Offloading Image Recognition Processing to FPGA Using Resource Manager for Multi-access Edge Computing

HAYATO MORI[†1] EISUKE OKAZAKI[†2] GAI NAGAHASHI[†2]
MIKIKO SATO[†2] TAKESHI OHKAWA[†3] MIDORI SUGAYA[†1]

Abstract: To reduce the power consumption of autonomous robots with Artificial Intelligence (AI), an edge computing method called multi-access edge computing (MEC) is expected to offload processing to high-performance computers in close proximity via high-speed 5G wireless communications. Furthermore, field programmable gate arrays (FPGAs) are anticipated to play a crucial role as computing resources within MEC due to their low power consumption and high-speed parallel processing capabilities. In this research, we introduce an offloading method employing MEC-RM, a resource management middleware designed for MEC, aimed at reducing response times for image recognition processing on robots. MEC-RM serves as middleware enabling the offloading of processing tasks to compute resources like FPGAs and GPGPUs through the transmission of JSON-RPC requests from edge devices to a server responsible for resource management. This paper presents the evaluation results of response times when employing the proposed method to offload image recognition processing to MEC's FPGA and communication performance in a local 5G environment.

Keywords: Multi-access Edge Computing, FPGA, Autonomous Robots

1. Introduction

The edge computing method known as Multi-access Edge Computing (MEC) has garnered significant attention for its potential to reduce power consumption in autonomous robots equipped with artificial intelligence [1]. MEC aims to handle computationally intensive tasks that are challenging to execute on a robot's onboard computer or are hampered by latency issues in the cloud. This is achieved by leveraging high-performance computational resources located in proximity via 5G high-speed wireless communication. This approach is anticipated to significantly decrease response times in autonomous robots by offloading tasks such as image recognition to high-performance computing resources.

MEC utilizes accelerator-type computing resources such as GPGPU (General Purpose computing with Graphics Processing Unit) and FPGA (Field Programmable Gate Array). These resources can be optimized for specific applications and tasks. Given the power constraints in 5G wireless base stations and local 5G environments, there is a pressing need for high power efficiency and low latency. Among these accelerator-based computational resources, FPGAs stand out due to their ability to provide high-speed parallel processing with low power consumption [2], making them a promising solution for achieving higher power efficiency compared to GPUs [3].

In this study, we propose an offloading method that leverages MEC-RM (Multi-access Edge Computing-Resource Manager), which serves as a resource management middleware designed for MEC, with the aim of reducing response times for image recognition tasks in robots. MEC-RM is a middleware that facilitates task offloading from edge devices to computing resources like FPGAs and GPGPUs through the transmission of JSON-RPC requests to a resource management server. This paper presents the results of our evaluation of the response times for offloading face detection and OpenPose-based pose estimation processing to MEC's FPGA using the proposed method, along with an assessment of the communication performance when offloading is carried out within a local 5G environment.

†1 Shibaura Institute of Technology
†2 Tokai University
†3 Kumamoto University

2. Related Works

The field of MEC has seen a significant body of related research, encompassing practical implementations, application evaluations, and task management for computational resources like FPGAs and GPGPUs. In this section, we provide a list of relevant studies on MEC and elaborate on their relevance to our research.

2.1 Applications Utilizing MEC

Queralta et al. introduced a distributed robot system leveraging MEC through 5G communication [4]. In their proposed approach, MECs handle autonomous operations for both robots and vehicles, resulting in a high level of autonomy. While their method incorporates various computational resources, including CPUs and GPUs, the utility of FPGAs as computational resources within MEC has not yet been thoroughly assessed.

Furthermore, Inage et al. presented a method involving the creation of an FPGA cluster tailored for MEC applications [5]. In this methodology, they employed the M-KUBOS board [6], a medium-sized FPGA board, and achieved a remarkable power efficiency improvement ranging from 1.92 to 46.24 times that of AMD's Ryzen7 5800X CPU. This highlights the superiority of FPGA clusters as computational resources for MEC. However, it's worth noting that specific application evaluations within communication environments involving robots, as required by MEC, have not been conducted, nor have experiments taken place within a 5G environment.

