Joint Foundation Model Caching and Inference of Generative AI Services for Edge Intelligence

Minrui Xu[1], Dusit Niyato[1], Hongliang Zhang[2], Jiawen Kang[3], Zehui Xiong[4], Shiwen Mao[5], and Zhu Han[6,7]
[1]Nanyang Technological University, [2]Peking University, [3]Guangdong University of Technology,
[4]Singapore University of Technology and Design, [5]Auburn University,
[6]University of Houston, [7]Kyung Hee University
Email: minrui001@e.ntu.edu.sg, dniyato@ntu.edu.sg, hongliang.zhang@pku.edu.cn, kavinkang@gdut.edu.cn,
zehui_xiong@sutd.edu.sg, smao@ieee.org, hanzhu22@gmail.com.

GLOBECOM 2023 - 2023 IEEE Global Communications Conference | 979-8-3503-1090-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/GLOBECOM54140.2023.10436771

***Abstract*—With the rapid development of artificial general intelligence (AGI), various multimedia services based on pretrained foundation models (PFMs) need to be effectively deployed. With edge servers that have cloud-level computing power, edge intelligence can extend the capabilities of AGI to mobile edge networks. However, compared with cloud data centers, resource-limited edge servers can only cache and execute a small number of PFMs, which typically consist of billions of parameters and require intensive computing power and GPU memory during inference. To address this challenge, in this paper, we propose a joint foundation model caching and inference framework that aims to balance the tradeoff among inference latency, accuracy, and resource consumption by managing cached PFMs and user requests efficiently during the provisioning of generative AI services. Specifically, considering the in-context learning ability of PFMs, a new metric named the *Age of Context (AoC)*, is proposed to model the freshness and relevance between examples in past demonstrations and current service requests. Based on the AoC, we propose a least context caching algorithm to manage cached PFMs at edge servers with historical prompts and inference results. The numerical results demonstrate that the proposed algorithm can reduce system costs compared with existing baselines by effectively utilizing contextual information.**

***Index Terms*—Mobile edge computing, generative artificial intelligence, pretrained foundation models, joint foundation model caching and inference**

I. Introduction

Moving towards Artificial General Intelligence (AGI) in mobile edge networks [1], [2], pre-trained foundation models (PFMs), such as generative pre-trained transformers (GPTs) [3], have achieved great successes in a variety of fields over the past few years. As building blocks of AGI, PFMs with billions of parameters are essential due to their effectiveness in demonstrating emergent capabilities in downstream tasks with various data modalities [4]. The pre-training approach provides an efficient parameter initialization for a wide range of downstream tasks, including semantic segmentation, content generation, and information retrieval. As a result, language/visual/multimodal foundation models belong to the paradigm of transfer learning, which can adapt to new tasks and domains without any task-specific data during pre-training.

Multimedia services based on edge intelligence, such as intelligent digital twins (DTs), autonomous driving, and AI-generated content (AIGC), can be greatly enhanced by deploying PFMs on edge servers, benefiting from edge computing's low latency and flexible features. For instance, in autonomous driving, PFMs can generate traffic simulations and provide driving assistance in making complex driving decisions [5]. Additionally, during immersive human-avatar interactions in the Metaverse, PFMs can assist in comprehending and reacting to human emotions and behaviors. For example, ChatGPT facilitates consistent and fluent interactions with humans, fine-tuned based on GPT-3 to release its contextual awareness [3], which is an LFM with 175 billion parameters. Beyond executing PFMs in cloud data centers, edge servers can support fine-tuning and inference processes of PFMs requested by AI services, thus igniting the sparks of AGI in mobile edge networks.

However, unlike cloud data centers, resource-constrained edge servers are unable to concurrently load all PFMs to serve users' AI service requests [6], [7]. In literature, existing research generally focuses on offloading AI services to cloud data centers for remote execution or caching inference results at edge servers for low-latency response [8]. On one hand, offloading inference requests of PFMs to cloud data centers introduce additional latency, traffic overhead, and privacy threats to serving AI services over core networks and public cloud infrastructure. On the other hand, merely caching inference results at edge servers is no longer effective for satisfying users' interactive requirements. To enable mobile AI services with the computing and GPU resources currently loaded into the GPUs of edge servers, effective deployment of PFMs at

