Communication-Oriented Model Fine-Tuning for Packet-Loss Resilient Distributed Inference Under Highly Lossy IoT Networks

AUTHOR(S):

Itahara, Sohei; Nishio, Takayuki; Koda, Yusuke; Yamamoto, Koji

CITATION:

Itahara, Sohei ...[et al]. Communication-Oriented Model Fine-Tuning for Packet-Loss Resilient Distributed Inference Under Highly Lossy IoT Networks. IEEE Access 2022, 10: 14969-14979

ISSUE DATE:

2022

URL:

http://hdl.handle.net/2433/277852

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 21, 2021, accepted January 30, 2022, date of publication February 7, 2022, date of current version February 10, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3149336

Communication-Oriented Model Fine-Tuning for Packet-Loss Resilient Distributed Inference Under Highly Lossy IoT Networks

SOHEI ITAHARA[1], (Graduate Student Member, IEEE), TAKAYUKI NISHIO[2], (Senior Member, IEEE), YUSUKE KODA[3], (Member, IEEE), AND KOJI YAMAMOTO[1], (Senior Member, IEEE)

[1]Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan
[2]School of Engineering, Tokyo Institute of Technology, Meguro-ku, Tokyo 152-8550, Japan
[3]Centre of Wireless Communications, University of Oulu, 90014 Oulu, Finland

Corresponding author: Takayuki Nishio (nishio@ict.e.titech.ac.jp)

This work was supported in part by JST PRESTO under Grant JPMJPR2035.

ABSTRACT The distributed inference (DI) framework has gained traction as a technique for real-time applications empowered by cutting-edge deep machine learning (ML) on resource-constrained Internet of things (IoT) devices. In DI, computational tasks are offloaded from the IoT device to the edge server via lossy IoT networks. However, generally, there is a communication system-level trade-off between communication latency and reliability; thus, to provide accurate DI results, a reliable and high-latency communication system is required to be adapted, which results in non-negligible end-to-end latency of the DI. This motivated us to improve the trade-off between the communication latency and accuracy by efforts on ML techniques. Specifically, we have proposed a communication-oriented model tuning (COMtune), which aims to achieve highly accurate DI with low-latency but unreliable communication links. In COMtune, the key idea is to fine-tune the ML model by emulating the effect of unreliable communication links through the application of the dropout technique. This enables the DI system to obtain robustness against unreliable communication links. Our ML experiments revealed that COMtune enables accurate predictions with low latency and under lossy networks.

INDEX TERMS Distributed inference, communication-efficiency, machine learning, packet loss tolerant, delay-aware system.

I. INTRODUCTION

The Internet of things (IoT) is employed to enable multiple novel applications by combining the physical sensing of IoT devices with deep learning-based data analysis. Although deep learning technology is developing rapidly, it satisfying privacy and latency demands of the applications on resource-constrained IoT systems continue to pose a challenge. For example, factory automation and smart grids require latency of less than 10 ms and 20 ms, respectively [1]. In contrast, in smart home applications, IoT sensors such as visual and audio sensors obtain privacy-sensitive data that should not be exposed [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Moinul Hossain.

Distributed inference (DI) frameworks have been researched to address the privacy and latency challenges of deep learning deployment on IoT systems. In the DI framework for deep neural networks (DNNs) [3], [4], computationally expensive tasks are offloaded from the IoT devices to the locally located edge servers to reduce computation latency and the risk of data leakage, as compared with cloud computing. In the DI framework, the IoT devices and the edge server collaboratively process the portion of DNN, otherwise known as sub-DNN, by exchanging messages (e.g., outputs of sub-DNN) via IoT networks. Details of the DI are explained as follows: A well-trained DNN is divided into sub-DNNs through layers. The IoT device stores the input-side sub-DNN, while the edge server stores the output-side sub-DNN. The device obtains the output of the sub-DNN (i.e., the activations of the original DNN) from the raw input. Next,

the activation is transmitted to the edge server, and the server generates an inference result from the activations using its sub-DNN.

Although the DI reduces the computation latency and preserves the data privacy, the problem of the communication latency of the DI is posed [5]–[7]. This is because the communication payload size of the DI is typically larger than that of local computing and cloud computing. Moreover, the bandwidth of the IoT network is generally narrow; thus the communication latency is non-negligible in DI on IoT systems.

To achieve low communication latency, there are two general solutions: 1) adapting low-latency, which is a generally unreliable communication protocol, such as user datagram protocol (UDP) and higher physical transmission rate, and 2) lossy compression of the transmitted message. To realize ultra-low-latency DI (e.g., lower than 10 ms), it is necessary to simultaneously adopt both solutions. However, there is a trade-off in the solutions between the communication latency and prediction accuracy of the DI. The transport layer, for example, in the narrow-band and lossy IoT networks, the UDP transmission causes non-negligible packet losses, which degrades the inference accuracy by causing defects in the exchange of sub-DNN output between devices and edge servers. In contrast, reliable communication protocol (i.e., transport control protocol (TCP) transmission) causes non-negligible communication latency due to the retransmissions of dropped packets. Moreover, the lossy compression reduces the redundancy of the message, which increases the negative effect of packet loss (i.e., degrades the accuracy) on the DI.

This motivated us to improve the trade-off between communication latency and prediction accuracy by efforts on ML techniques. This study aims to design a DI method that achieves high accuracy using unreliable communication protocol on lossy IoT networks, where a considerable percentage of the transmitted packet is dropped. To this end, we have proposed communication-oriented model tuning (COMtune) to achieve robustness against the packet loss due to the non-retransmission policy of the unreliable communication protocol. Using COMtune, even when a part of the message exchanged between the nodes is dropped by the packet loss, one can obtain accurate inference results using the successfully received message.

To achieve such robustness against the packet loss, our key idea is to train the DNN through emulation of the effect of drops in the IoT network using the dropout technique [8], which randomly drops the activation in DNN. Through the training, the DNN would be able to provide accurate predictions using the dropped information. Moreover, the dropout technique [8] is well known as a regularization method; thus, the DNN receives the benefits of the regularization effect, and simultaneously achieves robustness against packet loss. Furthermore, to achieve even lower communication latency, COMtune employs lossy compression methods, which reduce the payload size of the message. We should note that the lossy compression reduces the redundancy of the message, results in the degradation of the robustness to the packet loss; thus, the COMtune, which improves the robustness against the packet loss, has further significant role in achieving high accuracy when the compression is applied. The performance evaluation using the image classification task CIFAR-10 demonstrated that the COMtune achieved higher accuracy than existing methods under lossy communication links, even while employing lossy compression methods.

The contributions of this study are summarized as follows:

- We have proposed COMtune to improve the trade-off in the DI framework, on the unreliable communication link between communication-latency and accuracy, using strong message compression. The message compression and robustness to the unreliable communication link are highly dependent on each other; the message compression can reduce the redundancy of the message, which further degrades the system robustness to the unreliable communication link. To the best of our knowledge, existing research has only addressed, either the message compression, or the robustness to the unreliable communication link.
- To improve the trade-off, COMtune tunes the DNN model by emulating the effects of the unreliable communication links using the dropout technique. The performance evaluation using CIFAR-10 demonstrated that the COMtune achieved higher accuracy than existing methods, under unreliable communication links even while employing lossy compression methods.

