Exploring Collaborative Diffusion Model Inferring for AIGC-enabled Edge Services

Weijia Feng, Ruojia Zhang, Yichen Zhu, Chenyang Wang*, Chuan Sun, *Member, IEEE*, Xiaoqiang Zhu, *Member, IEEE*, Xiang Li, and Tarik Taleb, *Senior Member, IEEE*

***Abstract*—With the advancements in AI-generated content (AIGC) technologies and edge cloud networks, the demand for AIGC services is increasing. The diffusion model stands out for its ability to generate images from text. However, its high computational requirements challenge user devices, leading to ongoing efforts to improve inferring speed while preserving image quality. In this study, we propose a novel edge-user collaborative inferring framework. By capitalizing on collaboration among the devices in edge and user, the proposed framework optimizes the service delay, network resource consumption, and user device computing resource consumption of the user's AIGC service, and improves the user's QoE. To demonstrate the efficiency of our proposed framework, we conduct comparative experiments and ablation experiments on a variety of datasets. Experimental results show that the proposed framework achieves better-generated image quality and reduces a large number of computing resources and network consumption, improving user QoE.**

***Index Terms*—Edge computing, task offloading, knowledge distillation, Artificial Intelligence Generated Content**

I. Introduction

The increasing demand for high-quality content has rendered traditional input-output models inadequate for contemporary users. To address this shortcoming, researchers increasingly utilize artificial intelligence to fulfill user requirements more effectively [1]–[3]. Originally developed to address the challenges of data intelligence within the digital economy, Artificial Intelligence-Generated Content (AIGC) presents an innovative approach to content creation and manipulation [4]. AIGC utilizes AI algorithms to automatically generate, modify, and enhance diverse data in novel ways. Its impressive performance has driven widespread adoption across diverse fields, including natural language generation (NLG) and image generation. Notable applications exemplifying these capabilities include ChatGPT [5] and Stable Diffusion [6].

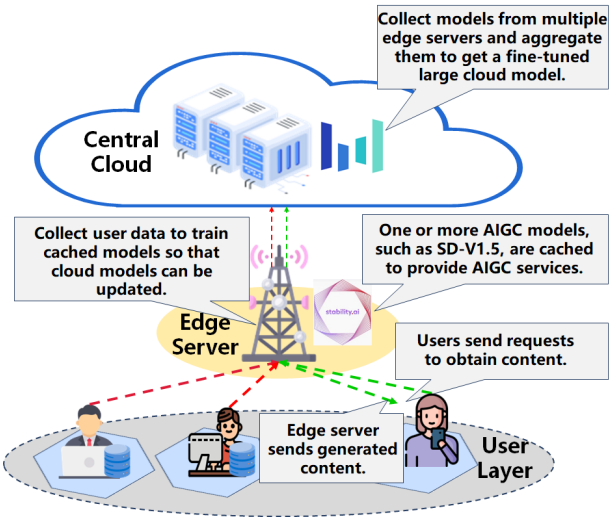


Fig. 1. Overview of Edge-cloud Collaborative AIGC service

AIGC is reshaping conventional production workflows, yet significant computational and network constraints limit its deployment on mobile devices. Large AIGC models, such as Stable Diffusion V1.5, require high-performance GPUs and substantial memory, far exceeding the capabilities of typical mobile devices [7]. Additionally, the extensive data traffic involved in AIGC model inference and training strains wireless networks, leading to congestion and unpredictable latencies that conventional network management techniques are ill-equipped to handle [8], [9]. These limitations restrict AIGC accessibility and necessitate alternative computational frameworks that reduce both resource demand and network load. To address these barriers, researchers are increasingly looking toward Edge Computing (EC) as a solution, offering

This study was funded by the NSFC (Grant No. 61602345, 62002263); National Key Research and Development (Grant No. 2019YFB2101900); Application Foundation and Advanced Technology Research Project of Tianjin (Grant No. 15JCQNJC01400); TianKai Higher Education Innovation Park Enterprise R&D Special Project (Grant No. 23YFZXYC00046); the Science and Technology Research Program of Chongqing Municipal Education Commission (Grant No. KJQN202400526). This work was also conducted at ICTFICIAL Oy, Finland; and supported in part by the EU's HE research and innovation program HORIZON-JUSNS-2023 under the 6G-Path project (Grant No. 101139172) and by the AerOS project funded by the EU's Horizon Europe, the EU's key funding program for research and innovation under Grant No. 101069732. The paper reflects only the authors' views, and the European Commission bears no responsibility for any utilization of the information contained herein.

W. Feng, R. Zhang, Y. Zhu, and X. Li are with the School of Computer and Information Engineering, Tianjin Normal University, Tianjin, China (e-mail: weijiafeng@tjnu.edu.cn, zrj20001127@163.com, zhuyiichen@163.com, yylixiang@tjnu.edu.cn).

C. Wang is with the College of Computer Science and Software Engineering, Shenzhen University; and the Guangdong Laboratory of Artificial Intelligence and Digital Economy, Shenzhen 518000, China.

C. Sun is with the College of Computing and Data Science, Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore (e-mail: chuan.sun@ntu.edu.sg).

X. Zhu is with the School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China; and also with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, China (e-mail: xqzhu@bjtu.edu.cn).

T. Taleb is with the Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum, 44780, Germany (e-mail: tarik.taleb@rub.de).

*Chenyang Wang is the corresponding author (e-mail: chenyangwang@ieee.org).

a distributed approach to computation that brings processing closer to data sources and alleviates network burden [10]–[12].

Unlike traditional cloud computing, which relies on centralized resources, EC offers localized processing, thus reducing latency and enhancing efficiency in resource-constrained environments [13]–[15]. Recent works demonstrate this potential: Wang *et al.* [16] proposed a joint optimization algorithm for 6G mobile edge networks, which efficiently offloads computing tasks and dynamically adjusts the diffusion steps of diffusion-based AIGC models. Likewise, Wang *et al.* [17] investigated the integration of EC with artificial intelligence-generated image content (AIGIC) to reduce terminal device loads and enhance processing efficiency.

This paper addresses the high computational demands and load challenges associated with edge servers by introducing an innovative Edge-User Collaborative Inference (EUCI) framework, as shown in Fig. 2. This proposed EUCI partitions the inference process of the diffusion model between users and edge servers, allowing for distributed and efficient computation, and operates across three layers: a central cloud, edge servers, and users. The central cloud aggregates and fine-tunes models from edge servers and periodically updates cached models to maintain performance. Edge servers host AIGC models (*e.g.,* SD-V1.5) to provide localized content generation services, while users send requests to retrieve generated content. The main contributions of this paper are listed as follows:

- **Edge-User Collaborative Inferring Framework:** We propose a collaborative framework, namely EUCI, that effectively addresses the resource limitations of user devices. By distributing inference tasks between users and edge servers, this approach enables high-quality image generation even under constrained network conditions, providing a robust and efficient method for AIGC on edge-enabled devices.
- **Reduction in Cloud Server Load and Network Energy Consumption:** Traditional diffusion model inference typically depends on intensive cloud-to-user communications, significantly burdening cloud resources. The proposed EUCI offloads part of the inference computation to edge servers and users, substantially reducing the reliance on cloud resources and decreasing network energy consumption.
- **Enhanced Performance and Resource Efficiency:** Comprehensive evaluations of the proposed EUCI against state-of-the-art approaches demonstrate its effectiveness across multiple datasets. Through extensive ablation studies, we show that our framework achieves competitive or superior image quality, as measured by SSI and PSNR, and outperforms existing methods regarding inference speed and resource efficiency.

The remainder of this article is organized as follows: Section II reviews the preliminaries of diffusion-based AIGC models for distributed feasibility. Section III introduces the system model for distributed inference. Section IV formulates the optimization problem, and Section V details our proposed framework. Experimental results are discussed in Section VI, with conclusions in Section VII.

II. Preliminaries on Diffusion-based AIGC

A. Development of Diffusion Model

Diffusion models (DMs), also widely known as diffusion probability models, are a class of Markov chain generative models trained via variational inference [18]. These models are designed to learn a process of gradually adding noise to data, a process known as diffusion, which is then reversed for sample generation. The concept of DMs has its roots in earlier works such as score-based generative models (SGM) [19] and diffusion probabilistic models (DPM) [20], which laid the foundation for the more advanced denoising diffusion probabilistic models (DDPM). The denoising diffusion probabilistic model (DDPM) marked a significant milestone in generative modeling. It introduced a parameterized Markov chain that generates images from noise through a series of denoising steps during inference. The training involves learning to reverse the noise addition process, which is added to natural images in each step [21]. This approach has sparked considerable interest and development in the field of generative models [22].

Incorporating guidance into diffusion-based image synthesis is crucial to improving the quality and relevance of generated images. Earlier studies have shown that labels can improve the quality of image synthesis in generative adversarial models (GANs) [21], [23]. As a core model in the field of image generation [24], diffusion models combine this principle to develop conditional and guided diffusion models to exploit additional information such as category labels or text to achieve better synthesis.

One of the pioneering frameworks based on diffusion models is GLIDE [25], which operates directly in high-dimensional pixel space. GLIDE adopts a classifier-free guidance mechanism to replace class labels with text to regulate sample generation. Compared with previous methods such as DALL-E, this method performs well in fidelity and human evaluation. Concurrent with pixel space methods, there has been significant progress in frameworks that operate in the latent space. Stable Diffusion is a notable example, which uses a variational autoencoder (VAE) to compress images into a low-dimensional space before applying the diffusion model. This approach effectively reduces complexity and preserves details in the generated images.

B. Workflow of Diffusion-based AIGC

In this section, we use Stable Diffusion (SD) to demonstrate the diffusion-based AIGC model:

a) The user's prompt is processed to create a text embedding. Stable Diffusion uses a tokenizer and a text encoder, typically CLIP [26], to transform the input text into a machine-readable embedding representation.
b) Latent Noise Image Generation: A random noise image is generated within the latent space. The same noise image can be reproduced for identical seed values by configuring the seed for the random number generator.