Our research aims to address these gaps in the existing

research by proposing an offloading method using MEC-RM, with a particular focus on FPGAs as computational resources for MEC. We evaluate this method by assessing its impact on the response times of image recognition tasks in robots and explore its performance within a local 5G environment. By doing so, we contribute valuable insights into the practical use of FPGAs within MEC scenarios that involve robot communication, bridging a crucial knowledge gap in the field.

2.2 Computational Resource Management Methods in MEC

Osaki et al. introduced the Giocci platform as a method for resource allocation to offload tasks from end devices to external computational resources within network configurations that incorporate MEC servers [7]. Giocci is structured into four layers: Client, Relay, Engine, and Store. Relay is responsible for transmitting data collected from the Client to the Engine, employing a defined algorithm for the transmission process. The Engine, on the other hand, handles computational processing tasks and is equipped with v-contact information that mirrors the p-contact information sent by the Client. Meanwhile, the Store is tasked with preserving the state of the Engine, ensuring its persistence. Giocci operates by forwarding data gathered from end devices through the Client to computing resources in the Engine layer via the Relay, subsequently returning the processing results to the Client.

Similarly, as the MEC-RM, which is the focus of this study, Giocci explores resource management methods during offloading. However, it differentiates itself by employing the functional programming language Elixir for communication. In contrast, MEC-RM facilitates offloading of processing tasks through JSON-RPC via HTTP communication, rendering it a more versatile choice for offloading tasks. This versatility can be attributed to its communication protocol, which may offer advantages in various scenarios, including the one presented in our research involving robot communication within a local 5G environment.

3. Purpose/Proposal

In this study, we present a novel approach for offloading image processing tasks to FPGA resources through the utilization of MEC-RM (Multi-access Edge Computing-Resource Manager), which serves as the resource management middleware within the MEC. The primary objective of this method is to significantly reduce the response time required for image recognition processes in autonomous robots.

Specifically, our research focuses on offloading image processing tasks to a medium-sized FPGA board. To facilitate this offloading process, we leverage the capabilities of the MEC-RM middleware [9]. MEC-RM is designed to function as a comprehensive system for managing multiple computational resources utilized within the MEC ecosystem. This strategic deployment of MEC-RM aligns perfectly with the central objective of our study.

3.1 Processing Tasks to be Offloaded to FPGA

In this experiment, we envisioned an autonomous robot designed to serve as a patrol robot within nursing care facilities. With a diminishing working-age population due to demographic aging, there is a growing expectation that robots will play a vital role in reducing the workload in nursing care facilities. The functions of nursing care robots encompass autonomous mobility, resident interaction, and the crucial task of detecting falls among the elderly. Consequently, two critical processing tasks, namely face detection and posture estimation, were employed to fulfill the requirements of these functions in our evaluation application.

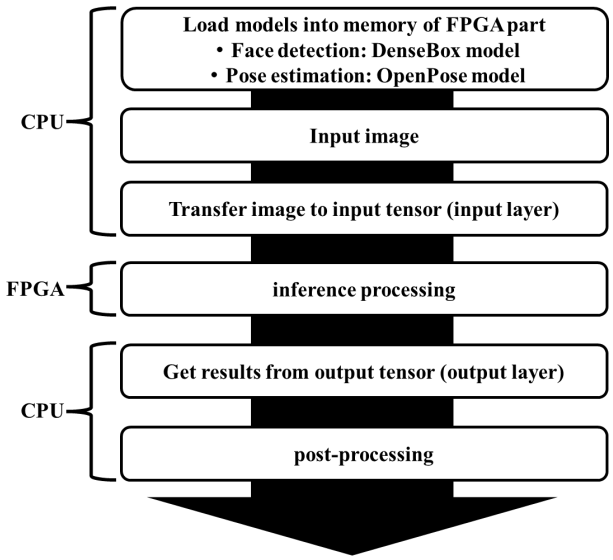

Figure 1 Processing Flow of an Application Using DPU.

Face detection processing was selected because it is essential for the robot to determine whether it has detected a person when greeting residents or caregivers. Robots operating in nursing care facilities must be capable of effective communication with residents and caregivers during their patrols. A fundamental requirement for this interaction is a greeting function, which necessitates the robot's ability to recognize individuals as people. Thus, face detection processing is a prerequisite.

