Optimal Model Placement and Online Model Splitting for Device-Edge Co-Inference

Jia Yan, *Member, IEEE*, Suzhi Bi, *Senior Member, IEEE*, and Ying-Jun Angela Zhang, *Fellow, IEEE*

***Abstract*— Device-edge co-inference opens up new possibilities for resource-constrained wireless devices (WDs) to execute deep neural network (DNN)-based applications with heavy computation workloads. In particular, the WD executes the first few layers of the DNN and sends the intermediate features to the edge server that processes the remaining layers of the DNN. By adapting the model splitting decision, there exists a tradeoff between local computation cost and communication overhead. In practice, the DNN model is re-trained and updated periodically at the edge server. Once the DNN parameters are regenerated, part of the updated model must be placed at the WD to facilitate on-device inference. In this paper, we study the joint optimization of the model placement and online model splitting decisions to minimize the energy-and-time cost of device-edge co-inference in presence of wireless channel fading. The problem is challenging because the model placement and model splitting decisions are strongly coupled, while involving two different time scales. We first tackle online model splitting by formulating an optimal stopping problem, where the finite horizon of the problem is determined by the model placement decision. In addition to deriving the optimal model splitting rule based on backward induction, we further investigate a simple one-stage look-ahead rule, for which we are able to obtain analytical expressions of the model splitting decision. The analysis is useful for us to efficiently optimize the model placement decision in a larger time scale. In particular, we obtain a closed-form model placement solution for the fully-connected multilayer perceptron with equal neurons. Simulation results validate the superior performance of the joint optimal model placement and splitting with various DNN structures.**

***Index Terms*— Edge inference, deep neural network, model splitting, model placement, optimal stopping theory.**

Manuscript received 20 May 2021; revised 2 December 2021; accepted 28 March 2022. Date of publication 15 April 2022; date of current version 11 October 2022. This work was supported in part by the National Key Research and Development Program under Project 2019YFB1803305, in part by the General Research Fund from the Research Grants Council of Hong Kong under Project 14201920 and Project 14202421, in part by the National Natural Science Foundation of China under Project 61871271, in part by the Key Project of Department of Education of Guangdong Province under Grant 2020ZDZX3050, in part by the Shenzhen Science and Technology Program under Project JCYJ20210324093011030, and in part by the Open Research Project Program of the State Key Laboratory of Internet of Things for Smart City (University of Macau) under Grant SKL-IoTSC(UM)-2021-2023/ORPF/A03/2022. The associate editor coordinating the review of this article and approving it for publication was T. Q. S. Quek. *(Corresponding author: Suzhi Bi.)*

Jia Yan and Ying-Jun Angela Zhang are with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: yanj@umn.edu; yjzhang@ie.cuhk.edu.hk).

Suzhi Bi is with the College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: bsz@szu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2022.3165824.

Digital Object Identifier 10.1109/TWC.2022.3165824

I. Introduction

A. Motivation and Contributions

WITH recent advancements in artificial intelligence (AI) [1], [2], many deep neural network (DNN)-based applications have emerged in mobile systems [3], [4], such as human face recognition and augmented reality. Due to the tremendous amount of computation workload, the DNN-based applications cannot be fully executed at the wireless devices (WDs) with low-performance computing units and limited battery life [5]–[9]. Alternatively, the WDs can choose to offload the computations to a nearby server located at the network edge, referred to as edge inference [1]. Typically, the edge server can execute the whole DNN-based application on the WD's behalf after receiving the raw input data from the WD. However, due to the massive original input data (e.g., 3D images and videos), the excessive communication overhead makes it impractical to support delay-sensitive services [10], [11]. Such difficulty can be overcome by performing *device-edge co-inference*, where a large DNN is splitted into two parts. The first part with computation-friendly workload is executed on the WD, while the remaining part is computed on the edge server. The WD needs to transmit the output of the first part (i.e., the intermediate feature) to the edge server for further execution.

It is essential to determine at which layer the WD splits the DNN model, i.e., stops local computing and offloads the intermediate feature. The prior work on model splitting [12]–[17] showed that by carefully selecting the model splitting point, one can strike a balance between the on-device computation workload and the offloading communication overhead. Take the AlexNet [18] for example. Fig. 1 shows that a deeper splitting point (i.e., splitting the DNN at a later layer) in the AlexNet leads to a larger local computation workload and lower offloading data size. Besides, the model splitting decisions are affected by the time-varying wireless channel fading, e.g., deep fading may lead to a large offloading cost from the WD to the edge server. The existing model splitting methods [13]–[17] are based on offline optimization assuming non-causal channel knowledge. However, in practice, it is difficult for a WD to predict the channel state information (CSI) at a forthcoming model splitting point. Therefore, the model splitting point selection is an online decision process, where decisions must be made based on the past and present observations of wireless channel conditions without any future channel knowledge.

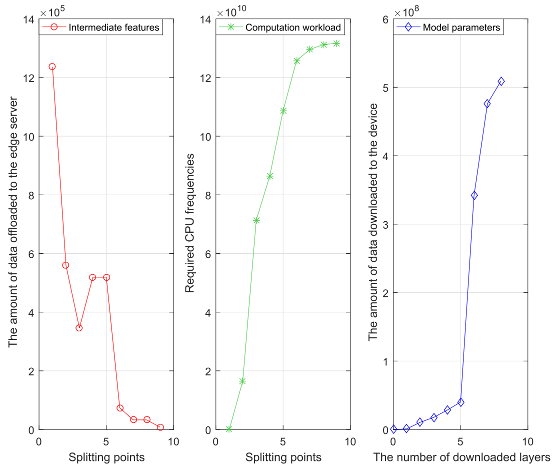

Fig. 1. The intermediate feature size, computation workload and model parameter data size in the AlexNet.

Prior work on device-edge co-inference assumes that the DNN model is completely stored at the WD. In practice, the DNN model needs to be updated from time to time through a training process at the edge server [1], [2], [18]. Once the model parameters are regenerated, the WD needs to download them from the edge server to facilitate on-device inference. Such model parameter downloading process is time-consuming in a wireless system due to the large model parameter size. For example, the total model parameter size in bytes is in the order of 10^8 in the AlexNet. We argue that for device-edge co-inference, it is not necessary to place all the DNN layers at the WD. This is because the layers after the model splitting point are never executed at the WD. Noticeably, the model placement decision must be jointly optimized with the model splitting strategy, because the WD is not able to split at a layer that is not downloaded.

The model placement and splitting decisions are made on two different time scales. On one hand, we need to make the model splitting decision for every DNN inference process due to the fast variation of wireless channel conditions. On the other hand, the DNN model parameters are updated at a much lower frequency than the DNN inference requests, and thus the model placement decision is made on a much larger time scale [1], [2], [18].

In this paper, we are interested in answering the following two key questions:

1) *On a large time scale, how many layers of the DNN model shall be placed at the WD, so that the expected device-edge co-inference cost is minimized?*
2) *On a fast time scale, how to choose the model splitting point to achieve the optimal tradeoff between the on-device computation and communication overhead when the future CSI is unknown?*

The main contributions of this paper are summarized in the following.

- *Online Model Splitting Strategy:* For given model placement decision, we formulate the optimal model splitting point selection problem as an optimal stopping problem [19]–[22] with finite horizon. We then solve the optimal stopping problem by backward induction to find the optimal model splitting strategy (i.e., the optimal stopping rule). Besides, we propose a suboptimal one-stage look-ahead (1-sla) stopping rule, where the analytical expressions of the model splitting strategy can be derived. Accordingly, we further analyze the optimality probability of the 1-sla stopping rule.
- *Optimal Model Placement Algorithm:* Based on the optimal stopping rule, we derive the long-term expected cost of the WD as a function of the model placement decision. Accordingly, the optimal model placement can be obtained by enumerating the $N+1$ possible decisions, where N is the total number of layers of the DNN. The brute-force search based model placement algorithm is computationally expensive, mainly because evaluating each model placement decision involves the full process of backward induction. To reduce the complexity, we propose an efficient 1-sla stopping rule based model placement algorithm. In particular, for a fully-connected multilayer perceptron with equal neurons at all the layers, we show that the optimal model placement solution can be obtained in closed form.
- *Performance Improvement:* Our simulation results show that the optimality probability of 1-sla model splitting strategy is as high as 0.9 in the AlexNet for any model placement decision. Besides, we demonstrate that the joint model placement and splitting algorithm significantly reduces the overall device-edge co-inference cost under various DNN structures.

