SPARKS OF GENERATIVE PRETRAINED TRANSFORMERS IN EDGE INTELLIGENCE FOR THE METAVERSE

Caching and Inference for Mobile Artificial Intelligence-Generated Content Services

Minrui Xu, Dusit Niyato, Hongliang Zhang, Jiawen Kang, Zehui Xiong, Shiwen Mao, and Zhu Han

Aiming at achieving artificial general intelligence (AGI) for the metaverse, pretrained foundation models (PFMs), e.g., generative pretrained transformers (GPTs), can effectively provide various artificial intelligence (AI) services, such as autonomous driving, digital twins (DTs), and AI-generated content (AIGC) for extended reality (XR). With the advantages of low latency and privacy-preserving, serving PFMs of mobile AI services in edge intelligence is a viable solution for caching and executing PFMs on edge servers with limited computing resources and GPU memory. However, PFMs typically consist of billions of parameters that are computation- and memory-intensive for edge servers during loading and execution. In this article, we investigate edge PFM serving problems for mobile AIGC services of the metaverse. First, we introduce the fundamentals of PFMs and discuss their characteristic fine-tuning and inference methods in edge intelligence.

Digital Object Identifier 10.1109/MVT.2023.3323757
Date of current version: 3 November 2023

Then, we propose a novel framework of joint model caching and inference for managing models and allocating resources to satisfy users' requests efficiently. Furthermore, considering the in-context learning ability of PFMs, we propose a new metric to evaluate the freshness and relevance between examples in demonstrations and executing tasks, namely the *Age of Context* (*AoC*). Finally, we propose a least-context (LC) algorithm for managing cached models at edge servers by balancing the tradeoff among latency, energy consumption, and accuracy.

Introduction

Toward AGI in the metaverse [1], [2], PFMs, e.g., GPTs [3], with billions of parameters have achieved great success in a variety of fields in recent years, effectively demonstrating emergence abilities in downstream tasks with different data modalities [4]. The pretraining approach provides reasonable parameter initialization for various downstream applications, such as object detection, image generation, and text retrieval. Therefore, PFMs, including language foundation models (LFMs), visual foundation models (VFMs), and multimodal foundation models (MFMs), are in the paradigm of transfer learning that can generalize to new tasks and domains without any task-specific data during pretraining.

The metaverse, as a collective virtual shared space, heavily relies on AI services for seamless interactions and realistic simulations. AIGC [5] brings life to the metaverse by creating content dynamically. Mobile edge computing [6] ensures that these AI services are delivered with low latency, while PFMs act as the underlying AI models driving these services. Together, they create an ecosystem that powers the metaverse. PFMs can empower a multitude of intelligent services for the metaverse, such as autonomous driving, DTs, and AIGC for XR. For instance, PFMs can facilitate complex driving decisions and generate traffic simulations for autonomous driving [7]. Moreover, PFMs can help understand and respond to human emotions and behaviors during immersive human–avatar interactions. For example, based on the GPT-3 [3], which is an LFM with 175 billion parameters, ChatGPT (https://openai.com/blog/chatgpt/) enables long and fluent conversations with humans using world knowledge and contextual awareness. In addition to serving PFMs at cloud servers, edge servers equipped with GPU resources can also support fine-tuning and inference processes of metaverse services, which brings the sparks of GPTs to mobile edge networks. Therefore, the deployment of PFMs in black enables the delivery of localized AI services with low latency.

However, compared to cloud servers, resource-constraint edge servers cannot load all PFMs simultaneously to satisfy the requests of services in the metaverse. Aiming at provisioning mobile AI services in edge networks, existing works primarily focus on offloading AI services to cloud servers for remote execution or caching inference outputs at edge servers for low-latency access [8]. On the one hand, offloading PFMs of AI services to cloud servers leads to additional latency in the core network, additional traffic, and privacy risks for users of AI services. On the other hand, caching reasoning results on edge servers is no longer efficient for real-time AI service delivery. Therefore, direct deployment of PFMs on edge servers requires effective and fine-grained resource and request management for AI query execution with the available computational and energy resources.

Specifically, in contrast to existing works on joint service caching and task offloading [9], there are several unique difficulties for joint PFM caching and inference to balance the tradeoff among accuracy, latency, and energy consumption in edge intelligence, as follows [6]:

- *Dynamic runtime configuration:* During the execution of PFMs, there are a varying number of requests and performance requirements for downstream tasks, such as accuracy and latency [8].
- *Equivalent model adaptation:* Different PFMs are adaptively applied to similar downstream tasks in different metaverse services [4]. This presents a challenge for edge servers, as cached PFMs can be used interchangeably for inference to minimize model miss.
- *Continuous in-context learning:* PFMs, like GPT-3, can continuously learn and adapt to new domains and tasks based on interactive demonstrations for personalization and customization [10]. The ability of in-context learning enables cached PFMs to improve their performance during inference without parameter updates. This increases the complexity of cache replacement and deployment decisions by introducing a new tradeoff among inference latency, resource consumption, and accuracy.