This study is an expanded version of [9] and evaluates the performance of COMtune when message compression is applied, and reveals that the COMtune is more efficient when the message compression is combined.

Correspondingly, however, independent of this work, a similar concept to improve the trade-off between unreliable communication and prediction accuracy through training of DNN by emulating the effect of unreliable communication has been presented in [10]. Meanwhile, there are two primary differences between [10] and our research, that is the communication link assumption and model training scheme. This study focuses on the end-to-end communication link and assumes packet loss, while [10] focuses on one-hop wireless links and assumes bit-error. Thus, the proposed COMtune can be applied to any network that experiences packet loss due to queue or buffer overflow, as well as bit errors. Second, our model training procedure is comparatively simpler to implement and more efficient in terms of accuracy. This is because [10] uses custom non-differentiable functions in DNN to emulate the effect of the unreliable communication. This procedure increases the implementation cost and decreases model training efficiency. In contrast, our training methods only utilize a dropout layer for the emulation; thus, the proposed method is easier to implement and can accommodate the link emulation layer in the back-propagation

TABLE 1. Distributed inference frameworks toward low communication latency.

Name	Analog or digital	Communication reliability	Communication link Model	Approach
[10]–[12]	Analog	-	-	-
[5], [13]–[20]	Digital	Reliable	-	-
[10]	Digital	Unreliable	one hop wireless	ML training
[21]–[23]	Digital	Unreliable	End-to-end	Tensor completion
Proposed method	Digital	Unreliable	End-to-end	ML training

process, which enables the model to benefit from the regularization effect caused by the model training using the dropout.

II. RELATED WORKS

This section summarizes the existing research that addresses the problems of communication overhead in DI. The summary of the related works is given in Table 1. First, without specifying the DI framework, a vast majority of research [24], [25] has been addressed to improve the trade-off between the latency and reliability of the communication systems. The proposed COMtune is orthogonal to these researches and improves the trade-off beyond the limits of those improved by the efforts on the communication system [24], [25].

In DI frameworks, the inference task generated in the IoT device is offloaded to the other nodes by sharing the raw inputs, or the results of the local computation, which are referred to as vertical and horizontal DIs, respectively. Unlike the research on vertical DI [26], [27], we have focused on the horizontal DI, because the sharing of the raw input in the vertical DI includes a critical privacy risk. In the horizontal DI literature, some works have addressed the achievement of low communication latency by optimizing the division point [5], [14], [15], leveraging multiple sink nodes [13], pruning the DNN model [14], quantizing the message [15]–[17], dimensional reduction of the message [18]–[20], and combining multiple inference tasks into a single one [16]. However, these works assumed a reliable communication link and aimed to reduce the communication payload size. The problem of the trade-off between reliability and latency has not been addressed by these works; thus, they are orthogonal to this research.

Another direction is to adapt an analog communication system [10]–[12]. [10], [11] used analog communication to reduce the cost of channel encoding in digital communication, where multiple nodes transmit signals in the same time slot by leveraging superimposition [12]. However, this study focuses on digital communication, which is more widely used than analog communication.

The impact of the unreliable communication link on DNN inference was evaluated in [6]. This work demonstrated the feasibility and effectiveness of employing unreliable but low-latency communication protocols for AI-empowered time-critical applications. [6] transmits the raw input from the device to the server, which includes the critical privacy risks. In the DI literature, to achieve robustness against the unreliable communication link, certain researches addressed estimation of the clean transmitted message from the received message, which is corrupted by the unreliable communication link, by joint source-channel coding [28], linear tensor completion [21], low-rank tensor completion [22], and image inpainting based completion [23]. Orthogonal to these works, which estimate the clean message from the corrupted message, we aimed to achieve a split model that achieves highly accurate predictions from the corrupted message, and proposed a joint model training method.

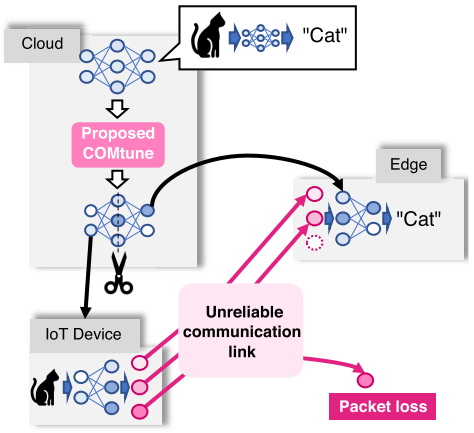


FIGURE 1. Over view of distributed inference with the proposed communication-oriented model tuning. The red arrows indicate the upload of the message from the device to the edge server via the unreliable communication link, in which the message is corrupted. The edge server obtains the prediction results using the corrupted message.

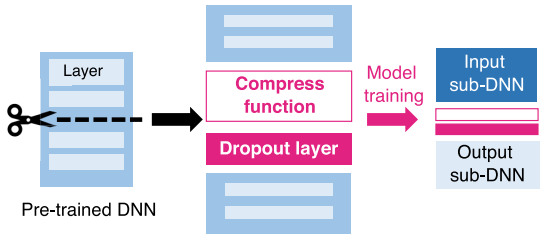
III. PROPOSED METHOD: COMMUNICATION-ORIENTED MODEL FINE-TUNING

A. SYSTEM MODEL

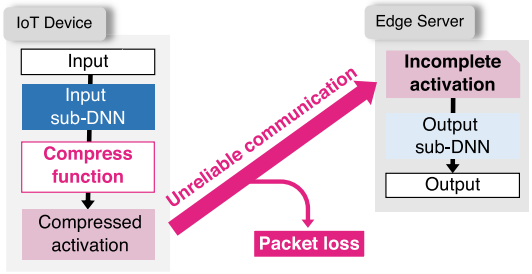
We assume an application scenario of automated surveillance in public places, roadsides, or factories, where IoT devices and edge servers cooperatively predict accidents, such as collisions, to avoid their occurrence. IoT devices equipped with cameras, monitor the target area and send information to the edge server. Based on the information, the edge server detects objects and their movement and further predicts the probability of the accident. In the application scenario, latency is a critical issue because the edge server is required to complete the prediction before the incident occurs.

Fig. 1 shows the system model consisting of a cloud server, an edge server, and an IoT device.[1] The edge server and

[1]We assume that it is predetermined which server the IoT device will communicate with, and a server selection problem is out of scope.



(a) Proposed: communication-oriented model tuning



(b) Subsequently distributted inference

FIGURE 2. Detailed procedure of the proposed communication-oriented model tuning. Red arrows indicate the upload of the activation from the device to the edge server via the unreliable link, which does not retransmit dropped packets. The edge server obtains prediction results using only the successfully transmitted activations.

IoT device are connected via unreliable communication links such as lossy and narrow-band IoT networks. This communication link is abstracted as an end-to-end communication link between the server and the device, which is detailed in the next section. The cloud server obtains a pre-trained DNN model from public repositories that are suitable for the inference tasks generated on the IoT device, for example, VGG [29] for image recognition tasks, YOLO [30] for object detection tasks, and BERT [31] for neural language tasks. As shown in Fig. 2 (a), the pre-trained model is fine-tuned by the proposed COMtune method to provide accurate inference while conducting DI via the unreliable protocol under lossy and narrow-band networks with ultra-low latency. The detailed COMtune procedure has been explained in Section III-C. Following the fine-turning, the DNN is divided at a division layer into two portions and the portions (sub-DNNs) are distributed to the IoT device and the edge server.