This research is supported by the National Key RD Project of China under Grant No. 2020YFB1807100, the NSFC underGrant 62102099, in part by the Pearl River Talent Recruitment Program underGrant 20210N02S643, in part by the Guangzhou Basic Research Program underGrant 2023A04J1699, in part by the NSFC under Grants U22A2054 and 62101594, the National Research Foundation, Singapore, and Infocomm Media Development Authority under its Future Communications Research Development Programme, DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-019), Energy Research Test-Bed and Industry Partnership Funding Initiative, Energy Grid (EG) 2.0 programme, DesCartes and the Campus for Research Excellence and Technological Enterprise (CREATE) programme, MOE Tier 1 (RG87/22), the SUTD SRG-ISTD-2021-165, the SUTD-ZJU IDEA Grant (SUTD-ZJU (VP) 202102), the Ministry of Education, Singapore, under its SUTD Kickstarter Initiative (SKI 20210204), NSF CNS-2148382, CNS-2107216, CNS-2128368, CMMI-2222810, ECCS-2302469, US Department of Transportation, Toyota and Amazon.

Authorized licensed use limited to: Auburn University. Downloaded on January 13,2025 at 18:15:50 UTC from IEEE Xplore. Restrictions apply.

edge servers requires flexible and context-aware management on computing resources and user requests.

Differing from the existing works on joint service caching and task offloading, several unique challenges arise for joint foundation model caching and inference to balance the tradeoff among inference latency, accuracy, and resource consumption in mobile edge networks [9]. First, different quantities of requests and performance requirements of the downstream tasks, such as accuracy and latency, are present during the fine-tuning and inference of PFMs [8]. Additionally, a variety of PFMs can be applied to comparable downstream tasks in a range of AI services. This presents a challenge for edge servers in that the cached PFMs may be called interchangeably to handle model misses. Furthermore, PFMs can continuously learn and adapt to new domains and tasks through prompts of instruction and interactive demonstrations [10]. Due to the in-context learning ability of PFMs, cached models can enhance their inference accuracy during inference without parameter updates. These challenges make decisions about cached model management and request offloading increasingly difficult for optimizing the performance of the framework, which is a tradeoff among inference latency, accuracy, and resource consumption.

To address these issues, in this paper, we investigate the important but rarely studied problem of joint foundation model caching and inference of generative AI services for edge intelligence in mobile edge networks. We propose a joint foundation model caching and inference framework to serve PFMs for provisioning generative AI services. Furthermore, to balance the tradeoff among inference latency, accuracy, and resource consumption, we propose a new metric named *Age of Context* (AoC) to indicate the freshness and relevance between examples in historical demonstrations and current inference requests. With a context vanishing factor, the AoC follows the non-increasing utility function that affects the effective examples in context from instruction, demonstrations, and outputs of past interactions. Based on the AoC, we propose a Least Context (LC) algorithm to manage cached PFMs at edge servers. Simulation experiments demonstrate that the proposed LC algorithm can reduce the total system cost by utilizing contextual information for improving the service accuracy and utilizing the computing power and GPU memory of edge servers efficiently.

The main contributions of this paper are summarized as follows.

- For the first time, we formulate the joint foundation model caching and inference problem in mobile edge networks, for minimizing service cost and accuracy loss under limited computing and GPU memory capacity of edge servers.
- Considering the in-context learning ability of PFMs, we propose a new metric named age of context to measure the freshness and relevance of historical examples in context and current inference requests.
- Based on the AoC, we develop the least context algorithm to efficiently manage the cached models by utilizing the contextual information and thus reducing model switching, inference, and accuracy costs.

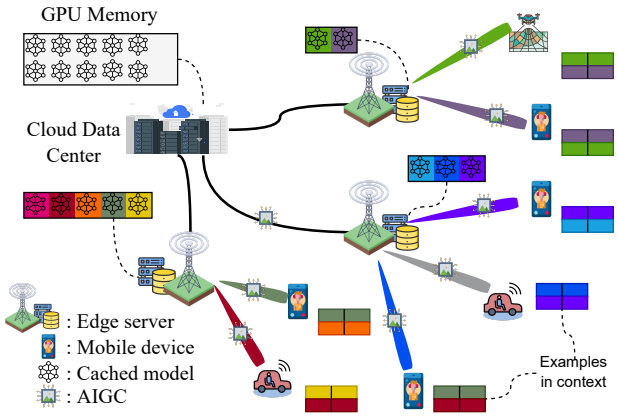

Fig. 1: Joint foundation model caching and inference of generative AI services for edge intelligence.