To address these issues, this article investigates the potential but scarcely studied problems of PFM caching and inference in black. We first introduce the fundamentals of PFMs for serving mobile AIGC services of the metaverse, and their fine-tuning and inference methods in edge networks. Then, we present a joint model caching and inference framework in edge networks to serve PFMs of mobile AI services of the metaverse. Furthermore, we discuss potential applications and challenges of serving PFMs for metaverse services. Finally, we propose a novel metric to indicate the freshness and relevance of examples in demonstrations and current tasks, namely the AoC, to balance the tradeoff between inference latency, resource consumption, and accuracy. The AoC follows the nonincreasing utility function that affects the effective examples in context from the entirety of demonstrations resulting from historical interactions. Based on this metric and the number of examples in context, we propose an LC algorithm to manage PFMs at edge servers. Experimental results demonstrate that

the proposed LC algorithm can reduce the total system cost by improving the accuracy of edge-cached PFMs, reducing offloading latency, and utilizing the caching and computing resources of edge servers efficiently.

Serving PFMs in Edge Intelligence for the Metaverse

Fundamentals of PFMs

PFMs belong to the transfer learning paradigm that is used to initialize parameters for downstream tasks. PFMs [4]—such as BERT, GPT-3, Stable Diffusion, CLIP, and ChatGPT—leverage large-scale datasets and pretraining techniques to provide reasonable parameter initialization for various AI services [4]. As shown in Figure 1, there are primarily three types of PFMs: i.e., LFMs, VFMs, and MFMs.

LFMs

LFMs, also known as *large-scale language models*, are PFMs designed to understand, process, and generate human languages. LFMs are trained on massive amounts of text data and can develop a broad understanding of language, including grammar, syntax, semantics, and even some aspects of common knowledge. Two examples of LFMs are GPT and ChatGPT, which have demonstrated impressive abilities in natural language understanding and generation. GPT-3 can enable conversations with humans based on world knowledge and contextual awareness, while ChatGPT is designed to generate human-like responses in a chatbot setting. LFMs employ self-attention mechanisms to better understand the context and relationships between words in a given text and can be adopted in various downstream tasks, such as sentiment analysis, machine translation, text summarization, question answering, and text generation.

VFMs

VFMs specialize in understanding and generating complex images and videos, which are designed to process visual information and generate target outputs. VFMs have shown great potential in advancing the field of computer vision, but they are computing-intensive, particularly during the inference stage. For example, the U-Net in Stable Diffusion [11] is a generative model that can produce high-quality images by iteratively refining a noise vector.

MFMs

MFMs can process multiple types of data—such as text, images, and audio—simultaneously. They are trained on datasets containing various data modalities to learn the relationships, patterns, and structures within and across different data types. For instance, CLIP is one of the MFMs that classify images based on textual descriptions [12], which uses contrastive learning to train on text and image pairs, distinguishing between positive and negative pairs. During inference, the model takes in an image and a textual description and outputs a score representing the likelihood that the image matches the description, calculated through a dot product.

Fine-Tuning of PFMs

Fine-tuning refers to the process of improving the performance of PFMs to a specific downstream task by updating its parameters. Since PFMs usually consist of billions of parameters, the fine-tuning process is computationally

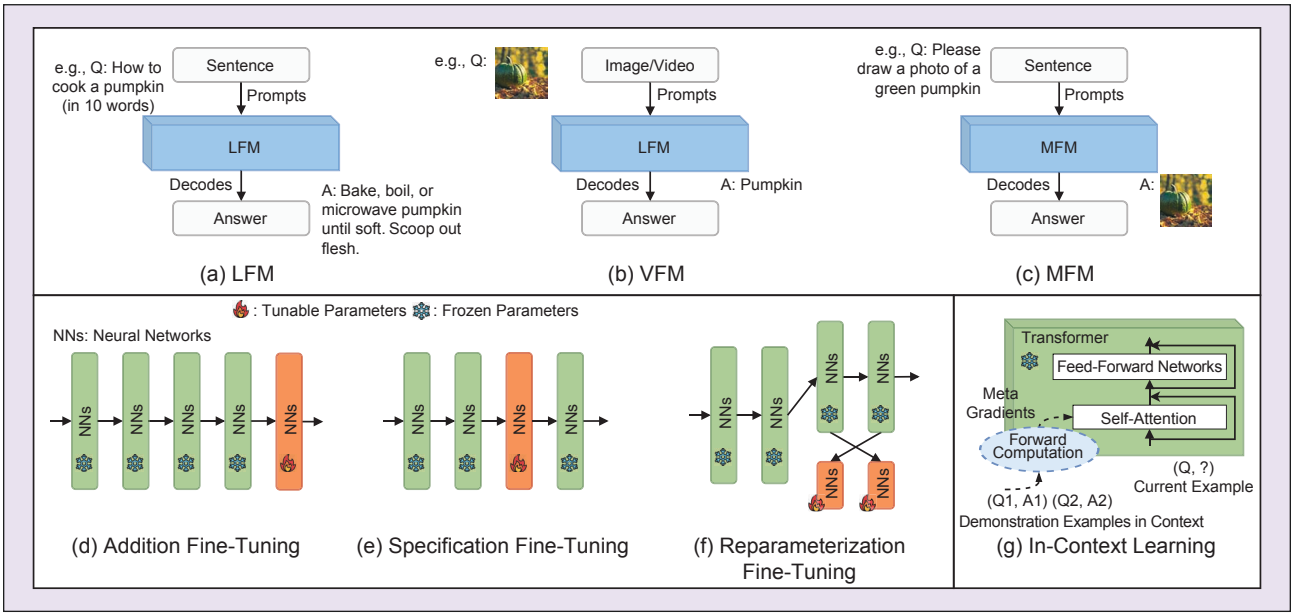


Figure 1 Categories of PFMs and their characteristic fine-tuning and inference methods. (a)–(c) The workflows of LFMs, VFMs, and MFMs. (d)–(f) The illustration of parameter-efficient fine-tuning. (g) An example of in-context learning.