As shown in Fig. 2 (b), when an inference task is generated in the IoT device, the device and the server collaboratively solve the inference task using the distributed sub-DNNs, as follows: The IoT device generates activation by feeding the input sample to the input sub-DNN, compresses the activation, and sends the compressed activation to the edge server via the unreliable communication link. Note that computational delays to process the sub-DNNs are not considered in this study since we focus on communication latency. In the unreliable communication link, a non-negligible amount of packets are dropped; however, the dropped packets are not retransmitted. The edge server obtains the prediction results through the output sub-DNN by inputting the successfully received activation from the IoT device. The detailed DI procedure has been explained in Section III-D.

B. UNRELIABLE COMMUNICATION LINK ASSUMPTION

We assumed that the transmitted messages are probabilistically dropped owing to the non-retransmission policy of the unreliable communication protocol, where one does not retransmit the packets even when the packets are dropped. More formally, considering that the device sends a vector $\boldsymbol{x}$ via the communication link with a packet loss rate p, the edge server successfully receives a vector $f^{\mathrm{c}}(\boldsymbol{x} \mid p)$ denoted as follows:

$$f^{\mathrm{c}}(\boldsymbol{x} \mid p) = \boldsymbol{x} \odot \boldsymbol{m}(p), \tag{1}$$

where operator $\odot$ indicates the element-wise product and $\boldsymbol{m}(p)$ is a binary vector following the Bernoulli distribution with an expected value of $1 - p$.

In a real-world communication system, the vector of the activation $\boldsymbol{x}$ is divided into multiple packets and transmitted. Therefore, when a packet is dropped, the consecutive elements of $\boldsymbol{x}$ are lost. To avoid the burst loss, the device shuffles the vector elements and stores them in packets. The edge server constructs vector $\boldsymbol{x}$ from the successfully received packets, which results in (1). That is, the device permutes the elements randomly and stores them in packets. A packet $\boldsymbol{p}_i$ is represented as follows:

$$\boldsymbol{p}_i := \{x_{k_j} \mid i \leq j < i + s\}, \tag{2}$$

where k_j and s are the permuted identification of the element and the number of elements stored in a packet, respectively. The edge server reconstructs the vector of activations from a subset of transmitted packets $\boldsymbol{P}^{\mathrm{r}}$, where

$$\boldsymbol{P}^{\mathrm{r}} = \{\boldsymbol{p}_i \mid \boldsymbol{p}_i \text{ is received successfully}\}. \tag{3}$$

Thus, the reconstructed vector is expressed as $\boldsymbol{x} \odot \boldsymbol{m}(p)$.

Assuming the aforementioned communications model, the number of the received packets and the latency are denoted as follows. In the unreliable communication link, if n^{t} packets are transmitted using the communication link with a packet loss rate of p, the probability mass function (PMF) of the number of received packets n^{r} is expressed as follows:

$$\mathrm{PMF}(n^{\mathrm{r}}) = \begin{cases} \binom{n^{\mathrm{t}}}{n^{\mathrm{r}}} p^{n^{\mathrm{t}}-n^{\mathrm{r}}}(1-p)^{n^{\mathrm{r}}}, & \text{if } 0 \leq n^{\mathrm{r}} \leq n^{\mathrm{t}}; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The expected number of received packets is denoted by $(1-p)n^{\mathrm{t}}$. Assuming throughput b and packet size l, the latency is calculated as $n^{\mathrm{t}}l/b$. In contrast, all the transmitted packets are received when using a reliable communication link; thus, $n^{\mathrm{r}} = n^{\mathrm{t}}$. The PMF of latency is

$$\mathrm{PMF}(\tau) = \begin{cases} \binom{\lceil \tau/T \rceil - 1}{n^{\mathrm{t}}-1} p^{\lceil \tau/T \rceil - n^{\mathrm{t}}}(1-p)^{n^{\mathrm{t}}}, & \text{if } \lceil \tau/T \rceil \geq n^{\mathrm{t}}; \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

C. DETAILS OF COMMUNICATION-ORIENTED MODEL TUNING

To achieve high accuracy DI prediction under an unreliable communication link in highly lossy networks with activation compression, the pre-trained DNN is trained through emulation of the effect of packet loss and lossy activation compression. The overview of the COMtune is depicted in Fig. 2 (a). To emulate the effect of packet loss, we determined that the behavior of the dropout layer is similar to the effect of the packet loss in the unreliable communication link, as defined in (1), and used the dropout layer to emulate the effect of the packet loss. Further, the dropout layer and the activation compression function are inserted into the division layer of the pre-trained model. Subsequently, the model with the dropout layer and the activation compression is trained.

First, the cloud server obtains a pre-trained DNN model from the public repository, which is denoted by $f^{\mathrm{pre}}(\cdot \mid \boldsymbol{w}^{\mathrm{pre}})$, where $\boldsymbol{w}^{\mathrm{pre}}$ are the parameters. The pre-trained DNN is divided into input-sub DNN $f^{\mathrm{in}}(\cdot \mid \boldsymbol{w}^{\mathrm{in}})$ and output-sub DNN $f^{\mathrm{out}}(\cdot \mid \boldsymbol{w}^{\mathrm{out}})$ as

$$f^{\mathrm{pre}}(\cdot \mid \boldsymbol{w}^{\mathrm{pre}}) = f^{\mathrm{out}}(\cdot \mid \boldsymbol{w}^{\mathrm{out}}) \circ f^{\mathrm{in}}(\cdot \mid \boldsymbol{w}^{\mathrm{in}}), \tag{6}$$

where $f(\cdot) \circ g(\cdot)$ denotes the composite function of $f(\cdot)$ and $g(\cdot)$. We tuned the DNN $f^{\mathrm{trn}}(\cdot \mid \boldsymbol{w}^{\mathrm{trn}})$, which consists of two sub-DNNs, the dropout layer, and compression functions. The following section details the DNN $f^{\mathrm{trn}}(\cdot \mid \boldsymbol{w}^{\mathrm{trn}})$. Following the training, the input sub-DNN $f^{\mathrm{in}}(\cdot \mid \boldsymbol{w}^{\mathrm{in}})$ is sent to the IoT device, and an output sub-DNN $f^{\mathrm{out}}(\cdot \mid \boldsymbol{w}^{\mathrm{out}})$ is sent to the edge server, respectively.

The dropout was originally proposed as a regularization method in DNN literature, which enables training of the DNN for longer periods without overfitting and improves the test accuracy [8]. Thus, the dropout technique has been used in various DNN architectures and is available in multiple deep learning frameworks. In each training iteration using the dropout technique, the outputs of the hidden units are set to zero using a dropout layer with a dropout rate r. In addition to omitting the hidden unit outputs, the surviving (non omitted) hidden units are multiplied by $1/(1-r)$. Hence, the dropout behavior $f^{\mathrm{d}}(\cdot \mid r)$ is represented as follows:

$$\boldsymbol{x}_{i+1} = f^{\mathrm{d}}(\boldsymbol{y}_i \mid r) = \frac{1}{1-r}\boldsymbol{y}_i \odot \boldsymbol{m}(r), \tag{7}$$

where $\boldsymbol{y}_i$ is the hidden unit of the ith layer, and $\boldsymbol{x}_{i+1}$ is the input of the $i+1$th layer. Comparing equations (1) and (7), we determine that the dropout technique can emulate the drops of activation due to packet loss, in the model training. Therefore, the model trained using the dropout technique can provide accurate inferences even when the activations are dropped.