Moreover, the posture estimation process was implemented because it is vital for fall detection. Detecting falls among elderly individuals requires an estimation of the person's posture to determine whether they are standing, sitting, or falling. To meet this requirement, we implemented the posture estimation process on an FPGA using the OpenPose-based Vitis AI library [10], effectively offloading the processing from the robot to the FPGA.

Both applications were implemented using the Deep-Learning Processing Unit (DPU) [11], a programmable engine designed specifically for convolutional neural networks and provided by AMD Xilinx. The DPU was chosen for its compatibility with a wide range of convolutional neural network architectures, including ResNet and YOLO, making it well-suited for the high-speed image recognition demands inherent in autonomous robots, which is the central objective of this research.

Figure 1 outlines the processing flow of an application utilizing the DPU. Initially, a pre-trained model is loaded into the internal memory of the FPGA. Subsequently, image input is

initiated, and the image data is transferred to the input tensor. Following this, the inference process is executed utilizing the DPUs within the FPGA circuit. Once the process is complete, the results are retrieved from the output tensor and undergo post-processing tailored to the specific requirements of the application.

3.2 Design of Offloading System Using MEC-RM

Figure 2 illustrates the process flow of offloading through MEC-RM, while Figure 3 presents the system configuration. MEC-RM operates on MEC servers, where a Requester initiates a processing request in JSON-RPC format directed at the MEC server. The MEC server then forwards the processing request to a waiting worker for execution. Subsequently, the worker carries out the processing task and transmits the results back to the Requester via the MEC server. This approach is designed to enable the computational resources within MEC to efficiently handle the intensive processing load originally placed on the edge device, the Requester, achieving high-speed processing.

To facilitate discussions concerning the response times of both the CPU and FPGA, we conducted the face detection and posture estimation processes described in Section 3.1 on a Raspberry Pi 4B, which serves as the onboard computer on the robot. Additionally, we evaluated the response time of MEC-RM running on a Raspberry Pi 4B also installed on the robot. Table 1 provides an overview of the evaluation environment. The edge device employed was the Raspberry Pi 4B, and the FPGA responsible for offloading processing was the M-KUBOS board.

Furthermore, we conducted measurements not only in a Wi-Fi environment but also in a local 5G environment. This approach allows us to assess the impact of offloading on response times while accounting for communication latency within the 5G environment, which aligns with the assumptions made by MEC. For the measurements conducted in the local 5G environment, we utilized a Laptop environment, as indicated in Table 1, for the sake of experimental convenience, in lieu of the Raspberry Pi 4B.

3.3 Offloading the Image Recognition Process to the M-KUBOS board

Figure 4 presents the system configuration for the evaluation application. The process begins by capturing images from the camera mounted on the Raspberry Pi 4B, designated as the MEC Requester. These images are then transmitted to a ROS (Robot Operating System) node responsible for communication with the MEC server through a ROS topic. Subsequently, processing requests and image data are sent to the MEC server via HTTP communication. The MEC server, upon receiving these requests, assigns a computation task to the M-KUBOS board, which functions as the MEC worker.

The M-KUBOS board, upon receiving the processing request, executes the target application using the DPU located within the FPGA circuit. After completing the processing, it transmits the processing results back to the MEC server. Once the results are available on the MEC server, they are then transferred to the Raspberry Pi 4B. In this evaluation, we measured several time intervals, including the time taken by the Raspberry Pi 4B to upload the input data (upload), the time required for sending the processing request (query), the duration from sending the request to its completion (job), and the time taken for downloading the results (download). The execution time on the Raspberry Pi 4B is categorized as part of the job since it pertains to the application's execution time.

Table 1 Evaluation Environment

	RaspberryPi 4B	M-KUBOS	Laptop
CPU	ARM Cortex-A72 (4core,1.5GHz)	ARM Cortex-A53 (1.4GHz)	Intel Corei7-11800H (2.30GHz)
FPGA	-	XCZU19EG-2FFVC1760	-
RAM	4GB	4GB DDR4-2400	32GB DDR4-3200 16GB x2
OS	Ubuntu 20.04 LTS	PYNQ v2.5	Ubuntu 22.04 LTS

