Optimized Cooperative Inference for Energy-Efficient and Low-Latency Mobile Edge Computing

Hyun-Ho Choi
ICT and Robotics Engineering
Hankyong National University, South Korea
hhchoi@hknu.ac.kr

Kisong Lee
Information and Communication Engineering
Dongguk University, South Korea
kslee851105@gmail.com

***Abstract*—To overcome the limitations of standalone inference on edge devices or servers, we propose a cooperative inference method for mobile edge computing (MEC) systems. Using dual confidence thresholds on a small neural network (NN) at the edge, ambiguous images are filtered and sent to a larger NN on the server for reevaluation. We evaluate the method's accuracy, delay, and energy consumption, accounting for confidence score distributions that could trigger false alarms. A joint optimization problem is formulated to minimize delay and energy consumption by selecting optimal confidence thresholds, transmit power, and duty cycle while ensuring accuracy. Experimental results show that this approach significantly reduces delay and energy consumption while achieving higher accuracy than device-only inference and lower costs than server-only inference in various MEC scenarios.**

***Index Terms*—Cooperative inference, mobile edge computing (MEC), confidence thresholds, joint optimization.**

I. Introduction

Artificial intelligence (AI) functions, often implemented through deep neural networks (DNNs), demand substantial computational resources for precise inference. This presents a significant challenge when deploying DNN models directly on edge devices such as sensors, cameras, and mobile phones, which typically have limited processing power and memory. Running standard-sized DNNs on these resource-constrained devices increases inference time, making real-time detection impractical. While using compressed or smaller DNNs designed for edge devices can alleviate this issue, it inevitably results in a decline in performance concerning accuracy and reliability.

To tackle this challenge, mobile edge computing (MEC) introduces powerful servers located near base stations (BS) [1]. In this architecture, edge devices offload intensive tasks to edge servers, enabling faster and more accurate processing. However, this increases data traffic to the BS, resulting in transmission delays and higher energy consumption. Unreliable wireless channels can corrupt data, requiring retransmissions, and server overloads can cause queuing delays. Uploading raw data also introduces privacy concerns [2].

To address the limitations of standalone inference on either edge devices or servers, cooperative inference has emerged as a hybrid method [3]–[6]. Various architectures and frameworks now support DNN training and inference across end devices, edge servers, and cloud centers [3]. Collaborative DNN inference architectures balance computational cost and communication overhead in MEC networks through model splitting, compression, and feature encoding techniques [5]. Additionally, distributed computing systems dynamically partition DNN inference across heterogeneous edge devices based on their capabilities and network conditions [6].

In practical MEC applications, cooperative inference techniques have been successfully applied [7]–[10]. An edge-cloud co-inference method using model splitting and compression was evaluated for real-time video processing [7]. For risk detection on construction sites, a method employed tiny-YOLO for coarse edge detection, escalating to YOLOv3 on the central server for precise detection [8], later extended to an intelligent edge surveillance system [9]. Additionally, a Raspberry Pi-based prototype performed initial motion detection on edge, with suspicious images processed by a cloud server for object detection [10].

Previous studies have focused on system architecture, algorithms, and evaluations [3]–[6], but have overlooked the optimization of confidence thresholds, a key parameter affecting inference performance. Typically, confidence thresholds were either set to default values or adjusted heuristically. Additionally, most MEC systems have not considered negative images that may be similar to or mistaken for the positive images they are supposed to detect [7]–[10]. For instance, fog or clouds may be misinterpreted as smoke; sunlight or car lights might be perceived as fire. Considering such potential negative images is important in system design because false alarms triggered by these images can inconvenience users, and if repeated, the system could lose the trust of its customers.

To resolve these gaps, we propose a novel cooperative inference method where the edge device uses two confidence thresholds to filter ambiguous input images. Only these ambiguous images are sent to the edge server, which applies a single confidence threshold for final evaluation. We analyze inference accuracy, delay, and energy consumption based on the confidence score distributions of both positive and negative images, as well as the device's transmit power and duty cycle. We then formulate an optimization problem to find the optimal confidence thresholds, transmit power, and duty cycle that minimize the weighted sum of delay and energy consumption while maintaining accuracy. Results indicate a trade-off between accuracy and cooperation costs, and confirm that