In addition to the unreliable communication link, the lossy compression reduces communication latency; however, it may degrade inference accuracy. To adapt the DNN model to the activation compression, COMtune fine-tunes the DNN model by inserting the compression function and dropout layer to the division layer. In this study, we used either of the two general lossy compression methods, quantization and dimensional reduction, which are detailed in Appendix A. Here, we have described COMtune with the general compression method. The compression and decompression function are denoted by $f^{\mathrm{cmp}}(\cdot)$ and $f^{\mathrm{dec}}(\cdot)$, respectively. Given the raw activation as $\boldsymbol{a}^{\mathrm{raw}}$, the compressed activation $\boldsymbol{a}^{\mathrm{cmp}}$ is denoted as $\boldsymbol{a}^{\mathrm{cmp}} := f^{\mathrm{cmp}}(\boldsymbol{a}^{\mathrm{raw}} \mid M)$, where, M is the data size of the compressed activation. From the compressed activation, the uncompressed activation is estimated by $\boldsymbol{a}^{\mathrm{dec}} = f^{\mathrm{dec}}(\boldsymbol{a}^{\mathrm{cmp}'})$. Therefore, using the above defined functions, the DNN $f^{\mathrm{trn}}(\cdot \mid \boldsymbol{w}^{\mathrm{trn}})$ that fine-tuned in the COMtune is denoted as follows:

$$f^{\mathrm{trn}}(\cdot \mid \boldsymbol{w}^{\mathrm{trn}}) = f^{\mathrm{out}}(\cdot \mid \boldsymbol{w}^{\mathrm{out}}) \circ f^{\mathrm{dec}}(\cdot) \circ f^{\mathrm{d}}(\cdot \mid r) \circ f^{\mathrm{cmp}}(\cdot \mid M) \circ f^{\mathrm{in}}(\boldsymbol{x} \mid \boldsymbol{w}^{\mathrm{in}}), \tag{8}$$

where r is a dropout rate.

We should further note that the dropout rate and message size used in the model training corresponds to the packet loss rate and the message size in the DI procedure. Thus, training using a larger dropout rate implies that the DNN is trained to adapt to a more lossy communication link, thus improving packet loss tolerance. Training with a smaller message size in the fine-tuning implies adapting to use a smaller message size in the DI, thus reducing communication latency. In contrast, as mentioned in [8], a larger training dropout rate degrades the achievable model performance (i.e., performance without any packet loss); similarly, a smaller message size degrades the achievable model performance, as well [32]. Therefore, the dropout rate and the message size are selected based on the packet loss rate of the communication link, desired communication latency, and model performance requirements.

Moreover, the dimensional reduction may strongly degrade the accuracy than quantization in highly unreliable communication link. This is because, in the dimensional reduction, this paper adopts principal component analysis (PCA) to compress the message; the message is represented by a linear combination of the small number of basis vectors, and the coefficients of basis vectors are transmitted as the compressed message, leading to a significant difference in the contribution of each element of the compressed message, which is detailed in Appendix. Thus, when the elements of the compressed message that correspond to important principal components (e.g., first principal components) are dropped, the accuracy is significantly degraded. On the other hand, in the quantization, an element of the compressed message corresponds to an element of an uncompressed message. Thus, the difference in the contribution between elements in the quantization is smaller than that in the dimensional reduction; this is a reason for the robustness of the quantization against packet loss, which will be validated in Section IV-D2.

D. DETAILS OF DISTRIBUTED INFERENCE

The DI is conducted when an inference task with input $\boldsymbol{x}$ is generated in the IoT device, which is depicted in Fig. 2 (b). First, the device generates and compresses the activation as

follows:

$$\boldsymbol{a} = f^{\mathrm{cmp}}(\cdot \mid M) \circ f^{\mathrm{in}}(\boldsymbol{x} \mid \boldsymbol{w}^{\mathrm{in}}). \tag{9}$$

Subsequently, the activation is transmitted by the communication link denoted in (1). Thus, the reconstructed vector is calculated as

$$\boldsymbol{a}' = f^{\mathrm{c}}(\boldsymbol{a} \mid p) = \boldsymbol{a} \odot \boldsymbol{m}(p). \tag{10}$$

To compensate the drops of the activation, the activation is multiplied by $1/1-p$. From the compensated activation, the uncompressed activation is estimated as

$$\boldsymbol{a}^{\mathrm{r}} = f^{\mathrm{dec}}\left(\frac{1}{1-p}\boldsymbol{a}'\right). \tag{11}$$

Subsequently, $\boldsymbol{a}^{\mathrm{r}}$ is fed to the output-sub DNN, and we obtain the prediction result. The prediction result $\boldsymbol{y}$ can be written as

$$\boldsymbol{y} = f^{\mathrm{trn}}(\cdot \mid \boldsymbol{w}^{\mathrm{trn}}) = f^{\mathrm{out}}(\cdot \mid \boldsymbol{w}^{\mathrm{out}}) \circ f^{\mathrm{dec}}(\cdot) \circ f^{\mathrm{c}}(\cdot \mid p) \circ f^{\mathrm{cmp}}(\cdot \mid M) \circ f^{\mathrm{in}}(\boldsymbol{x} \mid \boldsymbol{w}^{\mathrm{in}}). \tag{12}$$

If $f^{\mathrm{d}}(\cdot \mid r)$ in the model training is close to $f^{\mathrm{c}}(\cdot \mid p)$ in the DI, the model is expected to accurately predict from the corrupted activation. Comparing (7) and (1), when the parameter r is similar to the parameter p, $f^{\mathrm{d}}(\cdot \mid r)$ is similar to $f^{\mathrm{c}}(\cdot \mid p)$. Thus, when r is similar to p, the COMtune is expected to improve the prediction accuracy from the corrupted activation. Moreover, our evaluation revealed that even when the difference between r and p is large (e.g., $(r, p) = (0.5, 0.0)$), the COMtune achieved higher accuracy than the previous DI.

IV. EVALUATION

A. SETUP

We conducted a simulation evaluation in which an IoT device and edge server are connected by an abstracted communication link, and packets transmitted between them are randomly discarded with a certain probability. Since this study focuses on the trade-off between communication latency and accuracy, the simulation omitted the computation latency for processing the DNN model and the activation compression. The details of the simulation are as follows.

1) COMMUNICATION SETUP

An IoT device and an edge server were assumed to be connected via a lossy IoT network, which was abstracted as a communication link, in which packets were randomly dropped with the probability p. Hence, the elements of the activation vector transmitted by the IoT device were randomly dropped. To calculate the communication latency, the packet size and throughput of the communication link (including MAC and network layer overheads) were set to 100 bytes and 9.0 Mbit/s .[2] We considered two communication protocols; unreliable protocol (i.e., without retransmissions) and reliable protocol (i.e., with retransmissions).

[2]Note that this parameter is an example of the parameters defined in IEEE 802.11ah.