intensive. Therefore, parameter-efficient fine-tuning of PFMs is utilized for achieving comparable performance to traditional fine-tuning while reducing resource consumption [6]. As shown in Figure 1, parameter-efficient fine-tuning can be categorized into three types, including addition-based, specification-based, and reparameterization-based methods, as follows [13]:

- *Addition-based* methods involve adding a small number of parameters to the PFMs and fine-tuning them. These methods, which include scalar addition, vector addition, and layer addition, add parameters to the PFMs that are specific to the fine-tuning data. For instance, such parameters include additional layers or heads after the output layer of PFMs.
- *Specification-based* methods modify the architecture of PFMs to better suit downstream tasks. These methods, such as layer removal, layer replacement, and layer scaling, adjust the PFMs' parameters and architecture to improve performance.
- *Reparameterization-based* methods reduce the number of tunable parameters in PFMs by reparameterizing their parameters. These methods, such as low-rank factorization, matrix decomposition, and subspace projection, reparameterize the PFMs to reduce the number of tunable parameters while preserving the PFMs' expressiveness.

In edge networks, due to resource constraints, fine-tuning and inference methods prioritize efficiency. Techniques like early stopping or using a subset of data might be adopted [6]. In cloud servers, with abundant resources, the focus is on achieving the highest accuracy, even if it requires more extensive fine-tuning and larger datasets. Depending on applications such as the metaverse, the fine-tuning methods can be selected adaptively depending on the resource and performance requirements.

Inference of PFMs

Different from fine-tuning that updates the parameters of PFMs, the inference is to make predictions on input service requests without changing the parameters. Instead of injecting or updating neural modules in AI models, PFMs can provide accurate output for the task that does not exist in the training, fine-tuning, and inference from instructions and demonstrations from interaction without parameter updates. As shown in Figure 2, there are three scenarios during the inference of PFMs [3], including *zero-shot*, *one-shot*, and *few-shot* learning.

First, zero-shot learning refers to the PFMs that are evaluated on a task for which it has not been explicitly trained. Then, one-shot learning indicates the PFMs need to perform the inference for a new task based on only one example of that task. Finally, few-shot learning implies that a few demonstrations are provided before the inference of the new task. Based on the few-shot learning, the PFMs can perform a metagradient in the self-attention layer for adaptation to the new task. Different from fine-tuning, few-shot learning or in-context learning can perform metagradient in the attention layers during inference without changing its model parameters. Therefore, few-shot learning can improve the model performance based on examples in instructions and/or demonstrations. However, extra computation consumption and latency are required by processing the examples that depend on the size of the context window in PFMs. These learning techniques can be applied in different applications where the model is expected to perform inference on classes that have never been seen during training (i.e., zero-shot learning), and is provided with just one or very few examples of a new class (i.e., one-shot learning), or is provided with a small dataset of a new class and is expected to generalize well on that class (i.e., few-shot learning).

Joint Model Caching and Inference Framework

To serve PFMs in edge intelligence for the metaverse, we developed a framework of joint model caching and inference to satisfy service-level objectives by utilizing caching, computing, and communication resources in black.

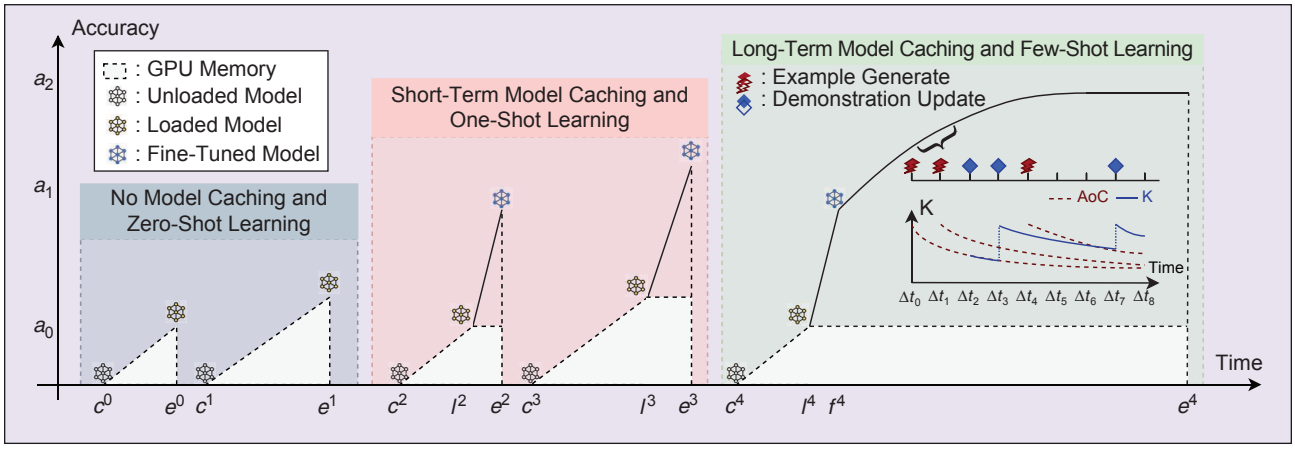


FIGURE 2 An illustration of the performance of zero-, one-, and few-shot accuracy under different model caching settings [3].