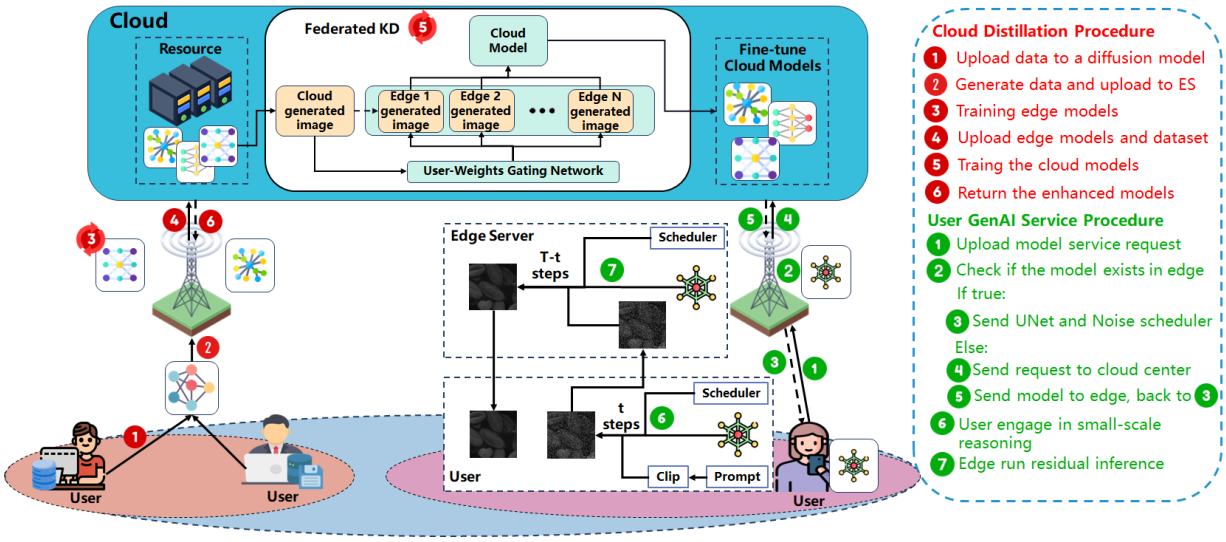


Fig. 2. Illustration of Edge-User Collaborative Diffusion-based AIGC Framework

This initial noise image contains only random noise without any coherent structure.

c) Noise Prediction Application: The noise predictor, a neural network, processes the latent noisy image and the text embedding to estimate the amount of noise that needs to be removed in the next step.

d) Refinement of Latent Noise Image: The initial noise image is refined by subtracting the predicted noise, with the Unet model in Stable Diffusion facilitating this denoising step.

e) Iterative Enhancement and Final Image Generation: Steps (b) through (d) are repeated to improve image quality progressively. Once sufficient refinement is achieved, the Variational Autoencoder (VAE) decoder transforms the refined latent representation back into pixel space, generating the final image output.

C. Knowledge Distillation and Distributed Inferring of DMs

Tian *et al.* [27] proposed a novel edge-cloud collaborative framework that integrates large cloud models with small edge models. This framework was designed to establish a distributed training architecture and a task-oriented deployment scheme, enabling the efficient provision of local Generative AI (GenAI) services. Yan *et al.* [28] proposed an edge-cloud collaborative inferring framework named Hybrid SD, which allocated inference tasks between large models hosted in the cloud and smaller models deployed on edge devices, facilitating collaborative inference between the edge and cloud environments. Chen *et al.* [8] proposed an innovative 6G native AI framework, which achieved a smarter task experience through cloud-edge-end collaboration, and discussed the challenges of implementing this framework and future research directions. Huang *et al.* [29] proposed a framework called "Collaborative Diffusion", which achieves multimodal face generation and editing through the collaboration of multiple pre-trained unimodal diffusion models without retraining. Allmendinger *et al.* [30] proposed a new distributed collaborative diffusion model approach, named CollaFuse, which aims to address some challenges in generative artificial intelligence (GenAI), especially in terms of data availability, computational requirements, and privacy. Du *et al.* [31] discussed that the integration of artificial intelligence-generated content (AIGC) services within wireless edge networks represented a significant advancement. They addressed the critical need for dynamic AIGC service provider (ASP) selection schemes.

In the field of exploring diffusion model distillation, Luhman *et al.* proposed a method that compresses multi-step denoising processes into single steps through knowledge distillation techniques [32], significantly improving sampling speed and making it closer to single-step generative models such as GANs and VAEs. Salimans and Ho *et al.* [33] used a progressive distillation method to distill a trained deterministic diffusion sampler (such as DDIM) into a new model that reduces the required sampling steps by half while maintaining sample quality. Sauer *et al.* introduced a novel training method called Adversarial Diffusion Distillation (ADD) in their study [34]. By combining adversarial loss and score distillation loss, the ADD method not only improves image fidelity under low-step sampling conditions but also significantly outperforms existing few-step methods in single-step sampling.

In this paper, our goal is to show how our work is distinguished from existing AIGC service frameworks through unique contributions and innovations. The main difference between the proposed framework and the previous studies is the distributed processing of computational tasks, which reduces the reliance on the central cloud by distributing tasks between user devices and edge servers, thereby reducing latency and improving service quality.

III. System Model

A. Network Architecture

As shown in Fig. 2, the proposed framework of Edge-User collaborative inferring involves several users, denoted as $N =$

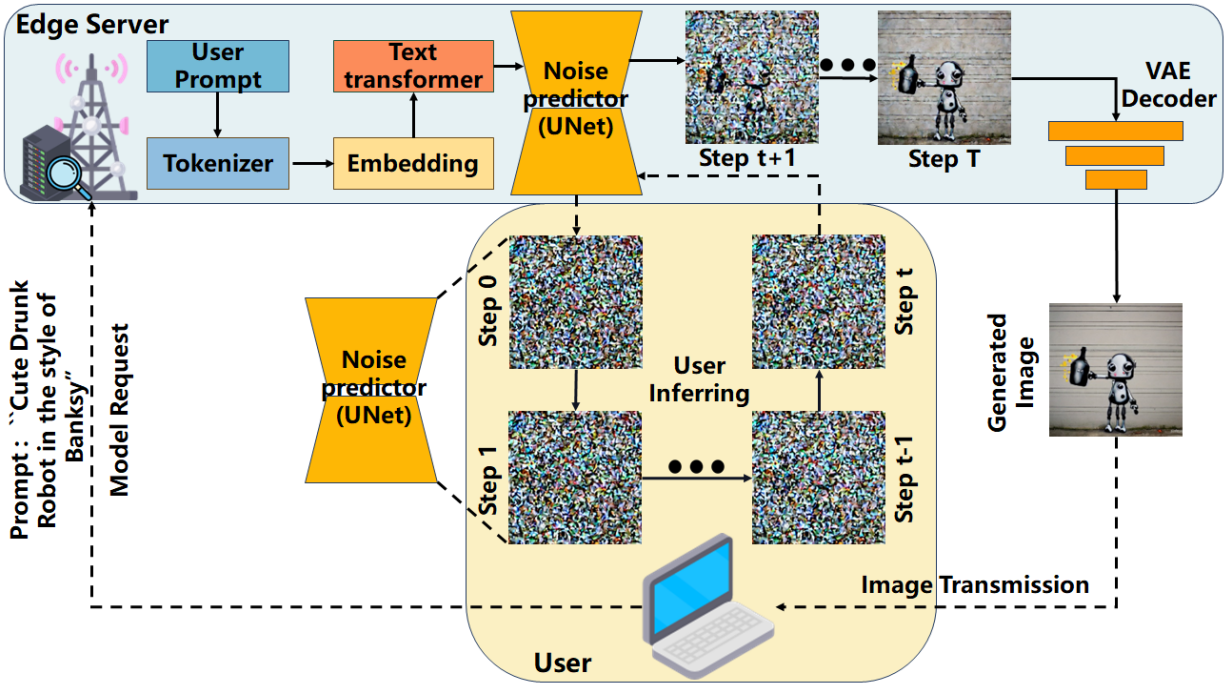


Fig. 3. Workflow of Edge-User Diffusion Model Collaborative Inferring

$\{n|1,2,\cdots,\mathcal{N}\}$, who are randomly distributed across the network. Each user is equipped with a performance-constrained GPU, capable of $\mathcal{C}_n$ floating-point operations per second (FLOPS) and an energy consumption of $\mathcal{P}_n$. To be specific, several base stations (BSs), denoted as $\mathcal{B} = \{b|1,2,\cdots,\mathcal{B}\}$, are evenly distributed across the network. Each BS is equipped with an edge server (ES) that contains a high-performance GPU with $\mathcal{C}_b$ FLOPS, an energy consumption of $\mathcal{P}_b$, and limited storage resources $\mathcal{S}_b \in R^+$. It is important to note that $\mathcal{C}_b$ is significantly larger than $\mathcal{C}_n$, which results in $\mathcal{P}_b$ being substantially higher than $\mathcal{P}_n$. Additionally, we assume that cloud C maintains a model library within the network, which contains $\mathcal{M}_C^m, m \in \{m|1,2,\ldots,M\}$ popular Stable Diffusion (SD) models. The size of each SD model is denoted as $\kappa_m \in \mathbb{R}^+$. At the same time, we assume that each edge server caches a subset of the models stored on the cloud server, denoted as $\mathcal{M}_E^b, b \in \mathcal{B}$.

We assume that each base station (BS) is connected to cloud center C via backhaul links, facilitated by network devices such as switches, wireless controllers, and gateways, all of which possess high computational capabilities. Additionally, the system's user equipment (UE) is classified into two categories: indoor UEs and outdoor UEs, based on their respective service locations. The indoor user equipment (UEs) consists of smart home devices (*e.g.,* laptops, PCs) and Internet of Things (IoT) devices, while the outdoor UEs include mobile devices and Internet of Vehicles (IoV). We assume that indoor UEs are connected to routers with a power supply, enabling access to remote cloud centers via network devices and wired links. Outdoor UEs, on the other hand, connect to base stations (BSs) through cellular links and are subject to battery life constraints. Finally, we assume that cloud center C has sufficient computational resources.

The model training process is shown in the Cloud Distillation Procedure in Fig. 2. Firstly, the shared dataset $\mathcal{D}_n$ is processed by a diffusion model $\mathcal{M}_l$ through steps ① and ② to generate a new training dataset $\mathcal{D}_n^{'}$ and send it to edge server b. After receiving $\mathcal{D}_n^{'}$ from the users, edge server b performs steps ③ and ④, training the cached model $\mathcal{M}_E^b$ and combining the trained model with the user-generated training dataset $\mathcal{D}_n^{'}$ and upload to cloud center C. After receiving the training models uploaded by each edge server, cloud center C performs step ⑤ to update the cached models $\mathcal{M}_C$ and store them in the cloud. The purpose of doing so is to ensure that the model benefits from the collective knowledge of edge servers and users while protecting data privacy. In step ⑥, the cloud center C will return the trained model $\mathcal{M}_C$ to edge server b to meet the user's AIGC service requirements.

