximum cache capacity D_e on the system performance. The results in Fig. 10 (a) indicate that as the maximum cache capacity D_e increases, the response time decreases significantly. Compared to the NCLDC algorithm, our proposed algorithm reduces the response time by 22.35%, and the DRC algorithm has a lower response time cost than the other three benchmark strategies. It is also observed that when the maximum cache capacity D_e becomes larger, the cache hit rate increases, as shown in Fig. 10 (b). The reason is that a larger cache means that more intermediary outputs can be stored, and thus the probability of a request finding the required data in the cache (i.e., the hit rate) is higher. When the maximum cache capacity D_e is set to 5G, compared with the DNN algorithm, the DRC algorithm can reduce the time cost of each request by 28.57%.

6 Conclusion

In this work, we have proposed a novel edge computing-assisted GenAI framework to enable efficient GenAI service provision, where the intermediary output can be cached on edge servers and reused by different users. Assuming the existence of causally correlated auxiliary information, we have formulated a long-term caching problem under intra-time-slot caching constraints by considering various maximally reusable steps of diffusion model-based GenAI tasks and the available caching capacity at edge servers. By leveraging the Lyapunov optimization framework, we have transformed the multi-stage optimization problem into multiple deterministic sub-problems. To solve each deterministic optimization problem with low time complexity, the DRC algorithm with an actor-critic framework has been proposed, including a DNN-based caching decision generation module and a DNN model updating module. Finally, extensive simulation experiments has been conducted to demonstrate that the proposed DRC algorithm can reduce response time and improve cache hit rates compared to other benchmark strategies. In future work, the proposed DRC algorithm will be further validated for its applicability and performance enhancement in different types of content generation tasks.

Just Accepted

Acknowledgment

This work is supported in part by Guizhou Provincial Basic Research Program (Natural Science) (Grant No. ZK[2024]YiBan048 and No. ZK[2025]Mianshang627), in part by Youth Science And Technology Talent Growth Project of Guizhou Education Department (Grant No. QianjiaoJi[2024]22), in part by Guizhou Science and Technology Plan Project (Grant No. QKHRC-KJZY[2025]036 and QKHRC-KJZY[2025]021), in part by Guizhou Science and Technology Plan Project (Grant No. QKHPT-KXJZ[2024]002), and in part by National Natural Science Foundation of China (Grant No. U2333202).

References

[1] Alia Asheralieva and Dusit Niyato. 2020. Combining Contract Theory and Lyapunov Optimization for Content Sharing With Edge Caching and Device-to-Device Communications. *IEEE/ACM Transactions on Networking* 28, 3 (2020), 1213–1226. doi:10.1109/TNET.2020.2978117

[2] Bitan Banerjee, Adita Kulkarni, and Anand Seetharam. 2018. Greedy Caching: An optimized content placement strategy for information-centric networks. *Computer Networks* 140 (2018), 78–91. https://api.semanticscholar.org/CorpusID:13714858

[3] Riccardo Barbano, Alexander Denker, Hyungjin Chung, Tae Hoon Roh, Simon Arridge, Peter Maass, Bangti Jin, and Jong Chul Ye. 2023. Steerable Conditional Diffusion for Out-of-Distribution Adaptation in Medical Image Reconstruction. *arXiv e-prints*, Article arXiv:2308.14409 (Aug. 2023), arXiv:2308.14409 pages. doi:10.48550/arXiv.2308.14409 arXiv:2308.14409 [cs.CV]

[4] Zouhir Bellal, Boubakr Nour, and Spyridon Mastorakis. 2021. CoxNet: A Computation Reuse Architecture at the Edge. *IEEE Transactions on Green Communications and Networking* 5, 2 (2021), 765–777. doi:10.1109/TGCN.2021.3071497

[5] Suzhi Bi, Liang Huang, Hui Wang, and Ying-Jun Angela Zhang. 2021. Lyapunov-Guided Deep Reinforcement Learning for Stable Online Computation Offloading in Mobile-Edge Computing Networks. *Trans. Wireless. Comm.* 20, 11 (Nov. 2021), 7519–7537. doi:10.1109/TWC.2021.3085319

[6] Suzhi Bi, Liang Huang, and Ying-Jun Angela Zhang. 2020. Joint Optimization of Service Caching Placement and Computation Offloading in Mobile Edge Computing Systems. *IEEE Transactions on Wireless Communications* 19, 7 (2020), 4947–4963. doi:10.1109/TWC.2020.2988386

