In: GLOBECOM 2025 - 2025 IEEE Global Communications Conference, 8-12 December 2025,
Taipei, Taiwan, pp. 4131-4136
DOI: 10.1109/GLOBECOM59602.2025.11431783

Building AI Service Repositories for On-Demand Service Orchestration in 6G AI-RAN

Yun Tang
yun.tang@cranfield.ac.uk

Mengbang Zou
M.Zou@cranfield.ac.uk

Udhaya Chandhar Srinivasan
u.srinivasan@cranfield.ac.uk

Obumneme Umealor
obumneme.umealor@cranfield.ac.uk

Dennis Kevogo
Dennis.Kevogo@cranfield.ac.uk

Benjamin James Scott
benjamin.scott@cranfield.ac.uk

Weisi Guo
weisi.guo@cranfield.ac.uk

***Abstract*—Efficient orchestration of AI services in 6G AI-RAN requires well-structured, ready-to-deploy AI service repositories combined with orchestration methods adaptive to diverse runtime contexts across radio access, edge, and cloud layers. Current literature lacks comprehensive frameworks for constructing such repositories and the proposed orchestrators generally over-simplify key orchestration factors compared to real edge computing environments. To fill these gaps, this paper systematically reviews and categorizes critical attributes influencing AI service orchestration in 6G AI-RAN and introduces an open-source, LLM-assisted toolchain that automates service packaging, deployment, and runtime profiling. We validate the proposed toolchain through the Cranfield AI Service repository case study, demonstrating significant automation benefits, reduced manual coding efforts by up to 98%, and the necessity of infrastructure-specific profiling, paving the way for more production-ready service orchestration and provisioning frameworks.**

***Index Terms*—AI Service Provisioning, 6G, AI-RAN**

I. Introduction

AI-RAN, particularly the AI-on-RAN paradigm, aims to support a wide array of connected use cases, from autonomous vehicles to smart cities, by providing on-demand, context-aware AI services that adapt to user needs in real-time. However, we identify significant bottlenecks towards effective orchestration of on-demand AI services in real-world AI-RAN settings. Specifically, existing research has primarily concentrated on developing service orchestration and task scheduling algorithms, often employing heuristic methods [1], reinforcement learning (RL) [2], or large language models (LLMs) [3]. While these approaches have yielded promising results, they often consider a limited set of selected factors and AI tasks, which may not fully capture the complexity and flexibility of real-world scenarios.

Addressing these limitations requires a comprehensive understanding of the numerous factors that influence AI service orchestration in 6G networks. However, the current literature lacks a systematic compilation and analysis of these factors, creating a gap that hinders the advancement of effective AI service orchestration strategies.

To bridge this gap, our work thoroughly reviews the literature to identify and categorize the factors pertinent to AI service orchestration in 6G networks. Building upon this, we introduce an open-source LLM-assisted toolchain designed to assist AI service providers in constructing in-house AI service repositories. This toolchain automates the extraction of relevant factors and facilitates the packaging of AI services, thereby streamlining the deployment process within the 6G AI-RAN environment. As a proof of concept, we have constructed the Cranfield AI Service repository, incorporating a diverse range of AI services across multiple task categories. This repository leverages open pre-trained models from platforms like HuggingFace, demonstrating the practicality and efficiency of our approach. Our specific contributions are:

- A comprehensive review of AI service orchestration attributes crucial for 6G AI-RAN environments.
- An open-sourced, LLM-assisted toolchain for automated AI service repository construction, including model packaging and attribute profiling.
- Empirical validation through a case study demonstrating reduced manual coding (up to 98%) and highlighting the necessity of infrastructure-aware orchestrator algorithm design.

II. AI Service Orchestration Attributes in 6G

Effective orchestration of AI services in RAN requires a comprehensive understanding of various attributes that influence service selection, performance, resource allocation, and user satisfaction. We summarize these key attributes considered in the literature in Table I and highlight their significance in the orchestration process.

A. Functionality Attributes

Task Category and Detail: Specific functions (e.g., image classification or text analysis) the AI models are trained to perform, the expected input and generated output, and exemplar source code on how the AI model can be invoked. Accurately categorizing and describing the tasks ensures that services

The authors are with the Faculty of Engineering and Applied Sciences, Cranfield University, United Kingdom. The work is supported by EPSRC CHEDDAR: Communications Hub for Empowering Distributed clouD computing Applications and Research (EP/X040518/1) (EP/Y037421/1).
Corresponding Author: Yun Tang.

