This article has been accepted for publication in IEEE Network. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MNET.2024.3411772

Open-Source Edge AI for 6G Wireless Networks

Liqiang Zhao, Yunfeng Wang, Xiaoli Chu, Shenghui Song, Yansha Deng, Arumugam Nallanathan, George K. Karagiannidis

***Abstract*—Multi-access Edge Computing (MEC) has been recognized as a key enabler for next-generation networks in supporting a large variety of compelling applications with challenging requirements. With its widely proved strength and successes, AI has to become an integral part of MEC. In this paper, we present a novel open-source edge AI (OpenEAI) framework that introduces a native AI plane into the recently proposed open-source MEC framework. The AI plane is designed based on two principles: decoupling the edge AI services into independent AI functions; and recomposing the independent edge AI functions into customized OpenEAI instances based on users' specific requirements. Typical use cases of OpenEAI are characterized with the aid of a small-scale test network. Finally, we discuss the opportunities and challenges facing OpenEAI.**

***Index Terms*—Open Source, Multi-access Edge Computing, Artificial Intelligence, Edge AI, 6G**

I. Introduction

Mobile Internet traffic continues to escalate, driven by the rapid development of the internet of things (IoTs) and various attractive mobile applications in the vertical industries [1]. However, the conventional network architecture built upon specialized hardware and software may fail to keep up with the rapidly evolving deployment scenarios, service requirements and technologies. To circumvent this impediment, an open-source cellular network was proposed in [2] to allow operators to adaptively customize their networks based on users' specific needs.

Open-source cellular networks rely on the integration of next-generation (NG) technologies, such as network function virtualization, software-defined networking [3], multi-access edge computing (MEC) [4], etc. Among these technologies, open-source MEC serves as the core driver for cell-edge performance enhancement, and supports customized radio access networks (RANs), such as the IoT, and Ultra Reliable Low Latency Communications (URLLC), etc [5].

The Open-RAN (O-RAN) alliance promotes the development of open-source cellular networks and MEC that build on virtualized network elements, white-box hardware and standardized interfaces [6]. Flexibility and customization constitute the two basic requirements of open-source wireless networking. However, the heterogeneous NG networks rely on diverse resources, such that conventional network management methods based on human decisions become inadequate. Accordingly, O-RAN has adopted artificial intelligence (AI) as one of its core technologies [7] in support of NG networking capabilities by leveraging a universal infrastructure, open network architectures, open-source software/hardware, and other state-of-the-art technologies [8]. The AI enhancement of O-RAN has focused on the Intelligent RAN Controller (IRC) [9]. However, open-source MEC (OpenMEC) has not been widely studied in the context of O-RAN. In our previous treatise [2], OpenMEC was presented. Specifically, the monolithic MEC services were decomposed into multiple independent MEC functions, while the communication, caching and computing resources in the OpenMEC system were decoupled from the dedicated hardware and abstracted into a resource pool. Then, the MEC functions were recomposed to provide customized MEC services, and the resources in the resource pool were allocated to the recomposed MEC functions as needed. But AI was not embedded in the design of OpenMEC.

In [10], the deployment of monolithic AI services on the open-source MEC platforms was investigated, and an edge framework was proposed, where the 'containerized' AI and edge services run on the virtual machines in the MEC servers. Yang *et al.* [11] proposed an AI-enabled intelligent architecture of four layers, i.e., the intelligent sensing layer, data mining and analytics layer, intelligent control layer and smart application layer, to support diverse service requirements in 6G edge networks. A distributed edge AI model was studied in [12], where smart video surveillance services were decoupled into a set of virtual functions to create virtual function chains for processing video streaming data. We note that in the above works, there is a lack of a unified orchestration mechanism for the decoupling and recomposing of both the AI resources and functions to provide customized AI services for users.

To fill this research gap, we present the new paradigm of Open-Source Edge AI (OpenEAI) that leverages the advantages of open-source software/hardware to decouple the AI functions (AIFs) from the underlying physical network and

This work was supported in part by the National Key R&D Program of China (2020YFB1807700), Key Research and Development Program of Shaanxi under Grant 2022KWZ-09, Postdoctoral Fellowship Program of CPSF under Grant GZC20232058, Fundamental Research Funds for the Central Universities under Grant ZYTS24110, Postdoctoral Research Program of Shaanxi Province under Grant 2023BSHYDZZ100, Key-Area Research and Development Program of Guangdong Province (2020B0101120003).

Liqiang Zhao is with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China, and Guangzhou Institute of Technology, Xidian University, Guangzhou 510100, China (e-mail: lqzhao@mail.edu.xidian.cn).

