2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)
October 19-25, 2025. Hangzhou, China

2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3315-4393-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/IROS60139.2025.11246268

Masked Autoencoders are Robust Task Offloaders for Timely and Accurate Inference

Wonyeong Lee[1], Seunghoon Lee[1], Seungyeon Cho[1], Hyunwoo Koo[1], Hoon Sung Chwa[2] and Jinkyu Lee[1]*

***Abstract*— Edge devices for robotics in hazardous environments, such as rescue drones, navigate complex terrains while transmitting images to remote servers for anomaly detection, including wildfires. However, these devices operate under strict resource constraints, prioritizing operational-critical tasks (e.g., autonomous navigation) while handling image-processing workloads with minimal overhead. Offloading computation to a remote server can alleviate this burden, but unstable network conditions can degrade accuracy and timeliness. To address these challenges, this paper presents a novel offloading framework that balances computational efficiency and accuracy in image-processing tasks. Specifically, it ensures (R1) a minimum accuracy level for individual image-processing tasks associated with different camera sensors and (R2) maximizes the overall image-processing accuracy across all sensors. Our approach builds on an edge-server collaborative image reconstruction architecture, where images are divided into patches and selectively reconstructed. To achieve R1 and R2, we introduce: (i) a *hierarchical scheduler* that effectively prioritizes patch transmissions under resource constraints and (ii) a *feedback mechanism* that adapts to network instability, ensuring reliable offloading and inference. Experimental results demonstrate that our framework maintains high accuracy and timely processing, even under network failures.**

I. Introduction

The demand for deploying machine learning on edge devices is rapidly growing across various industries [1] including robotics in hazardous fields [2], [3]. In practical applications, edge devices must handle multiple tasks simultaneously, processing essential real-time operations on-device while offloading secondary tasks to a server for further computation. Consider a mountain search and rescue (SAR) drone [4], which is equipped with several specialized camera sensors, each performing a unique function. In disaster scenarios, the SAR drone autonomously navigates wooded areas, transmitting images to headquarters to detect anomalies like wildfires or injured individuals. As an edge device, the SAR drone allocates most of its computational resources to operation-critical tasks like autonomous navigation, necessitating the offloading of image-processing mission tasks, such as object detection and classification, to external servers. Such an edge-server collaborative system must ensure that offloaded tasks are completed within strict time constraints on the server side, enabling real-time decision making in the field, which poses the following challenges.

*Jinkyu Lee is the corresponding author.
[1]Department of Computer Science and Engineering, Sungkyunkwan University (SKKU), Republic of Korea. {lwy970327, seunghoon.l, seungyeoncho, koowoo3, jinkyu.lee}@skku.edu
[2]Department of Electrical Engineering and Computer Science, DGIST, Republic of Korea. chwahs@dgist.ac.kr

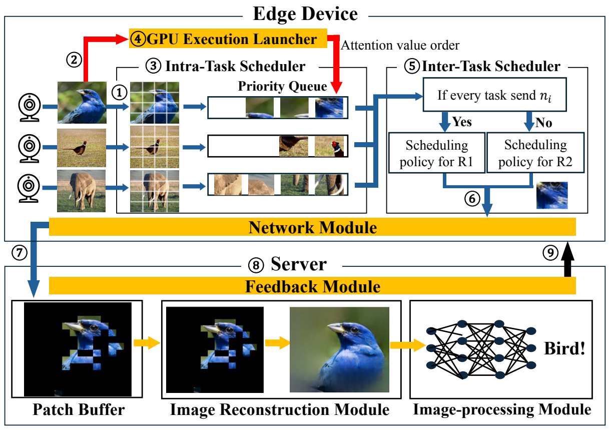


Fig. 1: System overview

C1. Limited and shared computing capability of each edge device: image-processing mission tasks must be executed or preprocessed for offloading while consuming as little as on-device CPU/GPU resources.
C2. Network instability: as data from camera sensors need consistent transfer to the server for processing, the system must address unstable network conditions to ensure timely completion of mission tasks on the server side under extreme environments, e.g., disaster scenarios.

To be effective in scenarios like the SAR drone, this paper targets a framework designed to meet the following requirements while addressing challenges C1 and C2:

R1. The proposed framework should ensure a minimum level of accuracy of the image-processing mission task (e.g., object detection, classification) for each camera sensor, which provides consistent minimal operational performance for different functions associated with their respective camera sensors, thereby achieving system stability and reliability.
R2. The proposed framework should maximize the aggregated accuracy of image-processing mission tasks for different camera sensors, which optimizes the overall system performance.

Related work unsupportive of R1/R2 under C1/C2. Many prior studies have investigated methods for achieving effective image-processing within the constraints of edge devices. A naive approach is to run image-processing mission tasks directly on the edge device and transmit only the results to the server [5], [6]. However, only limited capacity remains for additional processing in our target system, where the edge device (SAR drone) must prioritize operation-critical tasks consuming most of computing resources (e.g., autonomous navigation). To address this, some studies propose transmitting raw image data to the server for image-processing [7]. Yet, in environments with limited bandwidth and unstable connections, it is difficult to ensure the successful offloading of all data [8]. Partial data transfer disrupts inference of image-processing tasks, and re-requesting missing data introduces latency, compromising timing guarantees. Other studies [9]–[11] suggest compression methods to reduce bandwidth usage and improve performance while protecting privacy; however, these approaches require encoding on the edge device, which is challenging to implement within resource-constrained environments and therefore unsuitable for our target scenario.