[7] Lorenzo Bracciale and Pierpaolo Loreti. 2020. Lyapunov Drift-Plus-Penalty Optimization for Queues With Finite Capacity. *IEEE Communications Letters* 24, 11 (2020), 2555–2558. doi:10.1109/LCOMM.2020.3013125

[8] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. 2023. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 9 (2023), 10850–10869. doi:10.1109/TPAMI.2023.3261988

[9] Kun Deng, Shangping Zhong, and Kaizhi Chen. 2024. Diffusion Generation of Lace Texture Images with Self-Attention Mechanism Introduced in Higher Resolution Layers. In *Proceedings of the 2024 7th International Conference on Computer Information Science and Artificial Intelligence (CISAI '24)*. Association for Computing Machinery, New York, NY, USA, 220–225. doi:10.1145/3703187.3703224

[10] Tao Deng, Dongyu Chen, Juncheng Jia, Mianxiong Dong, Kaoru Ota, Zhanwei Yu, and Di Yuan. 2024. Optimizing Resource Allocation and Request Routing for AI-Generated Content (AIGC) Services in Mobile Edge Networks With Cell Coupling. *IEEE Transactions on Vehicular Technology* 73, 11 (2024), 17911–17916. doi:10.1109/TVT.2024.3421351

[11] Hongyang Du, Zonghang Li, Dusit Niyato, Jiawen Kang, Zehui Xiong, Huawei Huang, and Shiwen Mao. 2024. Diffusion-Based Reinforcement Learning for Edge-Enabled AI-Generated Content Services. *IEEE Transactions on Mobile Computing* 23, 9 (2024), 8902–8918. doi:10.1109/TMC.2024.3356178

[12] Hongyang Du, Zonghang Li, Dusit Niyato, Jiawen Kang, Zehui Xiong, Xuemin Shen, and Dong In Kim. 2024. Enabling AI-Generated Content Services in Wireless Edge Networks. *IEEE Wireless Communications* 31, 3 (2024), 226–234. doi:10.1109/MWC.004.2300015

[13] Hongyang Du, Ruichen Zhang, Yinqiu Liu, Jiacheng Wang, Yijing Lin, Zonghang Li, Dusit Niyato, Jiawen Kang, Zehui Xiong, Shuguang Cui, Bo Ai, Haibo Zhou, and Dong In Kim. 2023. Enhancing Deep Reinforcement Learning: A Tutorial on Generative Diffusion Models in Network Optimization. *arXiv e-prints*, Article arXiv:2308.05384 (Aug. 2023), arXiv:2308.05384 pages. doi:10.48550/arXiv.2308.05384 arXiv:2308.05384 [cs.NI]

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. *arXiv e-prints*, Article arXiv:2006.11239 (June 2020), arXiv:2006.11239 pages. doi:10.48550/arXiv.2006.11239 arXiv:2006.11239 [cs.LG]

[15] Linh T. Hoang, Chuyen T. Nguyen, and Anh T. Pham. 2023. Deep Reinforcement Learning-Based Online Resource Management for UAV-Assisted Edge Computing With Dual Connectivity. *IEEE/ACM Transactions on Networking* 31, 6 (2023), 2761–2776. doi:10.1109/TNET.2023.3263538

[16] Liang Huang, Suzhi Bi, and Ying-Jun Angela Zhang. 2020. Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks. *IEEE Transactions on Mobile Computing* 19, 11 (2020), 2581–2593. doi:10.1109/TMC.2019.2928811

[17] Yi Jia, Cheng Zhang, Yongming Huang, and Wei Zhang. 2022. Lyapunov Optimization Based Mobile Edge Computing for Internet of Vehicles Systems. *IEEE Transactions on Communications* 70, 11 (2022), 7418–7433. doi:10.1109/TCOMM.2022.3206885

[18] Divya Jyoti Bajpai and Manjesh Kumar Hanawal. 2024. Distributed Inference on Mobile Edge and Cloud: An Early Exit based Clustering Approach. *arXiv e-prints*, Article arXiv:2410.05338 (Oct. 2024), arXiv:2410.05338 pages. doi:10.48550/arXiv.2410.05338 arXiv:2410.05338 [cs.LG]

[19] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, Article arXiv:1412.6980 (Dec. 2014), arXiv:1412.6980 pages. doi:10.48550/arXiv.1412.6980 arXiv:1412.6980 [cs.LG]

