This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/LNET.2025.3599196, IEEE Networking Letters

Federated Learning Assisted Edge Caching Scheme Based on Lightweight Architecture DDPM

Xun Li, Qiong Wu, *Senior Member, IEEE*, Pingyi Fan, *Senior Member, IEEE*,
Kezhi Wang, *Senior Member, IEEE*, Nan Cheng, *Senior Member, IEEE*, and Khaled B. Letaief, *Fellow, IEEE*

***Abstract*—Edge caching is an emerging technology that empowers caching units at edge nodes, allowing users to fetch contents of interest that have been pre-cached at the edge nodes. The key to pre-caching is to maximize the cache hit percentage for cached content without compromising users' privacy. In this letter, we propose a federated learning (FL) assisted edge caching scheme based on lightweight architecture denoising diffusion probabilistic model (LDPM). Our simulation results verify that our proposed scheme achieves a higher cache hit percentage compared to existing FL-based methods and baseline methods.**

***Index Terms*—Federated learning, denoising diffusion probabilistic model, edge caching.**

I. Introduction

A. Background

In recent years, the surge of smart devices has led to a significant increase in mobile data traffic, putting enormous pressure on wireless networks. As users become increasingly reliant on user devices such as smartphones and computers to access content, ensuring satisfactory service quality has become challenging [1]. To address this challenge, edge caching has become an effective solution. By caching user interested content in advance at wireless network edge nodes such as base stations (BS), user can directly obtain requested content from nearby BS instead of remote cloud server. This method can significantly alleviate network congestion, reduce traffic load, reduce service latency, and improve overall system performance [2].

This work was supported in part by Jiangxi Province Science and Technology Development Programme under Grant 20242BCC32016; in part by the National Natural Science Foundation of China under Grant 61701197; in part by the National Key Research and Development Program of China under Grant 2021YFA1000500(4); in part by the Research Grants Council under the Areas of Excellence Scheme under Grant AoE/E-601/22-R; and in part by the 111 Project under Grant B23008. (Corresponding author: Qiong Wu.)

Xun Li and Qiong Wu are with the School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China, and also with the School of Information Engineering, Jiangxi Provincial Key Laboratory of Advanced Signal Processing and Intelligent Communications, Nanchang University, Nanchang 330031, China (e-mail: xunli@stu.jiangnan.edu.cn; qiongwu@jiangnan.edu.cn).

Pingyi Fan is with the Department of Electronic Engineering, State Key Laboratory of Space Network and Communications, and the Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: fpy@tsinghua.edu.cn).

Kezhi Wang is with the Department of Computer Science, Brunel University, London, Middlesex UB8 3PH, U.K (e-mail: Kezhi.Wang@brunel.ac.uk).

Nan Cheng is with the State Key Laboratory of ISN and School of Telecommunications Engineering, Xidian University, Xi'an 710071, China (e-mail: dr.nan.cheng@ieee.org).

Khaled B. Letaief is with the Department of Electrical and Computer Engineering, the Hong Kong University of Science and Technology, Hong Kong (e-mail: eekhaled@ust.hk).

However, BSs have limited cache capacity, which makes it crucial to efficiently cache content that users are interested in [3]. Denoising diffusion probabilistic models (DDPM) have garnered significant attention for their superior generative capabilities, which generate samples through a step-by-step denoising process [4]. Compared to earlier mainstream generative models (e.g., generative adversarial networks (GANs) and Flow-based models), DDPM exhibits more stable training processes and higher sample fidelity. However, the training of DDPM requires significant computational resources and is not suitable for resource-constrained devices. Recently, several lightweight architecture DDPM (LDPM) have been proposed to address this challenge. In [5], Li *et al.* achieved sub-second text-to-image generation on mobile devices for the first time through the design of an efficient U-Net architecture and improved step distillation techniques. In [6], Chen *et al.* proposed a LDPM suitable for edge devices through a lightweight U-Net architecture design, which only requires 4 denoising steps to generate high-quality speech.