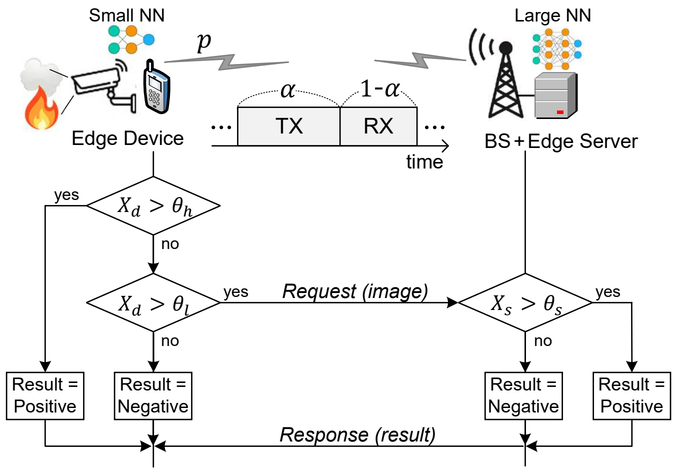

Fig. 1. Operation flow of proposed cooperative inference.

the proposed method achieves higher accuracy than device-only inference while reducing costs compared to traditional approaches in various MEC environments.

The rest of this paper is organized as follows: Section II details the proposed cooperative inference method. Section III analyzes its performance in terms of accuracy, delay, and energy consumption. Section IV presents the evaluation results, and Section V concludes the study.

II. Proposed Cooperative Inference

Fig. 1 illustrates the operation flow of the proposed cooperative inference in a network with an edge device, a base station (BS), and an edge server. The edge device (e.g., a camera or smartphone) connects wirelessly to the BS, and the edge server is directly connected to the BS [11]. Inference tasks, such as object detection or image classification, are performed collaboratively between the edge device and server using DNNs. The edge device, limited by its processing power and memory, uses a small NN, while the server uses a larger NN without such constraints [9]. Although the edge server can handle multiple devices, we simplify the model by considering a single device and server, as each operates independently with different scenes [7].

In the proposed cooperative inference, the edge device uses two confidence thresholds (θ_l and θ_h, where $\theta_l \le \theta_h$), and the edge server uses a single threshold (θ_s). Let X_d denote the confidence scores obtained from the edge device and X_s those obtained from the server. First, if $X_d > \theta_h$ at the device, the device determines that the confidence in the inference is sufficient and independently decides to consider the inference result as *positive*, i.e., an event of interest has occurred. If not, and if X_d is greater than θ_l (i.e., $\theta_l < X_d \le \theta_h$), the device determines that the confidence score is too ambiguous to decide and therefore delegates this decision to the server by transmitting the corresponding image data to it. Lastly, if $X_d \le \theta_l$, the device determines that the inference result is *negative*, i.e., an event has not occurred. That is, if the confidence scores are sufficiently low or high, the device trusts its own judgment and concludes the inference by itself. However, if the confidence scores fall into an ambiguous range, the device requests additional inference to the server. This is motivated by the fact that the edge device uses a small NN that results in lower inference accuracy compared with the server. To achieve this, the edge device must operate with *dual thresholds*, which is a distinguishing feature from the conventional inference methods that use a single confidence threshold to decide whether to proceed with further inference [7]–[10].

Meanwhile, the edge server performs inference using its large NN on the images received from the edge device. If the server's confidence score X_s is greater than θ_s (i.e., $X_s > \theta_s$), the server determines the inference result as *positive* and replies with this result to the edge device. Otherwise (i.e., $X_s \le \theta_s$), the server responds with the result set as *negative* to the edge device. In this way, the server acts as the final decision-maker for ambiguous images. This is reasonable because the server can employ a large NN and perform more accurate inferences. Therefore, this collaboration between the edge device and the edge server can enhance accuracy. However, it may increase energy consumption due to uplink transmission and cause longer delays from transmission and server processing.

To minimize cooperation overhead, efficient use of wireless resources is essential. In this context, we examine the allocations of the device's transmission power and transmission/reception time. As shown in Fig. 1, we adjust the device's transmit power (p) and duty cycle (α) when an α portion of the fixed frame is used for transmission, while the remaining $1-\alpha$ portion is used for reception [12]. For instance, increasing the transmit power p can improve transmission rate and reduce delay but increases energy consumption. Additionally, the duty cycle α must account for uplink/downlink rate differences and the size disparity between transmitted image data and received messages. Thus, transmit power, duty cycle, and confidence thresholds must be jointly optimized to balance accuracy, delay, and energy consumption.