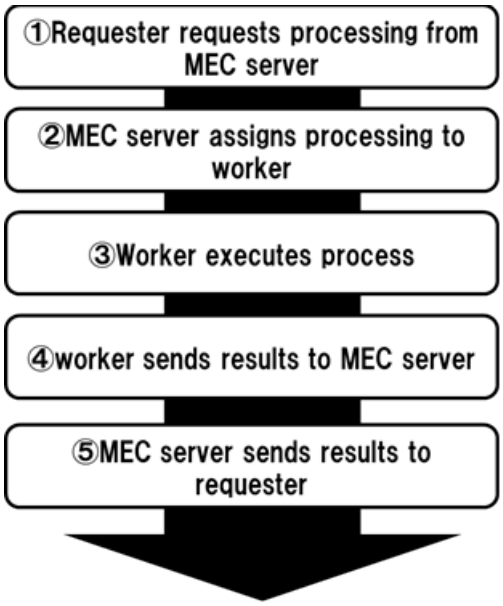

Figure 2 Process Flow of Offloading Using MEC-RM

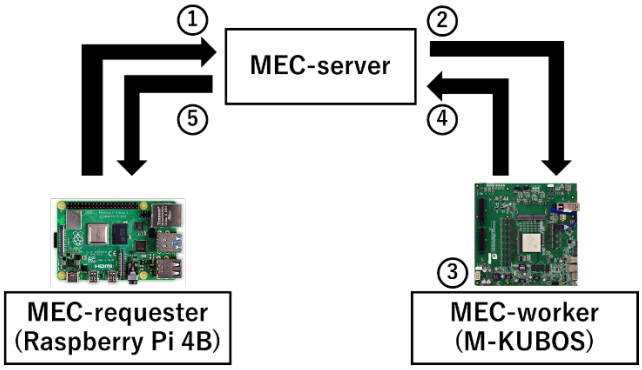

Figure 3 System Configuration for Offloading Using MEC-RM

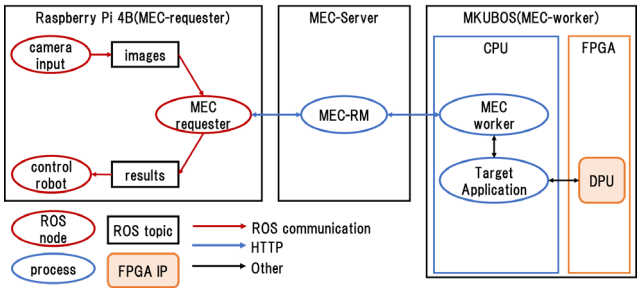

Figure 4 System Configuration for Evaluation Application Using DPU

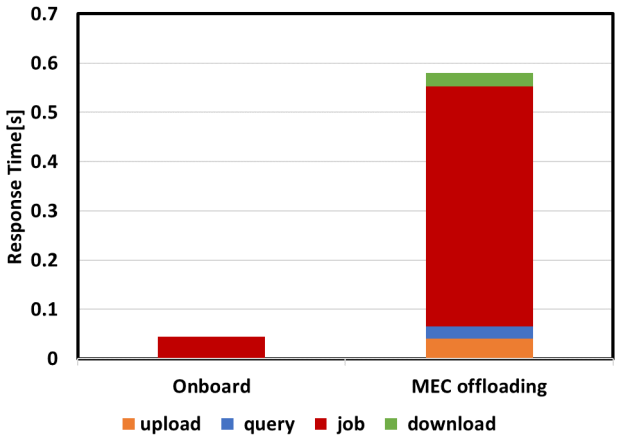

Figure 5 Response Time of Offloading Face Detection Processing to FPGA Using MEC-RM

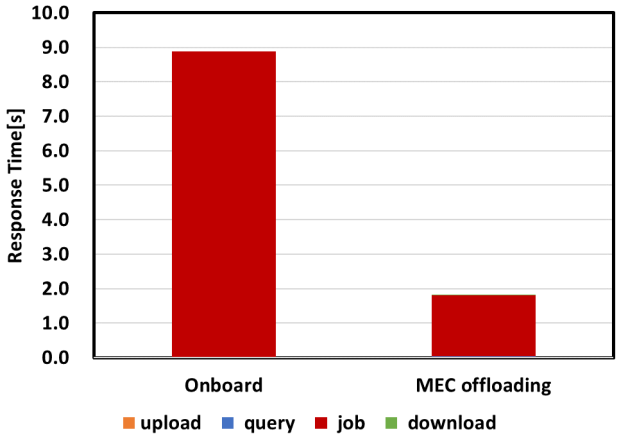

Figure 6 Response Time of Offloading Pose Estimation Processing to FPGA Using MEC-RM