Compared with our prior work in [11], this paper provides formal mathematical formulations for the joint foundation model caching and inference problem, the new age of context metric, and the least context algorithm.

II. System Model

As shown in Fig. 1, we consider an edge intelligence system model consisting of service providers, including one cloud data center and a set of edge servers, and a massive amount of users. The cloud data center and edge servers can serve generative AI services. The cloud data center is represented by 0 and the set of edge servers is represented by $\mathcal{N} = \{1, 2, \ldots, N\}$. In this system, edge servers and the cloud center provide generic AI services such as AIGC, depending on different PFMs. We use a set $\mathcal{I} = \{1, 2, \ldots, I\}$ to denote the available generative AI services based on a set of PFMs $\mathcal{M} = \{1, 2, \ldots, M\}$. As PFMs are capable of performing multiple downstream tasks in generative AI services simultaneously, and thus we consider $I \gg M$ that the number of AI servers is far greater than the number of PFMs.

Mobile users must request generative AI services from edge servers or cloud data centers when their devices are insufficient for executing PFMs. The inference requests of the diverse services might request different PFMs when they are serving different functions. Typically, a generative AI service requires the collaboration of several PFMs to process users' requests. For instance, in Stable Diffusion [12], text-related conditioning is based on a pre-trained CLIP ViT-L/14 model. Then, a variational autoencoder compresses images into a smaller dimensional latent space. Finally, a U-Net block is used to denoise the output from forward diffusion backwardly to obtain a latent representation. This is a typical process of serving text-to-image generation service requests. We use $R^t_{n,i,m}$ to denote the number of inference requests generated by AI service i to execute foundation model m at edge server n. The configuration of PFM m consists of the amount of

Authorized licensed use limited to: Auburn University. Downloaded on January 13,2025 at 18:15:50 UTC from IEEE Xplore. Restrictions apply.

runtime GPU memory, which is proportion to model size s_m, the inference cost per token e_m, the model accuracy a_m, and the size of context window w_m. The inference process of AI services can put certain context information into the context window of models. Then, the number of examples in context is denoted by $K_{i,m}^t$ of model m for application i which is zero initially, i.e., $K_{i,m}^0 = 0$.

A. Decision Variables

To offer AI services based on PFMs, we propose a joint foundation model caching and inference framework. Edge servers need to make model caching and request offloading decisions to utilize the existing edge computing resources for accommodating generative AI service requests of mobile users. Specifically, edge server n needs to determine the following variables: (i) Let $a_{n,i,m}^t \in \{0,1\}$ denote the binary variable indicating whether model m of application i is cached at edge server n at time slot t; (ii) Let $b_{n,i,m}^t \in [0,1]$ denote the continuous variable on whether model m of application i is cached at edge server n at time slot t. Let $\mathbf{a}_n^t = \{a_{n,1,1}^t, \ldots, a_{n,I,M}^t\}$ denote the model caching decisions of edge server n and $\mathbf{a}^t = \{\mathbf{a}_1^t, \ldots, \mathbf{a}_n^t\}$. In addition, the request offloading decision of edge server n can be denoted as $\mathbf{b}_n^t = \{b_{n,1,1}^t, \ldots, b_{n,I,M}^t\}$ and the request offloading decisions of all edge servers can be denoted as $\mathbf{b}^t = \{\mathbf{b}_1^t, \ldots, \mathbf{b}_n^t\}$.

The generative AI service requests of users can be executed at edge servers if the required components of models are loaded at the GPU memories. Let G_n denote the capacity of GPU memory of edge server n. Then, the model caching decision variables are subjected to the following constraint

$$\sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{M}} a_{n,i,m}^t s_m \leq G_n, \forall n \in \mathcal{N}. \quad (1)$$

The models of AI services can be executed at the edge server after they are loaded into the GPU memory. Therefore, the constraint of model execution at edge servers is

$$b_{n,i,m}^t \mathbf{1}(R_{n,i,m}^t > 0) \leq a_{n,i,m}^t, \forall n \in \mathcal{N}, i \in \mathcal{I}, m \in \mathcal{M} \quad (2)$$

at time slot t and $\mathbf{1}(\cdot)$ is the indicator function. Let E_n denote the resource capacity of edge server n. The total resource consumption of servers is constrained by the total energy capacity, which can be represented as

$$\sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{M}} e_m a_{n,i,m}^t b_{n,i,m}^t R_{n,i,m}^t \leq E_n, n \in \mathcal{N}. \quad (3)$$

In cloud data centers, there are no GPU memory constraints or energy constraints for cached PFMs.