The model inferring process is shown on the right side of Fig. 2. Firstly, in step ①, user n sends a model request U_{req} and a text prompt U_{prt} to edge server b. Then the edge server b performs step ② to check if the model requested by user n has been cached. If edge server b has already cached the requested model, then edge server b performs step ③ to send the Unet and Noise scheduler components in the requested model to user n. After receiving the Unet and Noise scheduler components, user n performs small inference steps t in step ⑥ to obtain an intermediate latent noise image $\mathcal{X}_t$. Afterward, the user performs step ⑦ to upload the intermediate latent noise image $\mathcal{X}_t$ to edge server b, which conducts the remaining inference steps $T-t$ to obtain the final generated image I_g. Suppose edge server b does not have the model. In that case, it performs step ④ to send the model request U_{req} to cloud center C, and then cloud center C performs step ⑤ to transfer

the model to edge server b, which will continue to perform the above steps.

B. Task Computation Model

In the energy consumption calculation framework of our system, task P is deployed to both user n and edge server b, denoted as $\mathbf{a}_P = \{a_P^{user}, a_P^{edge}\}$, where $a_P^{user}, a_P^{edge} \in \{0, 1\}$. These values indicate the proportion of inference steps performed on user n and edge server b respectively, in total inference steps T. The indicator function directly affects service latency and our system's energy consumption. It is subject to constraints based on the specific deployment location, whether on the user, the edge server, or the cloud, influencing the overall performance of our system.

$$a_P^{user} + a_P^{edge} = 1, \forall P \in \mathcal{P}. \tag{1}$$

We consider the overall energy consumption of GPUs in our proposed framework, which is calculated by multiplying the running time of the GPUs on both edge server b and user n by their respective energy consumption rates and then summing the results. Regarding the FLOPs generated by users and edge servers during inferring, we calculate the total FLOPs by adding the FLOPs generated by the Unet model, text encoder, and VAE components. Since the inferring process does not change $S_{\mathcal{X}}$, these fixed values are denoted as F_U, F_{te}, and F_{vae}, respectively. Therefore, we derive the formula for our system model task consumption as follows:

$$\begin{aligned} TC_{n,b} = &\frac{(F_U + F_{te} + F_{vae})t}{\mathcal{C}_n} * \mathcal{P}_n + \\ &\frac{(F_U + F_{te} + F_{vae})(T - t)}{\mathcal{C}_b} * \mathcal{P}_b, \end{aligned} \tag{2}$$

where $TC_{n,b}$ represents the total energy consumption of GPUs, $t = a_P^{user} * T$, and $T - t = a_P^{edge} * T$. When user n performs the image generation task independently, t equals T. Conversely, if edge server b handles the image generation task alone, t equals 0.

C. Service Delay Model

In our proposed framework, the transmission process involves models, requests, text, latent noisy images, and generated images. We assume that the size of the Unet model is S_{Unet} and the size of the noise scheduler is S_{NS}. It should be noted that compared to S_{SD} and S_U, the size of user requests U_{req} and text prompts U_{prt} can be ignored, so they are not included in the total transmission time. In addition, we define the physical channel distance between user n and edge server b as $L_{U,B}$, and the actual channel distance between edge server b and cloud center C as $L_{B,C}$. In addition, the size $\mathcal{X}$ is set to a fixed value $S_{\mathcal{X}}$, and the network bandwidth is set to W. Finally, we define the image size obtained through collaborative inference as S_I. Therefore, we can conclude that the transmission delay of our system model is as follows:

$$\begin{aligned} TL_{n,b} &= \frac{W * (S_{Unet} + S_{\mathcal{X}} + S_{NS})}{L_{U,B}} + \frac{W * \kappa_m}{L_{B,C}} + \frac{W * S_I}{L_{U,B}} \\ &= \frac{W * (S_{Unet} + S_{\mathcal{X}} + S_I + S_{NS})}{L_{U,B}} + \frac{W * \kappa_m}{L_{B,C}}, \end{aligned} \tag{3}$$

where $TL_{n,b}$ represents the sevice delay of our system model. $S_{\mathcal{X}}$ refers to the text condition input transmitted by edge server b to user n during Unet transmission, and latent noise image $\mathcal{X}_t$ is sent to edge server b by user n after inferring step t. Note that when edge server b does not need to request the model from cloud center C, κ_m is 0.

In practical application scenarios, service latency is not only limited to file transfer latency and transmission latency but also includes the time required for content generation. Therefore, on the basis of file transmission delay and transmission delay, our service delay should also include the time required for the image to be generated by user n and edge server b. The complete service delay formula is as follows:

$$\begin{aligned} TL_{n,b} = &TL_{n,b} + \frac{(F_U + F_{te} + F_{vae})t}{\mathcal{C}_n} \\ &+ \frac{(F_U + F_{te} + F_{vae})(T - t)}{\mathcal{C}_b}. \end{aligned} \tag{4}$$

D. Knowledge Distillation Model

Assume that each user possesses a unique dataset D_n, $n \in \{n|1, 2, \cdots, N\}$, and these datasets are processed through a diffusion model to generate a corresponding new dataset $D_n^{'}, n \in \{n|1, 2, \cdots, N\}$. Images in $D_n^{'}$ are similar but not identical to images in the original dataset D_n. Edge server b receives dataset $D_n^{'}$ and constructs a new training dataset $\mathcal{D}_E^b = \{D_n^{'}|n \in N\}$. Subsequently, edge server b uses the new dataset $\mathcal{D}_E^b$ to fine-tune diffusion model $\mathcal{M}_E^b$ cached in edge server b. After model $\mathcal{M}_E^b$ is fine-tuned or trained, cloud center C collects training dataset $\mathcal{D}_E^b$ from each edge server and combines them to form a new cloud center training dataset $\mathcal{D}_C$. Upon obtaining a new training dataset $\mathcal{D}_C$, model $\mathcal{M}_C^m$ cached in cloud center C is trained using a federated knowledge distillation method for diffusion models.

We define the loss of federated knowledge distillation at cloud center C as comprising two components: One component is the loss $\mathcal{L}_{task}$ of model $\mathcal{M}_C^m$ during training on dataset $\mathcal{D}_C$. The second loss, $\mathcal{L}_{distill}$, represents the difference between the images generated by model $\mathcal{M}_C^m$ and edge server cached model $\mathcal{M}_E^b$ on dataset $\mathcal{D}_C$. The formulas for the two types of losses are as follows:

$$\mathcal{L}_{task} = \mathbb{E}_{t, \mathcal{X}_0, \epsilon}[||\epsilon - \epsilon_\theta(\mathcal{X}_t, t)||^2], \tag{5}$$

$$\mathcal{L}_{distill} = \sum_{b=1}^{\mathcal{B}} \lambda_b \mathbb{E}_{t, \epsilon'}[c(t) d(\hat{x}_C(\mathcal{X}_t, t), \hat{x}_d(\hat{x}_{C,t}; t)], \tag{6}$$

where Eq.(6) holds $\sum_{b=1}^{\mathcal{B}} \lambda_b = 1$.

In Eq. (5), x_0 represents the original image in dataset $\mathcal{D}_C$, $\mathcal{X}_t$ denotes latent noise image during the forward diffusion process, and $\epsilon_\theta(\mathcal{X}_t, t)$ represents the noise predicted by the model, which is the noise the model aims to learn. In Eq.

TABLE I
KEY NOTATIONS

Notation	Meaning
T	Number of inference iterations of the framework
t	User inferring iterations
F_U	Floating point operations performed by Unet
F_{te}	Floating-point operations for text processing
F_{vae}	Floating-point operations for VAE converted images
$\mathcal{C}_n$	The FLOPS of the user's GPU
$\mathcal{C}_b$	The FLOPS of the edge server's GPU
$\mathcal{P}_n$	Energy consumption of user's GPU
$\mathcal{P}_b$	Energy consumption of edge server's GPU
$S_{\mathcal{X}}$	The size of the latent noise image $\mathcal{X}$
$S_{\mathcal{U}}$	The size of Unet model
$S_{\mathcal{NS}}$	The size of noise scheduler
S_I	The size of generated iamge
$L_{U,B}$	Distance between the user and the edge server
$L_{B,C}$	Distance between the edge server and the cloud
W	Network bandwidth
κ_m	The size of the mth diffusion model
I_g	The generated image
$\mathcal{L}_{task}$	The task loss of $\mathcal{M}_C^m$
$\mathcal{L}_{distill}$	The distillation loss of $\mathcal{M}_C^m$
Q_{user}	The user's QoE
$I_{quality}$	The quality of generated image I_g
λ	The weight parameter of $\mathcal{L}_{distill}$

(6), λ_b represents the weight of each edge server model, and $c(t)$ is a weight function that adjusts the weights of different steps based on the noise level t, d represents a similarity function, such as the structural similarity index (SSI) calculation function, ϵ and $\epsilon^{'}$ denote standard Gaussian distributions with a mean of 0 and a variance of 1. Common weight functions include Score Stochastic Differential Equations (Score SDEs) [35], among others. The primary objective of the distillation loss $\mathcal{L}_{distill}$ is to use the output of the teacher model as the target, guiding the student model to generate outputs that closely match those of the teacher model. In summary, the loss function can be defined as follows:

$$\begin{aligned}\mathcal{L} &= (1-\gamma)\mathcal{L}_{task} + \gamma\mathcal{L}_{distill} \\ &= (1-\gamma)\mathbb{E}_{t,\mathcal{X}_0,\epsilon}[||\epsilon - \epsilon_\theta(\mathcal{X}_t, t)||^2] + \\ &\gamma\sum_{b=1}^{\mathcal{B}}\lambda_b\mathbb{E}_{t,\epsilon'}[c(t)d(\hat{x}_C, \hat{x}_d(\hat{x}_{C,t}; t)],\end{aligned} \tag{7}$$

where γ is the specific gravity parameter of the distillation loss, and the model $\mathcal{M}_C^m$ in cloud center C is updated by performing a reverse gradient operation on loss $\mathcal{L}$.