B. Related Work

Existing work has extensively investigated the model splitting problem for device-edge co-inference [12]–[17]. Specifically, [14] proposed a three-step framework including the model splitting point selection, the communication-aware on-device model compression, and the task-oriented encoding for intermediate features. The model splitting point is selected via exhaustive search therein. The authors in [12] formulated the model splitting problem as an integer linear programming problem. In [15], a lightweight scheduler was designed to partition the DNN. Reference [16] investigated the encoding of the feature space for energy saving in edge-host partitioning of DNN. The authors in [17] proposed a 2-step pruning framework for DNN splitting in device-edge co-inference. The key assumption in [12]–[17] is that the WD knows the CSI at all the possible splitting points beforehand. This assumption, however, does not hold in practice since the wireless channel conditions at the forthcoming model decoupling points are random and unknown a priori.

Besides, the existing work [12]–[16] assumes that the WD has already stored the whole DNN model to enable device-edge co-inference. This incurs significant model placement cost when the DNN model is frequently updated. In [17], the WD downloads only part of the model that is needed for device-side computation. Nevertheless, the model placement decision is pre-determined and not optimized therein. In this

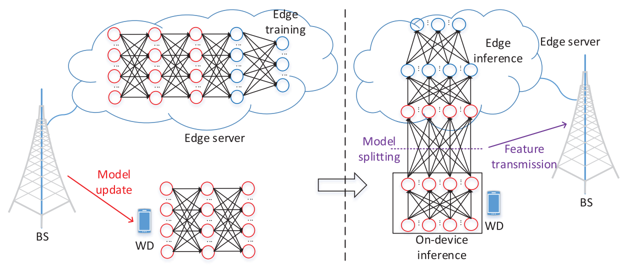

Fig. 2. An illustration of the model placement and model splitting for device-edge co-inference.

regard, the joint optimization of the model placement and online model partition strategy is largely overlooked in the literatures. This paper is among the first attempts to fill this gap.

C. Organization

The rest of the paper is organized as follows. Section II introduces the system model and the problem formulation. The optimal model splitting and placement strategies based on backward induction are proposed in Section III. In Section IV, we propose reduced-complexity algorithms based on the 1-sla stopping rule. We further propose a hybrid algorithm to balance the solution optimality and the computational complexity in Section IV. In Section V, simulation results are described. Finally, we conclude the paper in Section VI.

II. System Model and Problem Formulation

A. System Model

As shown in Fig. 2, we consider a mobile edge inference system with one base station (BS) and one WD. The BS is the gateway of the edge server and has a stable power supply. We consider a DNN-based application with the layered network structure, where the parameters of the DNN need to be periodically updated through a training process at the edge server. The edge server is interested in the inference result, i.e., the output of the DNN. On the other hand, the input of the DNN (e.g., image for the DNN-based human face recognition) is generated by the WD.

In most existing implementation, edge inference is either executed on device (device-only inference) or fully offloaded to the edge server (edge-only inference). To strike a balance between computation and communication overhead, we consider a device-edge co-inference framework. The WD executes the first few layers of the DNN and forwards the intermediate features to the edge server that executes the remaining layers. To enable device-edge co-inference, the BS needs to place the first few layers of the DNN at the WD. Note that once the DNN model is updated, the BS shall re-send the model to the WD. Take Fig. 2 for example. The BS deploys the first 3 layers (marked in red) at the WD. Then, the WD can choose to split the DNN (i.e., stop local execution and offload the intermediate features) after computing 0, 1, 2 or all the 3 layers.

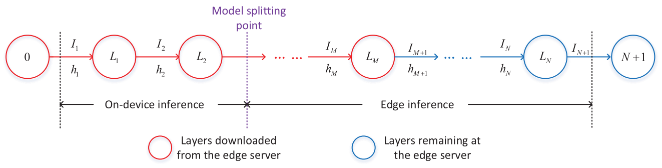

Fig. 3. Representation of the DNN as a sequential task graph.

Denote by N the total number of layers of the DNN, and by $M, 0 \leq M \leq N$, the number of layers placed at the WD. Suppose that the model can be used for K inference tasks before it is updated. Suppose that the downloading time of the i-th DNN layer is τ_i^m. Then, the average model placement cost per inference task is

$$\psi(M) = \frac{\sum_{i=1}^{M} \tau_i^m}{K}. \tag{1}$$

We denote by h_i^d the wireless channel gain when the i-th layer's parameters are downloaded. The noise at the receiver is additive white Gaussian noise (AWGN) with zero mean and variance σ^2. We denote the transmit power of the BS as P_d. Accordingly, the downloading data transmission rate for the i-th layer's parameters is $R_i^d = W_d \log_2(1 + \frac{P_d h_i^d}{\sigma^2})$, where W_d is the fixed downlink bandwidth. In this paper, we assume that the downlink transmission rate R_i^d of the BS is fixed. In practice, due to the strong transmit power and stable power supply at the BS, many power adaptation methods can be applied to achieve fixed target R_i^d by overcoming the effects of shadowing and small scale fading of downlink wireless channels. Then, by denoting the data size of the i-th layer's parameters in bits as D_i, the parameter downloading time of layer i is given by $\tau_i^m = \frac{D_i}{R_i^d}$.

In Fig. 3, we model the DNN based inference by a sequential task graph. Each vertex in the task graph represents a subtask, i.e., one layer of the DNN.[1] We denote the computational workload of subtask i in terms of the total number of CPU cycles as L_i. Besides, each edge in the task graph represents that the input data of subtask i is the output of the preceding subtask $i-1$. We denote the input data size in bits of subtask i as I_i. To reflect the fact that the input data is originated from the WD and the inference output is required by the edge server, we introduce two virtual subtasks 0 and $N+1$ as the entry and exit subtasks, respectively. In particular, subtasks 0 and $N+1$ must be executed at the WD and the edge server, respectively. Specifically, $L_0 = L_{N+1} = 0$.

We define the model splitting point n_k of inference task $k, k = 1, \ldots, K$, if subtasks 0 to $n_k - 1$ are executed on the WD and subtasks n_k to $N+1$ are computed at the edge server. The model splitting point n_k is determined by balancing the local computing cost, uplink transmission cost of the input

[1]In this paper, we consider the traditional sequential DNNs (e.g., AlexNet [18]), where input features flow layer by layer straightforward. In this case, each vertex (i.e., subtask) in the task graph represents one layer of such DNNs. It is worth noting that our proposed framework is applicable to all types of the DNNs [12-17]. Specifically, for the DNN models with branchy structures (e.g., ResNet [23]), each subtask in the sequential task graph represents an unit of such DNNs (e.g., one res-unit in ResNet).

data I_{n_k} of subtask n_k, and the edge computing cost. In the following, we focus on a tagged inference task and drop the subscript k for notational brevity. Note that the model splitting point is constrained by the number of layers placed at the WD. That is,

$$1 \leq n \leq M+1. \tag{2}$$

In particular, $n = 1$ implies edge-only inference, while $n = N+1$ implies device-only inference.

Denote by h_n the channel gain when the WD offloads the input data I_n at the model splitting point n. We assume that the transmit power of the WD is fixed as P. Suppose that the instantaneous signal-to-noise ratio (SNR) $\gamma_n = \frac{Ph_n}{\sigma^2}$ at model splitting point n is random with probability density function (PDF) $f_n(\cdot)$ and cumulative distribution function (CDF) $F_n(\cdot)$.[2] We assume that the γ_n is independent across different model splitting points [7], [24], [25]. Note that the time difference between splitting points n and $n+1$ is the local execution time of subtask n. In this paper, we assume that the channel coherence time is shorter than or comparable with the local computing time of one DNN layer,[3] such that the wireless channel conditions at two splitting points of the local DNN model may change and are considered independent.