Additionally, model training requires access to users' personal data. User personal data often contains a lot of privacy sensitive information, and users are unwilling to directly share their data with others, making it difficult to collect and train directly on user data [7]. Fortunately, federated learning (FL) can address this issue by enabling the sharing of local models instead of raw user data [8]. Therefore, it is necessary to introduce FL into edge caching in order to protect user privacy.

B. Related Work

Currently, there are many studies on edge caching. In [9], Meybodi *et al.* proposed a multi-model Transformer framework with parallel regression and classification branches that enables regression-based prediction of future content request probabilities. In [10], Yu *et al.* conducted edge caching for vehicle environments, integrating mobility prediction with FL to address the issues of high vehicle dynamics and user privacy. However, the aforementioned papers cannot accurately predict the content that users are interested in. The authors in [7], [8] adopted a "K-nearest neighbor selection mechanism," calculating similarity based on the interests between users, and using the interest lists of the K most similar "neighbor" users to the target user as auxiliary predictions of the target user's interests. Although it can predict the content of user interest more accurately, predicting the content of interest to the target user requires the use of the "neighbor" user's interest list, which to some extent exposes the privacy of the "neighbor" user. In [11], Wang *et al.* integrated Wasserstein GAN

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/LNET.2025.3599196, IEEE Networking Letters

(WGANs) into FL frameworks, enabling accurate prediction without exposing raw user data. However, the training of GAN models demands substantial computational resources, rendering them impractical for deployment on resource-constrained user edge devices.

C. Contributions

To address the above issues, we propose a FL assisted edge caching scheme based on LDPM [1], which achieves a high cache hit percentage without compromising user privacy. The main contributions of our work are as follows:

- We are the first to combine LDPM with FL for edge caching. Compared to previous FL schemes, our approach achieves higher cache hit percentage without compromising user privacy and is suitable for edge devices.
- To effectively learn the distribution of high-dimensional sparse user data, we use the pre-trained encoder to map the raw user data into a low-dimensional latent space, allowing LDPM to learn the user data distribution in this low-dimensional space.
- To accurately predict the content of interest to users while protecting their privacy, we first propose a federated-based LDPM training algorithm, and then propose a content popularity prediction method that generates data samples using the global LDPM at the BS to predict the content of interest to users.

II. System Model

A. System Scenario

The system scenario is shown in Fig. 1, where the edge computing network includes a BS, a remote cloud server, and I users. The BS and the remote cloud server are connected via a reliable backhaul link, and users are within the coverage range of the BS. Each user $i = 1, 2, \ldots, I$ has one smart device. The BS is equipped with a caching entity and FL scheduling module. The caching entity has a limited storage capacity and can accommodate up to N contents, while the remote cloud server caches all available content. The FL scheduling module is used to connect different devices for federated training and predict popular content. When the content requested by a user is cached in the BS, the BS will directly deliver the content to the user. Otherwise, the BS requests the content from the remote cloud server and then delivers it to the user, which results in higher request content delay. Our goal is to maximize the cache hit percentage of user requests by accurately predicting content popularity and proactively caching at the BS.

B. Denoising Diffusion Probabilistic Model (DDPM)

The theoretical foundation of diffusion models stems from the entropy increase-inverse process of non-equilibrium thermodynamic systems. DDPM achieves forward diffusion process and reverse diffusion process through parameterized Markov chains.

[1]The source code has been released at: https://github.com/qiongwu86/Federated-Learning-Assisted-Edge-Caching-Scheme-Based-on-Lightweight-Architecture-DDPM

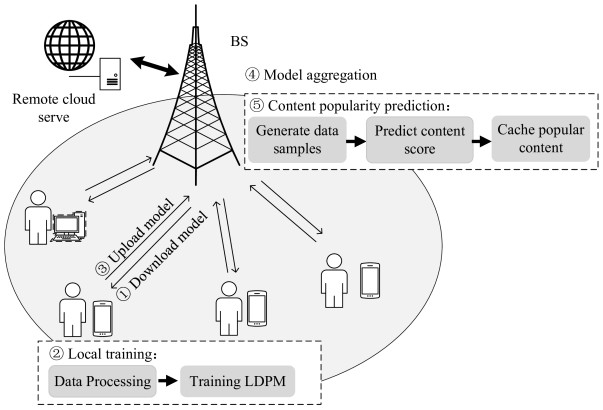

Fig. 1. System Model.