In this paper, we develop a framework that accomplishes R1 and R2 under C1 and C2, illustrated in Figure 1. As a foundation for our framework, we choose the edge-server collaborative image reconstruction architecture [12] with the reconstruction model of Masked AutoEncoder (MAE) [13], which provides a robust interface for achieving the goal, as it enables images to be divided into small patches, reducing transmission load, and allowing for partial reconstruction for image-processing tasks, even when not all patches are received. On top of the interface, we propose the *hierarchical scheduler* and the *feedback mechanism*, which are key components in achieving the goal, to be detailed now.

The hierarchical scheduler is a key component of our framework, designed to achieve both R1 and R2 by primarily addressing resource constraints of edge devices (i.e., C1). This scheduler incorporates two main parts: the intra-task and inter-task schedulers. The intra-task scheduler organizes the transmission order of patches within each image to enhance the accuracy of each image for a given number of patches, thereby contributing R2 by maximizing the accuracy of individual images. Meanwhile, the inter-task scheduler assigns transmission priorities among patches from different images, with each image's patches pre-sorted by the intra-task scheduler, so as to ensure a minimum number of patches of every image to be transmitted (achieving R1) and maximize the sum of the expected accuracy of all images (achieving R2).

Also, we propose a feedback mechanism within the framework designed to handle the challenges of network instability, which can result in discrepancies between patches transmitted by the edge device and those successfully received by the server. This mechanism ensures that accuracy requirements are met despite varying network conditions by providing periodic feedback from the server to the device regarding useful information such as the number of received patches. The feedback mechanism operates on both long-term and short-term scales. Feedback on a long-term scale adjusts the transmission margin (that controls minimum required patches) based on historical loss rates, adding stability without overcompensating for isolated failures. Feedback on a short-term scale, on the other hand, reacts immediately to current network conditions by calculating the difference between patches sent and received, dynamically changing scheduling policies to meet accuracy targets. This dual feedback approach maintains system efficiency while addressing performance requirements under network fluctuations.

We evaluate the performance of our system under various task periods and patch transmission configurations. Without network failures, our system achieves up to 11% higher accuracy compared to baseline systems while timely transmitting the patches required for the minimal operational performance of individual camera sensors. Even under network failure conditions, our framework prioritizes images subject to minimal operation performance in a time-predictable manner while achieving no deadline miss, which cannot be accomplished by the baselines.

This paper makes the following contributions.

- Developing the first framework that achieves R1 and R2, which is necessary for an edge-server collaborative robot system in hazardous fields;
- Presenting the architectural design that supports the proposed framework (Section II);
- Designing a hierarchical scheduler consisting of intra-task and inter-task schedulers (Section III);
- Proposing a feedback mechanism that provides resiliency for the proposed framework (Section IV); and
- Demonstrating the effectiveness of the proposed framework using real-world experiments (Section V).

II. System Design

In this section, we present the system design of our framework, which establishes the foundation for achieving R1 and R2 while addressing the challenges of C1 and C2.

Base architecture with reconstruction model. Due to the challenges C1 and C2 presented in Section I, we need to find a base architecture, which features as follows: instead of directly running the target application for each image and sending the result to the server, (i) the edge device not only minimizes its workload to process each image (addressing C1), (ii) but also some loss of packet transmitted to the server does not nullify the result of image-processing mission tasks (addressing C2). Therefore, we apply the edge-server collaborative image reconstruction architecture [12] with the MAE reconstruction model [13]. The MAE model's reconstruction capability enables it to restore original images from a limited number of patches, achieving objective (i). Additionally, its ability to reconstruct images from partial data supports efficient offloading, which minimizes network resource consumption and makes it well-suited for environments subject to network failure, achieving objective (ii).

Once MAE is selected as a reconstruction model to be incorporated into the edge-server collaborative image reconstruction architecture, our framework has the following components, illustrated in Figure 1.

Camera sensors. The edge device is equipped with multiple camera sensors that periodically send images (frames) to the hierarchical scheduler (① in Figure 1), each operating at a different period according to its criticality.

Hierarchical scheduler. When an image arrives, the edge device preprocesses it by dividing it into patches of a specified size and quantity. The following schedulers (or launcher) then determine the order in which patches are transmitted:

- *Intra-task scheduler* (③): After an image is divided into multiple patches, this component decides the sequence of which patches to process from the same image.
- *Inter-task scheduler* (⑤): This component selects which camera sensor's image patches to prioritize for transmission, based on the incoming images from multiple sensors. Then, it sends the selected patch to the network module.
- *GPU execution launcher* (④): To improve the quality of the selected patches in the intra-task scheduler, this component leverages the GPU, to get the attention information (to be detailed in Section III-B) as soon as an image is received from the camera sensor; then it sends the result to the intra-task scheduler.

Importantly, the hierarchical scheduler is central to achieving R1 and R2. Section III will describe the fundamental operations without accounting for network failures, while Section IV will present how we adapt the hierarchical scheduler for the environments subject to network instability.