III. Performance Analysis

In this section, we numerically analyze the performance of inference accuracy, total delay, and energy consumed by the end device for device-only inference, server-only inference, and proposed cooperative inference methods. We denote the confidence scores of images processed at the device and server as X_d^{p}, X_d^{n}, X_s^{p}, and X_s^{n}, respectively, according to whether the ground truth of the image is positive or negative. Then, their cumulative distribution functions (CDFs) are defined as $F_{X_d^{\mathrm{p}}}(x)$, $F_{X_d^{\mathrm{n}}}(x)$, $F_{X_s^{\mathrm{p}}}(x)$, and $F_{X_s^{\mathrm{n}}}(x)$, respectively. That is, $F_{X_i^{\mathrm{j}}}(x) = P(X_i^{\mathrm{j}} \le x)$ where $i \in \{d, s\}$ and $\mathrm{j} \in \{\mathrm{p}, \mathrm{n}\}$.

A. Device-Only Inference

The device-only inference method determines the occurrence of an event on the device itself without the assistance of the server, using a single confidence threshold, θ_d. If the confidence score of the input image exceeds θ_d, the device determines that an event has occurred (i.e., positive). Otherwise, it determines the result as negative.

From this typical classification, the confusion matrix at the device, i.e., true positive (TP), false negative (FN), false positive (FP), and true negative (TN), is expressed as

$$TP_d = 1 - F_{X_d^{\mathrm{p}}}(\theta_d), \quad FN_d = F_{X_d^{\mathrm{p}}}(\theta_d), \tag{1}$$

$$FP_d = 1 - F_{X_d^{\mathrm{n}}}(\theta_d), \quad TN_d = F_{X_d^{\mathrm{n}}}(\theta_d). \tag{2}$$

Subsequently, the accuracy is defined as

$$A = \frac{TP_d + TN_d}{TP_d + FN_d + FP_d + TN_d} = \frac{TP_d + TN_d}{2}. \tag{3}$$

Moreover, the delay is simply expressed as the time it takes for the edge device to perform the inference, and its inference time is modeled as

$$D = T_d^{\mathrm{in}} = \frac{N_d \mathcal{X}}{\nu_d}, \tag{4}$$

where N_d represents the number of floating point operations (FLOPs) required by the NN used in the edge device for inference, $\mathcal{X}$ is the number of central processing unit (CPU) cycles required to process one FLOP, and ν_d is the frequency of the CPU used in the edge device, measured in Hz.

Additionally, the energy consumed by the device for performing this inference can be expressed as

$$E = E_d^{\mathrm{in}} = T_d^{\mathrm{in}} P_p = \frac{N_d \mathcal{X} P_p}{\nu_d}, \tag{5}$$

where P_p represents the power consumed in processing at the edge device, measured in watts.

B. Server-Only Inference

In the server-only inference method, the edge device sends every input image to the server and fully delegates inference to the server. The server performs inference on the received images based on its confidence threshold θ_s to determine the occurrence of events. Hence, the confusion matrix at the server is similarly obtained as

$$TP_s = 1 - F_{X_s^{\mathrm{p}}}(\theta_s), \quad FN_s = F_{X_s^{\mathrm{p}}}(\theta_s), \tag{6}$$

$$FP_s = 1 - F_{X_s^{\mathrm{n}}}(\theta_s), \quad TN_s = F_{X_s^{\mathrm{n}}}(\theta_s). \tag{7}$$

Subsequently, the accuracy is determined as

$$A = \frac{TP_s + TN_s}{TP_s + FN_s + FP_s + TN_s} = \frac{TP_s + TN_s}{2}. \tag{8}$$

The transmission and reception rates, R_t and R_r, between the edge device and the BS can be expressed using the Shannon capacity as follows:

$$R_t = \alpha W \log_2\left(1 + \frac{gp}{N_0 W}\right), \tag{9}$$

$$R_r = (1-\alpha) W \log_2\left(1 + \frac{gP_b}{N_0 W}\right), \tag{10}$$

where g is the channel power gain between the device and the BS assuming channel reciprocity, W is the bandwidth of the wireless channel used, P_b is the constant transmit power of the BS, and N_0 is the power spectral density of the noise.