[20] Hui Li, Xiuhua Li, Chuan Sun, Fang Fang, Qilin Fan, Xiaofei Wang, and Victor C. M. Leung. 2024. Intelligent Content Caching and User Association in Mobile Edge Computing Networks for Smart Cities. *IEEE Transactions on Network Science and Engineering* 11, 1 (2024), 994–1007. doi:10.1109/TNSE.2023.3312369

[21] Siyuan Li, Xi Lin, Hansong Xu, Kun Hua, Xiaomin Jin, Gaolei Li, and Jianhua Li. 2024. Multi-Agent RL-Based Industrial AIGC Service Offloading over Wireless Edge Networks. *arXiv e-prints*, Article arXiv:2405.02972 (May 2024), arXiv:2405.02972 pages. doi:10.48550/

Just Accepted

arXiv.2405.02972 arXiv:2405.02972 [cs.NI]
[22] Ya Li, Kebing Jin, Jianhang Tang, Yang Zhang, and Yixiong Feng. 2024. Diffusion-Enabled Digital Twin Synchronization for AIGC Services in Space-Air-Ground Integrated Networks. *IEEE Internet of Things Journal* (2024), 1–1. doi:10.1109/JIOT.2024.3523324
[23] Junyan Lin, Feng Gao, Xiaochen Shi, Junyu Dong, and Qian Du. 2023. SS-MAE: Spatial–spectral masked autoencoder for multisource remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023), 1–14. doi:10.1109/TGRS.2023.3331717
[24] Jing Long, Guanhua Ye, Tong Chen, Yang Wang, Meng Wang, and Hongzhi Yin. 2024. Diffusion-Based Cloud-Edge-Device Collaborative Learning for Next POI Recommendations. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24)*. Association for Computing Machinery, New York, NY, USA, 2026–2036. doi:10.1145/3637528.3671743
[25] Yuyi Mao, Jun Zhang, and Khaled B. Letaief. 2016. Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices. *IEEE Journal on Selected Areas in Communications* 34, 12 (2016), 3590–3605. doi:10.1109/JSAC.2016.2611964
[26] M. S. Mekala, Gaurav Dhiman, Gautam Srivastava, Zulqar Nain, Haolin Zhang, Wattana Viriyasitavat, and G. P. S. Varma. 2024. A DRL-Based Service Offloading Approach Using DAG for Edge Computational Orchestration. *IEEE Transactions on Computational Social Systems* 11, 3 (2024), 3070–3078. doi:10.1109/TCSS.2022.3161627
[27] Axi Niu, Pham Xuan Trung, Kang Zhang, Jinqiu Sun, Yu Zhu, In So Kweon, and Yanning Zhang. 2023. ACDMSR: Accelerated Conditional Diffusion Models for Single Image Super-Resolution. *arXiv e-prints*, Article arXiv:2307.00781 (July 2023), arXiv:2307.00781 pages. doi:10.48550/arXiv.2307.00781 arXiv:2307.00781 [cs.CV]
[28] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *arXiv e-prints*, Article arXiv:2203.02155 (March 2022), arXiv:2203.02155 pages. doi:10.48550/arXiv.2203.02155 arXiv:2203.02155 [cs.CL]
[29] Xinyue Qi, Jianhang Tang, Jiangming Jin, and Yang Zhang. 2024. Diffusion-Based Multi-Agent Reinforcement Learning with Communication. In *Proceedings of the 2024 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS'2024)*. 1–6. doi:10.1109/APWCS61586.2024.10679289
[30] Abegaz Mohammed Seid, Gordon Owusu Boateng, Bruce Mareri, Guolin Sun, and Wei Jiang. 2021. Multi-Agent DRL for Task Offloading and Resource Allocation in Multi-UAV Enabled IoT Edge Network. *IEEE Transactions on Network and Service Management* 18, 4 (2021), 4531–4547. doi:10.1109/TNSM.2021.3096673