4. Evaluation

In this section, we present the outcomes of offloading two evaluation applications to FPGAs within the evaluation environment detailed in the preceding section. Additionally, we share the results of evaluating the impact of communication performance within a local 5G environment. Firstly, we provide an overview of the results pertaining to the offloading of image recognition processing to FPGAs. This analysis elucidates the effect on response times resulting from the migration of the implemented application from the edge device to the FPGA located within the MEC. Furthermore, it assesses the efficacy of the proposed offloading method. Subsequently, we present the evaluation findings obtained within a local 5G environment, illustrating the value of the proposed method within the local 5G context assumed by the MEC.

During the evaluation, we employed image sizes of 640x360 pixels for the face detection process and 368x368 pixels for the posture estimation process. These image sizes were chosen to accurately represent the processing demands of the applications under examination.

4.1 Offloading Image Recognition Processing to FPGA Using MEC-RM

Figure 5 and Figure 6 display the results of offloading the face detection and posture estimation processes to the FPGA on the MEC, respectively, in comparison to the same processes executed on the Raspberry Pi 4B (Onboard) and offloaded to the MKUBOS board using MEC-RM (MEC offloading).

For the face detection process, offloading resulted in a 12.9-fold increase in response time compared to the Onboard execution. This increase can be attributed not only to the communication overhead with the MEC-server but also to the delays introduced by the execution time of jobs.

In contrast, the posture estimation process exhibited a 79.5% reduction in response time compared to Onboard execution. Since the Onboard response time for the posture estimation process was already substantial, our method effectively reduces response time by offloading the computationally intensive task to FPGA from the edge device.

However, it's worth noting that the response time following the offloading of the posture estimation process is 0.55 frames per second (fps), indicating that while the response time has been improved, the performance remains insufficient for autonomous robot control. To address this, a more detailed analysis of the time required for resource allocation and a review of the current data flow, where input/output data is transferred via the MEC-server for each request, will be essential to mitigate the observed delays and enhance system performance.