Unlike content caching in content delivery networks (CDNs), such as text, images, and videos, the cached models have different cache structures. The cache structure in CDNs is static, with fixed cache sizes and independent of computation resources [9].

Model Caching Configuration

The configuration of each cached PFM consists of the following information:

- *Frequency of use* for PFMs refers to the rate at which a particular model is executed for services in the metaverse. It can be measured in terms of the number of requests per second, the total time spent on processing each request, and other metrics that measure how often a PFM is being utilized.
- *Model size* indicates the number of parameters, weights, and other necessary components of PFMs, which affects the latency and energy cost of edge servers for loading and executing PFMs [8].
- *Runtime GPU memory* measures how much random access memory (RAM) or video RAM is needed by loading the PFM to execute on a given edge/cloud server with its current configuration settings. The runtime GPU memory not only depends on the model sizes but also the runtime precision configuration of the precision. Therefore, there is a tradeoff between model precision and GPU memory usage.
- *Model speed* of PFMs refers to the time complexity or speed at which a particular model can process inference requests. It is usually measured in terms of inference times, i.e., how long it takes for a PFM to complete its task given certain inputs and parameters. Model speed also has implications on accuracy, as faster processing often leads to lower accuracy due to less computation power being available for each request.
- *Model accuracy* of PFMs refers to the degree of correctness or precision with which a model can predict outcomes. For PFMs, model accuracy has implications on speed as higher accuracy often requires more computation power for each request, leading to longer processing times overall.
- *Number of examples in context:* Cached PFMs can accumulate instructions and demonstrations while processing inference requests. The number of examples in context represents the number of related examples in demonstrations the PFMs have gathered. Due to the in-context learning ability of PFMs, the number of examples in context can also impact the accuracy of the models.

Model Caching and Eviction

Since the cache structure of PFMs is more complicated than traditional web/content caching, the model caching and eviction are also more intractable. Model caching and eviction mainly consist of two types of operations, i.e., the passive and active operation as well as binary and partial operation.

PEDESTRIANS AND VEHICLES CAN PROCESS THE SERVICES TO CACHE AND EXECUTE PFMS WITH THEIR LOCAL COMPUTING RESOURCES ON MOBILE DEVICES OR DEVICES NEARBY.

- *Passive and active caching and eviction:* Passive caching is a reactive approach where models are evicted from the cache only when there is not enough GPU memory to load a requested model. Additionally, active caching is a proactive approach where models are evicted and loaded into GPU memory based on predictions of future demand. Active caching can be more efficient than passive caching [8], but requires more sophisticated prediction algorithms and can be less responsive to sudden changes in demand.
- *Binary and partial caching and eviction:* Binary caching involves loading the entire model into GPU memory before starting inference. In contrast, with partial caching, only a portion of the model is loaded into memory, and inference can begin using that portion. This approach provides a lower level of inference but can be useful when memory resources are limited. When additional memory becomes available, the remaining portions of the model can be loaded into memory, improving inference quality.

Collaborative Mobile Edge–Cloud Caching and Inference

As shown in Figure 3, collaborative resource allocation among heterogeneous mobile edge–cloud infrastructures is critical in paving the way toward AGI at the edge.

Mobile Caching and Inference

Pedestrians and vehicles can process the services to cache and execute PFMs with their local computing resources on mobile devices or devices nearby. This solution can be useful in situations where Internet connectivity is limited or unreliable.

Edge Caching and Inference

When the local resources of mobile devices and vehicles are not enough for executing PFMs, offloading these services to edge servers via radio access networks becomes an alternative solution for enabling AI services on edge servers with limited resources. However, due to the limited GPU resources of edge servers, they can only cache several PFMs to react to the user's request. If the edge server does not cache the model requested by the user, it can migrate the user's request to the cloud for execution via core networks or load the model and then execute the model requested by the user. This approach can improve

response time and reduce the load on the cloud infrastructure, making it more scalable and cost-effective.

Cloud Caching and Inference

Cloud caching and inference solutions involve the utilization of powerful cloud servers to provide almost all PFMs for serving users' requests. However, offloading services to cloud servers incurs additional core network latency, which might cause congestion in core networks if there are too many service requests.

Deploying PFMs in mobile edge networks involves optimizing the models to meet the constraints of edge servers. Techniques like model pruning, quantization, and knowledge distillation can be used to reduce the model's size without significantly compromising performance. Once optimized, the models can be deployed on edge servers, close to the end-users, ensuring low latency responses.

Model Caching and Eviction Policy

To design the model caching and eviction policy, three issues should be considered carefully:

- *Reducing model miss rate:* Actively preloading models and optimizing GPU utilization through dynamic scheduling of AI models can minimize latency and model miss rates, streamlining memory use and request handling.
- *Addressing model misses:* Handling model misses at edge servers involves offloading service requests to cloud servers, incurring extra core network latency, or loading missing models, leading to switching costs, such as additional latency and energy consumption for allocating resources as well as hardware wear-and-tear.
- *Timing model cache decisions:* Making cache decisions when the model is first loaded and upon receiving new requests enables dynamic adjustments based on current conditions and usage patterns, promoting efficient caching and lower latency responses.