[31] Jiawei Shao, Yuyi Mao, and Jun Zhang. 2023. Task-Oriented Communication for Multidevice Cooperative Edge Inference. *IEEE Transactions on Wireless Communications* 22, 1 (2023), 73–87. doi:10.1109/TWC.2022.3191118
[32] Wei Song, Wen Ma, Ming Zhang, Yanghao Zhang, and Xiaobing Zhao. 2024. Lightweight diffusion models: a survey. *Artificial Intelligence Review* 57, 6 (2024), 161. doi:10.1007/s10462-024-10800-8
[33] Gan Sun, Wenqi Liang, Jiahua Dong, Jun Li, Zhengming Ding, and Yang Cong. 2024. Create your world: Lifelong text-to-image diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, 9 (2024), 6454 – 6470. doi:10.1109/TPAMI.2024.3382753
[34] Jianhang Tang, Jiangtian Nie, Jingpan Bai, Ji Xu, Shaobo Li, Yang Zhang, and Yanli Yuan. 2024. UAV-Assisted Digital-Twin Synchronization With Tiny-Machine-Learning-Based Semantic Communications. *IEEE Internet of Things Journal* 11, 17 (2024), 28437–28451. doi:10.1109/JIOT.2024.3401229
[35] Tao Wang, Tuo Shi, Xiulong Liu, Jianping Wang, Bin Liu, Yingshu Li, and Yechao She. 2024. Minimizing Latency for Multi-DNN Inference on Resource-Limited CPU-Only Edge Devices. In *Proceedings of the 2024 IEEE Conference on Computer Communications (INFOCOM'2024)*. 2239–2248. doi:10.1109/INFOCOM52122.2024.10621120
[36] Yun-Cheng Wang, Jintang Xue, Chengwei Wei, and C. C. Jay Kuo. 2023. An Overview on Generative AI at Scale With Edge–Cloud Computing. *IEEE Open Journal of the Communications Society* 4 (2023), 2952–2971. doi:10.1109/OJCOMS.2023.3320646
[37] Zhiyuan Wu, Sheng Sun, Yuwei Wang, Min Liu, Ke Xu, Wen Wang, Xuefeng Jiang, Bo Gao, and Jinda Lu. 2024. FedCache: A Knowledge Cache-Driven Federated Learning Architecture for Personalized Edge Intelligence. *IEEE Transactions on Mobile Computing* 23, 10 (2024), 9368–9382. doi:10.1109/TMC.2024.3361876
[38] Xiaoyu Xia, Feifei Chen, Qiang He, John Grundy, Mohamed Abdelrazek, and Hai Jin. 2021. Online Collaborative Data Caching in Edge Computing. *IEEE Transactions on Parallel and Distributed Systems* 32, 2 (2021), 281–294. doi:10.1109/TPDS.2020.3016344
[39] Xiaoyu Xia, Feifei Chen, Qiang He, John Grundy, Mohamed Abdelrazek, Jun Shen, Athman Bouguettaya, and Hai Jin. 2022. Formulating Cost-Effective Data Distribution Strategies Online for Edge Cache Systems. *IEEE Transactions on Parallel and Distributed Systems* 33, 12 (2022), 4270–4281. doi:10.1109/TPDS.2022.3185250
[40] Gaochang Xie, Renchao Xie, Xinyuan Zhang, Jiangtian Nie, Qinqin Tang, Qian Chen, and Dusit Niyato. 2024. Enhancing Vehicular Edge Intelligence through Distributed Collaborative Generative AI Inference. In *Proceedings of the 2024 IEEE International Conference on Communications (ICC'2024)*. 4560–4565. doi:10.1109/ICC51166.2024.10622951
[41] Changfu Xu, Jianxiong Guo, Jiandian Zeng, Shengguang Meng, Xiaowen Chu, Jiannong Cao, and Tian Wang. 2024. Enhancing AI-Generated Content Efficiency Through Adaptive Multi-Edge Collaboration. In *Proceedings of the 2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS'2024)*. 960–970. doi:10.1109/ICDCS60910.2024.00093