Accordingly, the data transmission rate from the WD to the edge server at model splitting point n is

$$R_n(\gamma_n) = W\log_2(1+\gamma_n), \tag{3}$$

where W is the fixed bandwidth allocated to the WD. Then, the offloading transmission time at model splitting point n is

$$\tau_n^u(\gamma_n) = \frac{I_n}{R_n(\gamma_n)}, \tag{4}$$

and the corresponding energy consumption is

$$e_n^u(\gamma_n) = P\frac{I_n}{R_n(\gamma_n)}. \tag{5}$$

As for the on-device inference, we denote by f_l the local CPU frequency for computing the subtasks. Then, the local execution time for subtask i is

$$\tau_i^l = \frac{L_i}{f_l}, \tag{6}$$

and the corresponding energy consumption is

$$e_i^l = \kappa L_i f_l^2, \tag{7}$$

where κ is the effective switched capacitance parameter depending on the chip architecture.

As for the edge inference, the edge computing time of subtask i is

$$\tau_i^c = \frac{L_i}{f_c}, \tag{8}$$

where f_c is the CPU frequency of the edge server.

[2]In this paper, we consider the channel statistics (i.e., $f_n(\cdot)$ and $F_n(\cdot)$) are known. Suppose that the wireless channel statistic is partially known. We can first apply data-driven methods to estimate the PDF and CDF of the SNR from historical observed data. That is, we have a warm-up period to learn the wireless channel statistics.

[3]In practice, the channel coherence time is about several hundred milliseconds, while executing one layer of the DNN usually takes several seconds.

B. Problem Formulation

In this subsection, we first formulate the optimal model splitting problem as an optimal stopping problem with finite horizon $M+1$. Then, we formulate the joint model placement and splitting optimization problem in (12).

1) Optimal Stopping Problem With Given Model Placement Decision: Suppose that the first M layers of the DNN are placed at the WD. That is, the WD has to offload the intermediate features for edge inference no later than stage $M+1$. Let $\boldsymbol{\gamma} = \{\gamma_1, \ldots, \gamma_{M+1}\}$ denote a sequence of random variables representing the uplink SNR. Suppose that the WD can only observe the past and current SNRs, but not the future ones. At stage n, having observed γ_n, the WD needs to decide whether to continue computing the n-th layer locally or stop local computing and offload the n-th layer's input data (i.e., the $(n-1)$-th layer's output data) to the edge server.

If the WD decides to stop at stage n (i.e., split the model at point n), then the total energy-time cost (ETC), defined as the weighted sum of the WD's energy consumption and time on inference, is given by

$$\begin{aligned}\eta_n(\gamma_n) &= \beta_t(\sum_{i=0}^{n-1}\tau_i^l + \sum_{i=n}^{N+1}\tau_i^c + \tau_n^u(\gamma_n)) \\ &\quad +\beta_e(\sum_{i=0}^{n-1}e_i^l + e_n^u(\gamma_n)) \\ &= \omega_n + (\beta_t I_n + \beta_e P I_n)\frac{1}{R_n(\gamma_n)},\end{aligned} \tag{9}$$

where

$$\omega_n = \beta_t(\sum_{i=0}^{n-1}\frac{L_i}{f_l} + \sum_{i=n}^{N+1}\frac{L_i}{f_c}) + \beta_e(\sum_{i=0}^{n-1}\kappa L_i f_l^2). \tag{10}$$

Notice that ω_n increases in n due to the more powerful computation capacity at the edge server (i.e., $f_l < f_c$). Besides, β_t and β_e denote the weights of total inference time and energy consumption, respectively. Note that the ETC $\eta_n(\gamma_n)$ is deterministically known to the WD since it observes the SNR γ_n at stage n.

The WD decides to continue local computing at stage n only when doing so incurs a lower ETC than stopping at stage n. Note that the ETC of stopping at a future stage is random to the WD, as it does not know the channel SNR in the future.

A stopping rule determines the model splitting point $S(M,\boldsymbol{\gamma}) \in \{1, 2, \ldots, M+1\}$ based on the number of downloaded layers M and the random SNR observations $\boldsymbol{\gamma}$. Note that $S(M,\boldsymbol{\gamma})$ is random, as it is a function of random variables $\boldsymbol{\gamma}$. Different realizations of observations may lead to different stopping decisions. Given the model placement decision M, our purpose is to find the optimal model splitting strategy (i.e., optimal stopping rule) $S^*(M,\boldsymbol{\gamma})$ to minimize the expected inference ETC, i.e., $E_{\boldsymbol{\gamma}}[\eta_{S^*(M,\boldsymbol{\gamma})}(\gamma_{S^*(M,\boldsymbol{\gamma})})]$. In optimal stopping theory, this problem is a stopping rule problem with a finite horizon [19].

2) Joint Optimization of Model Placement and Model Splitting Strategy: Our goal is to find the optimal model splitting strategy $S^*(M,\boldsymbol{\gamma})$ and select the optimal number of downloaded layers $M, M \in \{0,1,\ldots,N\}$, with the objective to minimize the overall expected cost, defined as the expected ETC of the WD plus the weighted average model downloading time cost, i.e.,

$$Z(M,S^*(M,\boldsymbol{\gamma})) = \beta_t\psi(M) + E_{\boldsymbol{\gamma}}[\eta_{S^*(M,\boldsymbol{\gamma})}(\gamma_{S^*(M,\boldsymbol{\gamma})})]. \tag{11}$$

Mathematically, the optimization problem is formulated as

$$\begin{aligned}\text{(P1)}\quad &\min_{S^*(M,\boldsymbol{\gamma}),M} Z(M,S^*(M,\boldsymbol{\gamma})),\\ &\text{s.t. } M \in \{0,1,\ldots,N\},\\ &\qquad S^*(M,\boldsymbol{\gamma}) \in \{1,2,\ldots,M+1\}.\end{aligned} \tag{12}$$

Notice that the above Problem (P1) is challenging due to the combinatorial nature of the integer model placement decision M and the underlaying stopping rule problem given M.

III. Optimal Model Placement and Online Model Splitting Strategy

In this section, we first assume a fixed model placement decision M and investigate the optimal model splitting strategy $S^*(M,\boldsymbol{\gamma})$. Then, based on the analysis of the expected ETC achieved by the optimal model splitting strategy, we propose the optimal model placement algorithm.

A. Backward Induction

Given a finite horizon $M+1$, we solve the optimal stopping problem by backward induction. Notice that when $M=0$, the WD has no choice but to offload the data I_1 to the edge server at the first stage. In the following, we consider the case where the number of downloaded layers $M \geq 1$.

With backward induction, we first find the optimal model splitting strategy at stage M. Then, we find the optimal splitting strategy at stage $M-1$, taking the decision at stage M as given. The process continues backward until the first stage. Let $V_n^{(M+1)}$ denote the minimum expected inference ETC for splitting the model starting from stage $n, n \leq M$, given the current observation γ_n. That is,

$$\begin{aligned}V_n^{(M+1)} &= \min\left\{\eta_n(\gamma_n), E(V_{n+1}^{(M+1)})\right\}\\ &= \begin{cases}\eta_n(\gamma_n), & \gamma_n > \hat{\gamma}_n(M);\\ E(V_{n+1}^{(M+1)}), & \gamma_n < \hat{\gamma}_n(M),\end{cases}\end{aligned} \tag{13}$$

where

$$\hat{\gamma}_n(M) = 2^{\frac{(\beta_t I_n + \beta_e P I_n)}{W\left[E(V_{n+1}^{(M+1)})-\omega_n\right]}} - 1. \tag{14}$$

At the last stage $M+1$, we have

$$\begin{aligned}&E(V_{M+1}^{(M+1)})\\ &= \int_0^\infty \left[\beta_t(\sum_{i=0}^{M}\tau_i^l + \sum_{i=M+1}^{N+1}\tau_i^c + \tau_{M+1}^u(\gamma_{M+1}))\right.\\ &\qquad \left.+\beta_e(\sum_{i=0}^{M} e_i^l + e_M^u(\gamma_{M+1}))\right] f_{M+1}(\gamma_{M+1})d\gamma_{M+1}\\ &= \omega_{M+1} + (\beta_t I_{M+1} + \beta_e P I_{M+1})\\ &\qquad \times \int_0^\infty \frac{1}{R_{M+1}(\gamma_{M+1})} f_{M+1}(\gamma_{M+1})d\gamma_{M+1}.\end{aligned} \tag{15}$$