Network module. The module in the edge device transmits a patch to the server (⑦), as soon as it receives a patch from the inter-task scheduler.

Image reconstruction module. This module in the server collects all patches transmitted from the edge device into its patch buffer and performs the image reconstruction process on a set of patches belonging to the same image.

Image-processing module. This module in the server utilizes the reconstructed image to execute the designated target image-processing mission tasks (e.g., image classification, object detection). No additional fine-tuning is applied to the reconstructed images prior to these tasks.

Feedback module. This module in the server provides feedback to the edge device, which is a key to be detailed in Section IV.

III. Hierarchical Scheduler

We design a hierarchical scheduler that achieves both R1 and R2, focusing solely on C1. The hierarchical scheduler mainly consists of *intra-task* and *inter-task* schedulers. The intra-task scheduler determines the order of patches of each image to be transmitted, in order to maximize the accuracy of each image under a given number of transmitted patches, contributing to R2. The inter-task scheduler determines the patch (from different images) to be transmitted in each time slot, such that (i) the number of transmitted patches for each image within its deadline is no smaller than the threshold given by the criticality of its camera sensor (achieving R1), and (ii) the sum of the expected accuracy of all images is maximized (achieving R2).

A. Task Model

Each camera sensor, which is a task of the system, $\tau_i \in \tau$ is modeled as follows. Each camera sensor τ_i captures an image every T_i time unit, where the x-th image from τ_i is represented as $I_{i,x}$. According to the approach in [13], we divide each image $I_{i,x}$ into 196 patches $\{P^q_{i,x}\}_{1\le q\le 196}$ (arranged in a 14×14 grid), where $P^q_{i,x}$ denotes the q-th patch of $I_{i,x}$ (which is expressed as $P_{i,x}$ when q is irrelevant). Let Δ represent the time spent waiting for a single patch to be placed onto the network medium. All patches $P_{i,x}$ from an image $I_{i,x}$ captured at time t must launch its transmission to the server before $t+T_i$ (called *transmission launch deadline*, or shortly *deadline*).

Each image captured by each camera τ_i shares a threshold n_i for the number of patches to be transmitted to ensure minimal operational performance associated with τ_i (to achieve R1); we call n_i the ground patch count of τ_i. In the absence of network failures, n_i is determined based on the relationship between the number of patches and the accuracy of the image-processing mission task (e.g., Figure 2(d)), reflecting the minimal operational performance required for each camera task's priority. However, when network failures occur, adjustments to n_i, are necessary to maintain performance, which will be provided in Section IV.

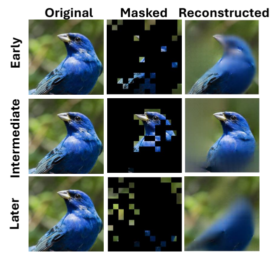
Using the notion of the ground patch count, we define the *state* of each image $I_{i,x}$ (denoted by $\mathcal{S}_{i,x}$) at t, whose deadline is later than t as follows: *ground state* ($\mathcal{S}_{i,x}$=GS) in which the number of patches of $I_{i,x}$ launched to be transmitted to the server at t is not larger than n_i, or *extended state* ($\mathcal{S}_{i,x}$=ES) in which the number at t is larger than n_i. Then, the *mode* of the edge device (denoted by $\mathcal{M}$) at t is defined as follows: *ground mode* ($\mathcal{M}$=GM) in which there is at least one image with $\mathcal{S}_{i,x}$=GS (whose deadline is later than t) at t, or *extended mode* ($\mathcal{M}$=EM) in which there is no such an image.

B. Intra-Task Scheduler

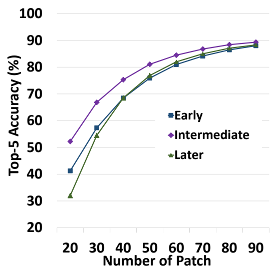
The intra-task scheduler aims to enhance accuracy by determining the order of patches within an image to be processed and sent to the server (after competing the use of the network medium according to the inter-task scheduler). We leverage two key directions (by observations) to rank the patches: (O1) using the attention values from the Vision Transformer (ViT) [14] to prioritize certain patches, and (O2) balancing the information content across patches to ensure a comprehensive representation.

O1. The attention mechanism allows models to learn relationships within data. In ViTs, this mechanism captures relationships between image patches, enhancing the model's understanding of the structure. Patches near key objects typically receive higher attention values, indicating their importance. In networked environments where image patches are transmitted, prioritizing patches with higher attention values ensures that essential parts are sent first (whereas lower-priority patches may not be selected for transmission by the inter-task scheduler). This approach preserves core information, enabling image recognition even with limited bandwidth or unstable network conditions, thereby improving transmission efficiency and retaining critical content.