Yunfeng Wang is with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China (e-mail: yunfengwang@stu.xidian.edu.cn).

Xiaoli Chu is with the Department of Electronic and Electrical Engineering, The University of Sheffield S10 2TN, U.K. (e-mail: x.chu@sheffield.ac.uk).

Shenghui Song is with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: eeshsong@ust.hk).

Yansha Deng is with the Department of Engineering, King's College London, WC2R 2LS London, U.K. (e-mail: yansha.deng@kcl.ac.uk).

Arumugam Nallanathan is with the School of Electronic Engineering and Computer Science, Queen Mary University of London, E1 4NS London, U.K. (e-mail: a.nallanathan@qmul.ac.uk).

G. K. Karagiannidis is with Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Greece and also with Artificial Intelligence and Cyber Systems Research Center, Lebanese American University (LAU), Lebanon (e-mail: geokarag@auth.gr).

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on September 23,2024 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Network. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MNET.2024.3411772

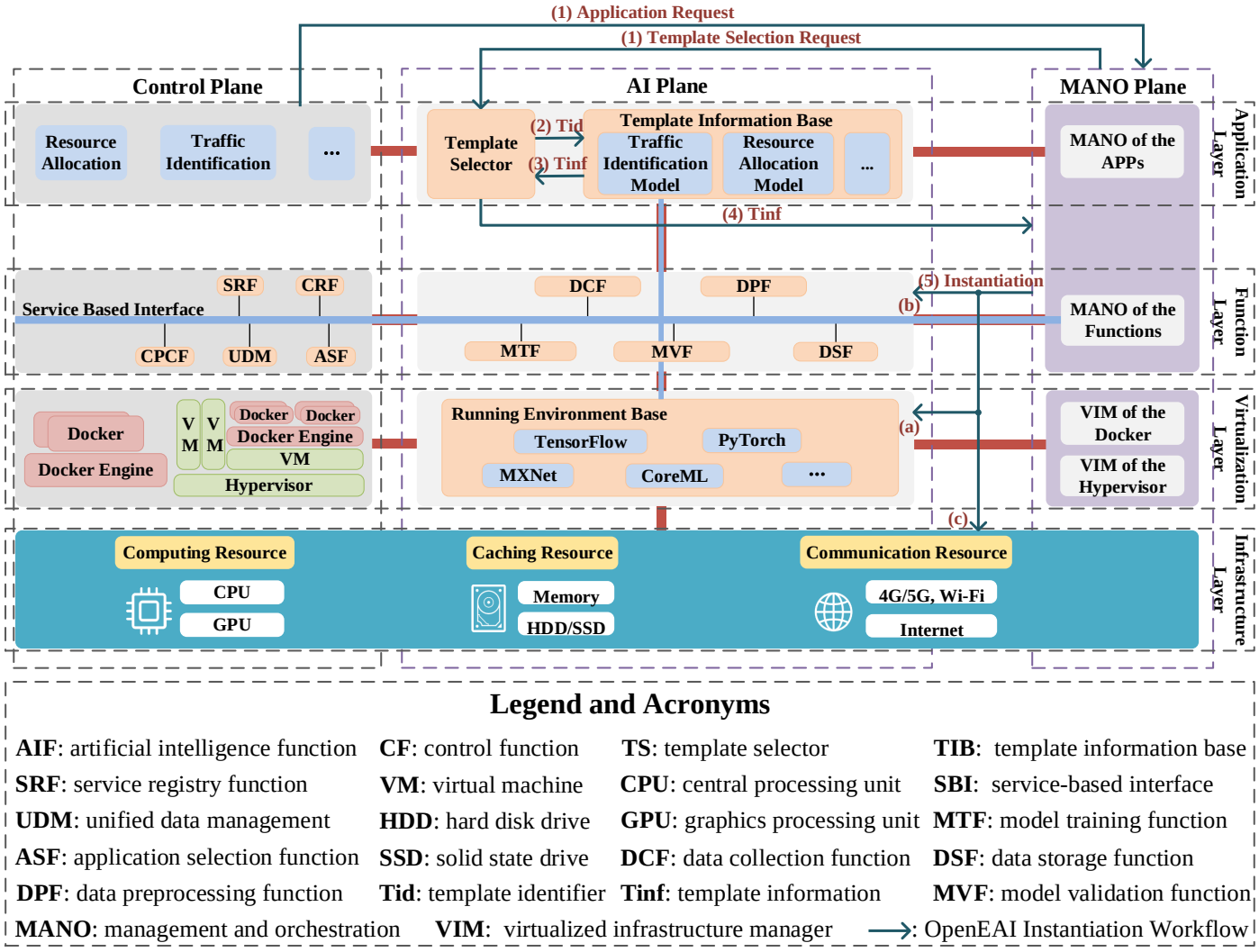

Fig. 1: OpenEAI Architecture

enable AIF reconfiguration, thus realizing native AI in MEC. Specially, open-source software/hardware allows for the establishment of an OpenEAI system, where the AI services are separated from the underlying physical network and they are further decoupled into independent AIFs. The independent AIFs can be recomposed as needed to provide multifarious AI services based upon the common physical networks. The contributions of this paper are summarized as follows. Firstly, we introduce a native AI plane into the previously proposed OpenMEC [2] and present the new framework of OpenEAI. This is different from 5G networks where AI is typically an additional function or application (APP) [13]. Secondly, we discuss in detail the key technologies of OpenEAI, including the microservice-based function layer, and the template as well as instantiation scheme. More specifically, the MEC and AI services in the function layer are decoupled into independent control functions (CFs) and AIFs, respectively. Then a unified stateless hypertext transfer protocol-aided service-based interface (SBI) connects the functions together to ensure that they can communicate with one another when needed. Thirdly, several typical use cases are investigated in the test network to validate the flexibility and latency of OpenEAI.

II. OpenEAI Architecture

To extend the associated intelligent control in the access domain, such as the IRC in O-RAN, to the edge domain, we propose the OpenEAI framework by introducing a native AI plane into OpenMEC. To provide customized edge AI, the tightly coupled AI resources and functions of the conventional AI-enabled networks have to be appropriately decoupled, so that the decoupled AI resources and functions can be reassembled dynamically according to the users' specific requirements. The OpenEAI architecture of Fig. 1 consists of four horizontal layers and three vertical planes, i.e., the infrastructure, virtualization, function, as well as application layers, and the control, AI, as well as the management and orchestration (MANO) planes. In the following, we describe in detail the four horizontal layers and the three vertical planes.

A. The four horizontal layers

The *infrastructure layer*, at the bottom of Fig. 1, contains all the resources of the system, including computing resources such as general purpose CPU/GPUs; caching resources; and communication resources such as NG nodeB.