Let f denote the frames per second (fps) of images captured by the camera on the edge device, S_t denote the bit size of an image transmitted to the BS, and S_r denote the bit size of a result message received from the BS. Then, the traffic amount transmitted or received by the device per second is fS_t and fS_r, respectively. Therefore, the transmission and reception times for exchanging data between the device and the BS are calculated as

$$T^{\mathrm{tx}} = \frac{fS_t}{R_t} = \frac{fS_t}{\alpha W \log_2\left(1 + \frac{gp}{N_0 W}\right)}, \tag{11}$$

$$T^{\mathrm{rx}} = \frac{fS_r}{R_r} = \frac{fS_r}{(1-\alpha) W \log_2\left(1 + \frac{gP_b}{N_0 W}\right)}. \tag{12}$$

Thus, the total delay for inference is the sum of the transmission delay, the server's inference time, and the reception delay, which can be expressed as

$$\begin{aligned} D &= T^{\mathrm{tx}} + T_s^{\mathrm{in}} + T^{\mathrm{rx}} \\ &= \frac{fS_t}{\alpha W \log_2\!\left(1+\frac{gp}{N_0 W}\right)} + \frac{N_s \mathcal{X}}{\nu_s} + \frac{fS_r}{(1-\alpha) W \log_2\!\left(1+\frac{gP_b}{N_0 W}\right)}, \end{aligned} \tag{13}$$

where N_s and ν_s indicate the number of FLOPs required by the NN and the frequency of the processing unit used in the edge server, respectively. Here, the transmission time between the BS and the edge server is ignored, as they are physically close to each other [].

For this server-only inference operation, the edge device does not consume energy for inference processing but must use energy for transmission and reception. Therefore, the energy consumption of the device is given by

$$\begin{aligned} E &= E^{\mathrm{tx}} + E^{\mathrm{rx}} \\ &= \mathcal{E}_c + \frac{fS_t p}{\alpha W \log_2\left(1+\frac{gp}{N_0 W}\right)} + \frac{fS_r P_r}{(1-\alpha) W \log_2\left(1+\frac{gP_b}{N_0 W}\right)}, \end{aligned} \tag{14}$$

where $\mathcal{E}_c$ is the constant energy consumed by the device's transceiver, and P_r is the power consumed by the device to receive data.

C. Proposed Cooperative Inference

Fig. 2 shows the probability density function (PDF) of confidence scores at both the edge device and the edge server for confusion matrix analysis in the proposed cooperative inference. First, at the device, the confusion matrix is given by

$$TP_d = 1 - F_{X_d^{\mathrm{p}}}(\theta_h), \quad FN_d = F_{X_d^{\mathrm{p}}}(\theta_l), \tag{15}$$

$$FP_d = 1 - F_{X_d^{\mathrm{n}}}(\theta_h), \quad TN_d = F_{X_d^{\mathrm{n}}}(\theta_l). \tag{16}$$

Moreover, as shown in Fig. 2(a), the use of two confidence thresholds leads us to define *uncertain positive (UP)* as an instance of positive data for which the confidence score is too uncertain to be considered positive. Similarly, *uncertain negative (UN)* is defined as an instance of negative data for

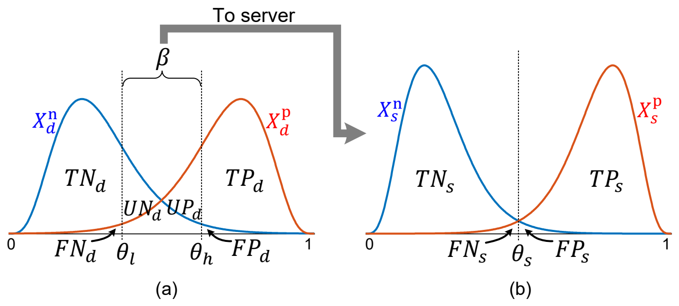

Fig. 2. Confusion matrix analysis for proposed cooperative inference at (a) edge device and (b) edge server.

which the confidence score is too uncertain to be considered negative. Hence, UP and UN at the device are respectively expressed as

$$UP_d = F_{X_d^{\mathrm{p}}}(\theta_h) - F_{X_d^{\mathrm{p}}}(\theta_l), \tag{17}$$

$$UN_d = F_{X_d^{\mathrm{n}}}(\theta_h) - F_{X_d^{\mathrm{n}}}(\theta_l). \tag{18}$$