IV. PROBLEM FORMULATION

In this paper, our goal is to minimize service latency $TL_{n,b}$ and energy consumption $TC_{n,b}$ of the user's text-to-image tasks while maximizing the quality of the final image generated, thereby enhancing the overall user Quality of Experience (QoE).

A. User's QoE for Text-to-Image Task

Quality of Experience (QoE) metrics play a key role in evaluating AIGC performance [36]. QoE measures user satisfaction with generated content, taking into account factors such as visual quality, relevance, and usefulness. In this article, we consider service latency $TL_{n,b}$ and task energy consumption $TC_{n,b}$ as key factors for measuring the user's QoE. The task of the text-to-image is to generate an image that aligns with the user's input text. Three indicators, MSE, SSI, and PSNR, are introduced to evaluate the quality of generated image I_g. These indicators comprehensively assess the generative requirements of users by evaluating each generated image across all three metrics. We define the user's QoE as Q_{user}, *i.e.,*

$$Q_{user} = w_1 \cdot I_{quality} - w_2 \cdot TL - w_3 \cdot TC, \tag{8}$$

where $w_1 + w_2 + w_3 = 1$. In Eq. (8), $I_{quality}$ represents the quality of the generated image I_g, defined as a combination of MSE, SSI, and PSNR. The weight coefficients w_1, w_2, and w_3 are used to adjust the relative importance of each indicator. Therefore, the image quality $I_{quality}$ is denoted as:

$$I_{quality} = \phi \cdot I_{PSNR} + \varphi \cdot I_{SSI} - (1 - \phi - \varphi) \cdot I_{MSE}, \tag{9}$$

where $\phi + \varphi < 1$. I_{MSE}, I_{SSI}, and I_{PSNR} represent the MSE, SSI, and PSNR of generated image I_g respectively. These indicators' weight parameters are ϕ and φ. Therefore, the objective of the optimization problem in our framework can be summarized as follows:

$$\begin{aligned}\max_{O,P,Q} Q_{user} =& w_1 \cdot (\phi \cdot I_{PSNR} + \varphi \cdot I_{SSI} \\ &- (1 - \phi - \varphi) \cdot I_{MSE}) \\ &- w_2 \cdot TL_{n,b} - w_3 \cdot TC_{n,b},\end{aligned} \tag{10}$$

where $w_1 + w_2 + w_3 = 1$, $\phi + \varphi < 1$, and denote O, P, Q as $I_{quality}$, TC and TL, represently.

B. Federated Knowledge Distillation Process

We consider a federated knowledge distillation method in our model training phase, which involves updating the student model by measuring the differences between the generated images of the student and teacher models $\mathcal{M}_C$. In our diffusion-based AIGC service framework, the goal is to train cloud model $\mathcal{M}_C$ through federated knowledge distillation using an end-to-end cloud network structure, which adapts to changing user needs while maximizing user's QoE. The training process aims to continuously reduce the distillation loss $\mathcal{L}$ between the cloud models and edge models. At the same time, the objective of maximizing QoE focuses on minimizing service latency $TL_{n,b}$ and system energy consumption $TC_{n,b}$, while improving image generation quality $I_{quality}$, *i.e.,*

$$\begin{aligned}&\arg\min_{\mathcal{M}_C^m} \mathcal{L} = \arg\min_{\mathcal{M}_C^m}(\mathcal{L}_{task} + \gamma\mathcal{L}_{distill}), \\ &s.t. \max_{O,P,Q} Q_{user}, \\ &TL_{n,b} \le TL_{max}, \forall(n.b) \in (\mathcal{N}, \mathcal{B}), \\ &TC_{n,b} \le TC_{max}, \forall(n.b) \in (\mathcal{N}, \mathcal{B}).\end{aligned} \tag{11}$$

The optimization problem is subject to the following constraints: service latency constraint, $TL_{n,b} \le TL_{max}, \forall(n, b) \in (\mathcal{N}, \mathcal{B})$, ensures that the latency between user n and edge server b does not exceed a maximum value. Previous research shows that deploying diffusion-based AIGC services

on user n results in high latency due to limited computational resources. The energy consumption constraint, $TC_{n,b} \leq TC_{max}, \forall(n,b) \in (\mathcal{N},\mathcal{B})$, ensures that the total energy consumption of users and edge servers does not exceed a maximum value. Earlier studies indicate that edge servers, despite having high instantaneous energy consumption, incur the highest total energy consumption, TC_{edge}, setting $TC_{max} = TC_{edge}$.

V. Edge-User Collaborative Diffusion-based AIGC model

A. Edge-User Collaborative Diffusion Model Inferring

To enhance user QoE and improve the quality of generated images, we propose an Edge-User Diffusion Model Collaborative Inferring framework. The framework consists of three phases: (1) the local user request phase, (2) the collaborative edge-user inference phase, and (3) the image generation and transmission phase.

1) Overview of Our Framework: In our framework, cloud center C caches diffusion models that cater to various user needs. We define the total number of inference steps as T, with t representing the inference steps for user n and $T-t$ representing the inference steps for edge server b.

- **Local user request phase:** At the start of the task, user n sends a diffusion model request U_{req} to edge server b. Edge server b first checks whether the requested diffusion model is cached. If the model is cached, edge server b sends the Unet model and noise scheduler from the diffusion model to user n while receiving text prompt U_{prt} from user n. If the model is not cached, edge server b forwards U_{req} to cloud center C, which then sends the diffusion model to edge server b. Edge server b subsequently caches the model for future use.
- **Edge-user collaborative inference phase:** In the user request phase, edge server b receives textual prompt U_{prt} sent by user n. Upon receiving U_{prt}, edge server b generates an initial latent noise image $\mathcal{X}_r$ containing the text information through step b). After receiving the Unet model, the noise scheduler, and $\mathcal{X}_r$, user n iterates steps c) and d) for several iterations corresponding to the user-side inference steps t to obtain the latent noise image $\mathcal{X}_t$. User n then sends the latent noise image $\mathcal{X}_t$ to edge server b, which continues to iterate steps c) and d) based on $\mathcal{X}_t$ to obtain the final latent noise image $\mathcal{X}_T$. The number of iterations performed by edge server b is $T-t$.
- **Image generation and transmission stage:** After obtaining the final latent noise image $\mathcal{X}_T$, which is collaboratively inferred by user n and edge server b, the VAE model on edge server b receives $\mathcal{X}_T$ as input to generate the final image I_g. Subsequently, edge server b transmits I_g to user n, thereby completing the task.

Alg. 1 outlines our edge-user collaborative inferring framework for executing a diffusion model to generate images that match text prompts. In our framework, edge servers and users share the inference task to optimize the computational load and improve the quality of service. Edge server b is responsible for processing the text prompt and generating the initial noisy image, then user n performs some of the inference steps. Finally, the edge server completes the remaining steps and generates the final image.

Algorithm 1 Edge-User Collaborative Diffusion Model Inferring

Initialize: Cloud diffusion model $\mathcal{M}_C^m$; Edge Stable Diffusion models $\mathcal{M}_E^b$; Collaborative inference steps T; Current inferring step $i=0$.
Output: The generated image I_g.
At Edge Server b***:***
Receive model requests U_{req} and text prompts U_{prt}.
if i=0 **then**
  Generate text vector z_{text} and initial noisy image $\mathcal{X}_r$ by (12) and (13).
end if
Send z_{text} and $\mathcal{X}_r$ to user n.
if i=t **then**
  Receive the initial noisy image $\mathcal{X}_t$.
end if
for $i = t+1$ to T **do**
  Generate the final noise-free image $\mathcal{X}_T$ by (15).
end for
Convert $\mathcal{X}_T$ to generate image I_g through U_{vae} in (16).
At User n***:***
Send model request U_{req} and text prompt U_{prt}.
if i=0 **then**
  Receive U_{Unet}, $U_{scheduler}$, $\mathcal{X}_r$, and z_{text} sent by edge server b.
end if
for $i = 0$ to t **do**
  Generate noisy image $\mathcal{X}_t$ at step t by (14).
end for
Send $\mathcal{X}_t$ to edge server b.

B. Edge-User Collaborative Inferring

The Edge User Collaborative Inferring Framework we proposed is shown in Fig. 3. In our inference process, we define a fixed inference step T, which represents the total number of inference steps performed by both edge server b and user n. In our proposed framework, we define t as the number of inference steps performed by user n, resulting in the edge server performing $T-t$ inference steps. At the beginning of the task, user n first sends model request U_{req} for model U_{model} and text prompt U_{prt} to edge server b. After receiving U_{req} from user n, edge server b first checks whether the model U_{model} requested by user n is cached on the server. If the model is already cached, edge server$b$ sends the Unet model U_{Unet} and the noise scheduler $U_{scheduler}$ to user n. If U_{model} is not cached, edge server b sends model request U_{req} to cloud center C for retrieval. Upon receiving U_{req}, cloud center C sends the complete model U_{model} to edge server b for caching, after which the communication process between edge server b and user n is repeated.

When the model transmission process is complete, edge server b first utilizes the text encoder $U_{encoder}$ to process U_{prt},

generating a fixed vector representation of U_{prt} as follows:

$$z_{text} = U_{encoder}(U_{prt}), \tag{12}$$

where z_{text} represents the fixed vector representation of U_{prt}. Next, a random noise image is generated for the reverse denoising process. The strategy for generating this random noise image is as follows:

$$\mathcal{X}_r \sim \mathcal{G}(0, I), \tag{13}$$

where $\mathcal{G}(0, I)$ represents a Gaussian distribution (normal distribution) with a mean of 0, and I denotes the covariance matrix (identity matrix). Note that I does not refer to the generated image I_g. $\mathcal{X}_r$ represents a completely noisy image generated through random sampling, and the reverse inferring process is performed gradually on this $\mathcal{X}_r$.