Published by IEEE. This is the Author Accepted Manuscript issued with: Creative Commons Attribution License (CC:BY 4.0).
The final published version (version of record) is available online at DOI:10.1109/GLOBECOM59602.2025.11431783. Please refer to any applicable publisher terms of use.

TABLE I
SUMMARY OF AI SERVICE ORCHESTRATION ATTRIBUTES IDENTIFIED FROM THE LITERATURE.

Category	Factor	Definition	References
Functionality	Task Category	High-level tasks the service performs, such as image classification, object detection, depth estimation, text classification etc.	-
	Task Detail	Detailed description of the expected input and output of the AI service.	-
	Accuracy	The accuracy of the AI model in relevant benchmarks.	[4], [5]
Resource	CPU Usage	The amount of CPU operations required per AI service execution.	[2], [4], [6], [7]
	CPU RAM Usage	The volume of CPU memory utilized by the AI service during the execution.	[8]
	DEVICE Usage	The amount of accelerator (e.g., GPU) operations required per execution.	[4], [6]
	DEVICE RAM Usage	The volume of accelerator memory required during the execution	[8]–[10]
	Disk I/O	The volume of the storage disk I/O during the AI service lifecycle.	[8]
	Service Storage Size	The disk space required to store the AI model or Docker image.	[7], [9]
	Energy Consumption	The amount of energy consumed by the AI service (idle and execution).	[1], [4]–[8], [10]–[12]
	Input Data Size	The data size required per AI service execution.	[1], [2], [7], [13]
	Output Data Size	The data size generated as output by the AI service	[2]
Latency	Inference Time	The time taken by the AI service to process one request.	[4]–[8], [10], [13]
	Initialization Time	The time the AI service takes to boot up before it is ready for user requests.	[9], [13]
	Eviction Time	The time taken to undeploy the AI service.	[8]
Flexibility	Cooperative Inference	Whether the AI service supports cooperative inference across multiple devices or nodes, accepting or outputting intermediate layer features.	[10], [11], [14], [15]
Trustworthiness	Feedback	Number of Likes and Dislikes, as well as general comments by end users.	[16]
	Explainable AI (XAI) Support	The degree to which the AI service supports integration with XAI methods, facilitating interpretability of its decisions and outputs.	[17]
Billing	Initialization Cost	The amount of credit required to deploy and initialize the AI service	-
	Keep-Alive Cost	The amount of credit cost per time unit to keep the AI service alive listening for user requests.	-
	Execution Cost	The amount of credit cost per AI service execution.	-

are precisely matched to the use case requirements, especially when queried using semantic searching in vector databases[1][2].

Accuracy: AI model performance against established benchmarks. High accuracy is crucial for applications where precision is paramount, while resource-constrained scenarios may prioritize less performant models for their reduced resource footprint and faster response time.

B. Resource Attributes

CPU and DEVICE Usage: The required computational load, i.e., the amount of CPU and accelerator device (e.g., GPU) operations such as floating-point operations [2], "frequency unit" [6] or multiply-and-accumulate operations [4]. While determining the accurate number of processor operations might be challenging, these metrics can be replaced by CPU or DEVICE time when profiled on edge or cloud nodes with known CPU/DEVICE frequency.

[1]https://www.trychroma.com/

[2]https://www.mongodb.com/products/platform/atlas-vector-search

CPU and DEVICE RAM Usage: The CPU and accelerator device memory needed, which may limit the deployment location of large AI models. Worth noting is that the memory profile also varies due to the size of the request data during service execution.

Disk I/O: The read and write (I/O) data volume on the hard disk, which may incur variable and non-trivial delay when the edge node has limited Disk I/O bandwidth.

Service Storage Size: The storage space required to cache the AI service (e.g., docker image) locally, which may constrain the deployment of large AI models or incur non-trivial download time during initialization. Worth noting is that the size of the AI service image may only reflect the dependencies of the AI services but not the AI model itself as the model parameters can be downloaded and stored in memory during service initialization.

Energy Consumption: The energy consumption when it is pending requests and performing inference. This metric has been a major factor under consideration in many works especially when there are battery-powered computation nodes

or the energy is costly.