Thus, the proportion of uncertain images sent to the server, β, is determined as

$$\beta = \frac{UP_d + UN_d}{2}. \tag{19}$$

Meanwhile, the server receives the images classified as uncertain from the device. Letting the confidence scores of these positive and negative images be X_s^{p} and X_s^{n}, respectively, the confusion matrix at the server is represented as

$$TP_s = 1 - F_{X_s^{\mathrm{p}}}(\theta_s), \quad FN_s = F_{X_s^{\mathrm{p}}}(\theta_s), \tag{20}$$

$$FP_s = 1 - F_{X_s^{\mathrm{n}}}(\theta_s), \quad TN_s = F_{X_s^{\mathrm{n}}}(\theta_s). \tag{21}$$

When the server identifies UP_d as positive and UN_d as negative, it has made an accurate decision. Therefore, the accuracy in the proposed cooperative inference is calculated as

$$\begin{aligned} A &= \frac{TP_d + TN_d + UP_d \times TP_s + UN_d \times TN_s}{TP_d + FN_d + FP_d + TN_d + UP_d + UN_d} \\ &= \frac{TP_d + TN_d + UP_d \times TP_s + UN_d \times TN_s}{2}. \end{aligned} \tag{22}$$

In the proposed cooperative inference, the performances of delay and energy consumption vary according to the following two cases. The first case corresponds to the edge device making its own decision without offloading to the server, which occurs with a probability of $1-\beta$. In this case, the delay and energy consumption are the same as those of the device-only inference, as shown in (4) and (5), and can be written respectively as

$$D_1 = T_d^{\mathrm{in}} = \frac{N_d \mathcal{X}}{\nu_d}, \tag{23}$$

$$E_1 = E_d^{\mathrm{in}} = \frac{N_d \mathcal{X} P_p}{\nu_d}. \tag{24}$$

The second case corresponds to the edge device not making a decision and offloading the task to the server, which occurs with a probability of β. In this case, the amounts of traffic transmitted from the device to the server and vice versa are given by $\beta f S_t$ and $\beta f S_r$, respectively. Thus, the transmission and reception time, T^{tx} and T^{rx}, between the device and the BS are expressed as

$$T^{\mathrm{tx}} = \frac{\beta f S_t}{R_t}, \quad T^{\mathrm{rx}} = \frac{\beta f S_r}{R_r}. \tag{25}$$

Hence, the delay in the second case is given by

$$\begin{aligned} D_2 &= T_d^{\mathrm{in}} + T^{\mathrm{tx}} + T_s^{\mathrm{in}} + T^{\mathrm{rx}} \\ &= \frac{N_d \mathcal{X}}{\nu_d} + \frac{\beta f S_t}{\alpha W \log_2\left(1 + \frac{gp}{N_0 W}\right)} + \frac{N_s \mathcal{X}}{\nu_s} \\ &\quad + \frac{\beta f S_r}{(1-\alpha) W \log_2\left(1 + \frac{g P_b}{N_0 W}\right)}. \end{aligned} \tag{26}$$

In addition, the energy consumption of the device is represented as

$$\begin{aligned} E_2 &= E_d^{\mathrm{in}} + E^{\mathrm{tx}} + E^{\mathrm{rx}} \\ &= \frac{N_d \mathcal{X} P_p}{\nu_d} + \mathcal{E}_c + \frac{\beta f S_t p}{\alpha W \log_2\left(1 + \frac{gp}{N_0 W}\right)} \\ &\quad + \frac{\beta f S_r P_r}{(1-\alpha) W \log_2\left(1 + \frac{g P_b}{N_0 W}\right)}. \end{aligned} \tag{27}$$

Finally, from (23), (24), (26), and (27), the average delay and energy consumption in the proposed cooperative inference are respectively obtained as

$$D = (1-\beta) D_1 + \beta D_2, \tag{28}$$

$$E = (1-\beta) E_1 + \beta E_2. \tag{29}$$

D. Proposed Optimization Problem

We need to increase accuracy while decreasing delay and energy consumption. In relation to these performance metrics, we can control the confidence thresholds of the edge device and server ($\vec{\theta} \triangleq [\theta_l, \theta_h, \theta_s]$), the transmit power of the edge device (p), and the duty cycle (α). Therefore, we formulate an optimization problem that minimizes the cooperation cost, $C(p, \alpha, \vec{\theta})$, defined as the weighted sum of delay and energy consumption, while maintaining a certain level of accuracy, as follows:

$$\min_{p, \alpha, \vec{\theta}} \quad C\left(p, \alpha, \vec{\theta}\right) = \omega D + (1-\omega) E \tag{30}$$

$$\text{s.t.} \quad A \geq \Gamma, \tag{31}$$

$$0 \leq p \leq P_{max}, \tag{32}$$

$$0 \leq \alpha \leq 1, \tag{33}$$

$$0 \leq \theta_l \leq \theta_h \leq 1, \quad 0 \leq \theta_s \leq 1, \tag{34}$$

where ω is the weighting factor to balance between delay and energy consumption, Γ is the required accuracy, and P_{max} is the maximum transmit power of the edge device. Note that we solved this optimization problem by applying traditional convex optimization techniques and utilizing a greedy search. However, due to space limitations, the detailed solution method is omitted from this study.

TABLE I
PARAMETER SETUP

Description	Value
Frame per second	f = 10 fps
Weight	ω = 0.5
Required accuracy	Γ = 0.86 or 0.825
Channel bandwidth	W = 1 MHz
Noise power spectral density	N_0 = –165 dBm/Hz
Transmit power at BS	P_b = 43 dBm
Maximum transmit power at device	P_{max} = 23 dBm
Receive power at device	P_r = 0.01 W
Constant energy at transceiver	$\mathcal{E}_c$ = 0.5 J
Processing power at device	P_p = 7 W
Inference time at device	T_d^{inf} = 100 ms
Inference time at server	T_s^{inf} = 50 ms
Size of result message	S_r = 64 bytes
Size of image transmitted	S_t = 32∼1024 KB (default = 256)
Channel gain	g = –110∼–50 dB (default = –100)

IV. RESULTS AND DISCUSSION

For performance evaluation, we utilized the parameters detailed in Table I. We employed object detection models based on YOLOv8, a prominent deep learning model known for real-time object detection [13]. Among the five available sizes of YOLOv8 NN structures, we chose the smallest nano model for the edge device and the largest xlarge model for the edge server, considering their respective capabilities [14]. Our experiment focused on the detection task for fire and smoke as a representative surveillance service. To achieve this, we trained the YOLOv8 nano and xlarge models using a fire-smoke dataset provided in [15]. For the evaluations, we used 500 fire and 500 smoke images not included in the training set as positive samples, and 250 images each of the sun, car lights, clouds, and smog, which are easily confused with fire or smoke, as negative samples [16]. We assumed that the device's camera captures 10 frames per second, which are compressed into jpg format with a resolution of 640×480 pixels for inference and transmission [17]. This compression results in an average image size of 256 kilobytes, which we set as the default value. In contrast, the size of a result message was set to 64 bytes, the minimum frame length for Ethernet [18].

Fig. 3 compares the considered inference methods in terms of accuracy, cost, delay, and energy consumption, according to the presence or absence of resource allocation (RA) and changes in required accuracy (Γ). Here, two required accuracies are considered: $\Gamma = 0.86$, which is the same as the accuracy of the server-only inference, and $\Gamma = 0.825$, which corresponds to the average accuracy of both the server-only and device-only inferences. As shown, the device-only inference has the lowest accuracy but the smallest cost (i.e., the shortest delay and the least energy consumption), while the server-only inference has the highest accuracy but the greatest cost. On the other hand, the proposed cooperative inference achieves the same accuracy as the server-only inference (i.e., $\Gamma = 0.86$) but demonstrates a significantly lower cost. Furthermore, for each method, applying RA reduces the cost by approximately 50%

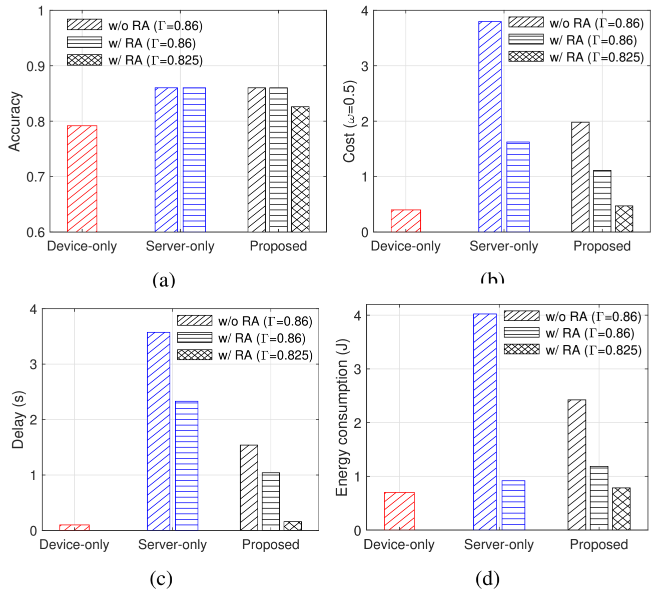

Fig. 3. Comparison of inference methods: (a) accuracy, (b) cost, (c) delay, and (d) energy consumption, according to RA and Γ.