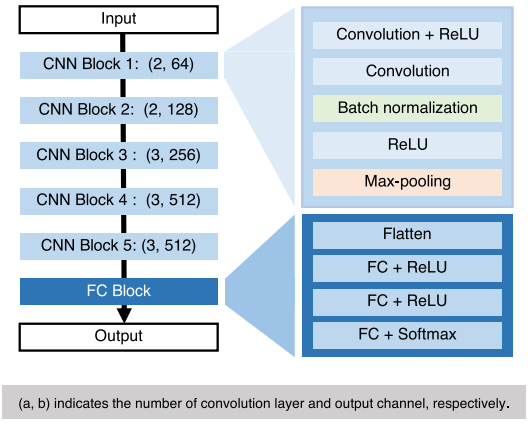


FIGURE 3. Architecture of DNN. Each convolutional neural network (CNN) block consists of two or three convolutional, batch normalization, and max-pooling layers. The number of convolutional layers *a* in each CNN block and the output channel *b* are denoted as (*a*, *b*) in each CNN block. The fully connected (FC) block consists of three FC layers.

2) DATASET AND MACHINE LEARNING MODELS

We used an image recognition dataset, CIFAR-10,[3] with 50,000 training and 10,000 testing images that represented 10 image classes, such as "dog" and "ship." The training dataset was used to fine-tune the pre-trained model in the COMtune. The test dataset was used to evaluate the inference performance of the DI phase.

The architecture of the DNN model used in the experiments is shown in Fig. 3. The model was designed with reference to VGG16 [29], which consists of five convolutional blocks and a FC block. Each convolutional block included two or three 3×3 convolutional layers activated by the rectified linear unit (ReLU), and the block was followed by a 2×2 max-pooling layer. The convolutional layers have the same number of output channels in each convolutional block. Additionally, one of the two convolutional layers is followed by the batch normalization layer. The FC block consists of three FC layers (256 and 128 units with ReLU activation and 10 other units activated by softmax).

3) MACHINE LEARNING TRAINING

The detailed ML training procedure is as follows: The training dataset is divided into updating and validation datasets in a ratio of 9:1. The DNN model is updated using only the updating dataset for multiple epochs. In each epoch, the model is evaluated using the validation dataset. The training is completed if 150 epochs are performed, or if the validation loss increased after 20 epochs consecutively, which indicates that the model is starting to overfit. The Adam optimizer, a training rate of 0.001, and a mini-batch size of 128 were selected as hyperparameters. Notably, this paper generates a pre-trained model by training a randomly initialed ML model using the aforementioned training procedure.

[3]https://www.cs.toronto.edu/ kriz/cifar.html

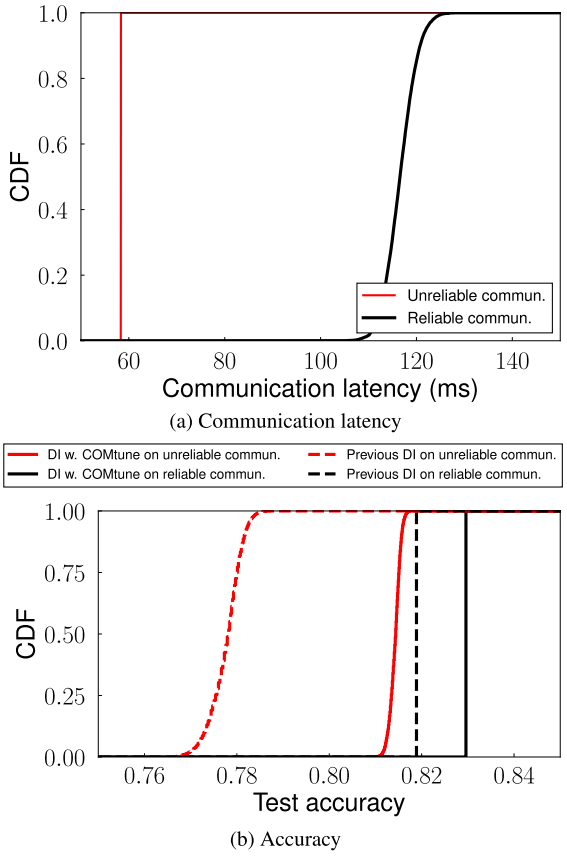


FIGURE 4. Cumulative distribution function of the accuracy and communication latency for the previous DI and proposed DI with COMtune on the reliable and unreliable communication links, respectively.

4) DISTRIBUTED INFERENCE

In this evaluation, major parts of the inference task were offloaded to the edge server because the computational capacity of the IoT device is generally much worse than the edge server. Specifically, the CNN was divided into CNN block 1, resulting in the inference tasks of CNN block 1 being conducted at the IoT device and that of the CNN blocks 2, 3, 4, and 5, and the FC block being conducted in the edge server. The dimensions of the activation of the CNN block 1 is 16,384, which is 65.5 kB in 32bit float point representation (e.g., the communication delay is 58.2 ms when no packet loss occurs). The packet loss in the communication link is emulated by the dropout, where the dropout rate is set to the packet loss rate, which ranges from 0 to 0.9. Additionally, we ran each method ten times from different random seeds and computed the average and standard deviation of the performance in ten trials.

B. CUMULATIVE DISTRIBUTION FUNCTION OF THE ACCURACY AND LATENCY

Fig. 4 (a) illustrates the cumulative distribution function (CDF) of the communication latency of the DI using reliable and unreliable protocols, respectively, where the activation compression is not applied. The CDF is obtained following the aforementioned discussion, with the parameters described in Section IV-A and the packet loss rate of 0.5. While using the unreliable protocol, 50% of a message is dropped. However, in case that a reliable protocol is used, the entire message is successfully received by retransmissions. As shown in Fig. 4 (a), due to the no-retransmission policy, the latency of the unreliable protocol is stable and lower than that of the reliable protocol. Moreover, the latency of the reliable protocol transmission is not stable.

Fig. 4 (b) shows the CDF of the accuracy of the proposed DI with COMtune, and previous DI using unreliable and reliable protocols, respectively. Regardless of the underlying communication system, the proposed DI with COMtune achieved higher accuracy than the previous DI. This is because of two reasons: regularization and robustness to the packet loss. In the case of the reliable protocol, the accuracy of the DI with COMtune and previous DI is stable because all the transmitted packets are successfully received because of the retransmissions. The DI with COMtune achieved 1% higher accuracy than the previous DI because of the regularization effect of the dropout technique used in the COMtune. In contrast, for the unreliable protocol, the accuracy of the DI with COMtune and previous DI is not stable due to the transmitted packets being dropped because of the non-retransmission policy of the unreliable protocol. In the unreliable protocol transmission, the DI with COMtune achieved 4% higher accuracy than the previous DI. Moreover, comparing the accuracy degradation from that on the reliable protocol to that of the unreliable protocol, the degradation of the DI with COMtune is smaller than that of the previous DI. Thus, we can conclude that the COMtune improved the trade-off between the prediction accuracy and the communication latency, due to the training involving emulation of corruptions of the message in the unreliable and low-latency communication system.