Input and Output Data Size: The data throughput by AI services, which may affect deployment location or cooperative inference (or semantic communication) arrangements under bandwidth limitations.

C. Latency Attributes

Inference Time: Either the inference time by the AI model or, more practically, the response time by the AI service to client requests. Like energy consumption, inference time is another major factor considered by the service orchestrators for its direct impact on the Quality-of-Service (QoS). Worth noting is that the inference time varies with the hardware infrastructure ($\propto$ CPU/DEVICE usage divided by CPU/DEVICE frequency) and the current hardware load (when multiple AI services compete for hardware scheduling priority).

Initialization and Eviction Time: The AI service boot-up and termination time. These attributes are often overlooked [9] but may pose a significant impact under scenarios with rapidly changing workloads. Similar to inference time, these metrics vary with the infrastructure such as network and memory bandwidth.

D. Flexibility Attributes

Cooperative Inference: Whether the inference process can be decomposed into subprocesses and distributed across multiple nodes. Such flexibility enables semantic communication which is beneficial upon limited bandwidth or stringent latency requirements.

E. Trustworthiness Attributes

Feedback: User ratings and comments provide valuable insights into service reliability and effectiveness, which can 1) influence the AI service selection and orchestration strategies, especially for LLM-assisted agents, and 2) inform future improvement and deployment strategies.

Explainable AI (XAI) Support: A list of supported XAI techniques optional as add-ons. Services that offer transparency and explainability in their decision-making processes build user trust and facilitate easier debugging and compliance with regulatory standards.

F. Billing Attributes

Initialization, Keep-Alive, and Execution Costs: Vital for budgeting and cost management, influencing decisions on service selection and deployment.

By systematically evaluating these attributes, use case innovators and AI service orchestrators can make informed decisions regarding resource allocation, deployment strategies, and service optimization, especially in the LLM-based service provisioning era [3].

III. LLM-assisted Toolchain for AI Service Repositories in AI-RAN

To facilitate the collection of attributes, we have developed an open-source[3], LLM-assisted toolchain designed explicitly for AI service providers in 6G ecosystems. This toolchain facilitates automated packing, functionality validation and attribute profiling of AI services, aiming at reducing the complexity and time required for building deployment-ready service repositories.

A. Toolchain Overview

Fig. 1 illustrates our proposed LLM-assisted toolchain pipeline, comprising three phases: **Service Compilation** phase collects and prepares AI service artefacts, including documentation parsing, service code generation, and service image compilation; **Runtime Validation** phase validates the AI service API endpoints and profiles runtime attributes such as resource usage and latency; and lastly, **Repository Update** phase updates the AI service repository and database with the generated artefacts and information.

B. Service Compilation Phase

This phase translates user-selected AI models into deployable AI services with standard service interfaces (e.g., RESTful endpoints):

1) Manual Model Selection: Initially, AI service providers manually select pre-trained AI models of diverse task categories, typically sourced from public repositories such as HuggingFace.

2) Automated Metadata Extraction: The model documentation and metadata to identify and categorize task-related information are crawled and parsed automatically, including *task category* and *task detail*.

3) LLM-Assisted Service Code Generation: Leveraging large language models (LLMs), the service code is generated automatically based on the AI model documentation and code samples, including:

- `model.py`: RESTful API endpoints encapsulating model inference and runtime profiling endpoints, to be imported by the main AI service application.
- `Dockerfile`: Containerization instructions install dependencies and package the AI service.
- `client.py`: Code examples for requesting AI services.
- Revised Task Detail: Enhanced task description for improved semantic searchability in vector databases, tailored particularly for those LLM agent-powered orchestrators [3]).
- Accuracy Information: Performance benchmarks extracted from model documentation if available.

To reduce the probability of hallucinations and cost, LLM agents are not required to generate the AI service from scratch; instead, all AI services share a common code base, providing utilities such as the main API server application (`server.py`), service health