B. Age of Context and In-context Learning Accuracy

PFMs, such as GPT-3, have the ability to perform in-context learning, which means that they can learn from past prompts and inference results when an unseen task is presented to them. Some primary experiments show that larger models are more effective at using in-context instructions and demonstrations, as demonstrated by their improved ability to learn a task from contextual information [3]. This is particularly useful in NLP tasks, where understanding the context of a sentence or paragraph is crucial for accurate interpretation. Based on the evidence that GPT-3 is capable of in-context learning, which contributes to its strong performance on a variety of language tasks, such as translation, basic arithmetic, and Q&A. Let $K_{i,m}^t$ denote the number of effective examples of model m for application i. The examples in the demonstration might have different impacts on the model performance in terms of relevance, quality, and freshness. We propose the AoC to measure the freshness of examples in demonstrations that have an impact on the quality of services provided by PFMs in tasks that are now being carried out downstream. For instance, the historical Q&A records that are recorded during PFM inference can be used to improve future inference accuracy. These examples can be used to increase the accuracy of PFMs, as PFMs can use meta-gradient learn during interaction to fit them [13]. However, depending on the caliber, applicability, and timeliness of examples, the meta-gradient may have favorable or unfavorable impacts on the model performance. Similar to the definition of age of information (AoI), the AoC measures the freshness of historical contextual examples in demonstrations between the cached PFMs and the inference requests. As shown in TABLE I, with a vanishing factor $\nu_{i,m}$ of context, the AoC is adjusted by the non-increasing age utility function. Therefore, the effective number of examples in context $K_{i,m}^t$ at edge server n can be represented as

$$K_{i,m}^t = \min\left(w_m, \{K_{i,m}^{t-1} + R_{n,i,m}^t a_{n,i,m}^t b_{n,i,m}^t - \nu_{i,m}\}^+\right), \quad (4)$$

for $t = 1, \ldots, T$. According to the AoC, the weighted total of the number of examples in demonstrations may be used to determine the number of examples in context.

TABLE I: The Parameters of Accuracy in Downstream Tasks of GPT3-13B/175B [3].

Task	Model	K	A^0	A^1	α
Translation	13B	64	15.45	11.8	0.0923
	175B	64	22.03	7.59	0.1565
Basic Arithmetic	13B	50	3.79	12.19	-0.0501
	175B	50	25.99	14.72	0.1813
SuperGLUE	13B	32	54.40	9.89	0.0969
	175B	32	58.20	10.70	0.1431

As shown in Table I, the in-context (few-show) accuracy $A_{i,m}$ of model m for the downstream task in application i can be fit by a logarithmic function as [3]

$$A_{i,m}(K_{i,m}^t) = A_m^0 + A_m^1 \log_2(1 + {K_{i,m}^t}^{\alpha_m}), \quad (5)$$

where A_m^0 is the zero-shot accuracy, A_m^1 is the one-shot accuracy, $K_{i,m}^t$ is the number of examples in context, and α_m is the coefficient of model m.

C. Cost Structure

As discussed above, the generative AI service requests can be executed by edge servers and offloaded to cloud data centers over the core network. Given the model caching and request offloading decisions, the total system cost of serving generative

Authorized licensed use limited to: Auburn University. Downloaded on January 13,2025 at 18:15:50 UTC from IEEE Xplore. Restrictions apply.

AI services consisting of the edge inference cost and cloud inference cost can be formulated as follows.

1) Edge Inference Cost: Specifically, the edge inference cost consists of the edge switching cost, the edge transmission cost, the edge computing cost, and the model accuracy cost. According to model caching decisions, each edge server needs to load models into the GPU memory before execution. During the loading process, the model switching cost consisting of the model loading latency and hardware wear-and-tear cost is incurred. Therefore, the switching cost l_n^s of edge server n to load and evict models can be calculated as

$$l_n^{switch}(\mathbf{a}^t) = \sum_{i\in\mathbb{I}} \sum_{m\in\mathcal{M}} \lambda \mathbf{1}(a_{n,i,m}^t > a_{n,i,m}^{t-1}), \tag{6}$$

where λ denotes the coefficient for loading and evicting the model and $\mathbf{1}(\cdot)$ is the indicator function. When $a_{n,i,m}^t > a_{n,i,m}^{t-1}$, i.e., $a_{n,i,m}^t = 1$ and $a_{n,i,m}^{t-1} = 0$, $\mathbf{1}(a_{n,i,m}^t > a_{n,i,m}^{t-1})$ indicates that the loading of an uncached model. Otherwise, there is no switching cost incurred at edge servers.