1) Forward Diffusion Process: By parameterizing a Markov chain with a scheduling strategy $\{\beta_t\}_{t=1}^{T}$, Gaussian noise is progressively added to the original data, causing the data distribution to gradually perturb towards random noise, where T is the time steps. A single diffusion step can be described as

$$q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}\right). \tag{1}$$

In this work, we employ a varying scheduling strategy, where $\{\beta_t\}_{t=1}^{T}$ increases linearly from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. Similar to [12], we define $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ and $\alpha_t = 1 - \beta_t$, and can obtain $q\left(\mathbf{x}_t|\mathbf{x}_0\right)$ and $\mathbf{x}_t$ as

$$q\left(\mathbf{x}_t|\mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right), \tag{2}$$

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, \boldsymbol{\epsilon}_t \sim \mathcal{N}\left(0, \mathbf{I}\right). \tag{3}$$

2) Reverse Diffusion Process: By training a neural network $\boldsymbol{\mu}_\theta$ to predict the noise ϵ_θ at each step, the goal is to recover the original data distribution from the noisy data. The reverse step is parameterized by conditional probability $p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)$, defined as

$$p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right) = \mathcal{N}\Big(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\Big), \tag{4}$$

$$\boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\Big). \tag{5}$$

In [4], Ho *et al.* proposed a simplified objective function for optimization, expressed as

$$\mathcal{L}_{t-1}^{simple} = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2\right]. \tag{6}$$

III. Edge Caching Scheme

This section introduces the proposed edge caching scheme. We first introduce the federated-based LDPM training algorithm, and then introduce the content popularity prediction algorithm.

User data is typically high-dimensional and sparse, causing the Euclidean distances between data points to become uniform, and the noise distribution to become extremely flat [8]. This makes it difficult for the model to distinguish between signal and noise, and the DDPM fails to effectively learn

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/LNET.2025.3599196, IEEE Networking Letters

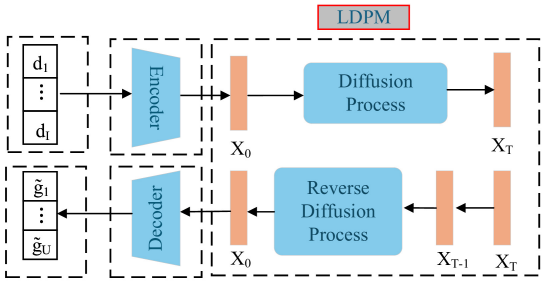

Fig. 2. Encoder, Decoder and LDPM framework.

the distribution of user data [13]. Therefore, we employ pre-trained encoder and decoder to process the data. Before performing local model training, we use the pre-trained encoder to map the raw user data into a low-dimensional latent space, allowing LDPM to learn the user data distribution in this low-dimensional space. Subsequently, when predicting content popularity at the BS, the pre-trained decoder is used to reconstruct the LDPM output data samples back to the original space dimensions. The pre-trained encoder and decoder are trained on the BS using publicly available datasets, and then fine-tuned on the edge using local data [14]. Fig. 2 illustrates the framework of the encoder, decoder, and LDPM. Furthermore, our strategy also supports dynamic user scenarios. After a user leaves, the BS will stop receiving model updates from the user, and a user entering the BS coverage area will be added to the training in a new round.

A. Federated-Based LDPM Training Algorithm

During the training process of FL, a total of $R_{\max}$ rounds of training are conducted. Each round of training $r = 1, 2, \ldots, R_{\max}$ consists of following four steps, corresponding to steps $1-4$ in Fig. 1.