[3]https://github.com/Cranfield-GDP/edge-ai-service-wrapper

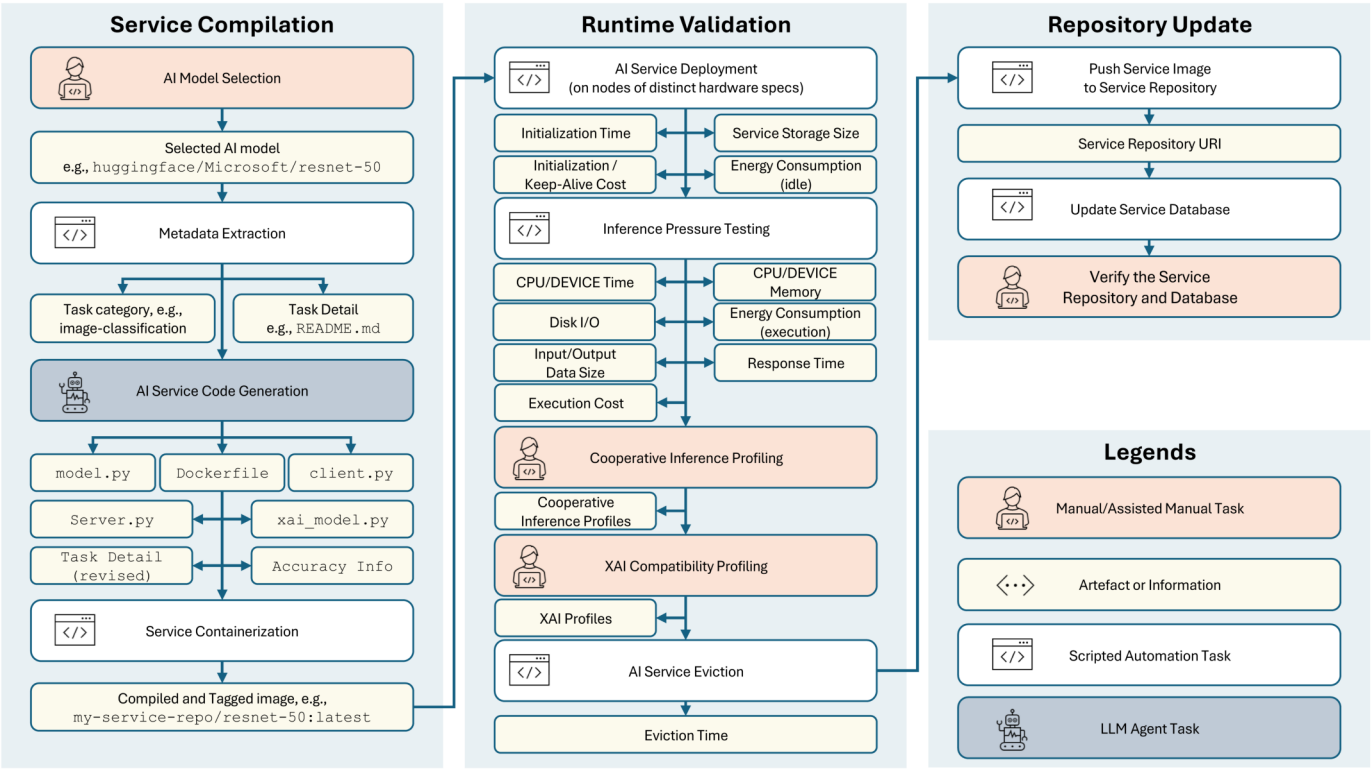

Fig. 1. LLM-assisted AI service repository toolchain overview.

checking (`healthcheck.py`), and explainability add-ons (`xai_model.py`) etc.

4) Automated Service Containerization: The generated service code and dependencies are automatically compiled into a containerized deployment-ready (Docker) image, properly tagged for repository management, such as `my-service-repo/resnet-50:latest`.

C. Runtime Validation Phase

Following service compilation, this phase rigorously evaluates the deployability, performance, and operational efficiency of the AI services under realistic network conditions, ensuring that critical runtime attributes are systematically measured, verified, and documented.

5) Automated Service Deployment: The containerized AI service image is automatically deployed on each edge or cloud node of distinct hardware specs. Upon container instantiation, the following runtime metrics can be captured: *initialization time*, *initialization cost*, *service storage size*, *keep-alive cost* and *idle energy consumption*.

6) Automated Pressure Testing: Following the deployment, pressure testing is performed using the LLM-generated client scripts to profile service performance and resource utilization at full workload. During this process, detailed runtime attributes can be collected, including: *CPU/DEVICE usage*, *CPU and DEVICE RAM usage*, *disk I/O*, *active energy consumption*, *input and output data size*, *inference time* and *execution cost*.