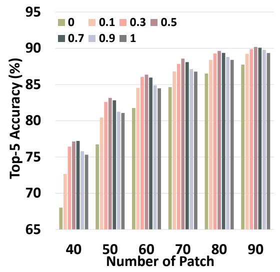
The early layers of the Vision Transformer focus on distributing attention evenly across all patches to capture



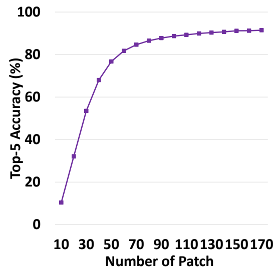
(a) Image reconstruction example across different layers



(b) Accuracy with varying ViT layer attention value



(c) Accuracy with varying ratios of patches selected by attention value order



(d) Relation between accuracy and # of randomly selected patches

Fig. 2: Motivational experiment results

general information about the entire image according to [15]. The intermediate layers, on the other hand, emphasize assigning higher attention to the important regions of the image, prioritizing accurate information extraction. Meanwhile, the later layers redistribute attention to less critical areas, contrasting them with the key patches to achieve a comprehensive understanding of the image.

Our system's reconstruction, relying on a limited number of image patches, benefits significantly from utilizing intermediate-layer attention values, which focus on critical regions. As shown in Figure 2(a), early and late layers are less effective at capturing essential areas, while intermediate layers achieve high-quality results even with only 15% of patches. Experiments using attention values from specific layers, as shown in Figure 2(b), confirm that intermediate layers deliver the best performance in reconstruction and classification, highlighting their superior ability to emphasize critical regions and achieve optimal accuracy. Additionally, exiting at intermediate layers, rather than processing all layers, improves computational efficiency.

Therefore, we calculate patch priority using the attention values from the intermediate layers of the ViT model.

O2. We hypothesized that it would enhance image reconstruction to combine two different types of patches: (i) patches with high attention values containing direct information about the objects, and (ii) patches from background areas. This hypothesis stems from the reasoning that solely focusing on patches near objects might limit the ability to reconstruct the entire image effectively. For instance, after capturing sufficient details about the object's regions, incorporating information from the background rather than continuing to prioritize the object area could provide the additional context necessary for a more complete reconstruction.

An experiment was conducted to combine the two types of patches in various ratios, each tested with different total numbers of patches. The results are illustrated in Figure 2(c). We observed that mixing the two approaches in approximately equal proportions consistently yielded the best overall performance, emphasizing the importance of balancing attention-based selection and randomness for effective image reconstruction. We can examine that as the number of patches increased to around 60, the object-related information became sufficient, and incorporating background details alongside object features further improved reconstruction results. This highlights the effectiveness of maintaining a balanced approach for varying patch counts.

Based on observations O1 and O2, the intra-task scheduler selects patches equal proportion (0.5 ratio of randomly chosen patches) balancing attention-based and randomly chosen patches. For each task, the scheduler tracks the number of patches selected from both the attention value order and the random order, ensuring this balance. If more patches are selected from one order, the next patch is chosen from the other to restore equilibrium (while being initially selected from the attention value order). For example, if three patches are selected from the attention order and two from the random order, the next patch will be chosen from the random order. These patches are then organized for task-wise selection and dispatched by the inter-task scheduler. It is important to note that the intra-task scheduler can leverage attention information from the ViT only when GPU resources are available, which is handled by the GPU execution launcher.

C. Inter-Task Scheduler

To satisfy R1, each image $I_{i,x}$ of τ_i must transmit at least n_i patches before the next image $I_{i,x+1}$ of τ_i is captured. To do this, we prioritize patches from images with $\mathcal{S}_{i,x}$=GS, which implies applying two different scheduling policies depending on the system mode. First, if the system mode is the ground mode (i.e., $\mathcal{M}$=GM) meaning that there is at least one image with $\mathcal{S}_{i,x}$=GS, we apply a scheduling policy that achieves R1. Second, if the system mode is the extended mode (i.e., $\mathcal{M}$=EM) meaning that there is no such image, we apply a scheduling policy that achieves R2.

Achieving R1. For the ground system mode, while we apply most (if not all) prioritization policies to the prioritized patches from images $\mathcal{S}_{i,x}$=GS, we choose to apply preemptive EDF (Earliest Deadline First) with a minimum preemption time unit of Δ, due to its optimality of scheduling periodic tasks in meeting their deadlines [16]. Then, we adapt the well-known EDF schedulability condition [16] to guarantee R1, as follows.

Lemma 1: If a set of camera sensors τ satisfies the following condition, every image $I_{i,x}$ of $\tau_i \in \tau$ is guaranteed to send at least n_i patches within its deadline.

$$\sum_{\tau_i \in \tau} \frac{n_i \cdot \Delta}{T_i - \Delta} \leq 1.0 \qquad (1)$$

Proof: Since our system operates similarly to preemptive EDF on a uniprocessor where the total execution time of each $I_{i,x}$ is $n_i \cdot \Delta$, we refer to Theorem 9 from [16]. According to [16], it is proven that in a preemptive EDF scheduling on a uniprocessor, if the total utilization of the task set does not exceed 1.0, the timely execution of every job of the task set is guaranteed.

However, unlike [16], our scheduler operates with a minimum time unit Δ and exhibits non-preemptive characteristics for sending patches. Once the sending of patch $P_{i,x}$ of τ_i begins, it cannot be interrupted by any other patches for the interval of Δ, which is equivalent to reducing T_i of Theorem 9 from [16] to $T_i - \Delta$, which proves the lemma. ■

Achieving R2. Once the system mode transitions from GM to EM, we need to select the patch to be transmitted among the patches of different images, in order to improve the sum of expected accuracy of all images whose deadlines have not yet passed. To this end, we analyze the relationship between the number of patches to be used for the image reconstruction and the resulting accuracy of the image-processing mission task, as follows.