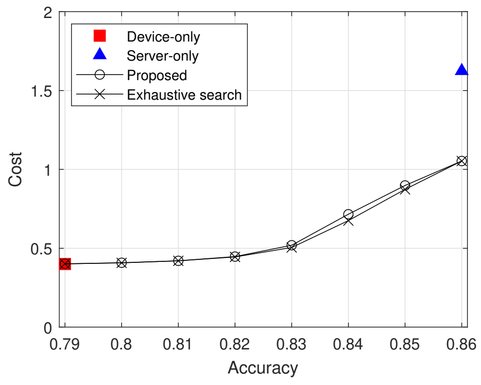

Fig. 4. Cost vs. required accuracy (Γ).

compared with not using RA. This verifies that optimizing not only confidence thresholds but also radio resources in a MEC environment significantly contributes to performance improvement. Notably, when the required accuracy is slightly lowered to 0.825, the cost significantly decreases. This is because the amount of offloading to the server considerably decreases as Γ decreases. Therefore, the proposed cooperative inference demonstrates a balanced performance between the device-only and server-only inferences, achieving higher accuracy than the device-only inference and lower cost than the server-only inference. The extent of this gain varies according to the accuracy requirement Γ.

Fig. 4 shows the cost versus the required accuracy (Γ). As Γ increases, the cost of the proposed inference initially remains low, comparable to the level of the device-only inference, but eventually increases once Γ exceeds a certain value. This implies that a significant amount of additional inference at the server is necessary to attain a certain high level of Γ. Nevertheless, the cost of the proposed inference is reduced to 65% of that of the server-only inference while achieving

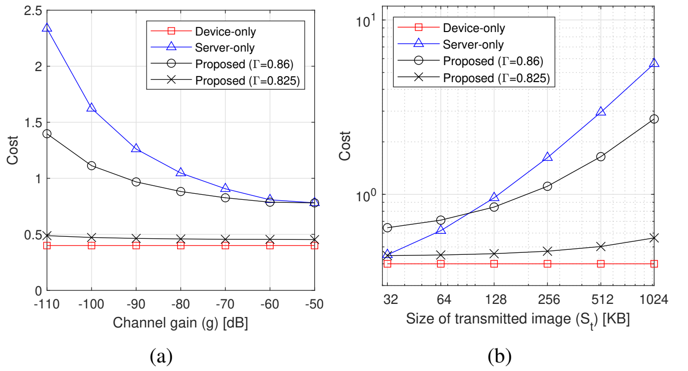

Fig. 5. Cost vs. (a) channel gain (g) and (b) size of transmitted image (S_t).

the same accuracy of $\Gamma = 0.86$. Furthermore, it is observed that the performance of the proposed method closely matches the performance found by the exhaustive search algorithm, indicating that the proposed optimization algorithm effectively determines parameters close to the optimal.

Fig. 5(a) shows the cost versus the channel gain (g). The device-only inference has the lowest constant cost because it is not affected by the channel, but the other methods experience a significant decrease in cost as g increases. In particular, the proposed inference exhibits lower cost than the server-only inference across most sections. Specifically, when $\Gamma = 0.825$, the proposed inference shows significantly lower cost at the expense of accuracy compared with when $\Gamma = 0.86$. However, for $\Gamma = 0.86$, its cost intersects with that of the server-only inference beyond $g = -50$ dB. This indicates that when g is extremely high (i.e., good channel quality), the server-only inference may offer a lower cost than the proposed inference. This is because the transmission and reception times (T^{tx} and T^{rx}) become negligible, and the device's inference time (T_d^{in}) is not incurred in the server-only method.