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on September 23,2024 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Network. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/MNET.2024.3411772

The $virtualization\ layer$, which is right above the infrastructure layer in Fig. 1, decouples the underlying resources from the hardware and abstracts them into a resource pool, where the communication resource (such as bandwidth), caching resource (such as SSD) and computing resource (such as CPU) are divided into logical resource groups. Upon receiving the application requests, the required AIFs are invoked, while the logical resource groups are re-organized as necessary for use by the AIFs invoked and allocated to the AIFs to optimize their operations.

On top of the virtualization layer of Fig. 1 lies the $function\ layer$. In contrast to the conventional monolithic MEC system, the function layer of OpenEAI contains a unified SBI and diverse CFs and AIFs. For instance, the MEC services are decoupled into independent CFs (e.g., Communication Protocol Conversion Function (CPCF) and CF Repository Function (CRF) [2]). Furthermore, the user plane function (UPF) is sunk from the 5G core network to OpenEAI. The CFs and AIFs can be recomposed, activated and released in a timely fashion based on the service requirements.

The $application\ layer$ is the top layer of Fig. 1 and contains various APPs for end-user support. Once called by users, the APPs will automatically call the associated CFs and AIFs at the function layer and Dockers at the virtualization layer.

B. The three vertical planes

As the core of OpenEAI, the vertical $AI\ plane$ of Fig. 1 sinks the model training and real-time reasoning processes of the AI algorithm from the cloud to the edge, thereby reducing the transmission delay of training data. The AI plane involves all the four horizontal layers of OpenEAI. Its application layer consists of the template selector (TS) and of the template information base (TIB). Its function layer builds on the idea of micro-services and decouples the AI service into several AIFs as well as a SBI. Its virtualization layer contains the environment library, where the running environment can be selected according to the current resource status, to the characteristics of the network applications, and to the users' service requests. The running environment is essentially the programming language used by the virtualization layer for abstracting the underlying resources. The infrastructure layer provides the resources required for AI services.

The vertical $control\ plane$ stems from software-defined networking that decouples the control signaling from the data transmission. It is mainly responsible for the control data processing and transmission from the infrastructure layer up to the application layer of Fig. 1.

The vertical $MANO\ plane$ is composed of the virtualized infrastructure manager (VIM) at the virtualization layer, the MANO of the functions at the function layer and the MANO of the APPs at the application layer. The MANO plane transforms the service requests from the control plane into the MANO's commands, which contain both the required runtime environment, the functions and resources of the service. The VIM manages the virtualized resources according to the MANO's commands, so as to ensure that the appropriate computing, caching and communication resources are supplied for the upper layers of Fig. 1. The MANO plane is responsible for managing and orchestrating the AIFs and APPs, as well as scheduling the VIM to allocate resources for supporting the MEC reconfiguration, where the OpenEAI's template and instantiation scheme are used for flexibly implementing the MEC reconfiguration per service requirement.