O3. We plot the accuracy improvement according to increasing the number of patches used for the image reconstruction. As shown in Figure 2(d), we observe that the function $f(x)$ is increasing and approximately concave, which provides a potential to establish a foundation for optimally selecting the patch for each time for maximizing the sum of the accuracy of all images.

From the empirical results in Figure 2(d), we utilize the property of the function, being increasing and concave. That is, as the number of patches used for the image reconstruction (denoted by x) increases, the resulting accuracy difference between applying x and applying $x+1$ decreases. Therefore, to maximize the expected accuracy improvement using one additional patch for the image reconstruction, we need to choose the patch from an image whose number of patches to be ready for the image reconstruction is the smallest, which is how the inter-task scheduler works.

In summary, under the ground system mode (GM), the inter-task scheduler selects the patch to be transmitted by the EDF policy among patches whose respective images are in the ground state, achieving R1. Under the extended system mode (EM), the inter-task scheduler selects the patch whose number of transmitted patches is the smallest, achieving R2.

D. Overall Workflow

According to the mechanisms of the intra-task and inter-task schedulers, we present the overall workflow with numbers in Figure 1. ①: When an image is generated by a camera sensor, it is divided into patches and added to the priority queue of the corresponding task in the intra-task scheduler. ②: At the same time, the image is sent to the GPU execution launcher to calculate attention values (only if the GPU is available). ③: Initially, as the patches do not yet have assigned priorities, they are temporarily prioritized at random. ④: Once the GPU execution launcher completes the attention value calculations, the results are sent to the intra-task scheduler. ⑤: The intra-task scheduler then updates the priority of each patch based on the attention value order. ⑥: The inter-task scheduler then checks the system mode. With the ground system mode, the scheduler uses EDF policy to prioritize and schedule patches belonging to images with the ground state. With the extened system mode, the scheduler selects the image with the fewest transmitted patches and sends the highest-priority patch of the image from the priority queue in the intra-task scheduler.

This process ensures that the hierarchical scheduler not only provides consistent minimal operational performance (achieving R1), but also maximizes overall system performance (achieving R2), by dynamically adapting the current system mode according to individual image states.

IV. Feedback Mechanism

Network instability may cause transmitted patches from the device to be lost or corrupted before reaching the server. The device relies on its transmitted patch count for scheduling decisions, but discrepancy can arise if the server receives fewer patches. This discrepancy hinders the system's ability to guarantee a minimum level of accuracy within the required time frame. For instance, the device might assume that all n_i patches for each task τ_i's image have been successfully transmitted and switch the state of the image to ES. At the same time, the server has yet to receive all n_i patches. Consequently, this can lead to a failure to transmit as many patches of an image as the ground patch count n_i. To address this issue, our system incorporates a feedback mechanism through which the server periodically provides the edge device with information on the number of patches received and more. Based on this feedback, the edge device then updates its transmitted patch count and adjusts its scheduling policy accordingly.

A. Information Exchange

Piggyback from edge device to server. The edge device maps two indices to each patch before transmission: one specifying the patch's position within the image, and another reflecting the priority of patches within the image, which is assigned by the intra-task scheduler. These indices are piggybacked onto the corresponding patch to be transmitted to the server.

Feedback from server to edge device. Upon receiving the patches, the server utilizes the position indices to execute the MAE reconstruction process effectively. Periodically, the server provides feedback to the edge device through *periodic feedback message*, detailing the priority indices of all received patches for a given image. This feedback enables the edge device to refine its scheduling and transmission strategies through the hierarchical scheduler.

Feedback information processing at edge device. With the server's feedback, the edge device identifies the largest priority index among the received patches as the latest one acknowledged by the server. Using this information, the device estimates how many patches the server has received and how many remain to be transmitted for the current image. By analyzing the scheduler priority indices in the feedback, the device compares the priority indices of sent patches with those received by the server to identify lost patches and prioritize their retransmission.

B. Feedback into Hierarchical Scheduler

To further maintain stable task performance, the edge device integrates feedback mechanisms, which adapt the hierarchical scheduler for robust operation under network failures. On a long-term scale, feedback analyzes historical loss rates received from the server, updating the expected loss rate and determining the additional patches needed for consistent performance. While on a short-term scale, it focuses on immediate discrepancies, comparing the patches sent by the device with those received by the server. This process allows the device to dynamically adjust its mode and recalculate the required additional patches by referencing the loss rate, ensuring timely and accurate delivery even in fluctuating network conditions. Together, these mechanisms enable the edge device to balance real-time adjustments with sustained efficiency.

Applying periodic feedback on a long-term scale. Utilizing the periodic feedback message on a long-term scale, the system adjusts the network loss rate based on past loss history, adding a margin to the ground patch count to support minimal operational performance. Since the feedback message on a long-term scale aims to add a margin to ensure the ground patch count, it should not fluctuate significantly in response to a single large network failure. Moreover, it is used solely for statistical purposes, ensuring that images whose deadlines have not yet passed remain unaffected during the period of receiving feedback. Instead, the adjustments are applied to subsequent tasks to maintain consistent performance over time.