When the requested models are cached into the GPU memory of edge servers, users communicate with the edge servers for requesting generative AI services. Let l_n^{trans} denote the transmission cost of input prompts and inference results. The transmission cost of edge server n can be calculated as

$$l_n^{trans}(\mathbf{a}^t, \mathbf{b}^t) = \sum_{i\in\mathcal{I}} \sum_{m\in\mathcal{M}} l_{n,m} R_{n,i,m}^t a_{n,i,m}^t b_{n,i,m}^t, \tag{7}$$

where $r_{i,m}$ is the unit transmission cost per input and result for model m of application i.

Let f_n denote the computing capacity of edge server n. The forward propagation process of AI services at edge servers incurs inference latency, which can be denoted as l_n^{comp} for edge server n. The edge computing cost can be calculated as

$$l_n^{comp}(\mathbf{a}^t, \mathbf{b}^t) = \sum_{i\in\mathcal{I}} \sum_{m\in\mathcal{M}} R_{n,i,m}^t a_{n,i,m}^t b_{n,i,m}^t \frac{c_n}{f_n}. \tag{8}$$

Finally, as edge servers might not have sufficient resources for executing the best match model requested by AI services, the requests processed by other PFMs with the equivalent function incur accuracy cost l_n^{acc}, which can be represented as

$$l_n^{acc}(\mathbf{a}^t, \mathbf{b}^t) = \sum_{i\in\mathcal{I}} \sum_{m\in\mathcal{M}} (1 - A_{i,m}) R_{n,i,m}^t a_{n,i,m}^t b_{n,i,m}^t. \tag{9}$$

By sacrificing some accuracy of generative AI services, the system can reduce the model missing rate. Therefore, the total edge inference cost of edge server n is

$$\begin{aligned} L_n^t(\mathbf{a}^t, \mathbf{b}^t) =& l_n^{switch}(\mathbf{a}_n^t) + l_n^{trans}(\mathbf{a}_n^t, \mathbf{b}_n^t) \\ &+ l_n^{comp}(\mathbf{a}_n^t, \mathbf{b}_n^t) + l_n^{acc}(\mathbf{a}_n^t, \mathbf{b}_n^t). \end{aligned} \tag{10}$$

The edge inference cost is jointly determined by the caching decisions and offloading decisions of edge servers. Nevertheless, the missed or offloaded requests are processed by the cloud data center.

2) Cloud Inference Cost: The edge servers are resource-constrained, such that they are unable to serve all PFMs. On one hand, due to the limited storage resources of the edge server, the model requested by a user may be too large to be loaded into the GPU of the edge server. On the other hand, the limited computing power of the edge server makes it necessary to actively migrate some requests to the cloud data center for execution. Therefore, when the requested models are missed at edge servers or offloaded to cloud data centers, this part of user requests are transmitted to the cloud data center, which needs to allocate resources for accomplishing such user requests. In line with [14], the cloud data centers can consider serving generative AI services in a serverless manner, which is charged in a "pay-as-you-go" manner. Therefore, users need to pay for executing AI services according to the number of requests instead of specific occupied resources. When the requests are missed at edge servers or edge servers do not have enough resources for serving the requests, the unaccomplished requests will be offloaded to the cloud data centers for remote execution. The cloud data centers can execute the models with their abundant computing and energy resources, and then return the inference results to edge servers. However, cloud inference incurs additional latency for data transmission in the core network, which is much higher than the data transmission latency at edge servers. Moreover, the accuracy cost of offloaded inference requests executed by the cloud data center is expected to be almost zero as they can be processed by the most accurate model with common in-context examples owned by the data center. Based on the above analysis, we use $l_{0,n}$ to denote the aggregated cost of offloading one request to the cloud data center for remote execution of model m. Then, the total cloud computing cost at time slot t is

$$L_0^t(\mathbf{a}^t, \mathbf{b}^t) = \sum_{n\in\mathcal{N}\setminus\{0\}} \sum_{i\in\mathcal{I}} \sum_{m\in\mathcal{M}} l_{0,m}(1 - a_{n,i,m}^t b_{n,i,m}^t) R_{n,i,m}^t. \tag{11}$$