Inductively, at stage n,

$$\begin{aligned}E(V_n^{(M+1)}) &= E\left[\min\left\{\eta_n(\gamma_n), E(V_{n+1}^{(M+1)})\right\}\right]\\ &= \int_{\hat{\gamma}_n(M)}^\infty \eta_n(\gamma_n)f_n(\gamma_n)d\gamma_n\\ &\quad + \int_0^{\hat{\gamma}_n(M)} E(V_{n+1}^{(M+1)})f_n(\gamma_n)d\gamma_n\\ &= \omega_n(1-F(\hat{\gamma}_n(M))) + E(V_{n+1}^{(M+1)})F(\hat{\gamma}_n(M))\\ &\quad +(\beta_t I_n + \beta_e P I_n)\int_{\hat{\gamma}_n(M)}^\infty \frac{f_n(\gamma_n)}{R_n(\gamma_n)}d\gamma_n.\end{aligned} \tag{16}$$

Accordingly, we can calculate $E(V_n^{(M+1)})$ and $\hat{\gamma}_n(M)$ in (14) for all $n \leq M$. According to (13), it is optimal to stop at stage n if the observed SNR $\gamma_n > \hat{\gamma}_n(M)$, and to continue otherwise. In other words, the optimal stopping rule $S^*(M,\boldsymbol{\gamma})$ is a threshold-based policy determined by $\hat{\boldsymbol{\gamma}}(M)$, where $\hat{\boldsymbol{\gamma}}(M) = \{\hat{\gamma}_1(M),\ldots,\hat{\gamma}_M(M)\}$. We therefore have the following Proposition 3.1.

Proposition 3.1: Given the number of downloaded layers M, the optimal stopping rule $S^*(M,\boldsymbol{\gamma})$ is given by

$$S^*(M,\boldsymbol{\gamma}) = \begin{cases}\min\{\Psi\}, & \Psi \neq \emptyset\\ M+1, & \text{otherwise},\end{cases} \tag{17}$$

where

$$\Psi = \{n | 1 \leq n \leq M, \gamma_n \geq \hat{\gamma}_n(M)\}, \tag{18}$$

and $\hat{\gamma}_n(M)$ is defined in (14).

According to Proposition 3.1, when $\Psi \neq \emptyset$, our proposed online model splitting strategy is to split the DNN model at the first stage that the observed SNR γ_n is larger than or equal to the corresponding threshold $\hat{\gamma}_n(M)$. In addition, $S^*(M,\boldsymbol{\gamma}) = M+1$ when $\Psi = \emptyset$.

B. Expected Inference ETC Performance

In this subsection, we analyze the expected ETC $E_{\boldsymbol{\gamma}}[\eta_{S^*(M,\boldsymbol{\gamma})}(\gamma_{S^*(M,\boldsymbol{\gamma})})]$ achieved by the optimal model spitting. If $M=0$, the expected ETC $E_{\boldsymbol{\gamma}}[\eta_{S^*(M,\boldsymbol{\gamma})}(\gamma_{S^*(M,\boldsymbol{\gamma})})] = E(V_1^{(1)})$. In the following, we consider the case where $M \geq 1$. Specifically, we denote the probability of stopping at stage n as $Pr\{S^*(M,\boldsymbol{\gamma}) = n\}$. Then, we have

$$\begin{aligned}&Pr\{S^*(M,\boldsymbol{\gamma}) = n\}\\ &= \begin{cases}1 - F_1(\hat{\gamma}_1(M)), & n = 1;\\ \left(\prod_{j=1}^{n-1} F_j(\hat{\gamma}_j(M))\right) & \\ \times\left(1 - F_n(\hat{\gamma}_n(M))\right), & 1 < n < M+1;\\ \prod_{j=1}^{M} F_j(\hat{\gamma}_j(M)), & n = M+1.\end{cases}\end{aligned} \tag{19}$$

Specifically, for $1 < n < M+1$, stage n will be reached if all the observed SNRs at its preceding stages $j, \forall j < n$, are smaller than the corresponding derived threshold $\hat{\gamma}_j(M)$. Then, we will stop at stage n if the observed SNR γ_n is larger than the threshold $\hat{\gamma}_n(M)$. Besides, the expected ETC $\eta_n(\gamma_n)$ when $S^*(M,\boldsymbol{\gamma})=n$ is calculated as

$$E_{\boldsymbol{\gamma}}[\eta_{S^*(M,\boldsymbol{\gamma})}|S^*(M,\boldsymbol{\gamma})=n] = \begin{cases} \omega_n + E_{\gamma_n}\left[(\beta_t I_n + \beta_e P I_n)\frac{1}{R_n(\gamma_n)}|\gamma_n > \hat{\gamma}_n(M)\right], \\ \quad n \le M; \\ \omega_n + E_{\gamma_n}\left[(\beta_t I_n + \beta_e P I_n)\frac{1}{R_n(\gamma_n)}\right], \\ \quad n = M+1, \end{cases} \tag{20}$$

where the expectation is taken over the random SNR γ_n at stage n. That is,

$$E_{\boldsymbol{\gamma}}[\eta_{S^*(M,\boldsymbol{\gamma})}|S^*(M,\boldsymbol{\gamma})=n] = \begin{cases} \omega_n + (\beta_t I_n + \beta_e P I_n)\times \\ \quad \dfrac{\int_{\hat{\gamma}_n(M)}^{\infty}\frac{1}{R_n(\gamma_n)}f_n(\gamma_n)d\gamma_n}{1-F_n(\hat{\gamma}_n(M))}, \quad n \le M; \\ \omega_n + (\beta_t I_n + \beta_e P I_n)\times \\ \quad \int_0^{\infty}\dfrac{1}{R_n(\gamma_n)}f_n(\gamma_n)d\gamma_n, \qquad n = M+1. \end{cases} \tag{21}$$

Therefore, the expected ETC given that the first M layers are downloaded to the WD is

$$\begin{aligned} &E_{\boldsymbol{\gamma}}[\eta_{S^*(M,\boldsymbol{\gamma})}(\gamma_{S^*(M,\boldsymbol{\gamma})})] \\ &= \sum_{n=1}^{M+1} Pr\{S^*(M,\boldsymbol{\gamma})=n\} \\ &\quad \times E_{\boldsymbol{\gamma}}[\eta_{S^*(M,\boldsymbol{\gamma})}|S^*(M,\boldsymbol{\gamma})=n], M \ge 1. \end{aligned} \tag{22}$$

By substituting (19) and (21) into (22), the expected ETC $E_{\boldsymbol{\gamma}}[\eta_{S^*(M,\boldsymbol{\gamma})}(\gamma_{S^*(M,\boldsymbol{\gamma})})]$, and hence the total expected cost $Z(M)$ in (11), can be expressed as a function of M.

C. Optimal Model Placement

In this subsection, we are ready to optimize the model placement decision based on the optimal stopping rule $S^*(M,\boldsymbol{\gamma})$ derived in Proposition 3.1 and the analysis of expected ETC in Section III B. Intuitively, one can enumerate all feasible $M \in \{0,1,\ldots,N\}$ and select the optimal one that achieves the minimal expected cost $Z(M)$.

Notice that the decision threshold $\hat{\gamma}_n(M)$ for the optimal stopping rule involves a nested expectation of ETC in future stages (see (14) and (16)). For given M, it takes $O(M)$ complexity to compute $\hat{\gamma}_n(M)$ using backward induction. Then, by exhausting all feasible model placement decisions, the joint optimization of model placement and splitting for solving Problem (P1) incurs a polynomial computational time complexity $O(N^2)$.