The calculation for feedback messages on a long-term scale is as follows: it is based on the average loss rate from the most recent H periodic feedback messages received from the server. If fewer than H number of records are available, the average is calculated from all available feedback. The loss rate calculated through feedback (denoted by γ) is continuously applied, adjusting n_i for each τ_i upon receipt at the server (i.e., $n_i \cdot \frac{1}{1-\gamma}$). This adjustment ensures that accuracy requirements are met efficiently, as the system maintains a stable margin for ground patch count transmissions without overreacting to isolated network failures.

Applying periodic feedback on a short-term scale. Server feedback enables the system to adjust the transmitted ground patch count n_i to meet target accuracy. Unlike long-term feedback, which smooths network fluctuations based on average drop rates, short-term feedback directly impacts images with unpassed deadlines during the feedback reception period. This allows the edge device to adjust n_i based on current transmission success rates.

Server feedback includes indices of successfully received patches, as assigned by the intra-task scheduler. Using this feedback, the device determines up to which index the server has received patches. If the latest index is smaller than n_i, the device identifies that the server has not yet received all required patches for the image. This allows the device to pinpoint missing patches, prioritize them at the top of the intra-task priority queue, and retransmit them efficiently.

Each image $I_{i,x}$ starts in the state $S_{i,x}$ set to GS. While in GS, the edge device operates in GM and continues sending patches. Upon receiving feedback, the system checks whether n_i patches have been successfully received by the server. Once the required number of patches (n_i) is confirmed to have been transmitted ($n_{\text{sent}} \geqslant n_{\text{recv}}$), $S_{i,x}$ transitions from GS to ES. When all images transition to ES, the system switches its mode to EM. This calculation incorporates the loss rate (γ) to ensure efficient and accurate delivery while accounting for network inconsistencies.

If the server's feedback indicates that n_{recv}, the number of patches received, is less than n_i and $S_{i,x}$ is in ES, this reveals a discrepancy where the system prematurely transitioned the image state from GS to ES despite incomplete patch delivery. In such cases, the system reverts the state of the image to GS and the overall mode to GM. The system then recalculates the total number of patches required, including patches still pending transmission and those requiring retransmission.

V. Evaluation

A. Experiment Setup

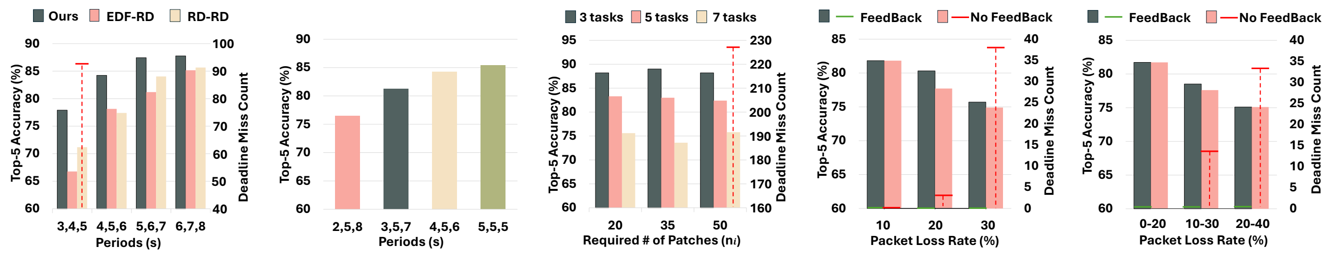
Hardware. The server operates on a Windows 11 operating system with an AMD Ryzen 5 7500F 6-Core Processor and an NVIDIA RTX 4060 Ti GPU. The edge device operates on Linux Ubuntu, powered by an Nvidia Jetson Orin Nano, which incorporates a 1024-core NVIDIA Ampere architecture GPU, 32 Tensor Cores, and a 6-core ARM Cortex-A78AE v8.2 64-bit CPU.

Dataset. For our experiments, we used 40,000 image samples from the ImageNet 1K validation set [17] to construct observation graphs and 1,000 random image samples for network experiments. Each image was resized to 224x224 pixels and normalized per the ImageNet standard. Performance was evaluated on an image classification task using the top-5 accuracy metric, widely used in previous studies [9], [18].

B. Evaluation Result

The edge device periodically transmits a single image patch every 30ms, maintaining a steady data flow. For this evaluation, we leverage the ViT [14] model with 12 attention layers, to calculate attention values for image patches.

ViT layer selection & GPU-independent operation. To determine a ViT layer that maximizes the accuracy, experiments were conducted with five settings: selecting patches randomly, and based on the attention values from the 3rd, 4th, 5th, and 6th ViT layers; each setting was tested with periods of 1, 1.5, and 2 seconds. As shown in Table I, selecting patches from the 5th ViT layer not only yields 3.7%–21.9% accuracy improvement over random patch selection, but also



(a) Accuracy with increasing periods (b) Accuracy with different period distributions (c) Accuracy and deadline miss with diferent number of tasks and n_i (d) Accuracy and deadline miss with fixed network loss rates (e) Accuracy and deadline miss under fluctuating network loss rates

Fig. 3: Evaluation results

maximizes the accuracy among those from different ViT layers. Therefore, subsequent experiments will use the 5th layer, balancing accuracy and resource efficiency.