Edge server b transmits the text matrix z_{text} and the initial random noise $\mathcal{X}_r$ to user n. As shown in User Inferring in Fig. 3, upon receiving these, user n uses the z_{text} and $\mathcal{X}_r$ provided by edge server b to perform the reverse inference process. In this step, user n infers the initial noisy image $\mathcal{X}_r$. Meanwhile, when edge server b performs reverse inferring, it utilizes noisy image $\mathcal{X}_t$ obtained by user n after t inference steps. Although the inference processes of user n and edge server b are not directly related, they follow a Markov chain to ensure consistency in the inferring processes.The formula for user n to perform reverse inferring on the initial noisy image $\mathcal{X}_r$ is as follows:

$$\mathcal{X}_i = \sqrt{\frac{\alpha_i}{\alpha_r}}\mathcal{X}_r + \sqrt{1 - \frac{\alpha_i}{\alpha_r}}\epsilon_i(\mathcal{X}_i, i, z_{text}), i \in (0, t), \tag{14}$$

where $\mathcal{X}_t$ is the noisy image obtained by user n during inference step t, and α_t is a predefined attenuation coefficient determined by the noise scheduler, which assists the model in understanding the amount of noise to be removed at each step and restoring a clearer image from the current noisy image. α_r represents the initial attenuation coefficient, which is set to a value approximately equal to 0 to facilitate the generation of a completely noisy image, and $\epsilon_t(\mathcal{X}_t, t, z_{text})$ denotes the predicted noise of U_{model} at the current time step, and the noise image at this time step is obtained by subtracting $\mathcal{X}_i$.

User n sends $\mathcal{X}_t$ to edge server b to continue the remaining $T-t$ steps of inferring. Similar to the inference process of user n, edge server b also performs reverse inferring by subtracting the noise predicted by the model at each time step. In contrast, the initial noise image for edge server b is the noise image $\mathcal{X}_t$ obtained from the inference performed by user n. By combining it with the text vectors stored on edge server b, the system maintains semantic consistency in the user's text generation task, ensuring that the edge server remains aligned with the user prompt throughout the inference process. The formula for inferring the final noisy image $\mathcal{X}_T$ by the edge server b is as follows:

$$\mathcal{X}_{i'} = \sqrt{\frac{\alpha_{i'}}{\alpha_t}}\mathcal{X}_t + \sqrt{1 - \frac{\alpha_T}{\alpha_t}}\epsilon_{i'}(\mathcal{X}_{i'}, i', z_{text}), i' \in (t, T). \tag{15}$$

After obtaining the final noisy image $\mathcal{X}_T$, the model inputs $\mathcal{X}_T$ to VAE U_{vae} to convert it into pixel space, generating the final image:

$$I = U_{vae}(\mathcal{X}_T). \tag{16}$$

C. Time and space complexity analysis

The time complexity of our proposed EUCI framework is mainly determined by the iterative inference steps on user n and edge server b. Specifically, user n performs t inference steps. The time complexity of each iteration is $\mathcal{O}(F_U)$, where F_U represents the number of FLOPs of Unet model in a single iteration; edge server b performs $T - t$ iterations, and its time complexity is $\mathcal{O}((T - t)F_U)$. It also includes the time complexity of text encoding, initial noisy image generation, and final image generation, where text encoding and initial noisy image involve a fixed number of operations, so their time complexity can be considered to be at a constant level, that is, $\mathcal{O}(1)$. The final image generation involves converting the final noisy image into an image in pixel space through VAE. Its time complexity depends on the complexity of the VAE model, which can be considered to be $\mathcal{O}(F_{vae})$, where F_{vae} represents the number of FLOPs required for VAE conversion. Therefore, the total time complexity is $\mathcal{O}(tF_U + (T - t)F_U + F_{vae})$.

The space complexity of our framework is primarily determined by the storage requirements for the text vector, initial noisy image, intermediate noisy images from the iteration process, and the final generated image. If the space complexity of each image is $\mathcal{O}(S_{\mathcal{X}})$, the total space complexity depends on the number of images that need to be stored. Considering that both the user device and the edge server need to store at least one noisy image, the space complexity is at least $\mathcal{O}(S_{\mathcal{X}})$. In addition, there are some intermediate variables involved in the algorithm, such as predicted noise, which are usually proportional to the image size, so the space complexity is also $\mathcal{O}(S_{\mathcal{X}})$.

VI. Experimental Results and Discussions

This section presents the numerical results and discusses our proposed collaborative edge-user AIGC framework. The discussion is organized into two parts: the implementation of the simulation environment and the performance analysis.

A. Experimental Settings

Our experimental settings include several mainstream stable diffusion models for cloud caching, such as SDXL, SD-V1.5, and SD2.1. The SD-V1.5 model is pre-cached in the edge server. In our experiment, the user is simulated as an iPhone 14 Pro Max, while the edge server is equipped with an Nvidia 4090 (24G) GPU. Both the user and the edge server are deployed in a 4G network environment, with the user sending model requests and text prompts to the edge server. The user, edge server, and cloud center generate the content collaboratively.

We conduct two types of experiments: comparative experiments and ablation experiments. The comparative experiment evaluates our framework against existing frameworks in terms of MSE, PSNR, and SSI during image generation. The ablation experiment compares the service latency and computational

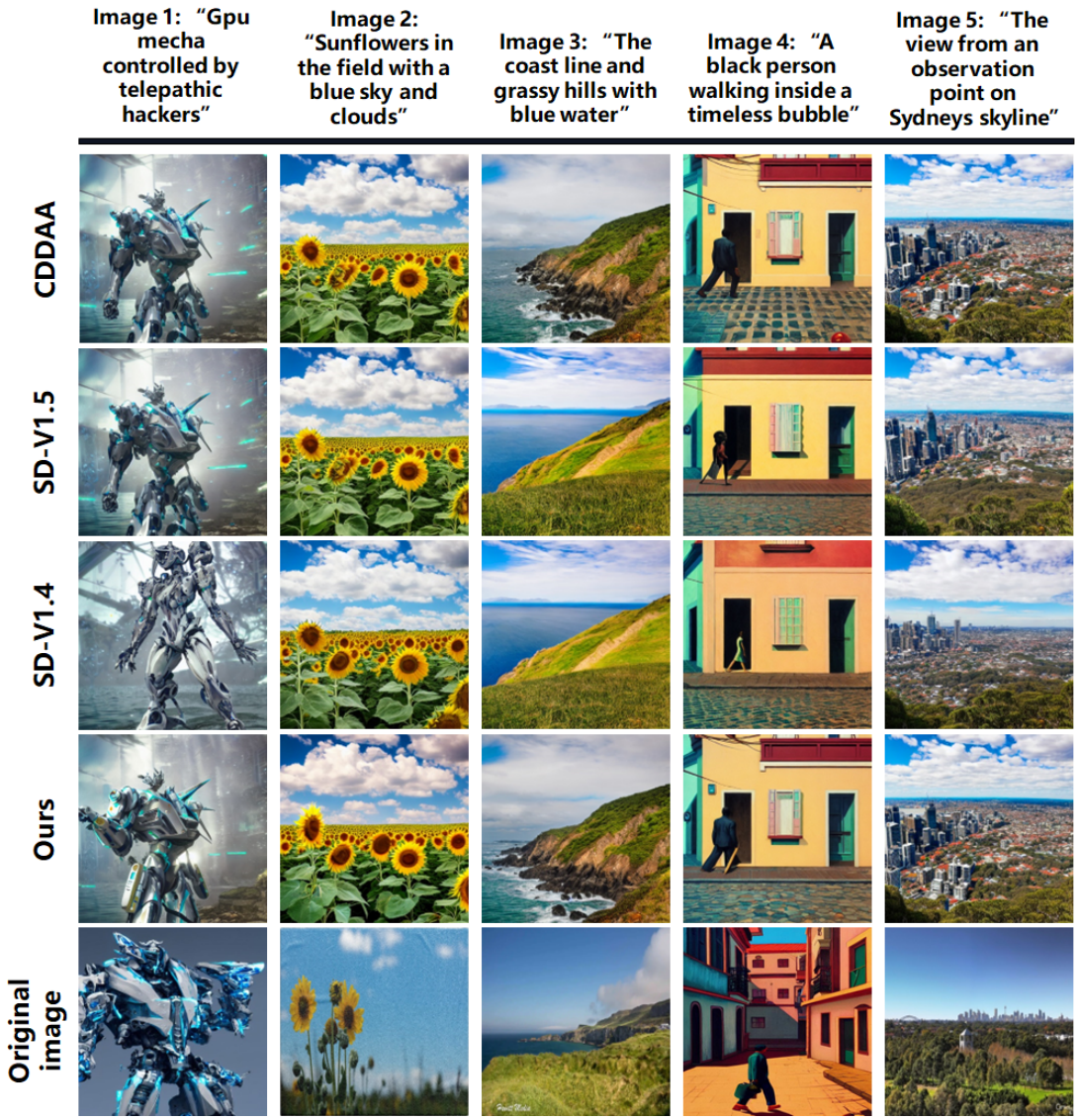


Fig. 4. Visualization of the generated images. We used prompts from the diffusiondb2m_first_5k_canny and laion-coco aesthetic datasets respectively and compared them with the corresponding prompt images in the datasets. For the sake of simplicity, some prompts have been omitted in this section. Although smaller models have slightly reduced semantic details compared to models such as SD-v1.4, our framework is better able to maintain consistency between images and semantics.

resource consumption between our framework and configurations where tasks are handled separately by the user or edge server.

Baselines: To demonstrate the superiority of our proposed framework, we compare it with several diffusion-based models in terms of image quality. Specifically, we evaluate the quality of generated images using three metrics: MSE, PSNR, and SSI. We select two baseline diffusion models and one collaborative inferring framework as the control group. The baseline models include Stable Diffusion v1.4 from Ommer Lab and Stable Diffusion v1.5 from RunwayML. For the collaborative inferring approach, we use the CDDAA framework, proposed by Du *et al.* [31]. The chosen diffusion model in our collaborative inferring framework is SD v1.5. For the datasets, we select the diffusiondb2m_first_5k_canny [37] and the Laion-coco aesthetic datasets [38] Both datasets are designed for text-to-image modeling, with diffusiondb2m_first_5k_canny containing 5,000 text-image pairs, and the Laion-coco aesthetic dataset comprising 600 million high-quality descriptive captions generated for publicly available images from the web.

Configurations: In our experiment, the FLOPS of user n is set to a maximum of 17 TFLOPS, while the FLOPS of GPU in edge server b could reach 1321 TFLOPS. At these FLOP levels, the user's energy consumption can reach 4.76W, while the energy consumption of the 4090 (24GB) can reach 450W. Practical application research indicates that the GPU utilization rate in existing diffusion models for inference can reach 95% or even 100%. Based on this, we set the parameters for the user n and edge server b in the experimental environment as described above.