Remark 3.1: Notice that the optimal model placement decision and online model splitting strategy obtained by solving Problem (P1) remain the same as long as the DNN model structure, the model update frequency parameter, and the statistic characteristics of wireless channels do not change. We claim that although solving Problem (P1) incurs $O(N^2)$ computational complexity, the online implementation for the model splitting follows a simple threshold-based policy with low complexity. That is, we simply need to compare the observed SNR related to the wireless channel gain with a threshold to decide whether to offload at the current model splitting point, where the overhead of such decision making is negligibly small. Once the optimal model placement decision is determined, the corresponding thresholds $\hat{\boldsymbol{\gamma}}(M)$ for the optimal stopping rule are simultaneously obtained and stored at the WD. This facilitates the online model splitting for every DNN inference according to Proposition 3.1.

IV. Reduced-Complexity Algorithms

The optimal model placement algorithm through exhaustive search results in $O(N^2)$ computational complexity due to the full process of backward induction when evaluating each feasible model placement decision. In this section, we are motivated to investigate linear-complexity, i.e., $O(N)$ algorithms based on a one-stage look-ahead stopping rule.

A. One-Stage Look-Ahead Stopping Rule for Model Splitting

First, we introduce the definition of one-stage look-ahead stopping rule.

Definition 1 (One-Stage Look-Ahead Stopping Rule): The one-stage look-ahead (1-sla) stopping rule is the one that stops if the cost for stopping at this stage is no more than the expected cost of continuing one stage and then stopping. Mathematically, the 1-sla stopping rule is described by the stopping time [19]

$$S_1 = \min\{n \ge 1 : \eta_n \le E_{\gamma_{n+1}}[\eta_{n+1}|\gamma_1,\ldots,\gamma_n]\}. \tag{23}$$

Note that the backward-induction based optimal stopping rule in Section III accounts for the ETC for all future stages until the last one. In contrast, the decision making at each stage under the 1-sla stopping rule only depends on the current observation and the expected performance of continuing for just one stage.

In the following, we first derive the 1-sla stopping rule for solving (P1) given the number of downloaded layers M.

Proposition 4.1: Given the number of downloaded layers M, the 1-sla stopping rule $S_1(M,\boldsymbol{\gamma})$ is given by

$$S_1(M,\boldsymbol{\gamma}) = \begin{cases} \min\{\Omega\}, & \Omega \ne \emptyset \\ M+1, & \text{otherwise}, \end{cases} \tag{24}$$

where

$$\Omega = \{n | 1 \le n \le M, \gamma_n \ge \hat{\gamma}_n^{1-sla}\}, \tag{25}$$

and $\hat{\gamma}_n^{1-sla}$ is given by (26), shown at the bottom of the page.

$$\hat{\gamma}_n^{1-sla} = 2^{\frac{1}{W}\frac{\beta_t I_n + \beta_e P I_n}{I_{n+1}(\beta_t+\beta_e P)\int_0^{\infty}\frac{1}{R_{n+1}}f_{n+1}(\gamma_{n+1})d\gamma_{n+1} + \beta_t(\frac{L_n}{f_l}-\frac{L_n}{f_c}) + \beta_e \kappa L_n f_l^2}} - 1. \tag{26}$$

Proof: According to Definition 1, the 1-sla stopping rule is

$$\begin{aligned}
&S_1(M,\boldsymbol{\gamma})\\
&\quad= \min\left\{1 \le n \le M : \eta_n \le E[V_{n+1}^{(n+1)}]\right\}\\
&\quad= \min\left\{1 \le n \le M : \eta_n \le E_{\gamma_{n+1}}[\eta_{n+1}|\gamma_1,\ldots,\gamma_n]\right\}\\
&\quad= \min\left\{1 \le n \le M : \omega_n + (\beta_t I_n + \beta_e P I_n)\frac{1}{R_n}\right.\\
&\quad\le \left.\int_0^{\infty} \eta_{n+1}(\gamma_{n+1}) f_{n+1}(\gamma_{n+1}) d\gamma_{n+1}\right\}\\
&\quad= \min\left\{1 \le n \le M : \gamma_n \ge \hat{\gamma}_n^{1-sla}\right\}, \qquad (27)
\end{aligned}$$

where $\hat{\gamma}_n^{1-sla}$ is given in (26).

When $\Omega = \{n | 1 \le n \le M, \gamma_n \ge \hat{\gamma}_n^{1-sla}\} = \emptyset$, then $S_1(M,\boldsymbol{\gamma}) = M+1$. ■

From Proposition 4.1, we have the following observations:

- The decision made in each stage $n \in [1, M]$ in the 1-sla stopping rule only depends on the expectation of the unit transmission delay $\frac{1}{R_{n+1}}$ at the next stage $n+1$ and the model parameters in the n-th layer (i.e., the input and output data sizes of the n-th layer I_n, I_{n+1} and the computation workload L_n of the n-th layer), regardless of M.
- Compared with the calculation of $\hat{\gamma}_n$ in the optimal stopping rule via backward induction, the decision threshold $\hat{\gamma}_n^{1-sla}$ in $S_1(M,\boldsymbol{\gamma})$ is much easier to calculate without the nested structure for the expected ETC in all future stages.
- The probability of stopping at stage n decreases when the input data size I_n of stage n increases, the input data size of the next stage I_{n+1} is smaller, or the difference between the local and edge execution costs for layer n is lower.

According to Proposition 4.1, when $\Omega \ne \emptyset$, the 1-sla stopping rule based online model splitting scheme is to split the model at the first point where $\gamma_n \ge \hat{\gamma}_n^{1-sla}$ occurs.

The 1-sla stopping rule is not optimal in general. According to [19], the 1-sla rule is optimal in a finite horizon monotone stopping rule problem. In the following Lemma 4.1 and Proposition 4.2, we show that the proposed 1-sla based model splitting strategy $S_1(M,\boldsymbol{\gamma})$ is optimal with certain probability.

Lemma 4.1: Let A_n denote the event $\{\eta_n \le E[\eta_{n+1}|\gamma_1,\ldots,\gamma_n]\}$, i.e., the 1-sla calls for stopping at stage n, for a given $n \in [1, M]$. Suppose that A_n holds. Then, if $A_{n+1},\ldots,A_M$ also hold, the 1-sla stopping rule $S_1(M,\boldsymbol{\gamma})$ is optimal, i.e., $S_1(M,\boldsymbol{\gamma}) = S^*(M,\boldsymbol{\gamma})$.

Proof: Recall that the optimal stopping rule is

$$S^*(M,\boldsymbol{\gamma}) = \min\{n \ge 1 : \eta_n \le E(V_{n+1}^{(M+1)}|\gamma_1,\ldots,\gamma_n)\},$$

where $V_{M+2}^{(M+1)} = +\infty$, $V_{M+1}^{(M+1)} = \eta_{M+1}$, and by backward induction,

$$V_n^{(M+1)} = \min\{\eta_n, E(V_{n+1}^{(M+1)}|\gamma_1,\ldots,\gamma_n)\},$$

for $n = 1,\ldots,M$.

Suppose that the 1-sla calls for stopping at stage n and $A_{n+1},\ldots,A_M$ also hold. First, for A_M, we have

$$\eta_M \le E(\eta_{M+1}|\gamma_1,\ldots,\gamma_M) = E(V_{M+1}^{(M+1)}|\gamma_1,\ldots,\gamma_M).$$

Hence,

$$V_M^{(M+1)} = \min\{\eta_M, E(V_{M+1}^{(M+1)}|\gamma_1,\ldots,\gamma_M)\} = \eta_M.$$

Then, for A_{M-1}, we have

$$\eta_{M-1} \le E(\eta_M|\gamma_1,\ldots,\gamma_{M-1}) = E(V_M^{(M+1)}|\gamma_1,\ldots,\gamma_{M-1}).$$

Hence,

$$V_{M-1}^{(M+1)} = \min\{\eta_{M-1}, E(V_M^{(M+1)}|\gamma_1,\ldots,\gamma_{M-1})\} = \eta_{M-1}.$$

Similarly, for $n < k < M-1$, we have $V_k^{(M+1)} = \eta_k$. Finally, for A_n, we have

$$\eta_n \le E(\eta_{n+1}|\gamma_1,\ldots,\gamma_n) = E(V_{n+1}^{(M+1)}|\gamma_1,\ldots,\gamma_n).$$

Hence,

$$V_n^{(M+1)} = \min\{\eta_n, E(V_{n+1}^{(M+1)}|\gamma_1,\ldots,\gamma_n)\} = \eta_n.$$

Therefore, the optimal stopping rule $S^*(M,\boldsymbol{\gamma})$ also calls for stopping at stage n. ■

Based on the above lemma, we are ready to derive the probability that 1-sla stopping rule is optimal.