The experiment results in Table I also indicate that the random patch selection maintains reasonable performance, achieving 59.0%–84.2% accuracy. The system can still function reliably even when the GPU computing resource is occupied by operation-critical tasks, which is different from prior studies that rely on encoder computations [9], [10], making a direct comparison less applicable.

Period	Layer 3	Layer 4	Layer 5	Layer 6	Random
1.0	64.5	67.8	**71.9**	68.4	**59.0**
1.5	78.8	81.7	**82.8**	82.4	**76.9**
2.0	85.2	88.0	**87.3**	87.5	**84.2**

TABLE I: Accuracy comparison of random and intermediate attention across different layers.

Performance for varying task periods. We evaluate the performance of our system by comparing it against two settings. In the first setting (EDF-RD), the inter-task scheduler in Ground Mode (GM) follows EDF, while in Extended Mode (EM), the inter-task scheduler chooses patches randomly. For both modes, the intra-task scheduler selects patches at random. In the second setting (RD-RD), both inter-task and intra-task schedulers randomly choose images and patches at any time.

First, we evaluate three offloading tasks with periods increasing by 1 second in four configurations: (3s,4s,5s), (4s,5s,6s), (5s,6s,7s), and (6s,7s,8s). For all configurations, n_i for each task is set to (30,25,20), respectively, and Lemma 1 is satisfied. Note that the value of n_i is chosen to be at most 20 according to the observation from Figure 2(b) for minimum level of accuracy. The settings reflect real-world scenarios, where critical tasks require shorter periods and higher guaranteed accuracy.

As shown in Figure 3(a), our system outperforms EDF-RD and RD-RD in all configurations. For the (3s,4s,5s) period configuration, our system achieves 6% and 11% higher accuracy compared to RD-RD and EDF-RD, respectively. Notably, RD-RD fails to guarantee the deadline of n_i in 92 instances, while our system and EDF-RD consistently meet the deadline of n_i across all cases, demonstrating the effectiveness of applying EDF in GM to the inter-task scheduler. Longer task periods allow more patches to be transmitted, improving reliability in object identification and narrowing accuracy gap between system and the settings. In contrast, shorter periods with fewer patches widen performance gap, highlighting the robustness of our approach.

We also conduct an evaluation to determine whether our system operates robustly even when the period distribution varies. The periods are set to (5s,5s,5s), (4s,5s,6s), (3s,5s,7s), and (2s,5s,8s), with n_i fixed at (30,25,20) in all cases. As shown in Figure 3(b), the system achieves an accuracy of 85.4% when the periods are (5s,5s,5s), demonstrating high performance, whereas it shows a lower accuracy of 76.5% for the periods (2s,5s,8s). This difference is likely due to the lower utilization of GS (i.e., the left-hand side of Eq. (1)) in (5s,5s,5s) compared to the higher utilization of GS in (2s,5s,8s). Thus, when the utilization of GS is high, the time the device spends in extended mode decreases. Consequently, the number of patches the device can transmit for a single image is reduced. Despite some variations caused by utilization, our system ensures a minimum level of accuracy, even when the period distribution changes.

Performance for varying the number of tasks. We evaluate the accuracy of our system by increasing the number of tasks (i.e., camera sensors) to 3, 5 and 7, respectively having periods of 6s–8s, 6s–10s and 6s–12s with a step size of 1s. We test 3 different settings of n_i: 20, 35 and 50 for all tasks of each case. As shown in Figure 3(c), the accuracy varies with 88.2%–89.1% when there are three tasks. As the number of tasks increases, the accuracy gradually decreases to 82.4%–83.3% for five tasks and 73.6%–75.9% for seven tasks, as each task transmits fewer patches; however, the system remains fully operational.

We also observe that all tasks meet their deadlines except when $n_i = 50$ with 7 tasks, where 227 deadline misses occur. The exception occurs because the left-hand side of Lemma 1 exceeds 1.0, making the task set unschedulable. This validates Lemma 1's schedulability test as a guideline for determining appropriate task set sizes.

Performance under unstable network condition. To evaluate the robustness of our system in unstable network conditions, we conducted two experiments: one with a fixed packet loss rate and another with a fluctuating packet loss rate. In both experiments, the task periods for three tasks were set to (4s,5s,6s), respectively, with n_i values (30,25,20). We set the value of H to 30, meaning it uses the latest 30 periodic feedback messages to calculate the average loss rate. While a shorter cycle would enhance performance, the resulting overhead becomes non-negligible. For this reason, the server is configured to send a feedback message once per second. The packet loss rate is set to 10%, 20%, and 30% in the fixed loss rate experiment, while during the fluctuating loss rate experiment, the packet loss rate changes every 5 seconds, alternating between ranges of 0%–20%, 10%–30%, and 20%–40%, thereby simulating realistic network dynamics such as burst losses. Results show that, in the fixed loss rate experiment in Figure 3(d), our system with a feedback mechanism achieved higher accuracy and consistently transmitted n_i patches within deadlines, unlike the system without feedback, which missed 38 deadlines at a 30% loss rate. In the fluctuating loss rate experiment in Figure 3(e), our system with the feedback mechanism performed significantly better in guaranteeing the transmission of n_i patches by dynamically adjusting to network conditions, while the system without feedback frequently failed to meet deadlines. Although the classification accuracy difference between the two systems was not significant, our feedback mechanism reliably transmitted n_i patches under high loss rates, demonstrating its effectiveness in maintaining performance and reliability in challenging network conditions.