III. Native AI in OpenEAI

In the OpenEAI framework, native AI is realized by the AI plane decoupling and recomposing. Specifically, the monolithic AI services are decoupled into independent functional modules (i.e., AIFs), each of which has its own complete business rationale and concentrates on a specific task, such as raw data collection, AI model training, etc. Any of the AIFs is able to communicate with other AIFs via the unified SBI based on the hypertext transfer protocol, and can be invoked to cooperate with other AIFs to provide customized AI services for users as and when needed. The OpenEAI template and instantiation scheme is devised to reconfigure the AIFs and resources according to the users' specific requirements, as commanded by the MANO. After receiving the application requests, the MANO of the functions sends the AIF activation signals to the AIFs required by the applications to invoke them. Meanwhile, the VIM allocates the communication, caching and computing resources in the resource pool to the AIFs invoked, and configures the runtime environment for the applications.

A. OpenEAI AIFs

The AIFs in the OpenEAI system are multi-faceted and the functional types are continuously evolving. For better understanding, we elaborate on five representative AIFs, including the Data Collection Function (DCF), Data Preprocessing Function (DPF), Model Training Function (MTF), Model Validation Function (MVF) and Data Storage Function (DSF), as shown in Fig. 1.

- **DCF** is responsible for collecting the raw data to form the training data sets that are required for training AI models. If all the collected data is valid and can be used directly, then the data will be transmitted to the MTF for training. Otherwise the data will be transmitted to the DPF for preprocessing before training.
- **DPF** preprocesses the raw data through data sampling, feature extraction, dimension reduction and by removing the invalid or biased contents and transforms the preprocessed data into the form required by the MTF.
- **MTF** is used for training the core models of all the AI algorithms. After receiving the required model type and application requests, the uniform resource locator (URL) of the request will be resolved by the application layer, and the related parameters of the URL will be extracted to match with the AI algorithm by the MTF. Then, the corresponding AI algorithm is selected for training the model to meet the application requirements. Specifically, the application requirements cover the application type (such as traffic identification, multi-dimensional resource allocation, etc.) and the key performance indicators such as latency, rate, etc.

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on September 23,2024 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Network. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MNET.2024.3411772

- **MVF** is in charge of evaluating the performance of the AI models explicitly. MVF performs both model validation during model training and model validation during real-time reasoning, where the former validates the training accuracy of the AI models when training has finished, while the latter validates the core models of the new AI algorithms based on the existing AI models. These new AI algorithms are derived from the existing AI algorithms.
- **DSF** stores and manages all the data in the AI plane centrally, including all the parameters of the AIFs. Other AIFs can access the data in the DSF through the unified SBI and can update the data dynamically according to the users' AI service requirements.

The above AIFs cover the main functions of the OpenEAI system. They can be activated by the MANO of the functions according to the service requirements.

B. OpenEAI Reconfiguration

The OpenEAI reconfiguration uses template and instantiation to activate the AIFs, configure the runtime environment, and allocate resources according to the types of applications so as to provide customized intelligent services for the users.

A template provides a universal solution for a certain type of problems by extracting and abstracting their commonalities. The template information (Tinf) of a certain AI application contains the constituent elements of the application, including the type of AIFs, plus the resource and runtime environment requirements. The template of each application has its unique template identifier (Tid), which is stored in the TS. The predefinition of a template for a specific application in the OpenEAI system defines parameters for the AIF activation, resource allocation and runtime environment configuration according to the specific requirement of the application.

Instantiation is a process that responds to a service request by creating an instance, i.e. an AI service for a certain application, according to the template. When receiving an instantiation request, the instance will be created according to the parameters in the template. Specially, if the corresponding template does not exist, the template will be established.

The workflow of OpenEAI instantiation is shown in Fig. 1 where they are numbered as follows:

(1) The MANO plane monitors the application layer continuously, and sends the template selection request to the TS when receiving an application request.
(2) The TS selects the template according to the application type, and sends the Tid of the selected template to the TIB to request the Tinf.
(3) The TIB extracts the corresponding Tinf and feeds the Tinf back to the TS.
(4) The TS sends the received Tinf to the MANO plane.
(5) The MANO plane implements the instantiation according to the Tinf:
   (a) Configure the runtime environment of the application.
   (b) Activate the related AIFs.
   (c) Allocate the required resources for the AIFs.