D. Problem Formulation

To optimize the performance of mobile edge intelligence, we jointly consider the cost of edge inference and cloud inference, including the switching cost, the accuracy cost, the transmission cost, and the inference cost over a time horizon T. The problem is formulated as follows:

$$\min_{\mathbf{a}^t, \mathbf{b}^t} \quad \frac{1}{T} \sum_{t\in\mathcal{T}} \left(L_0^t + \sum_{n\in\mathcal{N}} L_n^t \right) \tag{12a}$$

$$\text{s.t.} \quad (1), (2), (3), \tag{12b}$$

$$a_{n,i,m}^t \in \{0, 1\}, \tag{12c}$$

$$b_{n,i,m}^t \in [0, 1]. \tag{12d}$$

To solve the optimization problem described above, we must overcome the following challenges: (i) The problem involves time-coupling elements, such as GPU memories and in-context examples, as it considers both future request dynamics and historical inference contexts; (ii) Through historical statistical data, we can forecast future information before making a decision The problem becomes a mixed-integer programming problem, which is NP-hard. To address these challenges, a low-complexity heuristic algorithm is needed to make deci-

Authorized licensed use limited to: Auburn University. Downloaded on January 13,2025 at 18:15:50 UTC from IEEE Xplore. Restrictions apply.

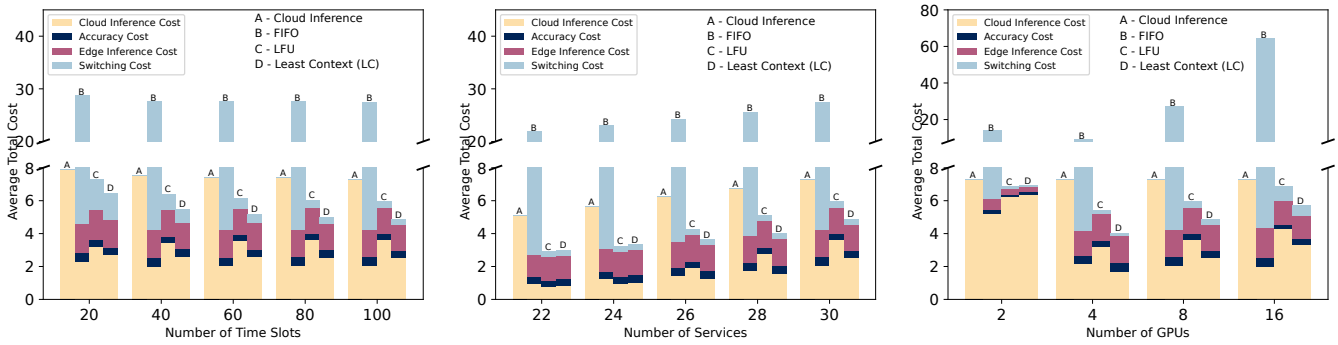

Fig. 2: Average total cost versus number of time slots.

Fig. 3: Average total cost versus number of services.

Fig. 4: Average total cost versus number of GPUs.

sions regarding model caching and request offloading, despite the lack of future information.

III. The Least Context Algorithm

To effectively serve PFMs for provisioning generative AI services, we propose the least context algorithm based on the AoC metric. When additional GPU memory is required for loading an uncached requested PFM, the LC algorithm counts the number of examples in context, calculates them, and removes the cached PFM with the fewest effective examples in context. Therefore, at each time slot t, the model caching decisions can be obtained by solving the maximization problem of the number of effective examples for the cached models, which can be represented as

$$\max_{\mathbf{a}^t} \quad \sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{M}} K_{i,m}^t \tag{13a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{M}} a_{n,i,m}^t s_m \leq G_n^t, \forall n \in \mathcal{N}, \tag{13b}$$

$$a_{n,i,m}^t \in \{0,1\}. \tag{13c}$$

The available capacity of GPU memory G_n^t of server n at time slot t can be calculated as $G_n^t = G_n - R_{n,i,m}^t a_{n,i,m}^t b_{n,i,m}^t s_m$. This optimization problem can be tackled with a complexity of $O(IM)$ with prior knowledge and statistical data. This algorithm gives the least important PFM for the current inference task priority for eviction. It works well with huge numbers of PFMs on edge servers with limited GPU memory. By using more contextual information during inference, the PFMs of mobile generative AI services are more accurate. Based on caching decisions $\mathbf{a}^t$ by solving the optimization problem (13a), offloading decisions $\mathbf{b}^t$ are obtained by solving the optimization problem (12a).