In our simulation environment, user n, edge server b, and cloud center C operate in a 4G network, resulting in a network

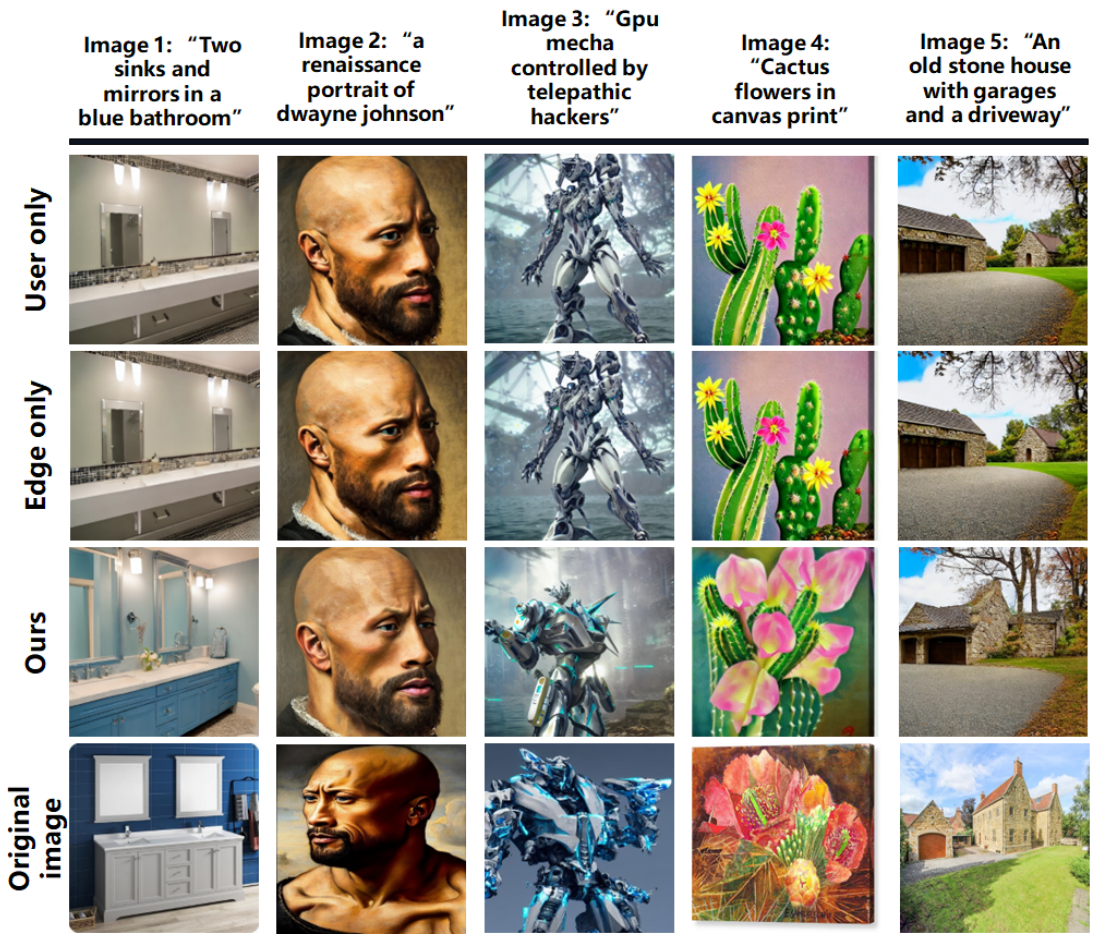


Fig. 5. Visualization of generated images of ablation experiments.

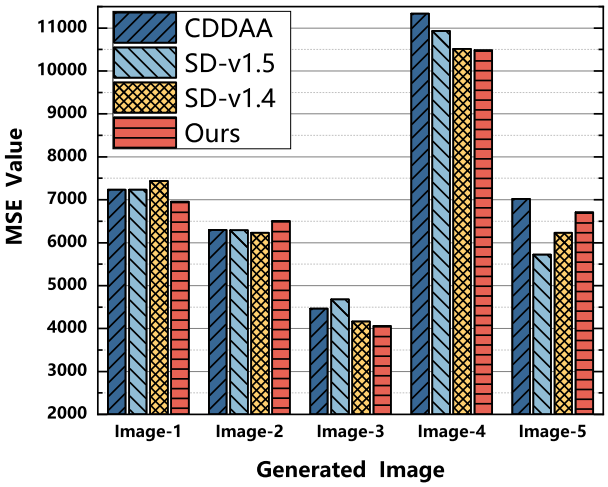
bandwidth range between the user, edge server, and cloud center, meaning that the theoretical maximum download speed can reach 125 Mbps. Additionally, we simulate the physical distances between user n, edge server b, and cloud center C, which impacts the transmission time of models and images. The distance between the user and the edge server is set to 5 kilometers, denoted as $L_{U,B}$, while the distance between the edge server and the base station is set to 50 kilometers, denoted as $L_{B,C}$.

Key Performance Indicators (KPI): In terms of key performance indicators, we measure the quality of image generation using MSE, SSI, and PSNR. MSE (Mean Squared Error) quantifies the difference between the predicted and true values for each pixel in an image. A lower MSE value indicates that the generated image is closer to the real image [39]. SSI (Structural Similarity Index) assesses the similarity between two images, with higher values indicating greater similarity [40]. PSNR (Peak Signal-to-Noise Ratio) is a widely used metric for evaluating image quality, particularly in terms of the differences between the generated and original images. Expressed in decibels (dB), a higher PSNR value indicates better image quality and a smaller difference from the original image [41]. In the ablation, we also introduce service latency and device consumption to further demonstrate the superiority of our framework compared to scenarios where tasks are handled independently by either the user or the edge server.

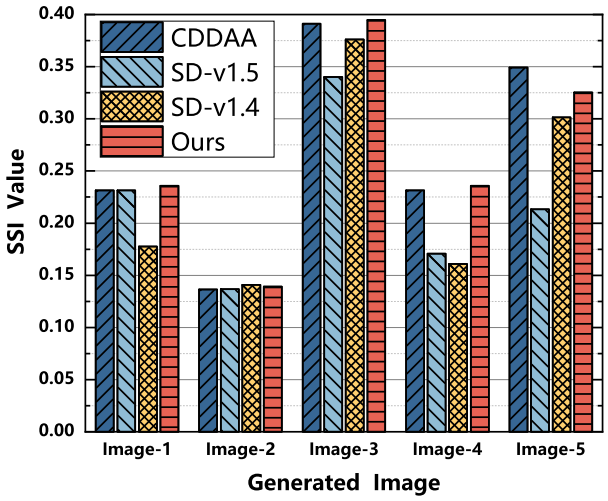
B. Performance of Different Baselines

Fig. 4 presents four different image generation models (CDAA, SD-V1.5, SD-V1.4, and our proposed framework), each generating images based on the same text description and comparing them with real images. Each row corresponds to a model, and each column represents a distinct scene description, including "GPU mecha controlled by telepathic hackers", "Sunflower fields under a blue sky and white clouds", "Coastline and green hills", "Black characters walking in time bubbles", and "Observation point perspective of the Sydney skyline.". All the prompts mentioned above are real prompts from the selected dataset. This comparison highlights differences in image generation quality, detail representation, and style across the models.

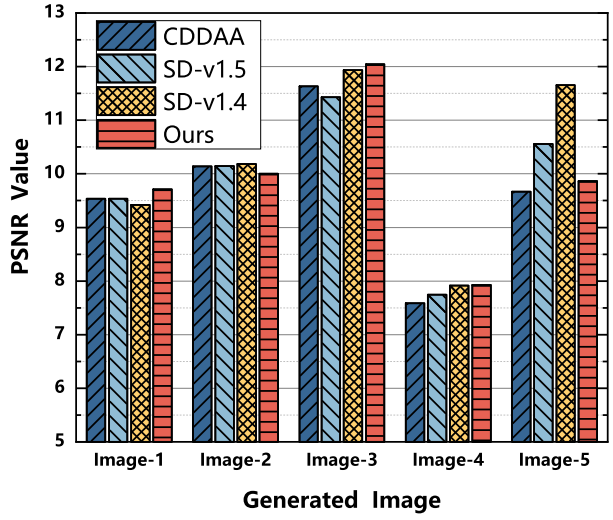
*1) **MSE results**:* Fig. 6(a) presents the experimental results comparing the mean square error (MSE) of different algorithms across five images. As shown in Fig. 6(a), the MSE values for our algorithm are lower than those of the comparison algorithms for most images, further demonstrating the effectiveness and superiority of our method. In particular, for Image 1, Image 2, and Image 5, our algorithm performs similarly to CDAA but significantly outperforms all other algorithms on Image 4. These results suggest that our



(a) MSE

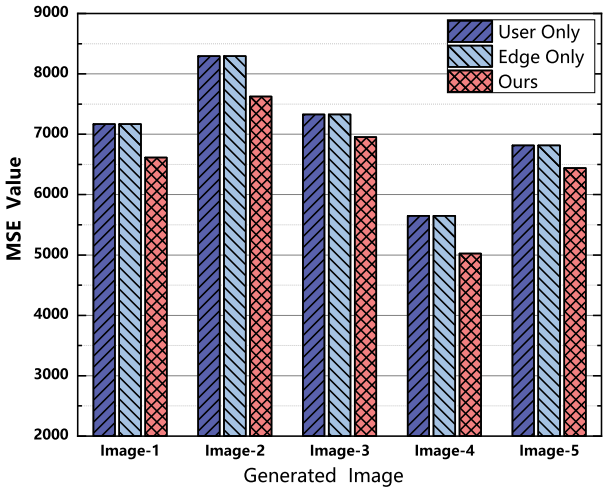


(b) SSI

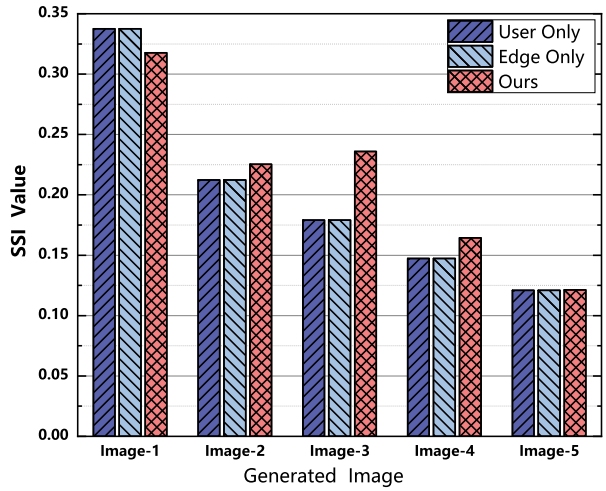


(c) PSNR

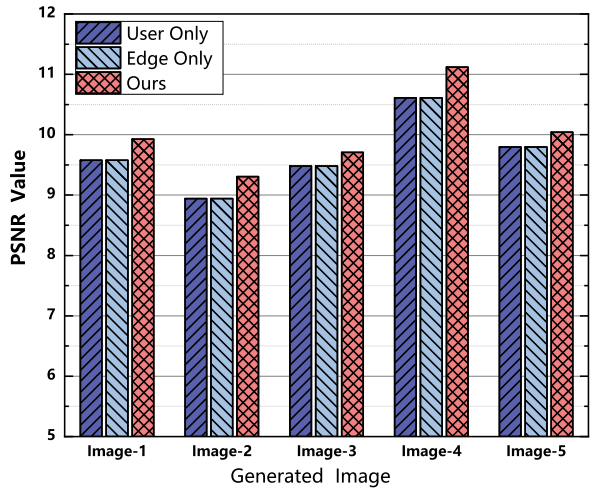
Fig. 6. Visualization of image quality generated by different baselines



(a) MSE



(b) SSI



(c) PSNR

Fig. 7. Visualization of image quality generated by different ablation studies

framework can consistently maintain low prediction errors across different images and scenes, showcasing its strong generalization ability.