1) Download Model: The BS first generates the global LDPM in this step. Let ω^r represent the global LDPM parameters for the r-th round. For the first round of training, the BS initializes global LDPM parameters ω^0. For subsequent rounds, the BS will update the global model at the end of the previous round. Then the BS distributes the global LDPM to users for training.

2) Local Training: The local training process includes data processing and training LDPM. Data processing is mainly used to map the raw user data into a low-dimensional latent space, and then let LDPM learn the distribution of user data in the low-dimensional latent space.

For iteration k, user i first performs data processing using a pre-trained encoder to map the raw user local data d_i into a low-dimensional latent space $\hat{d}_i = E(d_i)$, where $E(\cdot)$ represents the encoder parameters.

After completing data processing, let LDPM learn the distribution of user local data in the low-dimensional latent space. For iteration k, user i randomly samples a subset $\hat{b}^r_{i,k}$ from $\hat{d}_i$. Then, the local loss function for the LDPM can be described as

$$f(\omega^r_{i,k}) = \frac{1}{\left|\hat{b}^r_{i,k}\right|} \sum_{\hat{z} \in \hat{b}^r_{i,k}} L\left(\omega^r_{i,k}; \hat{z}\right), \tag{7}$$

where $\left|\hat{b}^r_{i,k}\right|$ is the size of the subset $\hat{b}^r_{i,k}$, $L(\cdot)$ is defined in Equation (6), $\hat{z}$ is a data point in $\hat{b}^r_{i,k}$, and $\omega^r_{i,k}$ refers to the LDPM parameters of user i at the r-th round in the k-th iteration. Then, the local LDPM is updated as

$$\omega^r_{i,k+1} = \omega^r_{i,k} - \eta_d \nabla f(\omega^r_{i,k}), \tag{8}$$

where η_d is the LDPM learning rate. After completing e iterations of local LDPM training, the local training process is complete.

3) Upload Model: Each user will upload the locally updated ω^r_i to the BS after completing the local training process.

4) Model Aggregation: After receiving all the local models uploaded by users, BS calculates the weighted sum of models for all users within the coverage area to obtain a new global model,

$$\omega^{r+1} = \omega^r - \eta \sum_{i=1}^{I} \frac{|d_i|}{d} \omega^r_i, \tag{9}$$

where $|d_i|$ is the size of the local data for user i, and d is the size of the total data for all users within the BS coverage. So far, the training of federated-based LDPM for the r-th round has been finished, and the BS has acquired a new global model ω^{r+1}. This model will be utilized for the next round of training. Once the number of training rounds reaches $R_{\max}$, the entire training process concludes.

B. Content Popularity Prediction

After completing the training process, the BS uses global LDPM to perform the reverse diffusion process, generating U data samples g_u, where U is the number of data samples and $u = 1, 2, \ldots, U$. These data samples are fed into the pre-trained decoder to produce reconstructed data samples $\tilde{g}_u = D(g_u)$ in the original data dimensions, where $D(\cdot)$ represents the dncoder parameters. We use these reconstructed data samples for content popularity prediction in the BS. Assuming the content library contains F items, the dimension of $\tilde{g}_u$ is F and can be expressed as $\tilde{g}_u(1, 2, ..., F)$. All reconstructed fake samples can be added by dimension to obtain the score $\tilde{g}(1, 2, ..., F)$ of all contents,

$$\tilde{g}(1, 2, ..., F) = \frac{1}{U} \sum_{u=1}^{U} \tilde{g}_u(1, 2, ..., F). \tag{10}$$

The score $\tilde{g}(1, 2, ..., F)$ reflects the overall preferences of users within the BS coverage area, which does not expose the privacy of individual users. The higher the score, the more popular the content is. Then, considering the cache capacity of the BS, cache the N most popular contents. The above process corresponds to step 5 in Fig. 1.

IV. Simulation

In this section, we conducted experiments on the widely used MovieLens 1M dataset. The MovieLens 1M dataset includes 1,002,099 ratings from 6,040 users on 3,952 movies, with each rating ranging from 0 to 5. The values of the parameters in the experiment are shown in Table I. Unless otherwise specified, the number of users participating in the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/LNET.2025.3599196, IEEE Networking Letters

training is 20, the time steps T is 50. and the BS cache capacity is 100 contents.