7) Assisted Manual Cooperative Inference Validation: This step verifies the AI service's capability to distribute the inference process across multiple nodes, vital for scenarios demanding semantic communication or real-time latency reduction under constrained network conditions. Results are documented as a list of cooperative inference profiles, each consisting of the start and end layer to be executed on the service side and the corresponding runtime resource metrics discussed above.

8) Assisted Manual XAI Compatibility Validation: The AI service is then tested against available XAI techniques, which is essential for debugging, user trust and regulatory compliance. The results are documented as a list of XAI technique profiles, each consisting of the name of the XAI technique as well as the corresponding runtime resource metrics discussed above.

9) Automated Service Eviction: Finally, the AI service is un-deployed from the hosting nodes, and the attribute *eviction time* is measured.

D. Repository Update Phase

10) After comprehensive validation, the AI service images are pushed to a service repository, each identified by a unique URI. Then, the service database is populated with detailed metadata, including all essential attributes collected during the service compilation and runtime validation phase. To maintain quality and trust, the AI service provider can conduct a final verification step, ensuring that the saved service knowledge is accurate, complete, and consistent.

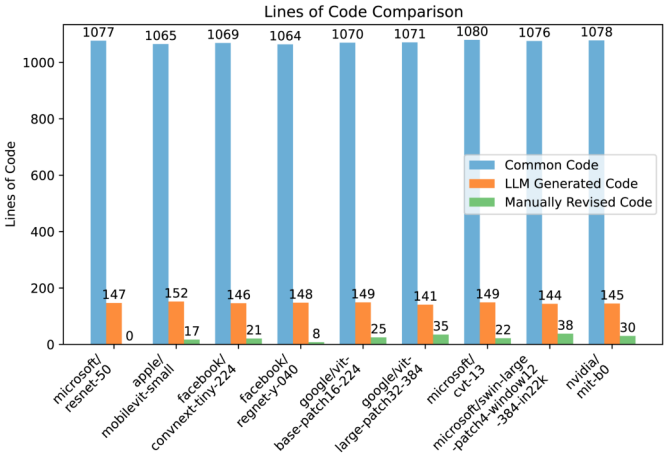

Fig. 2. Lines of code comparison for the first nine AI services (for image classification) in Cranfield AI Service Repository.

IV. Case Study: Cranfield AI Service Repository

To validate the practicality and effectiveness of our proposed toolchain, we implemented a proof-of-concept AI Service repository hosted at Cranfield University[4]. This repository currently integrates **more than 30** AI services performing **12 different tasks** using publicly available pre-trained models from HuggingFace. New AI services can be added within minutes.

A. AI Service Compilation Automation

Fig. 2 compares the lines of code on the common code base (which varies mainly due to formatting differences), LLM agent-generated AI model-specific code base, and human developer adjustments.[5]

Observation Thanks to the mature software engineering practices, a significant portion of the code base can be shared across AI services hosting different AI models. LLM-based agents are only required to fill those small model-specific pieces, effectively minimising the cost and probability of hallucinations. On average, human effort can be reduced by 86% **(without LLM) to** 98% **(from scratch)**. However, manual code revisions are still demanded in the following scenarios:

- to upgrade obsolete APIs due to obsolete code samples in the AI model README.
- to fix zero-day dependency breaks [6] (i.e., compilations may succeed today but fail tomorrow);
- to select the best neural network layer for XAI or cooperative inference[7]. After all, XAI techniques shall produce stakeholder-centred (e.g., AI service provider, AI service user, and legal auditor) explanations and keep the stakeholders in the loop.
- to update the code template and engineer prompts upon emerging model functionalities. For example, the Yolov8 object recognition model series offers a bounding box visualization API, while others don't.
- to fix LLM hallucinations, such as incorrect interpretation of abbreviations from README files.

Takeaway 1: Challenge of Full Automation

Manual effort (or stakeholder involvement in general) can be minimized utilizing such LLM-powered toolchains but may never be eliminated, because 1) most AI models are manually designed and trained, 2) most AI service dependencies are manually maintained and 3) trustworthiness is stakeholder-specific.

B. AI Service Resource Profiling

From the runtime resource profiling results in Fig 3, we learn that the edge nodes of different hardware and network infrastructures can produce significantly different AI service runtime profiles. Worth noting is that two XAI techniques (ScoreCAM and AblationCAM) are not even supported due to memory shortage on the laptop testbed.