*2) **SSI results**:* From Fig. 6(b), we compare our algorithm with three other algorithms: CDAA, Stable Diffusion V1.5, and Stable Diffusion V1.4. For most images, the SSI values of our algorithm are higher than or comparable to those of the other algorithms, further demonstrating the effectiveness and superiority of our method. Notably, for Image 1, Image 4, and Image 5, our algorithm performs similarly to CDAA, while significantly outperforming all other algorithms on Image 3. These results suggest that our framework can consistently maintain high image quality across various images and scenes, demonstrating its strong generalization ability.

*3) **PSNR results**:* Fig. 6(c) presents the experimental results comparing the peak signal-to-noise ratio (PSNR) of different algorithms on five images. From Fig. 6(c), it is evident that, for most images, the PSNR values of our algorithm are either higher than or comparable to those of the other algorithms, further validating the effectiveness and superiority of our method. Notably, for Image 1, Image 2, and Image 5, our algorithm performs similarly to CDAA, while significantly outperforming all other algorithms on Image 3.

The observed results can be attributed to the unique design of our algorithmic framework, which allocates complex inference tasks to edge servers, while user n handles relatively simpler operations. This distributed computing strategy effectively reduces overall computational errors, leading to improved performance in the MSE metric.

C. Performance of Different Ablation Studies

Fig. 5 compares the AIGC generation performance under different configurations (user-only, edge-only, and user-edge collaboration). The results indicate that the user-edge collaborative framework outperforms the single-end configurations regarding color reproduction, detail preservation, and style matching, with the generated images more closely resembling the original. Across various generation tasks, the collaborative framework consistently demonstrates high-quality performance, validating the advantages of this approach in enhancing the quality of diffusion-based AIGC services.

*1) **MSE results**:* Fig. 7(a) shows the comparison results of the mean squared error (MSE) under different deployment strategies in the ablation experiment. The experiment compares the performance of deploying inference on user n (User Only), solely on edge servers (Edge Only), and our distributed inferring framework (Our Work) across four images. In most cases, the MSE values of our framework are lower than those of the two single deployment strategies, further demonstrating the effectiveness and superiority of our approach. Notably, in Image 2, our framework performs similarly to the Edge Only strategy, but in Image 3, it outperforms the other two strategies by a significant margin. These results indicate that our framework maintains low prediction errors across various images and scenes, showcasing its strong generalization ability and efficiency.

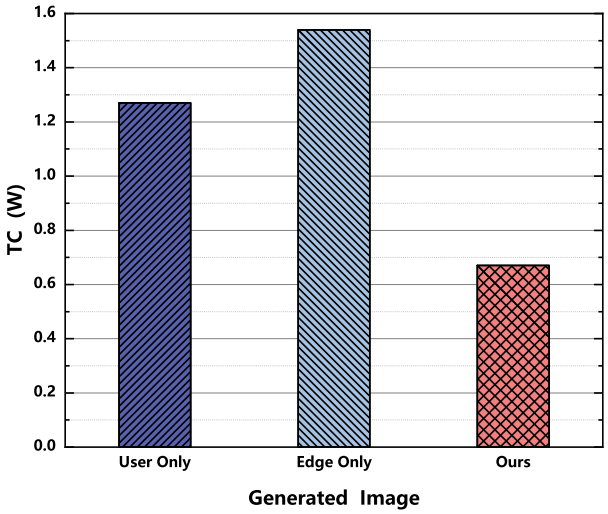
*2) **SSI results**:* Fig. 7(b) presents the comparison results of the Structural Similarity Index (SSI) across different deployment strategies in the ablation experiment. On most images, the SSI values of our algorithm are higher than those of the two single-deployment strategies, further demonstrating the effectiveness and superiority of our method. Notably, in Image 1, our framework performs similarly to the User Only strategy, but in Image 3, it outperforms the other two strategies significantly. These results indicate that our framework can maintain high image quality across diverse images and scenes while demonstrating strong generalization ability and efficiency.

*3) **PSNR results**:* Fig. 7(c) shows the comparison results of the Peak Signal-to-Noise Ratio (PSNR) under different deployment strategies in the ablation experiment. In most cases, the PSNR values of our algorithm are higher than those of the two single-deployment strategies, further demonstrating the effectiveness and superiority of our method. Notably, in Image4, our framework significantly outperforms the other two strategies.

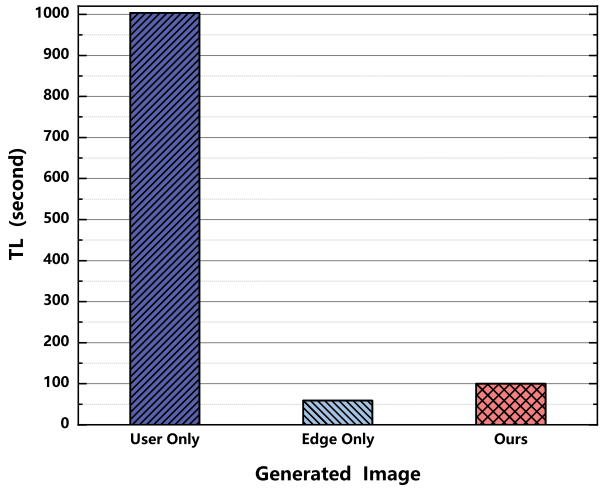
*4) **Service Delay and Device Energy Consumption**:* Since the resolution of the generated image is fixed at 512x512, and the total number of inference steps is also fixed at 110, the time required for image generation is essentially the same across the three methods. Therefore, it suffices to observe the performance of these three methods for the generation of a single image.

As shown in Fig. 8(a), when generating the same image, the energy consumption is highest when the edge server performs the process alone, reaching approximately 1.58W. Our method consumes the least energy, using less than 0.7W. The energy consumption when the user performs the process alone is second, at around 1.3W. Fig. 8(b) illustrates the time required to generate an image using the three methods. It is evident that the time required when the user performs the process alone is the longest, exceeding the time needed by the edge server alone by nearly 20 times, and is 10 times longer than when our method is used.

The observed results can be attributed to the fact that, in general, the computing resources of user n are significantly lower than those of the edge server. In this study, based on the assumed GPU computing power, the computational capacity of edge server b is many times greater than that of user n. This discrepancy causes the image generation time on user n to be much longer than when either of the other two methods is used. While it is theoretically true that the user alone takes much longer to generate the image compared to our method and the edge server alone, the energy consumption of user n is also significantly higher than that of the other two methods. However, due to the substantial energy consumption associated with the high computational capacity of edge server b, its energy consumption can match or even exceed that of user n in a short time. Although our method is not the fastest in terms of time, we have measured the tradeoff between time and energy consumption, achieving the goal of minimizing energy consumption without significantly increasing the task completion time.



(a) Energy Consumption



(b) Service Delay

Fig. 8. Visualization of energy consumption and service delay by different ablation studies

VII. Conclusion

In this study, we proposed a novel edge user collaborative inferring framework named EUCI for optimizing diffusion model-based artificial intelligence-generated content (AIGC) services. Our framework effectively balances the computational load by distributing computational tasks between user devices and edge servers, reduces the need for high communication bandwidth, and maintains image generation quality in a constrained network environment. Although our study mainly focuses on the performance optimization of the framework, we recognize the importance of user privacy protection in AIGC services, which is an issue that cannot be ignored.

In AIGC services, user-generated content and personal data may be processed and stored in edge computing environments. This involves the potential risk of leakage of sensitive information, especially when data is transmitted between user devices, edge servers, and cloud. To enhance the privacy protection capabilities of our framework, future work can explore the integration of advanced privacy protection techniques such as differential privacy and homomorphic encryption to ensure the security and privacy of user data without sacrificing user experience.

In summary, although this paper mainly focuses on improving the performance and efficiency of AIGC services, we also realize the importance of privacy protection and suggest that future research can further explore how to enhance user privacy protection while maintaining service quality. We believe that by combining the latest privacy protection technologies and privacy-aware design, we can build an AIGC service framework that is both efficient and secure, providing users with a better service experience.

References

[1] C. Wang, H. Yu, X. Li, F. Ma, X. Wang, T. Taleb, and V. C. M. Leung, "Dependency-aware microservice deployment for edge computing: A deep reinforcement learning approach with network representation," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 14 737–14 753, 2024.

[2] J. Wang, Y. Sun, and W. T. Ushio, "Mission-aware UAV deployment for post-disaster scenarios: A worst-case SAC-based approach," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 2, pp. 2712–2727, 2024.

[3] Y. Chen, Y. Sun, C. Wang, and T. Taleb, "Dynamic task allocation and service migration in edge-cloud iot system based on deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 16 742–16 757, 2022.

[4] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, "A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt," *arXiv preprint arXiv:2303.04226*, 2023.

[5] Y. Wang, Y. Pan, M. Yan, Z. Su, and T. H. Luan, "A survey on chatgpt: Ai-generated contents, challenges, and solutions," *IEEE Open Journal of the Computer Society*, 2023.