Therefore, an effective and efficient caching algorithm in this framework should address these three questions properly. Then, we can summarize two remarks for serving PFMs of mobile AI services as follows:

- *Remark 1:* Different from traditional edge content caching in CDNs, whose cache structures are static and independent from computation, the cache structures can be dynamic based on the service runtime configuration, such as batch size. This makes the cache loading and eviction of AI services more complex, which requires not only the consideration of user preferences but also the prediction of the intensity of future service requests.
- *Remark 2:* Unlike traditional computation and task offloading in mobile edge networks, where different computation tasks are independent, the inference tasks of PFM-related services are in-contextual. Therefore, before performing these inference tasks, the AIGC model needs to be preloaded into the edge servers' GPU memory, which can cache a limited number of models to provide AI services. Furthermore, as more in-context examples are collected during the interaction, the performance of cached services can be further improved [3].

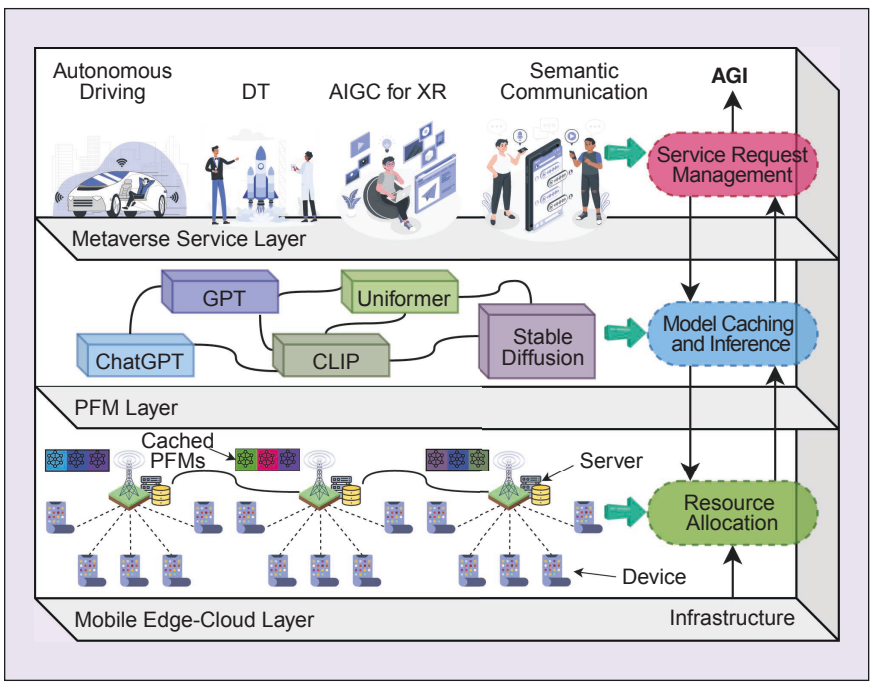


FIGURE 3 An illustration of the collaborative mobile edge-cloud computing architecture for serving PFMs for the metaverse.

Potential Applications and Challenges

Applications

Autonomous Driving

Autonomous driving in the metaverse necessitates AI services, such as traffic and driving simulation, which are dependent on computationally intensive PFMs [7]. To enable this on resource-limited edge servers, model caching and efficient inference scheduling are essential. In autonomous driving, active model switching can enhance traffic efficiency and safety by adapting to changing road conditions or traffic patterns.

DTs

DTs, virtual representations of physical objects or systems, utilize PFMs

for AI capabilities like predictive maintenance, anomaly detection, and optimization. While PFMs have shown great success in natural language processing applications, their vast parameter space and adaptability allow them to model complex systems, a feature that can be leveraged in DTs. DTs, being virtual replicas of physical entities, can benefit from PFMs' ability to predict and simulate physical systems under various scenarios.

Semantic Communication

Semantic communication, a novel paradigm that employs semantic representations, can transform wireless communication systems' design and operation. Its device-to-device pattern enables efficient and secure communication without centralized cloud infrastructure. However, this pattern necessitates advanced model caching algorithms to manage edge servers' limited resources while ensuring cached models' quality and consistency.

AIGC for XR

AIGC is generated by AI methods that utilize PFMs to create content that resembles human-produced content [6]. To provide AI-generated XR services, multiple PFMs are integrated to handle different types of data and produce relevant and meaningful 3D immersive content. The model caching algorithm ensures that the PFMs work smoothly together, maintaining seamless and immersive experiences for metaverse users.

Challenges

Dynamic User Service Requests and Objectives

Joint caching and inference services at edge servers face challenges due to dynamic user requests and objectives, such as service latency and accuracy. To tackle these challenges, edge servers must efficiently manage limited resources, ensure cached model quality and consistency, and design joint caching and inference policies to satisfy users' objectives, considering factors like model size, frequency of use, and accuracy.

Heterogeneous Model Configuration and Computing Resources

Heterogeneous model configuration and computing resources present challenges in proposing joint model caching and inference algorithms. In detail, PFMs' structure and available edge servers result in varying GPU memory and compute resource requirements, which is typically formulated as an NP-hard mixed-integer programming problem, complicating the optimization of caching and inference policies. Moreover, distinct model architectures and computation requirements add complexity.

When the local resources of mobile devices and vehicles are not enough for executing PFMs, offloading these services to edge servers via radio access networks becomes an alternative solution for enabling AI services on edge servers with limited resources.