C. IMPACT OF DROPOUT RATE ON ROBUSTNESS AGAINST PACKET LOSS

Fig. 5 shows the test accuracy of the DI with COMtune, and previous DI as a function of the packet loss rate while using the unreliable communication link. In the case of DI with COMtune, Fig. 5 shows the result for each dropout rate (i.e., 0.2 and 0.5) used in the COMtune prior to the DI. For both of the dropout rates, the DI with COMtune achieved higher accuracy than the previous DI even when the packet loss rate was low because of the regularization effect of the dropout technique. Moreover, COMtune mitigates the accuracy degradation caused by the packet loss, especially when the packet loss rate is high. In particular, the accuracy of the previous DI was degraded by more than 10%, when more than 70% of the packets were dropped,[4] while that of DI with COMtune with the dropout rate of 0.5 exhibited only a 3.8% degradation in accuracy. Thus, we can conclude that

[4]In this evaluation, a packet loss rate of 70% corresponds to the successful reception of about 190 packets out of 640 transmitted packets.

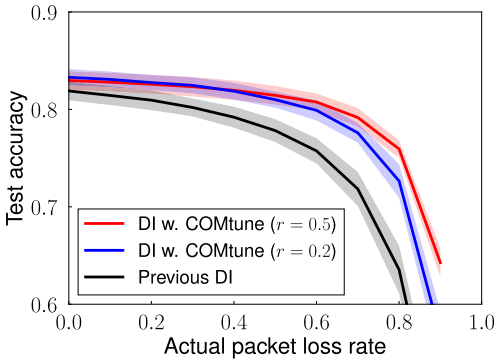


FIGURE 5. Test accuracy as a function of packet loss rate for each dropout rate r. The shaded regions denote the standard deviation of the performance among ten trials.

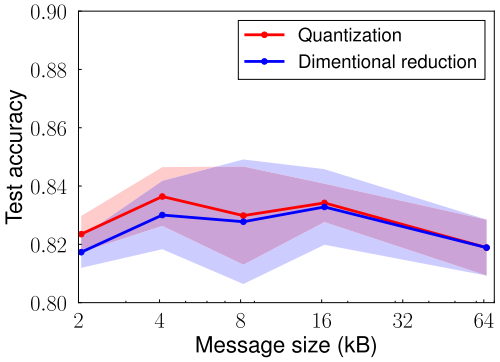


FIGURE 6. Effect of message compression on achievable accuracy. The message size without any compression is 64 kB.

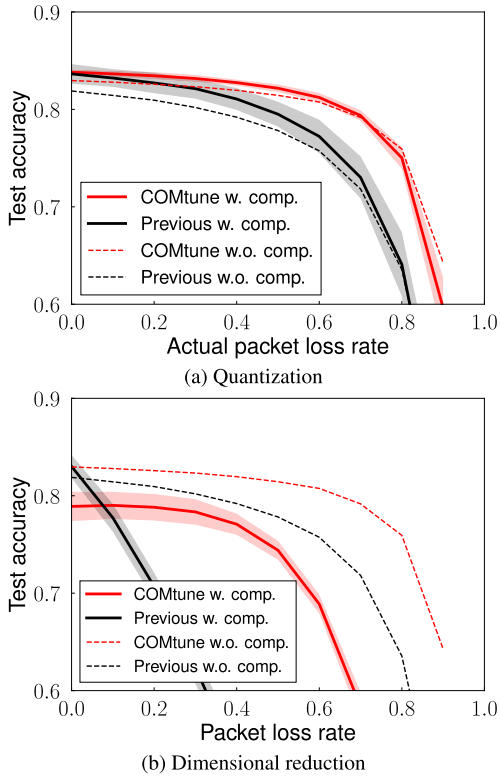


FIGURE 7. Test accuracy as a function of packet loss rate with or without message compression (message size is 4 kB and 64 kB, respectively). The black and red lines indicate the results obtained using the DNN tuned without any dropout layer and the COMtune with dropout rates of 0.5, respectively. The solid lines and shaded regions denote the average and standard deviation of the accuracy among ten trials with the message compression, respectively. The dots lines indicate the average accuracy without message compression.

COMtune improves the packet loss robustness, even when the dropout rate in COMtune differs from the packet loss rate.

Moreover, as the dropout rate increases, the accuracy degradation is better mitigated. Particularly, when the packet loss rate is 0.7, the model trained with a dropout rate of 0.5 and 0.2 demonstrated a 3.8% and 5.7% degradation in accuracy, respectively. This is because a larger dropout rate indicates emulation of the more lossy network in model training, which encourages the model to achieve high accuracy in the highly lossy network.

D. PERFORMANCE EVALUATION WITH COMtune USING ACTIVATION COMPRESSION

1) EFFECT OF ACTIVATION COMPRESSION ON ACHIEVABLE ACCURACY

Fig. 6 shows the test accuracy as a function of the message size without any packet loss, that is, that all transmitted packets were successfully received. The message is compressed by either quantization or dimensional reduction, which are both detailed in Appendix A. Even when the message is compressed, the accuracy is comparable to that when the message is not compressed (i.e., the 65.5 kB message), which is consistent with the existing works that have addressed DNN compression [32]. However, the following evaluation, as shown in Fig. 8, reveals that there is a trade-off between the message size and robustness to the unreliable communication link; when the message is highly compressed, the robustness is degraded.

2) EFFECT OF COMMUNICATION-ORIENTED MODEL TUNING ON ACCURACY WITH ACTIVATION COMPRESSION

Fig. 7 shows the test accuracy as a function of packet loss rate, with or without message compression (message size is 4 kB and 64 kB, respectively), using the unreliable protocol. Fig. 7 (a) and (b) show the results when the quantization and dimensional reduction are applied to compress the message, respectively. In Fig. 7 (a), when the compression is applied, the accuracy of DI with COMtune is higher than that of the previous DI regardless of the packet loss rate, which is consistent with Fig. 5. This demonstrated that COMtune improved the packet loss tolerance of the split model in case that the message is highly compressed, as well as the message is not compressed. In Fig. 7 (b), the DI with COMtune achieved higher accuracy than precious DI when the dropout rate and the packet loss rates are similar. In particular, DI with COMtune with the dropout rate of 0.5 does when the packet loss rate is larger than 0.1.

Comparing the accuracy with and without compression, when the dimensional reduction is applied, the accuracy with

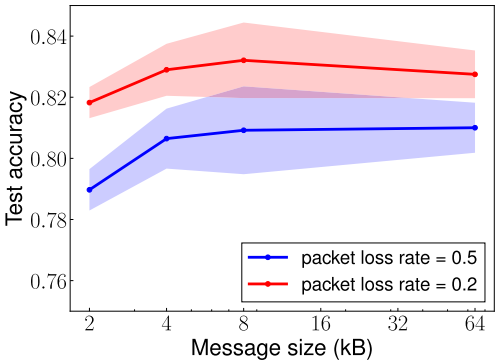


FIGURE 8. Effect of message size of DI with COMtune on robustness to the unreliable communication link. Quantization is applied as the message compression method, and DNNs are turned with dropout rates of 0.2.

compression is more degraded than without compression. For example, the accuracy degradation when the packet loss rate of 0.5 is 7.0% for DI with COMtune, and that is 34.6% for previous DI. On the other hand, when the quantization is applied, the accuracy with compression is comparable to that with compression. As discussed in Section III-C, this gap between the two message compression methods is explained in terms of the difference in the contribution of each element of the compressed message; the difference in dimensional reduction is significantly larger than the quantization. Thus, in the dimensional reduction, the accuracy is significantly degraded when the element of the compressed message has a high contribution. Therefore, we can conclude that quantization is a message compression method that achieves more robustness against the packet loss than the dimensional reduction.