Our overall U-Net architecture is similar to [15]. The main differences are the replacement of 2D convolutions with 1D convolutions to adapt to the user's interaction data structure, and the use of one-fourth of the number of channels and three feature map resolutions to reduce the model size [16], [17]. Therefore, our LDPM has only 770K parameters, making it suitable for resource-constrained edge devices. To evaluate the scheme, we adopt the cache hit percentage and request content delay as evaluation metrics [18]. The cache hit percentage represents the success rate of directly requesting content from the BS. The more accurate the predicted popular content, the higher the cache hit percentage. When the requested content is stored in the BS, it is regarded as a successful cache; conversely, if the content is not cached in the BS, it is termed a failed cache. The request content delay represents the average delay of all users getting content.

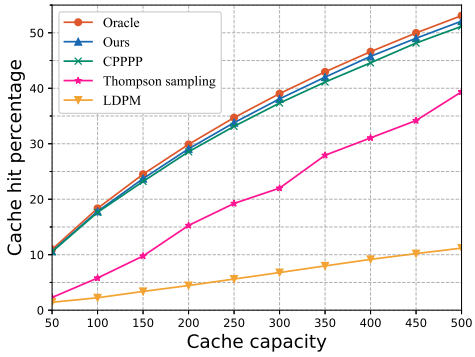

Fig. 3. Cache hit percentage versus cache capacity.

We compare our proposed edge caching scheme with other schemes, such as:

- Oracle [18]: The Oracle algorithm possesses full prior knowledge of users future requests, defining the theoretical maximum achievable cache hit percentage.
- CPPPP [11]: Claims to be the first to use FL with GAN to predict popular content.
- Thompson Sampling: In each iteration, the BS dynamically updates its cached contents by evaluating historical cache success/failure statistics and retains the top N highest-value items through Bayesian posterior probability updates.
- LDPM: Directly using raw user data for LDPM training without using the pre-trained encoder for data processing.

Fig. 3 illustrates the cache hit percentage of BS under different caching capacity across various schemes. It can be observed that as the caching capacity increases, the cache hit percentage improves for all schemes. This is because larger caching capacity enable the BS to store more content, making it more likely for users to retrieve the requested content from the BS. Oracle has the highest cache hit percentage because it knows the content of user requests in the future. Our proposed scheme outperform CPPPP because LDPM leverage a step-by-step denoising generation process, a stable training

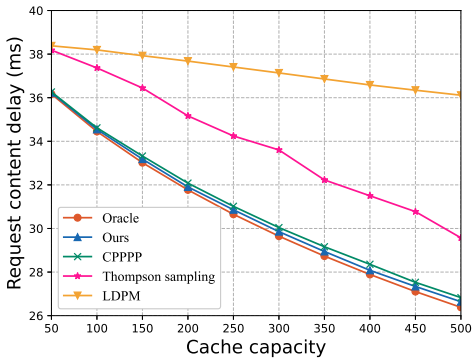

Fig. 4. Request content delay versus cache capacity.

TABLE I. Values of the parameters in the experiments.

Parameter	Value
η_d	0.0006
U	1000
F	3952
Structure of pre-trained encoder	3952-100-16
Structure of pre-trained decoder	16-100-3952

objective, and a more comprehensive ability to approximate data distributions, effectively overcoming the limitations of GAN in terms of training instability and mode collapse. The performance of CPPPP is superior to Thompson Sampling because Thompson Sampling does not rely on learning-based content prediction. LDPM has the worst performance because it is difficult for LDPM to learn an effective data distribution directly on the original high-dimensional sparse user data.

Fig. 4 shows the request content delay of BS under different caching capacity for various schemes. It can be observed that as the caching capacity increases, the request content delay decreases across all schemes. This is because a larger caching capacity allows the BS to store more content, increasing the likelihood that each user can obtain the desired content directly from the BS, thereby reducing the request delay. Furthermore, the request delay of our proposed scheme is lower than that of other schemes except for Oracle. This is attributed to the higher cache hit percentage of our proposed scheme, which enables more users to retrieve content from the BS, further minimizing the request delay.