Context-Aware Caching and Inference Algorithms

Codesigning caching and inference algorithms considering contextual information in mobile AI services at edge servers is challenging due to the indirect correlation between model caching and inference duration. Joint policies need to optimize resource allocation according to each model and inference requests' specific requirements while considering user objectives, model size, usage frequency, and accuracy. By codesigning caching and inference algorithms considering the number of examples in context, as shown in Figure 4, edge servers can utilize extra computation resources to improve the accuracy of PFMs.

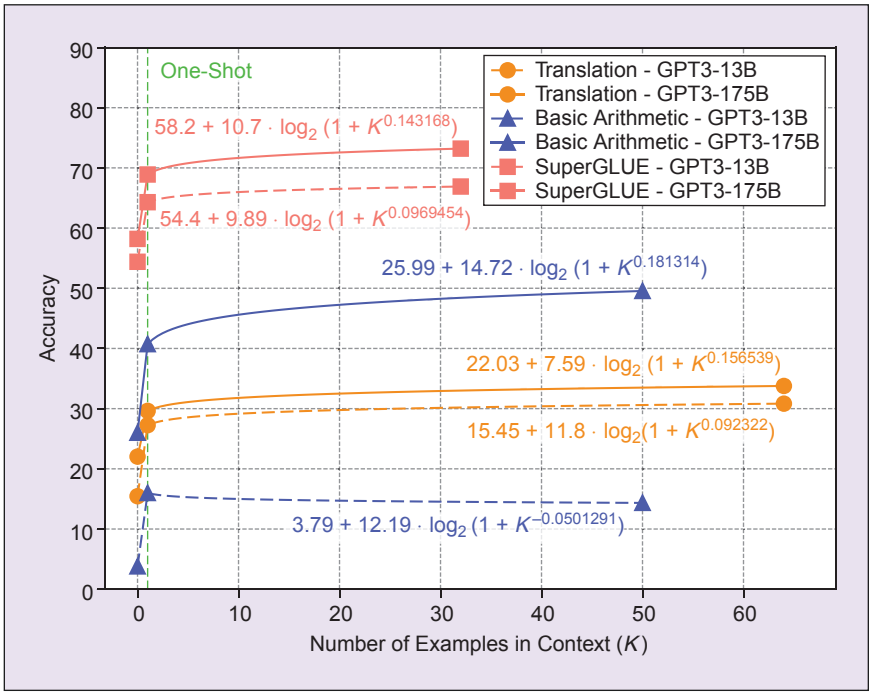


Figure 4 The accuracy in downstream tasks of GPT3-13B/175B versus number of examples in context. The few-shot accuracy $a_2 = a_0 + a_1 \log_2(1 + K^{\alpha})$, where a_0 is zero-shot accuracy, a_1 is one-shot accuracy, and α is coefficient.

Security and Privacy

PFM services encounter various security and privacy challenges. Data leakage can result from disclosing information about PFM's training data, while adversarial attacks can cause incorrect outputs. Moreover, PFMs might perpetuate biases inherent in the training data, which can have implications for privacy and fairness.

Use Case of Serving GPTs in Edge Intelligence for the Metaverse

Mobile AIGC Service Serving Model

We consider an intelligent transportation system in the metaverse system with a remote cloud center, an edge server, and multiple vehicles, serving different metaverse services, including autonomous driving, DTs, and AIGC-based XR, based on various PFMs. For instance, pedestrians and passengers can immerse themselves in the metaverse with XR by creating and interacting with AI-generated XR content synthesized by PFMs. When users do not have enough resources on their devices and onboard units for executing PFMs, they need to offload requests to edge servers or cloud servers for remote execution. Usually, an AIGC service requires multiple PFMs to work in synergy to satisfy the user's requirements in the metaverse. For example, the Stable Diffusion services consist of three types of PFMs [11], including a variational autoencoder that compresses images into a smaller dimensional latent space, a pretrained CLIP ViT-L/14 for conditioning, and a U-Net block that denoises the output from forward diffusion backward to obtain a latent representation.

The detailed parameters and performance of PFMs that need to be considered in intelligent transportation systems of the metaverse are listed in Table 1, including GPT3-13B [3], GPT3-175B [3], Uniformer-S [14], Uniformer-B [14], CLIP-ViT-L/14 [12], and CLIP-ViT-H/14 [12]. As we can observe, only LFMs are large enough to have in-context learning ability, while VFMs and MFMs are relatively small. As shown in Figure 4 and Table 1, PFMs utilize metagradients

TABLE 1 *Detailed parameters and performance of PFMs.*

						Model Performance Score		
	Models	Downstream Tasks	Model Size (M)	GFLOPs	K	Zero-Shot	One-Shot	Few-Shot
LFMs	GPT3-13B [3]	Translation	12,850	26.54	64	15.45	26.12	30.83
		Basic arithmetic			50	3.79	15.98	14.34
		SuperGLUE			32	54.4	64.3	66.9
	GPT-3-175B [3]	Translation	174,600	354.03	64	22.03	29.63	33.77
		Basic arithmetic			50	25.99	40.71	49.55
		SuperGLUE			32	58.2	68.9	73.2
VFMs	UniFormer-S [14]	Image classification	22	3.6	—	82.9	—	—
		Video classification	22	167		82.8	—	—
		Object detection and instance segmentation	41	269		45.6	—	—
		Semantic segmentation	25	247		46.6	—	—
		Pose estimation	25	4.7		74	—	—
	UniFormer-B [14]	Image classification	50	8.3	—	83.9	—	—
		Video classification	22	389		84	—	—
		Object detection and instance segmentation	69	399		47.4	—	—
		Semantic segmentation	54	471		48	—	—
		Pose estimation	54	9.2		75	—	—
MFMs	CLIP-ViT-L/14 [12]	Classification	428	175.5	—	75.20	—	—
		Image retrieval				71.08	—	—
		Text retrieval				84	—	—
	CLIP-ViT-H/14 [12]	Classification	986	381.9	—	77.97	—	—
		Image retrieval				73.43	—	—
		Text retrieval				86.04	—	—