Fig. 8 shows the test accuracy of DI using COMtune as a function of the message size for the packet loss rate of 0.2 and 0.5, respectively. In this evaluation, the quantization is applied as the message compression method. Regardless of the packet loss rate, the accuracy is degraded as the message size is reduced. Thus, we conclude that message compression degrades the robustness of the DI system to the unreliable communication link. This is because message compression reduces the redundancy of the message.

V. CONCLUSION

We have presented COMtune that aims to improve the prediction accuracy and communication latency by efforts on the application layer. Specifically, we aimed to achieve accuracy prediction under low-latency and unreliable communication link, such as UDP transmission. In COMtune, the key idea is to train the ML model by emulating the effect of the unreliable communication link, such that the model gains robustness to the unreliable communication system. Our experimental ML evaluation revealed that DI with COMtune obtains a more accurate prediction than previous DI on the highly unreliable communication link. Moreover, we revealed that the proposed COMtune is compatible with the general message compression methods. An interesting area for future work is an optimization framework that determines the parameters of the emulated communication systems to maximize the model accuracy in lossy wireless networks under the constraints of the total latency of communication and computation.

APPENDIX A
COMPRESSION METHODS

To reduce communication payload size of the message (i.e., activation of the input-sub DNN), we adapted general lossy compression methods that are quantization and dimensional reduction. Generally, the lossy compression method in DNN literature implies the compression of both, the activation and model parameters aiming to reduce the data size of the parameters and the computation cost of the inference. However, this study aims to reduce the data size of the activation, thus only the activation is compressed.

A. ACTIVATION QUANTIZATION

In the quantization, the elements of the activation are compressed from full-perception values (i.e., 32 bit float representation) to quantized values (i.e., n bit integer representation). The quantized activation is transmitted to the edge server as a message and dequantized to full-perception activation in the edge server, which is fed to output-sub network, in which the communication payload is reduced by $n/32$. Given the desired message size M and uncompressed message size M^{float}, which is the message size with 32bit float point, n is determined as $n = \lfloor 32M/M^{\text{float}} \rfloor$.

For more details of quantization, the elements are clipped into predefined ranges, and further represented by n bit integers. First, the full-perception activation elements are clipped into range from $s^{\min}$ to $s^{\max}$, where $s^{\min}$ and $s^{\max}$ are scale factors that indicate smallest and largest value represented by quantized value, respectively. The scale factors are determined for each activation element based on the range of the distribution of the element using the pre-trained dataset. Finally, the clipped value is quantized to n bit integer.

Hence, given ith element of full-perception activation as a_i^{float}, the corresponding clipped value a_i^{clip} is denoted as

$$a_i^{\text{clip}} = \max\left(\min\left(a_i^{\text{float}}, s_i^{\min}\right), s_i^{\max}\right). \tag{13}$$

Note that the scale factors are determined in the cloud server prior to the DI. The quantized activation a_i^{int} is represented as

$$a_i^{\text{int}} = \text{round}\left(\frac{2^n - 1}{s_i^{\max} - s_i^{\min}} a_i^{\text{float}}\right). \tag{14}$$

For shorthand notation, we denote a quantization function of a single element of the activation by $f^{\text{qut}}(a, s^{\min}, s^{\max})$, where $f^{\text{qut}}(a_i^{\text{float}}, s_i^{\min}, s_i^{\max}) = a_i^{\text{int}}$. From the quantized activation a_i^{int}, the unquantized activation is estimated as follows:

$$a_i^{\text{deq}} = \frac{s_i^{\max} - s_i^{\min}}{2^n - 1} a_i^{\text{int}}. \tag{15}$$

For shorthand notation, we denote a dequantization function of a single element of the activation by $f^{\text{deq}}(a, s^{\min}, s^{\max})$,

where $f^{\mathrm{deq}}(a_i^{\mathrm{int}}, s_i^{\mathrm{min}}, s_i^{\mathrm{max}}) = a_i^{\mathrm{deq}}$. Thus, given D dimensional vectors $\boldsymbol{s}^{\mathrm{min}}$, $\boldsymbol{s}^{\mathrm{max}}$, and $\boldsymbol{a}$ as the scale factors and uncompressed activation, the compression and decompression functions are denoted as

$$f^{\mathrm{cmp}}(\boldsymbol{a} \mid M) = \left\{ f^{\mathrm{qut}}(a_i, s_i^{\mathrm{min}}, s_i^{\mathrm{max}}) \mid 0 < i \leq D \right\}, \quad (16)$$

$$f^{\mathrm{dec}}(\boldsymbol{a}) = \left\{ f^{\mathrm{deq}}(a_i, s_i^{\mathrm{min}}, s_i^{\mathrm{max}}) \mid 0 < i \leq D \right\}. \quad (17)$$

B. DIMENSIONAL REDUCTION

In dimensional reduction, the activation is converted to a linear combination of basis vectors, where the number of the basis vectors is smaller than the dimensions of the activation. In the DI, the coefficients of basis vectors are transmitted rather than the elements of the activation, which reduces the communication payload size by D'/D, where the number of the basis vectors is D' and the dimensions of the activation is D. Thus, given the compressed message size M and uncompressed message size M', D' is determined as $D' = \lfloor MD/M' \rfloor$. The server estimates the uncompressed activation using the basis vector and the received coefficients. Formally, given a D dimensional vector $\boldsymbol{a}$ as the uncompressed activation and D' dimensional vector $\boldsymbol{a}'$ as a compressed activation, the compression and decompression functions are denoted as

$$f^{\mathrm{cmp}}(\boldsymbol{a} \mid M) = \boldsymbol{w}\boldsymbol{a}, \quad (18)$$

$$f^{\mathrm{dec}}(\boldsymbol{a}') = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{a}' + \boldsymbol{b}, \quad (19)$$

where $\boldsymbol{w}$ is a $D' \times D$ matrix and $\boldsymbol{b}$ indicates D dimensional bias vector, respectively.

To determine the parameters $\boldsymbol{w}$, PCA is used. In more detail, ith row of $\boldsymbol{w}$ is an eigenvector of the data covariance matrix $\boldsymbol{S}$ of the pre-trained dataset, which corresponds to ith largest eigenvalue. The data covariance $\boldsymbol{S}$ is denoted as

$$\boldsymbol{S} = \frac{1}{|\mathcal{A}|} \sum_{\boldsymbol{a} \in \mathcal{A}} (\boldsymbol{a} - \bar{\boldsymbol{a}})(\boldsymbol{a} - \bar{\boldsymbol{a}})^{\mathrm{T}}, \quad (20)$$

$$\bar{\boldsymbol{a}} := \frac{1}{|\mathcal{A}|} \sum_{\boldsymbol{a} \in \mathcal{A}} \boldsymbol{a}, \quad (21)$$

where

$$\mathcal{A} = \{ f^{\mathrm{in}}(\boldsymbol{x}_j \mid \boldsymbol{w}^{\mathrm{in}}) \mid \boldsymbol{x}_j \in \text{preobtained dataset} \}. \quad (22)$$

The bias vector $\boldsymbol{b}$ is denoted as

$$\boldsymbol{b} = \sum_{i=D'+1}^{D} (\bar{\boldsymbol{a}}^{\mathrm{T}}\boldsymbol{u}_i)\boldsymbol{u}_i, \quad (23)$$

where $\boldsymbol{u}_i$ is an eigenvector of $\boldsymbol{S}$, corresponding to ith largest eigenvalue.