[6] Y. Yi, Z. Zhang, L. T. Yang, X. Deng, L. Yi, and X. Wang, "Social interaction and information diffusion in social internet of things: Dynamics, cloud-edge, traceability," *IEEE Internet of things Journal*, vol. 8, no. 4, pp. 2177–2192, 2020.

[7] Y. Liu, H. Du, D. Niyato, J. Kang, Z. Xiong, C. Miao, A. Jamalipour *et al.*, "Blockchain-empowered lifecycle management for ai-generated content (aigc) products in edge networks," *arXiv preprint arXiv:2303.02836*, 2023.

[8] X. Chen, Z. Guo, X. Wang, H. H. Yang, C. Feng, J. Su, S. Zheng, and T. Q. Quek, "Foundation model based native ai framework in 6g with cloud-edge-end collaboration," *arXiv preprint arXiv:2310.17471*, 2023.

[9] Y. Liang, P. Yang, Y. He, and F. Lyu, "Resource-efficient generative ai model deployment in mobile edge networks," *arXiv preprint arXiv:2409.05303*, 2024.

[10] X. Wang, C. Wang, X. Li, V. C. Leung, and T. Taleb, "Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9441–9455, 2020.

[11] C. Sun, X. Li, C. Wang, Q. He, X. Wang, and V. C. Leung, "Hierarchical deep reinforcement learning for joint service caching and computation offloading in mobile edge-cloud computing," *IEEE Transactions on Services Computing*, 2024.

[12] C. Wang, R. Li, W. Li, C. Qiu, and X. Wang, "Simedgeintel: A open-source simulation platform for resource management in edge intelligence," *Journal of Systems Architecture*, vol. 115, p. 102016, 2021.

[13] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE access*, vol. 8, pp. 85 714–85 728, 2020.

[14] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[15] C. Wang, R. Li, X. Wang, T. Taleb, S. Guo, Y. Sun, and V. C. M. Leung, "Heterogeneous edge caching based on actor-critic learning with attention mechanism aiding," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 6, pp. 3409–3420, 2023.

[16] Y. Wang, C. Liu, and J. Zhao, "Offloading and quality control for ai generated content services in edge computing networks," *arXiv preprint arXiv:2312.06203*, 2023.

[17] S. Wang, Y. Deng, L. Hu, and N. Cao, "Edge-computing-assisted intelligent processing of ai-generated image content," *Journal of Real-Time Image Processing*, vol. 21, no. 2, p. 39, 2024.

[18] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[19] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.

[20] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[21] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[22] C. Zhang, C. Zhang, M. Zhang, and I. S. Kweon, "Text-to-image diffusion models in generative ai: A survey," *arXiv preprint arXiv:2303.07909*, 2023.

[23] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[24] D. Linsley, A. Karkada Ashok, L. N. Govindarajan, R. Liu, and T. Serre, "Stable and expressive recurrent vision models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 10 456–10 467, 2020.

[25] A. Q. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. Mcgrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," in *International Conference on Machine Learning*. PMLR, 2022, pp. 16 784–16 804.

[26] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[27] Y. Tian, Z. Zhang, Y. Yang, Z. Chen, Z. Yang, R. Jin, T. Q. Quek, and K.-K. Wong, "An edge-cloud collaboration framework for generative ai service provision with synergetic big cloud model and small edge models," *arXiv preprint arXiv:2401.01666*, 2024.

[28] C. Yan, S. Liu, H. Liu, X. Peng, X. Wang, F. Chen, L. Fu, and X. Mei, "Hybrid sd: Edge-cloud collaborative inference for stable diffusion models," *arXiv preprint arXiv:2408.06646*, 2024.

[29] Z. Huang, K. C. Chan, Y. Jiang, and Z. Liu, "Collaborative diffusion for multi-modal face generation and editing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6080–6090.

[30] S. Allmendinger, D. Zipperling, L. Struppek, and N. Kühl, "Collafuse: Collaborative diffusion models," *arXiv preprint arXiv:2406.14429*, 2024.

[31] H. Du, Z. Li, D. Niyato, J. Kang, Z. Xiong, D. I. Kim *et al.*, "Enabling ai-generated content (aigc) services in wireless edge networks," *arXiv preprint arXiv:2301.03220*, 2023.

[32] E. Luhman and T. Luhman, "Knowledge distillation in iterative generative models for improved sampling speed," *arXiv preprint arXiv:2101.02388*, 2021.

[33] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," *arXiv preprint arXiv:2202.00512*, 2022.

[34] A. Sauer, D. Lorenz, A. Blattmann, and R. Rombach, "Adversarial diffusion distillation," in *European Conference on Computer Vision*. Springer, 2025, pp. 87–103.

[35] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[36] M. Xu, H. Du, D. Niyato, J. Kang, Z. Xiong, S. Mao, Z. Han, A. Jamalipour, D. I. Kim, X. Shen *et al.*, "Unleashing the power of edge-cloud generative ai in mobile networks: A survey of aigc services," *IEEE Communications Surveys & Tutorials*, 2024.

[37] Z. J. Wang, E. Montoya, D. Munechika, H. Yang, B. Hoover, and D. H. Chau, "DiffusionDB: A large-scale prompt gallery dataset for text-to-image generative models," *arXiv:2210.14896 [cs]*, 2022. [Online]. Available: https://arxiv.org/abs/2210.14896

[38] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 278–25 294, 2022.

[39] C. Dewi, R.-C. Chen, Y.-T. Liu, and H. Yu, "Various generative adversarial networks model for synthetic prohibitory sign image generation," *Applied Sciences*, vol. 11, no. 7, p. 2913, 2021.

[40] M. Elasri, O. Elharrouss, S. Al-Maadeed, and H. Tairi, "Image generation: A review," *Neural Processing Letters*, vol. 54, no. 5, pp. 4609–4646, 2022.

[41] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 2366–2369.

Weijia Feng is a Vice Professor in the College of Computer and Information Engineering, at Tianjin Normal University. He received a Ph.D. from Tianjin University, China 2012, and was a Joint Training Ph.D student with the University of Oulu from 2010 to 2012. He has presided over several projects, including the National Natural Science Foundation of China, subtopics in National Key R&D Program Projects, and many Enterprise R&D Projects. He currently holds the position of Secretary General of the Tianjin Association for Artificial Intelligence. He is focusing on the research of Edge Computing, Edge AI, the Internet of Things, and Machine Vision.

Ruojia Zhang obtained his Bachelor's degree in Electrical Electronics and Computer Science from Guangxi University of Science and Technology in 2022 and is currently enrolled in the Software Engineering master's program at Tianjin Normal University, class of 2023. His current research areas are knowledge distillation, edge computing, and edge caching.

Yichen Zhu obtained his Bachelor's degree in Computer Science and Technology from Zhengzhou Technology and Business University in 2022 and is currently enrolled in the Software Engineering master's program at Tianjin Normal University, class of 2023. His research interests include edge computing and edge caching.

Chenyang Wang (Member, IEEE) received the B.S. and M.S. degrees in computer science and technology from Henan Normal University, Xinxiang, China, in 2013 and 2017, respectively. He got a PhD degree from the College of Intelligence and Computing, Tianjin University, Tianjin, China, in 2023. He has also been a visiting PhD student under the support of the China Scholarship Council (CSC) at the School of Electrical Engineering, Aalto University, Espoo, Finland in 2021. He is currently a postdoctoral researcher at the College of Computer Science and Software Engineering, at Shenzhen University. His current research interests include edge computing, big data analytics, reinforcement learning, and deep learning. He received the Best Student Paper Award of the 24th International Conference on Parallel and Distributed Systems from the IEEE Computer Society in 2018. He also received the Best Paper Award from the IEEE International Conference on Communications in 2021. In 2022, he received the "IEEE ComSoc Asia-Pacific Outstanding Paper Award".

Chuan Sun (Member, IEEE) received the B.S. degree from Wuhan University of Science and Technology, Wuhan, China, in 2017, his Ph.D. degree from the School of Big Data and Software Engineering, Chongqing University, Chongqing, China, in 2023. He is currently a research fellow in the College of Computing and Data Science, Nanyang Technological University, 50 Nanyang Avenue, Singapore. His current research interests include efficient LLMs, federated learning, and mobile edge computing. He has published more than 20 journal/conference papers in IEEE JSAC, TSC, TNSE, IEEE Network, and so on.

Xiaoqiang Zhu (M'23) received the Ph.D. degree in software engineering from Tianjin University, China, in 2022, and the M.S. degree in computer science from Dalian University of Technology, China, in 2018. He served as a joint Ph.D. student at ETH Zurich, Switzerland, supported by the China Scholarship Council in 2021. He is currently an Assistant Professor (Lecturer) at the School of Software Engineering, Beijing Jiaotong University, China. He has published scientific papers in international journals, such as IEEE COMST, TMC, TNSE, etc., and served as session chair for IEEE SmartIoT and PC member for IEEE CSCWD. He is also the reviewer of distinguished journals, including IEEE/ACM ToN, IEEE TMC, TWC, TNSE, IoT-J, etc. His research interests include the Internet of Things, machine learning, and privacy protection.

Xiang Li received the B.S and M.S. degrees in Computer Science and Technology by Tianjin Normal University, China, in 2007 and 2010 respectively. He has long been dedicated to the research of computer networks and communications. At present, his research interests encompass digital government governance and technical responses.

Tarik Taleb (Senior Member, IEEE) received the B.E. degree (with distinction) in information engineering and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively. He is currently a Professor at the Center of Wireless Communications, at the University of Oulu, Finland. He is the founder and the Director of the MOSA!C Lab, Espoo, Finland. He was an Assistant Professor with the Graduate School of Information Sciences, Tohoku University, in a laboratory fully funded by KDDI until 2009. He was a Senior Researcher and a 3GPP Standards Expert with NEC Europe Ltd., Heidelberg, Germany. He was then leading the NEC Europe Labs Team, involved with research and development projects on carrier cloud platforms, an important vision of 4G systems. From 2005 to 2006, he was a Research Fellow with the Intelligent Cosmos Research Institute, Sendai. He has also been directly engaged in the development and standardization of the Evolved Packet System as a member of the 3GPP System Architecture Working Group. His current research interests include architectural enhancements to mobile core networks (particularly 3GPP's), network softwarization and slicing, mobile cloud networking, network function virtualization, software-defined networking, mobile multimedia streaming, intervehicular communications, and social media networking.