IV. Demonstration in a Test Network

In this section, our proposed OpenEAI will be assessed in a small-scale test network for two typical use cases, i.e., native AI-enabled traffic identification and resource allocation. Fig. 2 shows the architecture of the test network, which is a 3GPP R15-based cellular network built on open-source software and open-air-interface. Our proposed OpenEAI is installed in a general x86 server that is connected to the 5G network through UPF. The Docker containers embark on abstracing and virtualizing the communication, storage and computing resources on the general x86 server so that the resources can be shared by the CFs and AIFs. Kubernetes is used centrally to orchestrate the deployment, scheduling and life-cycle management of the Docker containers. To verify the advantages of our proposed OpenEAI, we include the following benchmarks for performance comparison: 1) The Cloud-based AI (CloudAI) scheme, where the AI algorithms are deployed at the cloud server. 2) The Plugable edge AI (PlugEAI) scheme, where all the data and the parameters of the AI functional modules are packaged as a whole to be used as a plug-in edge AI application. 3) The Non-AI scheme, where the traditional optimization methods (such as convex optimization, etc.) are used to optimize the MEC system (such as communication, caching and computing resources allocation) when the application requests arrive.

In order to verify the feasibility of the proposed OpenEAI relying on the microservice-based AI plane decoupling as well as the Kubernetes-based template and instantiation scheme, traffic identification based on a Convolutional Neural Network is firstly exemplified as an instance, which classifies the services according to the characteristics of their data packets in the test network.

Having determined the traffic type, OpenEAI allocates communication, computing, and caching resources based on deep reinforcement learning (DRL) [14]. To validate the advantage of the microservice-based AI plane, the resource allocation model is tested for edge video caching and task offloading. Both the video caching and task offloading decisions are made by the DRL-based resource allocation algorithm, which is deployed at the OpenEAI server and the cloud server for the OpenEAI/PlugEAI system and the CloudAI system, respectively.

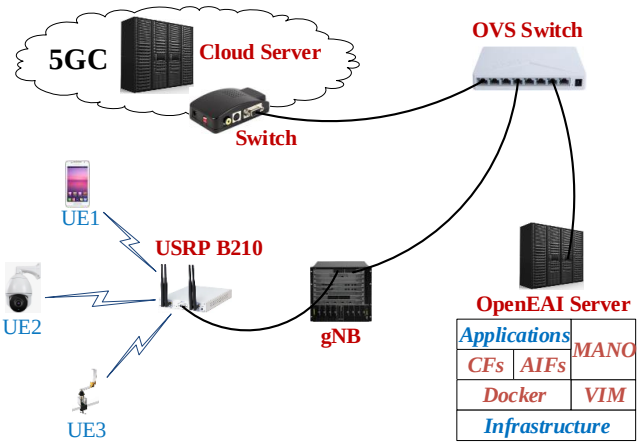

Fig. 2: Test network

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on September 23,2024 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Network. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/MNET.2024.3411772

Fig. 3 plots both the response time and training time of traffic identification against the data volume for the proposed OpenEAI system, the CloudAI system, and the PlugEAI system. For the CloudAI system, the response time is the time interval between the edge server receiving the application request from the user and the cloud server requesting the original data for training the AI model. As for the OpenEAI system, since the training data does not have to be transferred to the cloud server, the response time is the time between the MANO level receiving the application request and the AIF(s) required for successfully activating the application. The response time of the PlugEAI system is the time between the edge server receiving the application request and requesting the original data. The response times of both the OpenEAI and PlugEAI systems are shorter than that of the CloudAI system, and the training times of both the OpenEAI and PlugEAI systems are longer than that of the CloudAI system. This is because the AI model in CloudAI is trained on the cloud server using powerful computing resources, and the model training can be completed in a short period of time. By contrast, the AI model training of both the OpenEAI and PlugEAI takes longer time due to the limited computing resources of the edge server. We also note that both the training time and the response time of the PlugEAI system are longer compared to those of the OpenEAI system. This is because the AIFs in the OpenEAI system can operate in parallel, and the tasks of the MTF and MVF can be promptly executed without waiting for the data collection or preprocessing to be completed by the DCF and DPF, respectively. However, model training and response in the PlugEAI system associated with packaged functional modules can only be executed once the data collection or preprocessing is complete.