Fig. 5(b) illustrates the cost versus the size of the transmitted image (S_t). The device-only inference maintains the lowest constant cost, but the costs for the other methods significantly increase as S_t grows due to the increased overhead for transmitting image data. Nevertheless, the proposed inference generally shows reduced costs compared with the server-only inference in most sections. Notably, when the required accuracy $\Gamma = 0.825$, the proposed inference achieves a very low cost, albeit with some sacrifice in accuracy. However, at $\Gamma = 0.86$, the cost of the proposed inference crosses with that of the server-only inference around $S_t = 80$ KB. This implies that when S_t is very small (e.g., due to high data compression), the server-only inference might be more cost-effective than the proposed inference, as the transmission time T^{tx} required for offloading to the server becomes minor.

V. Conclusion

In this study, we proposed a cooperative inference method and analyzed its performance in terms of optimizing confidence thresholds, transmit power, and duty cycle to minimize delay and energy consumption while maintaining accuracy in MEC systems. The results demonstrated a trade-off between accuracy and energy-delay costs, with joint optimization of thresholds and radio resources significantly reducing both delay and energy consumption. The proposed method struck a balance between device-only and server-only inference, achieving higher accuracy than device-only and lower costs than server-only inference. We expect that this approach and optimization framework will be valuable in future MEC networks.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1A2C1011901).

References

[1] Y. Shi, K. Yang, T. Jiang, J. Zhang, and K. B. Letaief, "Communication-efficient edge AI: Algorithms and systems," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2167–2191, 2020.

[2] P. Joshi, M. Hasanuzzaman, C. Thapa, H. Afli, and T. Scully, "Enabling all in-edge deep learning: A literature review," *IEEE Access*, 2023.

[3] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[4] W.-Q. Ren, Y.-B. Qu, C. Dong, Y.-Q. Jing, H. Sun, Q.-H. Wu, and S. Guo, "A survey on collaborative DNN inference for edge intelligence," *Machine Intelligence Research*, vol. 20, no. 3, pp. 370–395, 2023.

[5] J. Shao and J. Zhang, "Communication-computation trade-off in resource-constrained edge inference," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 20–26, 2020.

[6] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 595–608, 2021.

[7] P. M. Grulich and F. Nawab, "Collaborative edge and cloud neural networks for real-time video processing," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 2046–2049, 2018.

[8] Y. Zhao, Q. Chen, W. Cao, W. Jiang, and G. Gui, "Deep learning based couple-like cooperative computing method for IoT-based intelligent surveillance systems," in *2019 IEEE 30th Annual International Symposium on PIMRC*. IEEE, 2019, pp. 1–4.

[9] Y. Zhao, Y. Yin, and G. Gui, "Lightweight deep learning based intelligent edge surveillance techniques," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1146–1154, 2020.

[10] A. A. Ahmed and M. Echi, "Hawk-Eye: An AI-powered threat detector for intelligent surveillance cameras," *IEEE Access*, vol. 9, pp. 63 283–63 293, 2021.

[11] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin *et al.*, "MEC in 5G networks," *ETSI white paper*, vol. 28, no. 2018, pp. 1–28, 2018.

[12] G. Vermeeren, L. Verloock, S. Aerts, L. Martens, and W. Joseph, "In situ assessment of uplink duty cycles for 4G and 5G wireless communications," *Sensors*, vol. 24, no. 10, p. 3012, 2024.

[13] F. M. Talaat and H. ZainEldin, "An improved fire detection approach based on YOLO-v8 for smart cities," *Neural Computing and Applications*, vol. 35, no. 28, pp. 20 939–20 954, 2023.

[14] A. Chaurasia and G. Jocher, "Ultralytics YOLOv8 documentation," https://docs.ultralytics.com/, 2023, accessed: 2024-02-02.

[15] Shahriar, "Fire smoke dataset," Sep 2022. [Online]. Available: https://universe.roboflow.com/shahriar-yytxo/fire-smoke-5wzrh

[16] "Overview of crawling and indexing topics," Mar 2024. [Online]. Available: https://developers.google.com/search/docs/crawling-indexing

[17] A. Rego, A. Canovas, J. M. Jiménez, and J. Lloret, "An intelligent system for video surveillance in IoT environments," *IEEE Access*, vol. 6, pp. 31 580–31 598, 2018.

[18] H.-H. Choi, K. Lee, and K.-H. Lee, "Optimizing confidence thresholds for cooperative inference in edge-AI surveillance systems: Avoiding the fate of 'the boy who cried wolf'," in *IEEE 22th Consumer Communications Networking Conference (CCNC)*, 2025, pp. 1–6.