REFERENCES

[1] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Müller, T. Elste, and M. Windisch, "Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 70–78, Feb. 2017.

[2] H. Lin and N. W. Bergmann, "IoT privacy and security challenges for smart home environments," *Information*, vol. 7, no. 3, pp. 1–15, Jul. 2016.

[3] N. D. Lane and P. Georgiev, "Can deep learning revolutionize mobile sensing," in *Proc. 16th Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2015, pp. 117–122.

[4] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud," in *Proc. ACM SIGKDD*, London, U.K., Jul. 2018, pp. 2407–2416.

[5] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 1, pp. 615–629, 2017.

[6] J. Liu and Q. Zhang, "To improve service reliability for AI-powered time-critical services using imperfect transmission in MEC: An experimental study," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9357–9371, Oct. 2020.

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

[8] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, *arXiv:1207.0580*.

[9] S. Itahara, T. Nishio, and K. Yamamoto, "Packet-loss-tolerant split inference for delay-sensitive deep learning in lossy wireless networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1–6.

[10] J. Shao and J. Zhang, "BottleNet++: An end-to-end approach for feature compression in device-edge co-inference systems," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.

[11] H. Xie and Z. Qin, "A lite distributed semantic communication system for Internet of Things," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 142–153, Jan. 2021.

[12] M. Krouka, A. Elgabli, C. ben Issaid, and M. Bennis, "Communication-efficient split learning based on analog communication and over the air aggregation," 2021, *arXiv:2106.00999*.

[13] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 328–339.

[14] W. Shi, Y. Hou, S. Zhou, Z. Niu, Y. Zhang, and L. Geng, "Improving device-edge cooperative inference of deep learning via 2-step pruning," in *Proc. IEEE INFOCOM Workshop*, Paris, France, Apr. 2019, pp. 1–6.

[15] H. Li, C. Hu, J. Jiang, Z. Wang, Y. Wen, and W. Zhu, "JALAD: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2018, pp. 671–678.

[16] Y. Matsubara and M. Levorato, "Neural compression and filtering for edge-assisted real-time object detection in challenged networks," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 2272–2279.

[17] X. Huang and S. Zhou, "Dynamic compression ratio selection for edge inference systems with hard deadlines," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8800–8810, Sep. 2020.

[18] Y. Koda, J. Park, M. Bennis, K. Yamamoto, T. Nishio, M. Morikura, and K. Nakashima, "Communication-efficient multimodal split learning for mmWave received power prediction," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1284–1288, Jun. 2020.

[19] M. Jankowski, D. Gunduz, and K. Mikolajczyk, "Joint device-edge inference over wireless links with pruning," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, May 2020, pp. 1–5.

[20] Y. Matsubara, D. Callegaro, S. Baidya, M. Levorato, and S. Singh, "Head network distillation: Splitting distilled deep neural networks for resource-constrained edge computing systems," *IEEE Access*, vol. 8, pp. 212177–212193, 2020.

[21] A. Dhondea, R. A. Cohen, and I. V. Bajic, "CALTEC: Content-adaptive linear tensor completion for collaborative intelligence," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 2179–2183.

[22] L. Bragilevsky and I. V. Bajic, "Tensor completion methods for collaborative intelligence," *IEEE Access*, vol. 8, pp. 41162–41174, 2020.

[23] I. V. Bajic, "Latent space inpainting for loss-resilient collaborative object detection," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.

[24] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3098–3130, 4th Quart., 2018.

[25] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 88–145, 1st Quart., 2019.

[26] J. Kim, Y. Park, G. Kim, and S. J. Hwang, "SplitNet: Learning to semantically split deep networks for parameter reduction and model parallelization," in *Proc. ICML*, vol. 70. Sydney, NSW, Australia, Aug. 2017, pp. 1866–1874.
[27] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "DeepThings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2348–2359, Nov. 2018.
[28] K. Choi, K. Tatwawadi, A. Grover, T. Weissman, and S. Ermon, "Neural joint source-channel coding," in *Proc. ICML*, vol. 97. Long Beach, CA, USA, Jun. 2019, pp. 1182–1192.
[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–14.
[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.
[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[32] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016, *arXiv:1602.02830*.

SOHEI ITAHARA (Graduate Student Member, IEEE) received the B.E. degree in electrical and electronic engineering from Kyoto University, in 2020, where he is currently pursuing the M.I. degree with the Graduate School of Informatics.

TAKAYUKI NISHIO (Senior Member, IEEE) received the B.E. degree in electrical and electronic engineering and the master's and Ph.D. degrees in informatics from Kyoto University, in 2010, 2012, and 2013, respectively. He had been an Assistant Professor with the Graduate School of Informatics, Kyoto University, from 2013 to 2020. From 2016 to 2017, he was a Visiting Researcher with the Wireless Information Network Laboratory (WINLAB), Rutgers University, USA. He has been an Associate Professor with the School of Engineering, Tokyo Institute of Technology, Japan, since 2020. His current research interests include machine learning-based network control, machine learning in wireless networks, and heterogeneous resource management.

YUSUKE KODA (Member, IEEE) received the B.E. degree in electrical and electronic engineering from Kyoto University, in 2016, and the M.E. and Ph.D. degrees in informatics with the Graduate School of Informatics, Kyoto University, in 2018 and 2021, respectively. In 2019, he visited the Centre for Wireless Communications, University of Oulu, Finland, to conduct collaborative research, where he is currently a Postdoctoral Researcher with the Centre for Wireless Communications. He received the VTS Japan Young Researcher's Encouragement Award, in 2017, and the TELECOM System Technology Award, in 2020. He was a recipient of the Nokia Foundation Centennial Scholarship, in 2019.

KOJI YAMAMOTO (Senior Member, IEEE) received the B.E. degree in electrical and electronic engineering and the master's and Ph.D. degrees in informatics from Kyoto University, in 2002, 2004, and 2005, respectively. From 2004 to 2005, he was a Research Fellow of the Japan Society for the Promotion of Science (JSPS). From 2008 to 2009, he was a Visiting Researcher at Wireless@KTH, Royal Institute of Technology (KTH), Sweden. Since 2005, he has been with the Graduate School of Informatics, Kyoto University, where he is currently an Associate Professor. His research interests include radio resource management, game theory, and machine learning. He is a Senior Member of the IEICE and a member of the Operations Research Society of Japan. He was a Tutorial Lecturer, in ICC 2019. He received the PIMRC 2004 Best Student Paper Award, in 2004, and the Ericsson Young Scientist Award, in 2006. He also received the Young Researcher's Award, the Paper Award, SUEMATSU-Yasuharu Award, Educational Service Award from the IEICE of Japan, in 2008, 2011, 2016, and 2020, respectively, and IEEE Kansai Section GOLD Award, in 2012. He is the Symposium Co-Chair of GLOBECOM 2021 and the Vice Co-Chair of IEEE ComSoc APB CCC. He serves as an Editor of IEEE Wireless Communications Letters, IEEE Open Journal of Vehicular Technology, and *Journal of Communications and Information Networks*.