From Table II, it can be observed that as the time steps T increases from 10 to 50, the cache hit percentage for different cache capacities show a significant improvement, while the total training time of the model and CPU cycles increase slightly. This is because the increase in the number of time steps T allows the model to learn more refined noise, thus producing better performance. Further increasing time steps T results in almost no change in the cache hit percentage for different cache capacities, but the total training time of the model and CPU cycles increase significantly. This is because the model has already achieved near-optimal performance, and further increasing the number of time steps T cannot significantly improve performance but will significantly increase the model training time and CPU cycles. Therefore, we choose $T = 50$.

Fig. 5 shows how cache hit percentage and request content delay vary with the number of user participating in training.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/LNET.2025.3599196, IEEE Networking Letters

TABLE II. Cache efficiency, total training time and CPU cycles under different time steps T.

T	Cache capacity										Time (s)	CPU cycles
	50	100	150	200	250	300	350	400	450	500		
10	10.18%	17.38%	22.85%	28.22%	32.94%	37.08%	40.81%	44.55%	47.80%	51.01%	18.46	57.05G
50	10.79%	17.67%	23.67%	29.06%	33.83%	38.23%	42.30%	45.75%	48.98%	52.20%	21.56	66.09G
100	10.83%	17.95%	24.02%	29.21%	34.09%	38.20%	42.13%	45.68%	49.20%	52.25%	33.88	103.97G
200	10.93%	18.00%	24.09%	29.37%	34.09%	38.34%	42.31%	45.92%	49.17%	52.18%	40.68	122.64G
500	10.87%	18.09%	24.20%	29.62%	34.31%	38.53%	42.44%	46.18%	49.38%	52.39%	92.10	283.03G

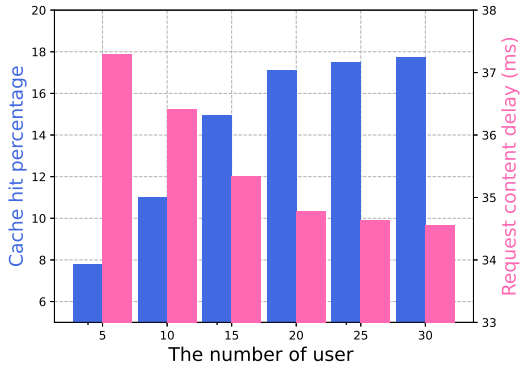

Fig. 5. Cache hit percentage and request content delay versus the number of user.

It can be seen that as the number of user participating in training increases, the cache hit percentage gradually increases and the request content delay gradually decreases. This is because more users provide more data and computational power, which allows for more accurate prediction of popular content. Furthermore, it can be observed that after the number of users reaches 20, further increasing the number of users leads to only a slight improvement in performance, while the communication overhead of FL will keep accumulating. Therefore, 20 users are selected to participate in the experiment for a better tradeoff between the scaling of user number and the complexity of communication overheads.

V. Conclusion

In this letter, we propose a FL assisted edge caching scheme based on LDPM, achieving a higher cache hit percentage compared to existing FL-based methods and baseline methods. To protect user privacy, we first propose a federated-based LDPM training algorithm. Afterwards, we propose an algorithm for predicting popular content on edge nodes. Finally, experiments are conducted to verify the scheme we proposed. Our current work only considers content caching for a single BS. We are considering introducing multi BSs collaborative caching in future work to further improve edge caching efficiency.

References

[1] Z. Shao, Q. Wu, P. Fan, N. Cheng, W. Chen, J. Wang, and K. B. Letaief, "Semantic-aware spectrum sharing in internet of vehicles based on deep reinforcement learning," *IEEE Internet of Things J.*, vol. 11, no. 23, pp. 38521–38536, 2024, doi: 10.1109/JIOT.2024.3448538.