Just Accepted

[42] Minrui Xu, Hongyang Du, Dusit Niyato, Jiawen Kang, Zehui Xiong, Shiwen Mao, Zhu Han, Abbas Jamalipour, Dong In Kim, Xuemin Shen, Victor C. M. Leung, and H. Vincent Poor. 2024. Unleashing the Power of Edge-Cloud Generative AI in Mobile Networks: A Survey of AIGC Services. *IEEE Communications Surveys & Tutorials* 26, 2 (2024), 1127–1170. doi:10.1109/COMST.2024.3353265

[43] Minrui Xu, Dusit Niyato, Hongliang Zhang, Jiawen Kang, Zehui Xiong, Shiwen Mao, and Zhu Han. 2023. Sparks of GPTs in Edge Intelligence for Metaverse: Caching and Inference for Mobile AIGC Services. *arXiv e-prints*, Article arXiv:2304.08782 (April 2023), arXiv:2304.08782 pages. doi:10.48550/arXiv.2304.08782 arXiv:2304.08782 [cs.NI]

[44] Chenqian Yan, Songwei Liu, Hongjian Liu, Xurui Peng, Xiaojian Wang, Fangmin Chen, Lean Fu, and Xing Mei. 2024. Hybrid SD: Edge-Cloud Collaborative Inference for Stable Diffusion Models. *arXiv e-prints*, Article arXiv:2408.06646 (Aug. 2024), arXiv:2408.06646 pages. doi:10.48550/arXiv.2408.06646 arXiv:2408.06646 [cs.CV]

[45] Jia Yan, Suzhi Bi, and Ying-Jun Angela Zhang. 2022. Optimal Model Placement and Online Model Splitting for Device-Edge Co-Inference. *IEEE Transactions on Wireless Communications* 21, 10 (2022), 8354–8367. doi:10.1109/TWC.2022.3165824

[46] Dongdong Ye, Shuting Cai, Hongyang Du, Jiawen Kang, Yinqiu Liu, Rong Yu, and Dusit Niyato. 2024. Optimizing AIGC Services by Prompt Engineering and Edge Computing: A Generative Diffusion Model-Based Contract Theory Approach. *IEEE Transactions on Vehicular Technology* (2024), 1–16. doi:10.1109/TVT.2024.3463420

[47] Jun Ye, Kai Li, Guo-Jun Qi, and Kien A. Hua. 2015. Temporal Order-Preserving Dynamic Quantization for Human Action Recognition from Multimodal Sensor Streams. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval (ICMR '15)*. Association for Computing Machinery, New York, NY, USA, 99–106. doi:10.1145/2671188.2749340

[48] Su-Kyung Yoon and Shin-Dug Kim. 2020. Pattern analysis based data management method and memory-disk integrated system for high performance computing. *Future Generation Computer Systems* 106 (2020), 185–198. doi:10.1016/j.future.2020.01.013

[49] Xingtong Yu, Chang Zhou, Yuan Fang, and Xinming Zhang. 2024. MultiGPrompt for Multi-Task Pre-Training and Prompting on Graphs. In *Proceedings of the ACM Web Conference 2024* (Singapore, Singapore) *(WWW '24)*. Association for Computing Machinery, New York, NY, USA, 515–526. doi:10.1145/3589334.3645423

[50] Weizhe Zeng, Jie Zheng, Hai Wang, Qian Sun, Rui Cao, Shuo Ji, Jie Ren, and Ling Gaol. 2024. Delay-Aware Parallel Offloading AIGC Service in Edge Computing. In *Proceedings of the 2024 IEEE/CIC International Conference on Communications in China (ICCC Workshops'2024)*. 208–213. doi:10.1109/ICCCWorkshops62562.2024.10693717

[51] Degan Zhang, Wenjing Wang, Jie Zhang, Ting Zhang, Jinyu Du, and Chun Yang. 2023. Novel Edge Caching Approach Based on Multi-Agent Deep Reinforcement Learning for Internet of Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 24, 8 (2023), 8324–8338. doi:10.1109/TITS.2023.3264553

[52] Shubin Zhang, Hui Gu, Kaikai Chi, Liang Huang, Keping Yu, and Shahid Mumtaz. 2022. DRL-Based Partial Offloading for Maximizing Sum Computation Rate of Wireless Powered Mobile Edge Computing Network. *IEEE Transactions on Wireless Communications* 21, 12 (2022), 10934–10948. doi:10.1109/TWC.2022.3188302

[53] Siqi Zhang, Na Yi, and Yi Ma. 2024. A survey of computation offloading with task types. *IEEE Transactions on Intelligent Transportation Systems* 25, 8 (2024), 8313 – 8333. doi:10.1109/TITS.2024.3410896

[54] Youhan Zhao, Chenxi Liu, Xiaoling Hu, Jianhua He, Mugen Peng, Derrick Wing Kwan Ng, and Tony QS Quek. 2025. Joint Content Caching, Service Placement and Task Offloading in UAV-Enabled Mobile Edge Computing Networks. *IEEE Journal on Selected Areas in Communications* 43, 1 (2025), 51 – 63.

[55] Ivan Zyrianoff, Lorenzo Gigli, Federico Montori, Luca Sciullo, Carlos Kamienski, and Marco Di Felice. 2024. CACHE-IT: A distributed architecture for proactive edge caching in heterogeneous IoT scenarios. *Ad Hoc Netw.* 156, C (June 2024), 16 pages. doi:10.1016/j.adhoc.2024.103413

Received 30 November 2024; revised 25 March 2025; accepted 22 April 2025

Just Accepted