Proposition 4.2: Given the number of downloaded layers M, the 1-sla stopping rule $S_1(M,\boldsymbol{\gamma})$ obtained by (24) is optimal with probability

$$\begin{aligned}
&Pr\{S_1(M,\boldsymbol{\gamma}) = S^*(M,\boldsymbol{\gamma})\}\\
&\quad= \sum_{n=1}^{M+1}\left[\prod_{j=1}^{n-1} F_j(\hat{\gamma}_j^{1-sla}) \prod_{k=n}^{M}[1 - F_k(\hat{\gamma}_k^{1-sla})]\right]. \qquad (28)
\end{aligned}$$

Proof: If the 1-sla stopping rule $S_1(M,\boldsymbol{\gamma})$ obtained by (24) calls for stopping at stage n, then it will also call for stopping at all future stages with probability

$$\begin{aligned}
&Pr\{\text{Stage } n \text{ is reached and } A_n, A_{n+1},\ldots,A_M \text{ hold}\}\\
&\quad= \prod_{j=1}^{n-1} F_j(\hat{\gamma}_j^{1-sla}) \prod_{k=n}^{M}[1 - F_k(\hat{\gamma}_k^{1-sla})]. \qquad (29)
\end{aligned}$$

Then, according to Lemma 4.1, we can obtain the probability $Pr\{S_1(M,\boldsymbol{\gamma}) = S^*(M,\boldsymbol{\gamma})\}$ as shown in (28). ■

Corollary 4.1: When the number of downloaded layers $M = 1$, the 1-sla stopping rule is optimal, i.e., $S_1(M,\boldsymbol{\gamma}) = S^*(M,\boldsymbol{\gamma})$.

Proof: According to Proposition 4.2, when $M = 1$, we have

$$\begin{aligned}
&Pr\{S_1(1,\boldsymbol{\gamma}) = S^*(1,\boldsymbol{\gamma})\}\\
&\quad= [1 - F_1(\hat{\gamma}_1^{1-sla})] + F_1(\hat{\gamma}_1^{1-sla}) = 1. \qquad (30)
\end{aligned}$$

Hence, when $M = 1$, the 1-sla stopping rule is optimal. ■

B. 1-Sla Stopping Rule Based Model Placement

Following the similar technique in Section III B, we can analyze the expected ETC performance $E_{\boldsymbol{\gamma}}[\eta_{S_1(M,\boldsymbol{\gamma})}(\gamma_{S_1(M,\boldsymbol{\gamma})})]$, and hence the overall expected cost $Z(M)$, as a function of model placement M under the 1-sla stopping rule. Then, we can enumerate all feasible $M \in \{0, 1, \ldots, N\}$ to find M^* that yields the minimal expected cost $Z(M)$.

It takes $O(N)$ complexity to compute all possible $\{\hat{\gamma}_n^{1-sla}, \forall n \in [1, N]\}$ in $S_1(M, \boldsymbol{\gamma})$ according to (26). Besides, the complexity of enumerating all feasible model placement decisions is $O(N)$. Notice that the decision threshold $\hat{\gamma}_n^{1-sla}$ under the 1-sla stopping rule is independent of the model placement M. Therefore, the overall computational complexity for solving Problem (P1) reduces to linear complexity $O(N)$ under the 1-sla stopping rule based algorithm.

C. Case Study: Fully-Connected Multilayer Perceptron

To obtain more engineering insights, we consider fully-connected multilayer perceptron (MLP) networks in this subsection. We will show that under certain assumptions, the closed-form expressions of the optimal model placement M^* can be derived when the model splitting decision is based on the 1-sla stopping rule.

Fully-connected MLP [26] is a class of feedforward artificial neural network, which consists of an input layer, multiple hidden layers and an output layer. Except for the nodes of the input layer, each node of the other layers is a neuron that uses a nonlinear activation function.

Suppose that layer i has X_i neurons with input data size

$$I_i = \lambda X_{i-1}, \tag{31}$$

where λ is the number of bytes to represent the original output value of each neuron. In Fig. 4, we illustrate the floating-point multiply-add calculations in one neuron of layer i. Each neuron in layer i first performs X_{i-1} multiply-add operations on the input data received from the previous layer using the weights (e.g., $\{w_1, w_2, \ldots, w_{X_{i-1}}\}$ in Fig. 4) and biases (e.g., b in Fig. 4) defined by the model parameters. Then, the result is further processed by a nonlinear activation function (e.g., $\phi(\cdot)$ in Fig. 4). It can be seen that the number of model parameters required for the inference of layer i and the total computation workload of layer i depend on the number of neurons in layer $i-1$ and i.

From the above, we see that the computation workload of each layer is proportional to $X_{i-1}X_i$, i.e.,

$$L_i = \alpha X_{i-1} X_i, \tag{32}$$

where α is the number of CPU cycles required to execute one floating-point multiply-add operation. Likewise, the number of model parameters needed by layer i, including the weights and biases, is $(X_{i-1}+1)X_i$. Hence, the parameter downloading time for layer i is

$$\tau_i^m = \frac{\mu(X_{i-1}+1)X_i}{R^d}, \tag{33}$$

where μ is the number of bytes to represent each model parameter and R^d is the fixed downlink transmission data rate.

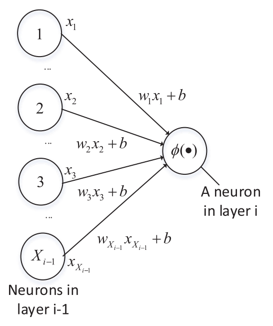

Fig. 4. An illustration of the floating-point multiply-add calculations in one neuron of layer i, where $\{w_1, w_2, \ldots\}$ and b are the parameters corresponding to that neuron in layer i and $\phi(\cdot)$ is the nonlinear activation function.

In the following, we assume that the number of neurons in each layer remains the same (i.e., $X_{i-1} = X_i, \forall i$). For the simplicity of analysis, we assume that the SNR γ_n in each stage is identically distributed, i.e., $f_1(\cdot) = f_2(\cdot) = \ldots = f(\cdot)$ and $F_1(\cdot) = F_2(\cdot) = \ldots = F(\cdot)$.

Lemma 4.2: For a fully-connected MLP with $X_{i-1} = X_i = X, \forall i$, the 1-sla stopping threshold $\hat{\gamma}_n^{1-sla}$ in (26) is a constant, i.e., $\hat{\gamma}_1^{1-sla} = \ldots = \hat{\gamma}_N^{1-sla} = \delta(X)$, where

$$\begin{aligned}\delta(X) &\\ &= 2^{\frac{1}{W}\frac{(\beta_t+\beta_e P)\lambda}{\lambda(\beta_t+\beta_e P)\int_0^\infty \frac{1}{R_{n+1}} f(\gamma_{n+1}) d\gamma_{n+1} + [\beta_t(\frac{1}{f_l}-\frac{1}{f_c})+\beta_e \kappa f_l^2]\alpha X}} - 1.\end{aligned} \tag{34}$$

Lemma 4.2 indicates that for a fully-connected MLP with fixed number of neurons in each layer, the 1-sla stopping rule is to compare the observed SNR γ_n with a constant $\delta(X)$.

To derive the model placement solution, the following lemma studies the relation between the expected ETC $E_{\boldsymbol{\gamma}}[\eta_{S_1(M,\boldsymbol{\gamma})}(\gamma_{S_1(M,\boldsymbol{\gamma})})]$ and the number of downloaded layers M under the 1-sla stopping rule.