[2] Q. Wu and J. Zheng, "Performance modeling of IEEE 802.11 DCF based fair channel access for vehicular-to-roadside communication in a non-saturated state," in *2014 IEEE Int. Conf. Commun. (ICC)*, 2014, pp. 2575–2580, doi: 10.1109/ICC.2014.6883711.

[3] A. B. Rahman, P. Charatsaris, E. E. Tsiropoulou, and S. Papavassiliou, "Information-centric networking cache memory allocation: A network economics approach," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, 2023, pp. 1259–1264. doi: 10.1109/GLOBECOM54140.2023.10437315.

[4] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 6840–6851.

[5] Y. Li, H. Wang, Q. Jin, J. Hu, P. Chemerys, Y. Fu, Y. Wang, S. Tulyakov, and J. Ren, "Snapfusion: Text-to-image diffusion model on mobile devices within two seconds," *NeurIPS* , 2024.

[6] J. Chen, X. Song, Z. Peng, B. Zhang, F. Pan, and Z. Wu, "LightGrad: Lightweight Diffusion Probabilistic Model for Text-to-Speech," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* , Rhodes Island, Greece: IEEE, Jun. 2023, pp. 1–5. doi: 10.1109/ICASSP49357.2023.10096710.

[7] Q. Wu, W. Wang, P. Fan, Q. Fan, H. Zhu, and K. B. Letaief, "Cooperative Edge Caching Based on Elastic Federated and Multi-Agent Deep Reinforcement Learning in Next-Generation Networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 21, no. 4, pp. 4179–4196, Aug. 2024.

[8] Q. Wu, Y. Zhao, Q. Fan, P. Fan, J. Wang, and C. Zhang, "Mobility-Aware Cooperative Caching in Vehicular Edge Computing Based on Asynchronous Federated and Deep Reinforcement Learning," *IEEE J. Sel. Top. Signal Process.*, vol. 17, no. 1, pp. 66–81, Jan. 2023, doi: 10.1109/JSTSP.2022.3221271.

[9] Z. H. Meybodi, A. Mohammadi, M. Hou, E. Rahimian, S. Heidarian, J. Abouei, and K. N. Plataniotis, "Multi-content time-series popularity prediction with multiple-model transformers in MEC networks," *Ad Hoc Networks*, vol. 157, p. 103436, 2024.

[10] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-Aware Proactive Edge Caching for Connected Vehicles Using Federated Learning," *IEEE Trans. Intell. Transport. Syst.*, vol. 22, no. 8, pp. 5341–5351, Aug. 2021, doi: 10.1109/TITS.2020.3017474.

[11] K. Wang, N. Deng, and X. Li, "An Efficient Content Popularity Prediction of Privacy Preserving Based on Federated Learning and Wasserstein GAN," *IEEE Internet Things J.*, vol. 10, no. 5, pp. 3786–3798, Mar. 2023, doi: 10.1109/JIOT.2022.3176360.

[12] X. Wang, K. Tao, N. Cheng, Z. Yin, Z. Li, Y. Zhang and X. Shen, "RadioDiff: An Effective Generative Diffusion Model for Sampling-Free Dynamic Radio Map Construction," *IEEE Trans. Cogn. Commun. Netw.*, pp. 1–1, 2025, doi: 10.1109/TCCN.2024.3504489.

[13] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 10674–10685. doi: 10.1109/CVPR52688.2022.01042.

[14] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, 2015, pp. 234–241.

[16] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail A. Kudinov, "Grad-tts: A diffusion probabilistic model for text-to-speech," in *ICML*, 2021, vol. 139, pp. 8599–8608.

[17] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "DiffWave: Aversatile diffusion model for audio synthesis," in *Proc. Int. Conf. Learn. Representations*, 2021.

[18] S. Muller, O. Atan, M. V. D. Schaar, and A. Klein, "Context-Aware Proactive Content Caching With Service Differentiation in Wireless Networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017, doi: 10.1109/TWC.2016.2636139.