Takeaway 2: Infra-specific AI Service Profile

Most literature assumes the latency, resource profile and the task (e.g., compatible XAI techniques) of the orchestrated AI services are infrastructure-agnostic while in reality, they vary with different hardware specs and real-time hardware load. The lack of such consideration renders many proposed service orchestrators ineffective in real settings.

V. Conclusions and Future Works

Conclusions We presented an LLM-assisted toolchain that automates AI service packaging and profiling for effective orchestration in 6G AI-RAN environments. The empirical case study on building the Cranfield AI Service Repository illustrated substantial manual effort reductions up to 98% and underscored the importance of infrastructure-specific profiling.

Future Works we cordially invite the community to join our endeavour in making the toolchain truly beneficial to the telecom ecosystem. Specifically, we believe the toolchain could benefit significantly in 1) supporting metrics or techniques to evaluate or enhance other essential trustworthy metrics than explainability, such as safety, security, robustness, privacy, and bias; 2) supporting AI training services, particularly federated learning which is widely adopted in 6G RAN and use cases; 3) supporting multi-AI pipelines incorporating multiple AI models for complex use cases; 4) supporting AI service transfer learning techniques to adapt pre-trained AI models to new use cases; and 5) supporting AI model quantization for different hardware and network infrastructures.

[4]https://hub.docker.com/repositories/cranfield6g

[5]The pipeline has been developed using the model *microsoft/resnet-50* as a reference, hence zero manual revisions counted for the model.

[6]https://github.com/huggingface/transformers/issues/37326

[7]https://jacobgil.github.io/pytorch-gradcam-book/HuggingFace.html

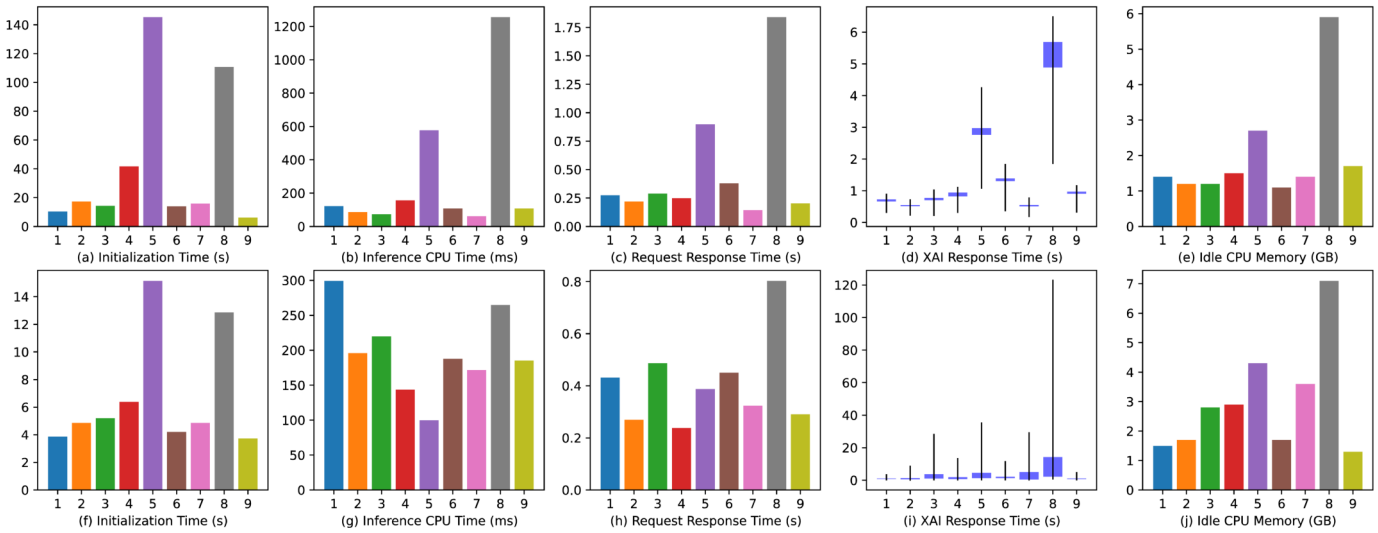

Fig. 3. Resource profile results of the first nine AI services. (a-e) are profiled on a CPU-only laptop (600Mb/s download) and (f-j) on an RTX A4000 GPU-powered workstation (900Mb/s). The candlestick plots for XAI response time are generated from all compatible XAI techniques for each AI service.