Lemma 4.3: With the increase of M, the expected inference ETC $E_{\boldsymbol{\gamma}}[\eta_{S_1(M,\boldsymbol{\gamma})}(\gamma_{S_1(M,\boldsymbol{\gamma})})]$ under the 1-sla rule decreases. That is,

$$\begin{aligned}\Theta^{1-sla}&(M)\\ &= E_{\boldsymbol{\gamma}}[\eta_{S_1(M,\boldsymbol{\gamma})}(\gamma_{S_1(M,\boldsymbol{\gamma})})]\\ &\quad -E_{\boldsymbol{\gamma}}[\eta_{S_1(M-1,\boldsymbol{\gamma})}(\gamma_{S_1(M-1,\boldsymbol{\gamma})})] < 0, \quad \forall M \in [1, N].\end{aligned} \tag{35}$$

Proof: The difference between the expected ETC when downloading M and $M-1$ layers is calculated in (36), shown at the bottom of the next page.

According to (24), $\forall \gamma_M < \hat{\gamma}_M^{1-sla}$, we have

$$\begin{aligned}\omega_M &+ (\beta_t I_M + \beta_e P I_M)\frac{1}{R_M(\gamma_M)}\\ &> \omega_{M+1} + (\beta_t I_{M+1} + \beta_e P I_{M+1})\\ &\quad \times \int_0^\infty \frac{1}{R_{M+1}(\gamma_{M+1})} f(\gamma_{M+1}) d\gamma_{M+1}.\end{aligned} \tag{37}$$

Therefore, we have $\Theta^{1-sla}(M) < 0, \forall M \in [1, N]$. ■

From Lemma 4.3, we observe that when the edge server places more layers to the WD (i.e., larger M), the expected

ETC decreases. Nevertheless, according to (1), a larger downloading time cost $\psi(M)$ occurs when M increases.

Corollary 4.2: When the model update frequency parameter K is sufficiently large, the optimal $M^* = N$.

Proof: When K is sufficiently large, the average model downloading time cost $\psi(M) = \frac{\sum_{i=1}^{M} \tau_i^m}{K}$ diminishes. Then, according to Lemma 4.3, we have $M^* = N$. ■

Based on Lemma 4.3, we are ready to derive the optimal model placement solution in the following proposition.

Proposition 4.3: If the fully-connected MLP satisfies $X_{i-1} = X_i = X, \forall i$ and the model splitting is based on the 1-sla stopping rule, then the optimal number of downloaded layers is given by (38), shown at the bottom of the page, where $\langle\cdot\rangle$ is the rounding function and $g(\delta(X))$ is given in (40).

Proof: For the fully-connected MLP with $X_{i-1} = X_i = X, \forall i$, we first rewrite $\Theta^{1-sla}(M)$ in (35) as

$$\Theta^{1-sla}(M) = X[F(\delta(X))]^M g(\delta(X)), 1 \leq M \leq N, \quad (39)$$

where

$$\begin{aligned} &g(\delta(X)) \\ &= \alpha X \beta_t (\frac{1}{f_l} - \frac{1}{f^c}) + \alpha X \beta_e \kappa f_l^2 + (\beta_t + \beta_e P)\lambda \\ &\quad \times (\int_0^\infty \frac{1}{R_{M+1}(\gamma_{M+1})} f(\gamma_{M+1}) d\gamma_{M+1} \\ &\quad - \frac{\int_0^{\delta(X)} \frac{1}{R_M(\gamma_M)} f(\gamma_M) d\gamma_M}{F(\delta(X))}) \\ &= (\beta_t + \beta_e P)\lambda \left(\frac{1}{R_M(\delta(X))} - \frac{\int_0^{\delta(X)} \frac{1}{R_M(\gamma_M)} f(\gamma_M) d\gamma_M}{F(\delta(X))} \right). \end{aligned} \quad (40)$$

Then, we have

$$\begin{aligned} &\Delta^{1-sla}(M) \\ &= Z(M) - Z(M-1) = \frac{\mu X(X+1)}{KR^d} + \Theta^{1-sla}(M) \\ &= X \left[[F(\delta(X))]^M g(\delta(X)) + \frac{\mu(X+1)}{KR^d} \right]. \end{aligned} \quad (41)$$

Since $g(\delta(X)) < 0$ and $0 < F(\delta(X)) < 1$, $[F(\delta(X))]^M g(\delta(X)) + \frac{\mu(X+1)}{KR^d}$ increases with M. Accordingly, if $\Delta^{1-sla}(N) < 0$, i.e., $[F(\delta(X))]^N g(\delta(X)) + \frac{\mu(X+1)}{KR^d} < 0$, $Z(M)$ decreases with M and we have $M^* = N$. If $\Delta^{1-sla}(1) > 0$, i.e., $[F(\delta(X))]g(\delta(X)) + \frac{\mu(X+1)}{KR^d} > 0$, $Z(M)$ increases with M, and thus $M^* = 0$.

Otherwise, the root of $\Delta^{1-sla}(M) = 0$ can be calculated as $M^* = \left\langle \log_{F(\delta(X))} \left(\frac{\mu(X+1)}{-KR^d g(\delta(X))} \right) \right\rangle$. We have $\Delta^{1-sla}(M) < 0$ when $M \leq M^*$ and $\Delta^{1-sla}(M) > 0$ when $M > M^*$. That is, $Z(M)$ decreases with M when $M \leq M^*$ and $Z(M)$ increases with M when $M > M^*$. In this case, the optimal number of downloaded layers is $\left\langle \log_{F(\delta(X))} \left(\frac{\mu(X+1)}{-KR^d g(\delta(X))} \right) \right\rangle$. ■

From Proposition 4.3, we have the following observations:

- When the total number of layers N increases, the BS tends to place part of the model to the WD, instead of downloading the whole DNN. Specifically, for a sufficiently large DNN model (i.e., $N \to \infty$), we have $[F(\delta(X))]^N g(\delta(X)) \to 0$ and the optimal model placement decision $M^* < N$.
- If the downlink data transmission rate increases (e.g., meeting better downlink wireless channel conditions or

$$\begin{aligned} \Theta^{1-sla}(M) &= E_{\boldsymbol{\gamma}}[\eta_{S_1(M,\boldsymbol{\gamma})}(\gamma_{S_1(M,\boldsymbol{\gamma})})] - E_{\boldsymbol{\gamma}}[\eta_{S_1(M-1,\boldsymbol{\gamma})}(\gamma_{S_1(M-1,\boldsymbol{\gamma})})] \\ &= \left(\prod_{j=1}^{M-1} F(\hat{\gamma}_j^{1-sla}) \right) \left[(1 - F(\hat{\gamma}_M^{1-sla})) \left(\omega_M + (\beta_t I_M + \beta_e P I_M) \frac{\int_{\hat{\gamma}_M^{1-sla}}^\infty \frac{1}{R_M(\gamma_M)} f(\gamma_M) d\gamma_M}{1 - F(\hat{\gamma}_M^{1-sla})} \right) \right. \\ &\quad + F(\hat{\gamma}_M^{1-sla}) \left(\omega_{M+1} + (\beta_t I_{M+1} + \beta_e P I_{M+1}) \int_0^\infty \frac{1}{R_{M+1}(\gamma_{M+1})} f(\gamma_{M+1}) d\gamma_{M+1} \right) \\ &\quad \left. - \left(\omega_M + (\beta_t I_M + \beta_e P I_M) \int_0^\infty \frac{1}{R_M(\gamma_M)} f(\gamma_M) d\gamma_M \right) \right] \\ &= \left(\prod_{j=1}^{M} F(\hat{\gamma}_j^{1-sla}) \right) \left[\omega_{M+1} + (\beta_t I_{M+1} + \beta_e P I_{M+1}) \int_0^\infty \frac{1}{R_{M+1}(\gamma_{M+1})} f(\gamma_{M+1}) d\gamma_{M+1} \right. \\ &\quad \left. - \left(\omega_M + (\beta_t I_M + \beta_e P I_M) \frac{\int_0^{\hat{\gamma}_M^{1-sla}} \frac{1}{R_M(\gamma_M)} f(\gamma_M) d\gamma_M}{F(\hat{\gamma}_M^{1-sla})} \right) \right] \end{aligned} \quad (36)$$

$$M^* = \begin{cases} N, & [F(\delta(X))]^N g(\delta(X)) + \frac{\mu(X+1)}{KR^d} < 0; \\ 0, & F(\delta(X)) g(\delta(X)) + \frac{\mu(X+1)}{KR^d} > 0; \\ \left\langle \log_{F(\delta(X))} \left(\frac{\mu(X+1)}{-KR^d g(\delta(X))} \right) \right\rangle, & \text{otherwise} \end{cases} \quad (38)$$

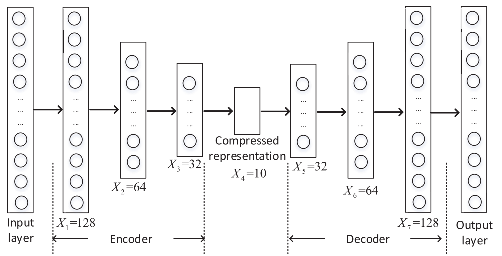

Fig. 5. The considered autoencoder structure.

stronger transmit power of the BS), more layers of the DNN will be placed to the WD.