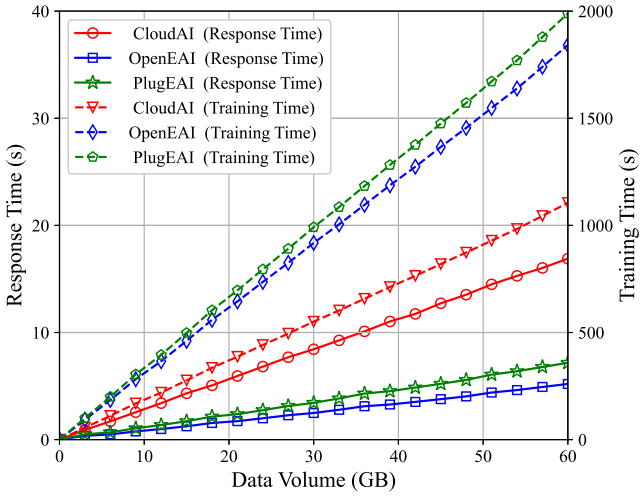

Fig. 3: Response time and training time for traffic identification

Fig. 4 depicts the accuracy and transmission overhead of traffic identification versus the number of requests. The accuracy is quantified by the proportion of services that are correctly classified [15]. It can be observed that the accuracy of the OpenEAI system approaches that of the CloudAI system, when the number of requests is below 1000. When the number of requests further increases, the OpenEAI system's accuracy becomes slightly higher than that of the CloudAI system. This is because in the CloudAI system, a considerable amount of data is transmitted to the cloud server, when the number of requests is high, while long-distance transmission makes the data susceptible to errors. The accuracy of the OpenEAI system is higher than that of the PlugEAI system, because the independent AIFs in the OpenEAI system can be flexibly orchestrated. More specifically, the OpenEAI system is capable of adjusting the allocations of communication and computing resources among the AIFs according to their respective workloads. This adaptability allows the OpenEAI system to achieve a higher level of accuracy. The transmission overhead is evaluated in terms of the data rate (Mbps) required for training data transmission. Fig. 4 shows that the OpenEAI system imposes a reduced transmission overhead in comparison to the CloudAI system, yet this is higher than that of the PlugEAI system and the Non-AI scheme.

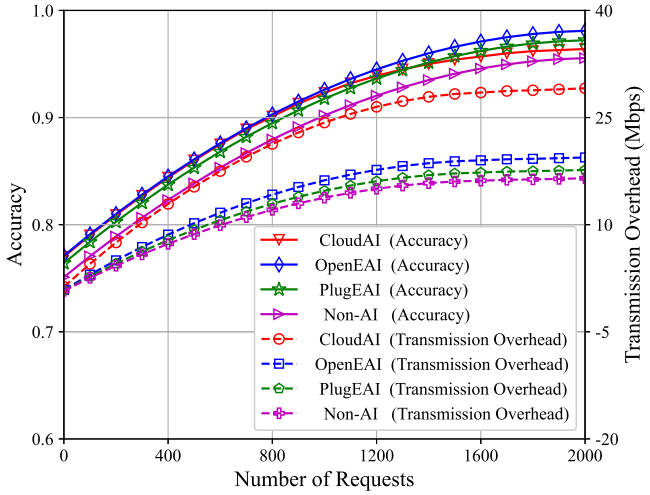

Fig. 4: Accuracy and transmission overhead for traffic identification

Fig. 5 depicts the throughput and service delay of the DRL-based resource allocation for edge video caching as a function of video size. It can be observed that both the throughput and service delay achieved by OpenEAI are nearly identical to those achieved by PlugEAI. Both OpenEAI and PlugEAI achieve higher throughput and lower service delay than the CloudAI and Non-AI schemes. This is due to the fact that the AI algorithm deployed in the OpenEAI system is situated in closer proximity of the users, and the video cached on the OpenEAI server can be updated at a higher flexibility in accordance with the specific requirements of the users.

Fig. 6 depicts the energy consumption and total completion time of the DRL-based resource allocation for task offloading versus the number of devices. The DRL-based resource allocation algorithm determines whether each task should be processed locally or offloaded to a server. Furthermore, the system is capable of allocating the requisite communication and computing resources in accordance with the user's service requirements. It can be observed that OpenEAI exhibits a comparable energy consumption and completion time to those of PlugEAI. Moreover, both OpenEAI and PlugEAI exhibit lower energy consumption and shorter completion times than CloudAI and Non-AI. This is due to the deployment of a DRL-based resource allocation algorithm at the edge server, which

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on September 23,2024 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Network. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/MNET.2024.3411772