IV. Numerical Results

In the experiment, we consider an edge intelligence system with $T = 100$ slots. The requests for generative AI services per time slot follow the Poisson process with an average of one. We consider three types of PFMs and select six representative models to serve in the experiments, i.e., GPTs, Uniformers, and CLIPs. The detailed model configuration can be found in [11]. The main parameters are listed in TABLE II.

TABLE II: The List of Main System Parameters.

Parameters	Value
Number of time slots	100
Number of services	30
Number of GPUs	8
Size of context window	2048
Size of examples	[10, 100]
GPU Memory	80 GB
GPU Computing Capacity	312000 GFLOPS
Edge transmission cost	0.0001
Cloud inference cost	0.0015
Switching cost coefficient	0.0001
Accuracy cost coefficient	0.01
GPU energy efficiency	810 GFLOPS/W
Energy capacity	300 W

We evaluate the proposed LC approach in comparison to several baselines including cloud inference, the first-in-first-out (FIFO) caching algorithm, and the least frequently used (LFU) caching algorithm. Initially, we examined the effectiveness of the LC algorithm by comparing the average total cost in various system settings. As we observe in Fig. 2, the switching cost of the LC algorithm gradually converges to a smaller value at around 1.3%, while the switching cost of the FIFO algorithm remains constant with system time. This indicates that the LC algorithm is able to cache most of the required models for inference services on the edge server in GPU memory. In addition, the LC algorithm achieves the lowest average total cost among all the algorithms. The LC algorithm can reduce the cloud inference cost by increasing the utilization of edge computing resources so that the requests can be executed at edge servers with low latency.

We then show that the proposed LC algorithm is robust under different system settings, such as a different number of services and a different number of GPUs. From Fig. 3, we can see that the total system cost increases with the number of services. This is because the resources in the edge servers become insufficient when more services need to be served on the edge servers. On one hand, the GPU memory on the edge

Authorized licensed use limited to: Auburn University. Downloaded on January 13,2025 at 18:15:50 UTC from IEEE Xplore. Restrictions apply.

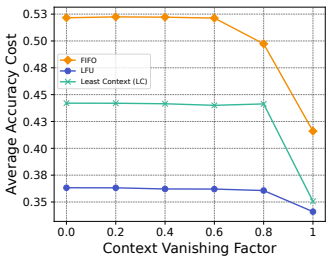

Fig. 5: Edge accuracy cost vs. context vanishing factor.

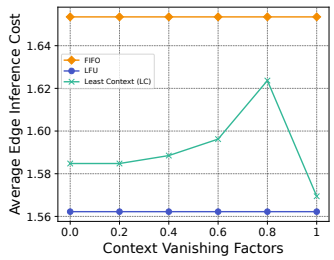

Fig. 6: Edge inference cost vs. context vanishing factor.

servers is limited, and as the number of services increases, more model switching will be required when running the model, so the switching cost becomes higher. On the other hand, when the resources on the edge servers are not sufficient, the requests for cloud inference have to be forwarded to cloud data centers, whose costs are higher than those of edge inference. In the meanwhile, the experimental results in Fig. 4 indicate that the number of GPUs has a complex impact on the total system cost. When the number of GPUs increases, the switching cost increases. The reason is that the edge servers can cache more models in the GPU memory. Without effective management of cached models, the switching cost is high for the FIFO algorithms. Though the cost of the proposed LC algorithm is always lower than those of the other algorithms, its cost increases when the number of GPUs increases. The reason behind this trend is that edge servers can cache larger models when the number of GPUs is large. However, such large models require intensive computing resources while incurring similar edge inference costs. Therefore, these user requests for large models are better offloaded to cloud data centers for remote execution.

After demonstrating the effectiveness of the proposed LC algorithm, we next investigate the impacts of the context vanishing factor. To make comparisons between models more noticeable, the size of the context window is set to 2^{14}. As shown in Fig. 5, as the context vanishing factor increases, the average accuracy cost of edge inference is first static and then decreases. When the context vanishing factor is small, the performance gap among these three algorithms becomes large. However, when the context vanishing factor is large, such as one in Fig. 5, the average accuracy cost decreases, and their performance gap starts to shrink. Another interesting finding shown in Fig. 6 is that the averse edge inference cost first increases as the context vanishing factor increases and then dramatically declines after a certain threshold.