K: number of examples in context.

to learn from context and improve performance as the user interacts with them. Then, the few-shot accuracy can be fit using the data in Table 1. Therefore, contextual information has a corresponding impact on the quality of service provided by AIGC, such as the accuracy of PFMs. Although the introduction of context in PFMs can improve the model performance, the size of the context window also affects the resource consumption and latency during the inference of the model. As shown in Figure 2, the freshness and relevance of the examples in demonstrations decrease over time until it is no longer pertinent to the current generation task, which is rarely measured in previous work.

AoC and LC Algorithm

Therefore, we propose the AoC for evaluating the relevance and freshness of examples in demonstrations that affect the quality of services of PFMs in currently executing downstream tasks. During inference of PFMs, the questions and answers can be recorded in the context windows as examples in demonstrations and instructions. These examples can be leveraged to improve the accuracy of PFMs as they can perform metagradient to fit these examples. However, the metagradient might have positive or negative effects on the accuracy, which depends on their quality, relevance, and timeliness. Similar to the age of information [8], the AoC indicates the relevance and timeliness of historical contextual examples in demonstrations to the cached PFM and the current inference task. As shown in Figure 2, the AoC follows the nonincreasing age utility function, factoring with a vanishing coefficient of context. Based on the AoC, the number of examples in content can be calculated as the weighted sum of the number of examples in demonstrations. Then, the accuracy of PFMs can be obtained by some function with respect to the number of examples in context as the functions demonstrated in Figure 4.

Finally, we propose the LC algorithm, based on the AoC, to manage PFMs for mobile AIGC services efficiently. The LC algorithm tracks the number of examples in context, calculating them and removing the cached PFM with the least contexts, i.e., number of examples in context, when GPU memory is needed for loading a new PFM. This approach is effective for large numbers of PFMs on edge servers with limited GPU memory, prioritizing the removal of the least-relevant PFM for the current inference task. Consequently, the accuracy of PFMs of mobile AIGC services is improved by leveraging more contextual information during inference.

In the experiment, we compare the proposed LC algorithm with random, cloud-only, first-in-first-out (FIFO), and least-frequently used (LFU) baselines. With the objective of minimizing service latency and accuracy loss, the system cost is calculated as the sum of the switching cost, the total accuracy cost, the edge inference latency, the edge offloading latency, and the cloud computing cost. As listed in Table 2, the performance of the proposed LC algorithm can achieve minimum total system cost while maintaining a high edge execution ratio, which indicates that most of the services are executed at edge servers. Especially, compared with the LFU algorithm, the LC algorithm can achieve a lower average service accuracy cost by efficiently leveraging the in-context learning ability of PFMs and contextual information.

***TABLE 2** Detailed system performance comparison for different caching algorithms.*

	Random	*Cloud*	*FIFO*	*LFU*	*LC*
System cost	25.67	7.29	27.51	5.93	**4.88**
Switching cost	18.72	0	23.28	.37	**.32**
Total accuracy cost	.13	0	.52	**0.36**	.44
Average accuracy cost	.0151	0	.0085	.0083	**.0076**
Inference latency	.12	0	1.30	1.32	**1.26**
Offloading latency	.04	0	0.35	**0.24**	.31
Cloud cost	6.63	7.29	**2.05**	3.63	2.52
Edge execution ratio	9.8%	0%	**70.7%**	49.4%	65%

Bold entries indicate the best performance in comparison.

Conclusions

In the article, we have studied edge caching and inference for serving PFMs in edge intelligence for the metaverse. We have proposed a joint model caching and inference framework for bringing the sparks of GPTs to mobile edge networks, toward achieving AGI. Specifically, we have proposed a new metric for evaluating the relevance and freshness of contextual examples and currently executing tasks. Furthermore, we have proposed the LC algorithm for cache replacement to improve the utilization of historical contextual information and thus to increase the accuracy of mobile AIGC services. The experimental results demonstrate that the LC algorithm can reduce system costs and improve the execution ratio at edge servers. In future work, we will extend this framework to a more general version for the evolving PFM services, such as the mixture -of-expert inference structure of GPT4.

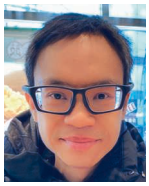
Acknowledgment

This work is supported by the National Science Foundation of China under Grants 62102099, U22A2054, and 62101594; the Pearl River Talent Recruitment Program under Grant 2021QN02S643; the Guangzhou Basic

Research Program under Grant 2023A04J1699; the National Research Foundation, Singapore; Infocomm Media Development Authority under its Future Communications Research & Development Programme; DSO National Laboratories under the Artificial Intelligence Singapore Programme (AISG Award AISG2-RP-2020-019); the Energy Research Test-Bed and Industry Partnership Funding Initiative; Energy Grid 2.0 Programme; DesCartes and the Campus for Research Excellence and Technological Enterprise (CREATE) Programme; and Ministry of Education Tier 1 (RG87/22). This work is partially supported by National Science Foundation CNS-2107216, CNS-2128368, CMMI-2222810, ECCS-2302469; the U.S. Department of Transportation; Toyota; and Amazon. S. Mao's work is supported in part by National Science Foundation Grant CNS-2148382. Jiawen Kang is the corresponding author of this article.