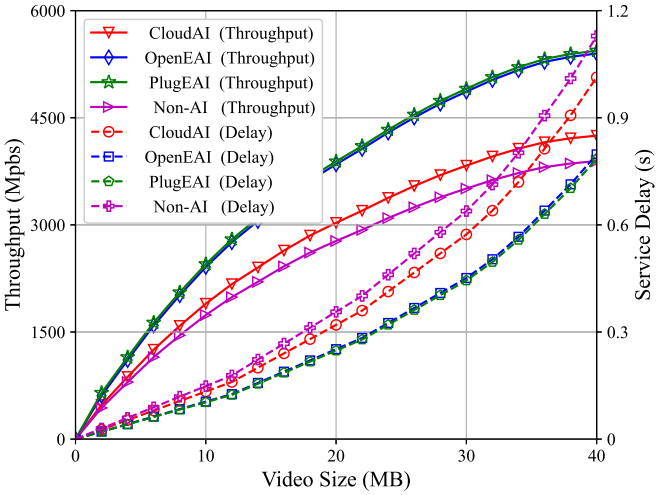

Fig. 5: Throughput and service delay for DRL-based resource allocation in edge video caching scenario

enables more timely offloading decisions.

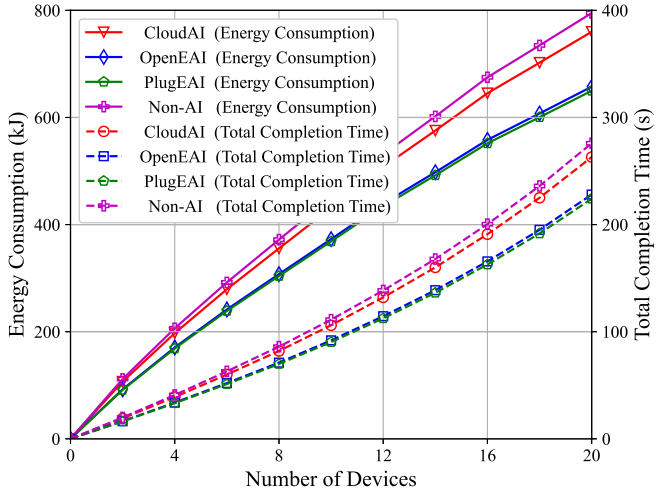

Fig. 6: Energy consumption and total completion time for DRL-based resource allocation in task offloading scenario

V. Challenges

The rapid development of AI may be exploited by introducing a native AI plane into MEC, which endows the edge network with the capability of self-optimization for accommodating the various users' service requirements and emerging NG applications based on the open-source architecture of Fig. 1 by harnessing AI plane decoupling and reconfiguration. However, there are still substantial challenges facing the intelligent open-source edge networks.

A. Improving the AIFs in Open-Source Networks

The proposed OpenEAI decouples the AI services into multiple AIFs, which significantly improves the flexibility and customization of the edge network. The native AI plane of the open-source architecture would be decoupled into more network function entities having fine granularity. For example, the scalability of the OpenEAI system can be improved by a 'self-reception' function, which automatically perceives the changes of network status and adaptively adjusts the runtime environment configuration, resource allocation and AIF activation.

B. Adaptability to Compound Scenarios

Moving on from 5G networks that focus on enhanced Mobile Broadband (eMBB), URLLC and massive Machine Type Communication (mMTC) services, NG open-source networks will face more challenging service requirements. For example, extended reality relies on a sophisticated amalgam of eMBB and URLLC services, and a smart city harnesses a combination of eMBB and mMTC services. Therefore, it is necessary for future intelligent edge networks to be flexible enough to support such challenging scenarios.

C. Long-term/Short-term Instantiation

The dynamic nature of edge AI service requirements is reflected by the variation of application requests on a long-term time-scale and by the data volume variation on a short-term time-scale. To improve the resource utilization, the instantiation process may be implemented on the basic of different time granularity. For example, the required AIFs are activated according to the type of application on a long-term time-scale, while the communication, storage and computing resources are allocated according to the tele-traffic on a short-term time-scale. In addition to the long-term/short-term, multiple spatial and resource granularity may also be used for the OpenEAI.

D. Collaboration of Cloud-edge-terminal AI

The edge server in the OpenEAI system can cooperate with the cloud server that possesses powerful computing capability and the terminal devices that have access to data sources for meeting the requirements of different AI applications. For example, the AI applications requiring computation-intensive training may be processed by the cloud server, while the AI applications requiring fast response may be processed by the edge server, and some AI applications with high security or privacy requirements may be processed by the terminal devices. However, the cooperation of cloud-edge-terminal AI to meet the distinct requirements of emerging AI applications while minimizing the transmission overhead is still an open challenge requiring further research.

E. Adaptability to Large Scale Networks

With the continuous emergence of new AI applications characterized by the massive amount of data conveyed by the NG networks (such as large language models), both the scales of the neural networks and the numbers of parameters are increasing exponentially, which imposes excessive computational burden. Hence, it is challenging to arrange for the future intelligent edge networks to be compatible with the large scale neural networks for AI model training and AI model validation.