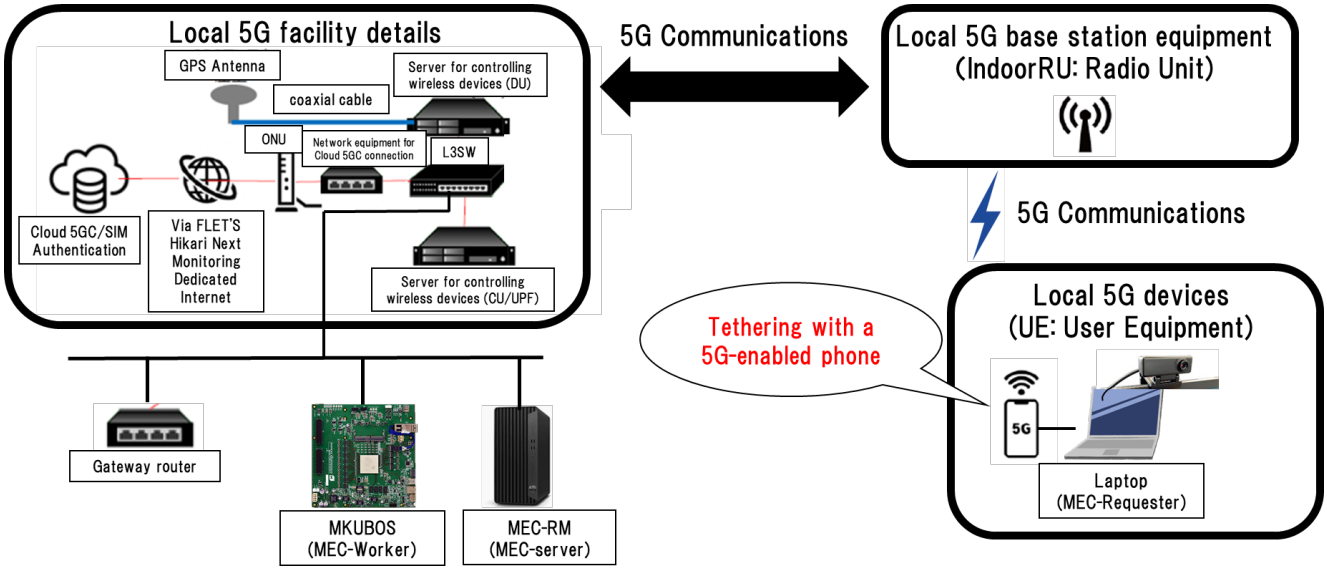

Figure 7 Network Configuration for Local 5G Environment

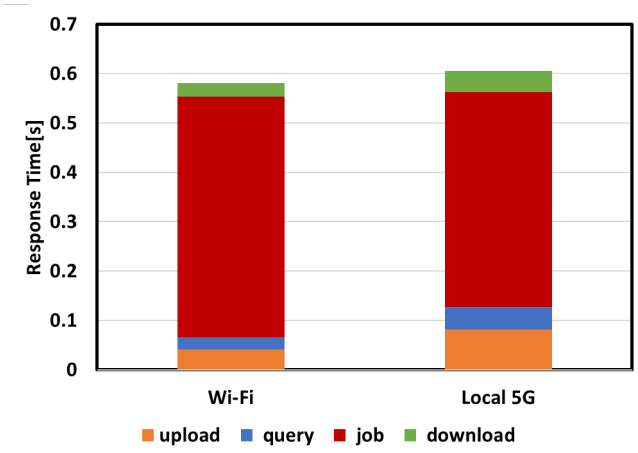

Figure 8 Response Time of Face Detection Processing in Local 5G Environment

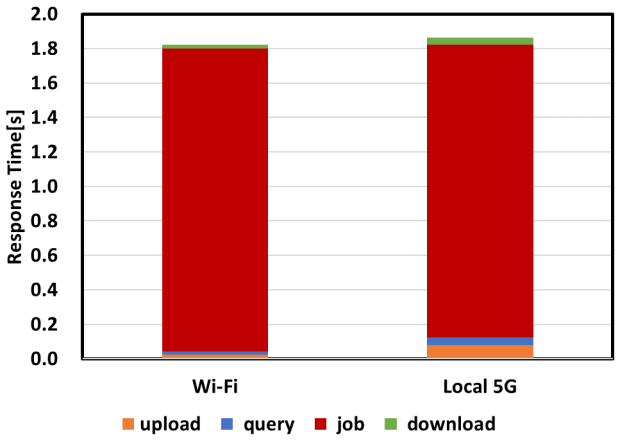

Figure 9 Response Time of Pose Estimation Processing in Local 5G Environment

4.2 Comparison of Communication Performance through Experiments in a Local 5G Environment

Figure 7 illustrates the network configuration of the local 5G environment, where the MEC-server and MKUBOS (MEC-worker) are connected to the DU (Distribution Unit) via an L3 switch. The MEC-requester, represented by the Laptop, establishes 5G wireless communication through the tethering function of a 5G-compatible smartphone connected to the local 5G base station device.

Figure 8 presents the results of offloading the face detection process, while Figure 9 displays the results of offloading the pose estimation process in both Wi-Fi (IEEE802.11b) and local 5G environments, mirroring the setup in the preceding section. Notably, the overall response times remained consistent, but the response times for upload, query, and download were greater in the local 5G environment compared to Wi-Fi for all processes. This difference can be attributed to the presence of multiple network devices between the MEC-requester, MEC-server, and MEC-worker in the local 5G environment, as opposed to the network configuration in the Wi-Fi environment, where only a router is used.

Conversely, the job time for both processes was reduced by approximately 50 milliseconds. This reduction is attributed to the fact that the MEC-server and MEC-worker are connected to the same L3 Switch, which likely mitigated delays compared to communication with the MEC-requester.

These results collectively demonstrate that offloading to FPGAs using the proposed method performs as effectively as in a Wi-Fi environment, even within the local 5G environment envisioned by MEC. This finding underscores the feasibility and robustness of the proposed offloading approach in scenarios characterized by local 5G connectivity.