Author Information

Minrui Xu (minrui001@e.ntu.edu.sg) is currently working toward his Ph.D. degree in the School of Computer Science and Engineering, Nanyang Technological University, 639798 Singapore. He received the B.S. degree from Sun Yat Sen University in Guangzhou, China in 2021. His research interests include mobile edge computing, deep reinforcement learning, and incentive mechanism design.



Dusit Niyato (dniyato@ntu.edu.sg) is a professor in the School of Computer Science and Engineering, Nanyang Technological University, 639798 Singapore. He received the Ph.D. degree in electrical and computer engineering from the University of Manitoba in Canada in 2008.His research interests include the Internet of Things (IoT), machine learning, and incentive mechanism design.

Hongliang Zhang (hongliang.zhang92@gmail.com) is an assistant professor in the School of Electronics at Peking University, Beijing 100084, China. He was the recipient of the 2021 IEEE ComSoc Heinrich Hertz Award.

Jiawen Kang (kjwx886@163.com) is a professor at Guangdong University of Technology, Guangdong 510006, China. He received the Ph.D. degree from Guangdong University of Technology in China, and was a postdoc at Nanyang Technological University in Singapore. His research interests include blockchain, security, and privacy protection in wireless communications and networking.

Zehui Xiong (zehui_xiong@sutd.edu.sg) is an assistant professor at Singapore University of Technology and Design, Singapore 487372. He received the Ph.D. degree

in computer science and engineering at Nanyang Technical University in Singapore, where he was a researcher with the Alibaba Nanyang Technical University Joint Research Institute. His research interests include wireless communications, network games and economics, blockchain, and edge intelligence.

Shiwen Mao (smao@ieee.org) is a professor and Earle C. Williams Eminent Scholar, and director of the Wireless Engineering Research and Education Center at Auburn University, Auburn, AL 36849 USA. He received the Ph.D. in electrical and computer engineering from Polytechnic University in Brooklyn, NY, USA. His research interests include wireless networks and multimedia communications.

Zhu Han (zhuhan22@gmail.com) is a professor in the Electrical and Computer Engineering Department at the University of Houston, Houston, TX 77204 USA. He is an American Association for the Advancement of Science Fellow and received the IEEE Kiyo Tomiyasu Award in 2020.

References

[1] S. Bubeck et al., "Sparks of artificial general intelligence: Early experiments with GPT-4," Mar. 2023, *arXiv:2303.12712*.

[2] P. Zhou et al., "Vetaverse: Technologies, applications, and visions toward the intersection of metaverse, vehicles, and transportation systems," Oct. 2022, *arXiv:2210.15109*.

[3] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2020, vol. 33, pp. 1877–1901.

[4] R. Bommasani et al., "On the opportunities and risks of foundation models," 2021, *arXiv:2108.07258*.

[5] H. X. Qin and P. Hui, "Empowering the metaverse with generative AI: Survey and future directions," Hong Kong Univ. Sci. Technol., Guangzhou, China, 2023. [Online]. Available: https://www.researchgate.net/profile/Hua-Xuan-Qin/publication/370132434_Empowering_the_Metaverse_with_Generative_AI_Survey_and_Future_Directions/links/6442b55376364938df622b08/Empowering-the-Metaverse-with-Generative-AI-Survey-and-Future-Directions.pdf

[6] M. Xu et al., "Unleashing the power of edge-cloud generative ai in mobile networks: A survey of AIGC services," Mar. 2023, *arXiv:2303.16129*.

[7] M. Xu et al., "Generative AI-empowered simulation for autonomous driving in vehicular mixed reality metaverses," *IEEE J. Sel. Topics Signal Process.*, early access, Jul. 10, 2023, doi: 10.1109/JSTSP.2023.3293650.

[8] M. Xu et al., "Joint foundation model caching and inference of generative AI services for edge intelligence," May 2023, *arXiv:2305.12130*.

[9] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Prof. IEEE Conf. Comput. Commun.*, Honolulu, HI, USA, May 2018, pp. 207–215, doi: 10.1109/INFOCOM.2018.8485977.

[10] Q. Dong et al., "A survey for in-context learning," Jan. 2023, *arXiv:2301.00234*.

[11] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, New Orleans, LA, USA, Jun. 2022, pp. 10,684–10,695, doi: 10.1109/CVPR52688.2022.01042.

[12] M. Cherti et al., "Reproducible scaling laws for contrastive language-image learning," Dec. 2022, *arXiv:2212.07143*.

[13] N. Ding et al., "Parameter-efficient fine-tuning of large-scale pre-trained language models," *Nature Mach. Intell.*, vol. 5, no. 3, pp. 220–235, Mar. 2023, doi: 10.1038/s42256-023-00626-4.

[14] K. Li et al., "UniFormer: Unifying convolution and self-attention for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 10, pp. 12,581–12,600, Oct. 2023, doi: 10.1109/TPAMI.2023.3282631. VT