V. Conclusions

In this paper, we investigated the joint foundation model caching and inference problem for deploying PFMs to serve AI-based multimedia services in mobile edge networks. We introduced a joint foundation model caching and inference framework designed to effectively provision generative AI services at edge servers, and thus advancing toward AGI. To this end, we proposed a new metric for measuring the relevance and freshness of contextual examples in relation to ongoing inference requests. Moreover, we have developed the LC algorithm for PFM management, which optimizes the utilization of historical contextual prompts and inference results, subsequently enhancing the performance of generative AI services. Experimental results indicated that the LC algorithm effectively reduces system costs by effectively leveraging historical demonstrates and managing cached models.

References

[1] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg *et al.*, "Sparks of artificial general intelligence: Early experiments with GPT-4," *arXiv preprint arXiv:2303.12712*, Mar. 2023, [Online]. Available: https://arxiv.org/abs/2303.12712.

[2] P. Zhou, J. Zhu, Y. Wang, Y. Lu, Z. Wei, H. Shi, Y. Ding, Y. Gao, Q. Huang, Y. Shi *et al.*, "Vetaverse: Technologies, applications, and visions toward the intersection of metaverse, vehicles, and transportation systems," *arXiv preprint arXiv:2210.15109*, Oct. 2022, [Online]. Available: https://arxiv.org/abs/2210.15109.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Proc. of the Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, Dec. 2020.

[4] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He *et al.*, "A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT," *arXiv preprint arXiv:2302.09419*, Feb. 2023, [Online]. Available: https://arxiv.org/abs/2302.09419.

[5] M. Xu, D. Niyato, J. Chen, H. Zhang, J. Kang, Z. Xiong, S. Mao, and Z. Han, "Generative AI-empowered simulation for autonomous driving in vehicular mixed reality metaverses," *arXiv preprint arXiv:2302.08418*, Feb. 2023, [Online]. Available: https://arxiv.org/abs/2302.08418.

[6] S. Zeng, H. Zhang, B. Di, Z. Han, and L. Song, "Reconfigurable intelligent surface (ris) assisted wireless coverage extension: Ris orientation and location optimization," *IEEE Communications Letters*, vol. 25, no. 1, pp. 269–273, 2020.

[7] Y. Huang, M. Xu, X. Zhang, D. Niyato, Z. Xiong, S. Wang, and T. Huang, "Ai-generated 6g internet design: A diffusion model-based learning approach," *arXiv preprint arXiv:2303.13869*, 2023.

[8] G. R. Gilman, S. S. Ogden, R. J. Walls, and T. Guo, "Challenges and opportunities of dnn model execution caching," in *Proc. of the Workshop on Distributed Infrastructures for Deep Learning*, Davis, CA, Dec. 2019, pp. 7–12.

[9] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. of the IEEE*, vol. 107, no. 8, pp. 1738–1762, Jun. 2019.

[10] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui, "A survey for in-context learning," *arXiv preprint arXiv:2301.00234*, Jan. 2023, [Online]. Available: https://arxiv.org/abs/2301.00234.

[11] M. Xu, D. Niyato, H. Zhang, J. Kang, Z. Xiong, S. Mao, and Z. Han, "Sparks of gpts in edge intelligence for metaverse: Caching and inference for mobile aigc services," *arXiv preprint arXiv:2304.08782*, Apr. 2023, [Online]. Available: https://arxiv.org/abs/2304.08782.

[12] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, LA, Jun. 2022, pp. 10 684–10 695.

[13] D. Dai, Y. Sun, L. Dong, Y. Hao, Z. Sui, and F. Wei, "Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers," *arXiv preprint arXiv:2212.10559*, Dec. 2022, [Online]. Available: https://arxiv.org/abs/2212.10559.

[14] K. Zhao, Z. Zhou, X. Chen, R. Zhou, X. Zhang, S. Yu, and D. Wu, "Edgeadaptor: Online configuration adaption, model selection and resource provisioning for edge dnn inference serving at scale," *IEEE Transactions on Mobile Computing*, pp. 1 – 16, Jul. 2022.

Authorized licensed use limited to: Auburn University. Downloaded on January 13,2025 at 18:15:50 UTC from IEEE Xplore. Restrictions apply.