5. Conclusion

In this study, we successfully employed MEC-RM to offload image recognition processing tasks onto an FPGA, effectively managing computational resources within the MEC framework. Our evaluation focused on two essential image recognition tasks: face detection and posture estimation, both critical for the functionality of a care robot. The results of our study revealed a significant increase in response time for the face detection process when compared to executing it on the Raspberry Pi 4B,

the edge device. Specifically, the face detection process exhibited a 12.9-fold increase in response time when offloaded to the FPGA. Conversely, the response time for the posture estimation process saw a remarkable reduction of 79.5% when offloaded. This outcome underscores the efficacy of our proposed method in offloading computationally intensive image recognition tasks to an FPGA, thereby substantially reducing response times. Overall, our findings demonstrate that our approach is capable of effectively offloading image recognition processing, which would otherwise be too high load for an edge device like the Raspberry Pi 4B, to an FPGA, resulting in significantly improved response times.

The same set of experiments was conducted in a local 5G environment to assess communication performance within this 5G context. The results indicated that the performance in the local 5G environment, including latency, was comparable to that of the Wi-Fi environment. In both scenarios, response times for upload, query, and download were slightly longer in the local 5G environment as compared to the Wi-Fi environment. However, there was a notable reduction of approximately 50 milliseconds in the job time. This reduction can be attributed to the fact that communication delays between the MEC-server and MEC-worker were mitigated through the utilization of 5G communication technology.

Moving forward, future tasks for this research include conducting a detailed measurement of the time required by MEC-RM for resource allocation, conducting a thorough analysis of potential bottlenecks, and enhancing system performance by reviewing the current data flow, where input and output data are transferred through the MEC-server for each request. These efforts aim to further optimize the efficiency of the proposed offloading method.

Reference

[1] Kekki, S., Featherstone, W., Fang, Y., Kuure, P., Li, A., Ranjan, A., ... & Scarpina, S. (2018). MEC in 5G networks. ETSI white paper, 28(2018), 1-28.

[2] K. Iizuka, H. Takagi, A. Kamei, K. Hironaka and H. Amano, "Power Analysis of Directly-connected FPGA Clusters," 2022 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS), Tokyo, Japan, 2022, pp. 1-6, doi: 10.1109/COOLCHIPS54332.2022.9772675.

[3] BIOOKAGHAZADEH, Saman; ZHAO, Ming; REN, Fengbo. Are fpgas suitable for edge computing? In: USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18). 2018.

[4] J. P. Queralta, L. Qingqing, Z. Zou and T. Westerlund, "Enhancing Autonomy with Blockchain and Multi-Access Edge Computing in Distributed Robotic Systems," 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 2020, pp. 180-187, doi: 10.1109/FMEC49853.2020.9144809.

[5] T. Inage et al., "M-KUBOS/PYNQ Cluster for multi-access edge computing," 2021 Ninth International Symposium on Computing and Networking (CANDAR), Matsue, Japan, 2021, pp. 95-101, doi: 10.1109/CANDAR53791.2021.00020.

[6] PALTEK, “FPGA computing platform M-KUBOS,” https://www.paltek.co.jp/design/original/m-kubos/ (accessed 2023-9-17).

[7] D. Sasaki, et al. "Resource allocation method among MEC servers in resource transparent platform for 5G communication network." Research Report Internet and Operational Technology (IOT) 2023.3 (2023): 1-8. (In Japanese).

[8] Nakagawa, I., et al.: Dripcast – Server-less Java Programming Framework for Billions of IoT Devices. In Proc. of COMPSAC, pp. 186–191 (2014).

[9] LI YANZHI, Midori Sugaya. (2022). Proposal of a resource management system for multi-FPGA/GPGPU mixed environment. Research report High Performance Computing (HPC), 2022(2), 1-10. (In Japanese).

[10] AMD Xilinx: Vitis AI, https://github.com/Xilinx/Vitis-AI, (accessed 2023-9-17).

[11] AMD Xilinx: DPU IP Details and System Integration, https://xilinx.github.io/Vitis-AI/3.5/html/docs/workflow-system-integration.html (accessed 2023-9-17).

Acknowledgments This work was supported by JST, CREST, JPMJCR19K1 and JTOWER. Inc.