VI. Conclusions

The OpenEAI concept has been proposed. To our best knowledge, it is first time to consider AI at the initial design of the edge network and to introduce a native AI plane into the OpenMEC system. Then, with the aid of network function virtualization, the AIFs and the resources have been decoupled from each other. Next, by using template and instantiation, the

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on September 23,2024 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Network. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MNET.2024.3411772

disaggregated AIFs and resources were reassembled according to the specific types of service requirements. Our proposed OpenEAI system has been investigated in a small-scale test network for two typical use cases. Finally, a suite promising future research directions have been highlighted.

REFERENCES

[1] L. Chettri and R. Bera, "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16-32, Jan. 2020.

[2] L. Zhao, G. Zhou, G. Zheng, C. -L. I, X. You and L. Hanzo, "Open-Source Multi-Access Edge Computing for 6G: Opportunities and Challenges," *IEEE Access*, vol. 9, pp. 158426-158439, 2021.

[3] M. Garrich, F. Moreno-Muro, M. Bueno Delgado and P. Pavn Mario, "Open-Source Network Optimization Software in the Open SDN/NFV Transport Ecosystem," *Journal of Lightwave Technology*, vol. 37, no. 1, pp. 75-88, 1 Jan.1, 2019.

[4] S. Das, F. Slyne and M. Ruffini, "Optimal Slicing of Virtualized Passive Optical Networks to Support Dense Deployment of Cloud-RAN and Multi-Access Edge Computing," *IEEE Network*, vol. 36, no. 2, pp. 131-138, Mar./Apr. 2022.

[5] A. Azari, M. Ozger and C. Cavdar, "Risk-Aware Resource Allocation for URLLC: Challenges and Strategies with Machine Learning," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 42-48, Mar. 2019.

[6] C. -L. I, S. Kuklinsk and T. Chen, "A Perspective of O-RAN Integration with MEC, SON, and Network Slicing in the 5G Era," *IEEE Network*, vol. 34, no. 6, pp. 3-4, Nov./Dec. 2020.

[7] L. Bonati, S. D'Oro, M. Polese, S. Basagni and T. Melodia, "Intelligence and Learning in O-RAN for Data-Driven NextG Cellular Networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21-27, Oct. 2021.

[8] C. -X. Wang, M. D. Renzo, S. Stanczak, S. Wang and E. G. Larsson, "Artificial Intelligence Enabled Wireless Networking for 5G and Beyond: Recent Advances and Future Challenges," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 16-23, Feb. 2020.

[9] B. Balasubramanian, E. S. Daniels, M. Hiltunen, R. Jana, K. Joshi, T. X. Tran, and C. Wang, "RIC: A RAN Intelligent Controller Platform for AI-Enabled Cellular Networks," *IEEE Internet Computing*, vol. 25, no. 2, pp. 7-17, 1 Mar.-Apr. 2021.

[10] D. C. G. Valadares, T. B. D. O. Filho, T. F. Meneses, D. F. S. Santos and A. Perkusich, "Automating the Deployment of Artificial Intelligence Services in Multiaccess Edge Computing Scenarios," *IEEE Access*, vol. 10, pp. 100736-100745, 2022.

[11] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao and K. Wu, "Artificial-Intelligence-Enabled Intelligent 6G Networks," *IEEE Network*, vol. 34, no. 6, pp. 272-280, Nov./Dec. 2020.

[12] V. Tsakanikas and T. Dagiuklas, "A Generic Framework for Deploying Video Analytic Services on the Edge," *IEEE Transactions on Cloud Computing*, vol. 11, no. 3, pp. 2614-2630, 1 Jul.-Sep. 2023.

[13] Darijo R., Ahmed H, Cormac J., Rakesh K., Emir H., Rittwik J., Vijay G., "On Leveraging Machine and Deep Learning for Throughput Prediction in Cellular Networks: Design, Performance, and Challenges," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 11-17, Mar. 2020.

[14] Y. Wang, L. Zhao, X. Chu, S. Song, Y. Deng, A. Nallanathan, and K. Liang, "Deep Reinforcement Learning-based Optimization for End-to-End Network Slicing with Control- and User-Plane Separation," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 11, pp. 12179-12194, Nov. 2022.

[15] Z. Wang, R. Chu, M. Zhang, X. Wang and S. Luan, "An Improved Selective Ensemble Learning Method for Highway Traffic Flow State Identification," *IEEE Access*, vol. 8, pp. 212623-212634, 2020.

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on September 23,2024 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.