References

[1] T. Song, W. Hu, X. Tan, J. Ren, S. Wang, and S. Xu, "Fast-ram: A fast ai-assistant solution for task offloading and resource allocation in mec," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[2] X. Song, L. Liu, J. Fu, X. Zhang, J. Feng, and Q. Pei, "A reinforcement learning-based dag tasks scheduling in edge-cloud collaboration systems," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 1771–1776.

[3] Y. Tang, U. C. Srinivasan, B. J. Scott, O. Umealor, D. Kevogo, and W. Guo, "End-to-end edge ai service provisioning framework in 6g oran," *arXiv preprint arXiv:2503.11933*, 2025.

[4] G. Pan, H. Zhang, S. Xu, S. Zhang, and X. Chen, "Joint optimization of dnn inference delay and energy under accuracy constraints for ar applications," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 2230–2235.

[5] H. Liu, H. Fu, S. Yuan, C. Wu, Y. Luo, and J. Li, "Adaptive processing for video streaming with energy constraint: A multi-agent reinforcement learning method," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 122–127.

[6] Z. Zhou, J. Feng, L. Liu, J. Wang, and Q. Pei, "Heterogeneous computation and resource allocation for wireless edge ai," in *GLOBECOM 2024-2024 IEEE Global Communications Conference*. IEEE, 2024, pp. 2641–2646.

[7] Z. Lin, S. Bi, and Y.-J. A. Zhang, "Optimizing ai service placement and computation offloading in mobile edge intelligence systems," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–7.

[8] J. Luo, B. Tang, and J. Zhang, "Container scheduling in hybrid cloud-edge collaborative system," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 5662–5667.

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

[10] L. Zhang, J. Wu, S. Mumtaz, J. Li, H. Gacanin, and J. J. Rodrigues, "Edge-to-edge cooperative artificial intelligence in smart cities with on-demand learning offloading," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[11] M. Tian, Z. Liu, C. Qiu, X. Wang, D. Niyato, and V. C. Leung, "Enabling collaborative and green generative ai inference in edge networks," in *GLOBECOM 2024-2024 IEEE Global Communications Conference*. IEEE, 2024, pp. 3709–3714.

[12] J. Heydari, V. Ganapathy, and M. Shah, "Dynamic task offloading in multi-agent mobile edge computing networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[13] D. Cao, Y. Wang, Y. Yang, and S. He, "Joint optimization of multi-type caching placement and multi-user computation offloading for vehicular edge computing," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 5593–5598.

[14] R.-J. Wang, C.-H. Wang, D.-N. Yang, G.-S. Le, and W.-T. Chen, "Social-inspired multicast feature selections with mobile edge computing," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 2287–2292.

[15] Q. Zeng, Z. Wang, Y. Zhou, H. Wu, L. Yang, and K. Huang, "Ultra-low-latency feature transmission for edge inference," in *GLOBECOM 2024-2024 IEEE Global Communications Conference*. IEEE, 2024, pp. 3128–3133.

[16] W. Ni, J. Tong, and L. Fu, "Online resource allocation for user experience improvement in heterogeneous mec systems," in *GLOBECOM 2024-2024 IEEE Global Communications Conference*. IEEE, 2024, pp. 3419–3424.

[17] J. Dutta, D. Puthal, and C. Y. Yeun, "Next generation healthcare with explainable ai: Iomt-edge-cloud based advanced ehealth," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. Ieee, 2023, pp. 7327–7332.

https://dspace.org/

Building AI service repositories for on-demand service orchestration in 6G AI-RAN

Tang, Yun; Zou, Mengbang; Srinivasan, Udhaya Chandhar; Umealor, Obumneme; Kevogo, Dennis ...

2025-12-08
Institute of Electrical and Electronics Engineers (IEEE)
https://doi.org/10.1109/globecom59602.2025.11431783
Tang Y, Zou M, Srinivasan UC, et al., (2025) Building AI service repositories for on-demand service orchestration in 6G AI-RAN. In: Proceedings of the GLOBECOM 2025 - 2025 IEEE Global Communications Conference, 8-12 Dec 2025, Taipei, Taiwan, pp. pp. 4131-4136

Downloaded from a DSpace repository. https://dspace.org/