Computing resource usage. Our system periodically sends one patch every 30ms, with the process of splitting into patches and sending to the network module taking about 0.1ms and scheduling around 0.05ms. This results in an average CPU utilization of about 0.1%. GPU utilization varies by layer, with approximately 1.17%, 1.85% and 2.22% respectively for the third, fourth and fifth layers, demonstrating efficient resource usage. The time-complexity of scheduling is $O(n)$, where n represents the number of tasks (i.e., camera sensors), and the feedback mechanism only requires updating the priority queue and estimating the network loss rate based on the received information. Considering the number of camera sensors in each edge device is typically small in real-world scenarios, the overhead remains marginal. The MAE reconstruction model, with its 1.25GB memory footprint and 12ms inference time, presents no operational issues given resources available on server.

VI. Conclusion

This paper proposed a robust offloading framework for resource-constrained edge devices operating under unstable network conditions. Leveraging Masked Autoencoders, the framework ensures a minimum accuracy level (R1) and maximizes aggregated accuracy (R2) for image-processing tasks. The hierarchical scheduler dynamically prioritizes tasks and patches, while the feedback mechanism adapts to network instability to maintain performance. Experimental results demonstrated higher accuracy and consistent task reliability compared to baseline systems. Future work could focus on optimizing feedback intervals and expanding the framework to accommodate diverse scenarios.

Acknowledgement

This work was supported in part by (1) the National Research Foundation of Korea (NRF) grant (RS-2024-00438248, RS-2023-00213309) and (2) Institute of Information & communications Technology Planning & Evaluation (IITP) grant (RS-2024-00398157) funded by the Korea government (MSIT).

References

[1] H. Hua, Y. Li, T. Wang, N. Dong, W. Li, and J. Cao, “Edge computing with artificial intelligence: A machine learning perspective,” *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[2] K. Feng, Z. Lu, J. Xu, H. Chen, and Y. Lou, “A safety filter for realizing safe robot navigation in crowds,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 9729–9736.

[3] L. Zhao, J. Hu, J. Bi, Y. Bai, E. Mas, and S. Koshimura, “Streamlining forest wildfire surveillance: Ai-enhanced uavs utilizing the flame aerial video dataset for lightweight and efficient monitoring,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 8063–8068.

[4] B. Mishra, D. Garg, P. Narang, and V. Mishra, “Drone-surveillance for search and rescue in natural disaster,” *Computer Communications*, vol. 156, pp. 1–10, 2020.

[5] T. Zhao, Y. Xie, Y. Wang, J. Cheng, X. Guo, B. Hu, and Y. Chen, “A survey of deep learning on mobile devices: Applications, optimizations, challenges, and research opportunities,” *Proceedings of the IEEE*, vol. 110, no. 3, pp. 334–354, 2022.

[6] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, “Challenges and opportunities in edge computing,” in *2016 IEEE international conference on smart cloud (SmartCloud)*, 2016, pp. 20–26.

[7] A. Islam, A. Debnath, M. Ghose, and S. Chakraborty, “A survey on task offloading in multi-access edge computing,” *Journal of Systems Architecture*, vol. 118, p. 102225, 2021.

[8] H. Yan, X. Zhang, H. Chen, Y. Zhou, W. Bao, and L. T. Yang, “Deed: Dynamic energy-efficient data offloading for iot applications under unstable channel conditions,” *Future Generation Computer Systems*, vol. 96, pp. 425–437, 2019.

[9] R. Wang, H. Liu, J. Qiu, M. Xu, R. Guérin, and C. Lu, “Progressive neural compression for adaptive image offloading under timing constraints,” in *IEEE Real-Time Systems Symposium (RTSS)*, 2023, pp. 118–130.

[10] C. Liebender, R. Bezerra, K. Ohno, and S. Tadokoro, “Region of interest loss for anonymizing learned image compression,” in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2024, pp. 3569–3576.

[11] J. Liu, H. Sun, and J. Katto, “Learned image compression with mixed transformer-cnn architectures,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 14 388–14 397.

[12] T. Liu, P. Li, Y. Gu, P. Liu, and H. Wang, “Adaptive offloading of transformer inference for weak edge devices with masked autoencoders,” *ACM Transactions on Sensor Networks*, 2024.

[13] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” in *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022, pp. 16 000–16 009.

[14] A. Dosovitskiy, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.

[15] L. Wu, S. Guo, Y. Ding, J. Wang, W. Xu, R. Y. Xu, and J. Zhang, “Demystify self-attention in vision transformers from a semantic perspective: Analysis and application,” *arXiv preprint arXiv:2211.08543*, 2022.

[16] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment,” *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[18] J. Qiu, R. Wang, A. Chakrabarti, R. Guérin, and C. Lu, “Adaptive edge offloading for image classification under rate limit,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 3886–3897, 2022.