- When the model is updated less frequently (i.e., a larger K), the optimal number of downloaded layers M^* increases. Specifically, when K is sufficiently large, $[F(\delta(X))]^N g(\delta(X)) + \frac{\mu(X+1)}{KR^d} < 0$ always holds, and we have $M^* = N$, which is consistent with Corollary 4.2.

According to Proposition 4.3, we can optimize the number of downloaded layers M using the closed-form expression (38) directly.

D. Hybrid Algorithm

Notice that it takes a linear computational complexity to solve Problem (P1) when the model splitting decision is based on the suboptimal 1-sla stopping rule. On the other hand, according to Section III C, the backward-induction based optimal model placement and splitting algorithm takes a polynomial time complexity. As an alternative to balance between solution optimality and computational complexity, we propose in this subsection a hybrid algorithm to solve the Problem (P1). In particular, we find the approximated optimal model placement decision M^{hybrid}, assuming the 1-sla stopping rule. That is

$$M^{hybrid} = \arg\min_{M} Z(M, S_1(M, \boldsymbol{\gamma})). \tag{42}$$

Then, we fix M^{hybrid} and find the optimal model splitting point $S^*(M^{hybrid}, \boldsymbol{\gamma})$ using the backward induction algorithm in Section III A. Based on the complexity discussions in Section III C and Section IV B, the overall computational complexity of the hybrid algorithm is $O(N)$. The proposed hybrid algorithm also has a linear computational complexity, while achieving a better performance compared with the 1-sla stopping rule based algorithm in Section IV B.

V. Simulation Results

In this section, we evaluate the proposed algorithms with different DNN architectures. Specifically, we consider the classical autoencoder and AlexNet. As shown in Fig. 5, the considered autoencoder consists of one input layer, one output layer, and seven hidden layers, where the number of neurons in the hidden layers is $\{X_i\} = \{128,\ 64,\ 32,\ 10,\ 32,\ 64,\ 128\}$. We set the number of neurons in the output layer as 784, which is the same as that in the input layer. Likewise, as shown in Fig. 6, the considered AlexNet contains eight layers: the first five are convolutional and the remaining three are fully-connected. The configuration of the AlexNet follows [18], where there are 1000 class lables and the input image size is $227*227*3$. We suppose that $\lambda = \mu = 8$ Bytes, where λ and μ are defined in (31) and (33), respectively. In addition, one floating-point multiply-add calculation requires $\alpha = 100$ CPU cycles.

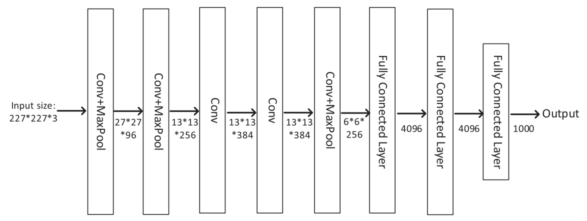

Fig. 6. The considered AlexNet structure.

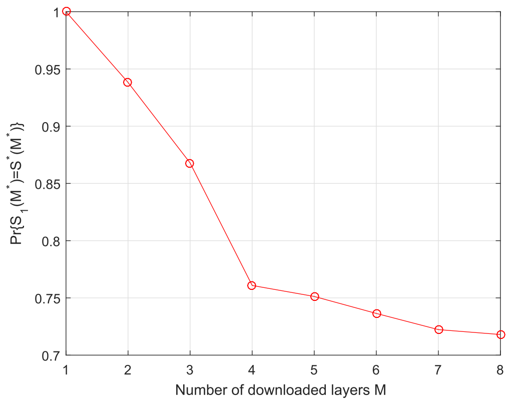

Fig. 7. Optimality probability for one-stage look-ahead rule versus the number of downloaded layers M in the considered autoencoder.

The wireless channels are modeled as the Rayleigh block fading with large-scale path loss. In this case, the instantaneous SNR γ_n is exponentially distributed with PDF

$$f_n(\gamma_n) = \frac{1}{\bar{\gamma}_n} e^{-\frac{\gamma_n}{\bar{\gamma}_n}}, \tag{43}$$

where $\bar{\gamma}_n = \frac{P}{\sigma^2} A_d (\frac{3 \cdot 10^8}{4\pi f^c d})^{PL}$ is the average SNR at model splitting point n. $A_d = 4.11$ denotes the antenna gain, $f^c = 915$ MHz denotes the carrier frequency, d in meters denotes the distance between the WD and the BS, and $PL = 3$ denotes the pass loss exponent. The transmit power of the WD and the BS is 100 mW and 1 W, respectively. The noise power $\sigma^2 = 10^{-10}$ W. We set the computing efficiency parameter $\kappa = 10^{-26}$, and the bandwidth $W = 2$ MHz. The weights of energy consumption and time of the WD are set as $\beta_t = \beta_e = 0.5$. The CPU frequencies at the WD and the edge server are 10^8 and 10^{10} cycles/second, respectively.

A. Optimality Analysis for One-Stage Look-Ahead Stopping Rule

In Fig. 7 and Fig. 8, we plot the optimality probability of the 1-sla stopping rule $Pr\{S_1(M, \boldsymbol{\gamma}) = S^*(M, \boldsymbol{\gamma})\}$ as a function of M when $d = 50$ meters with the autoencoder and AlexNet, respectively. We observe that the optimality

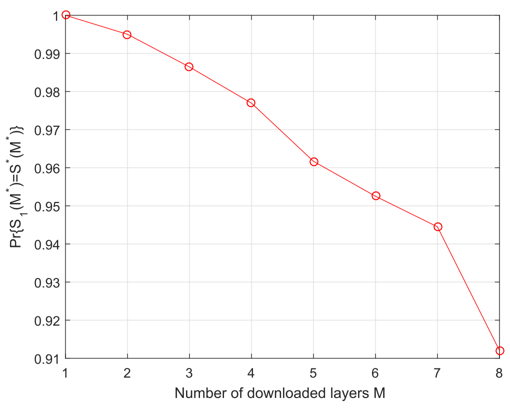

Fig. 8. Optimality probability for one-stage look-ahead rule versus the number of downloaded layers M in the considered AlexNet.

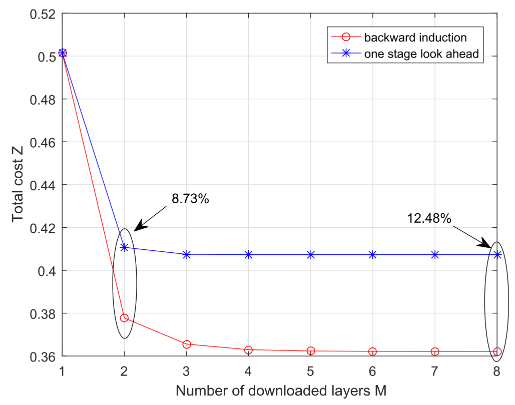

Fig. 9. Performance gap between backward induction and one-stage look-ahead rule when $K = \infty$ in the considered autoencoder.

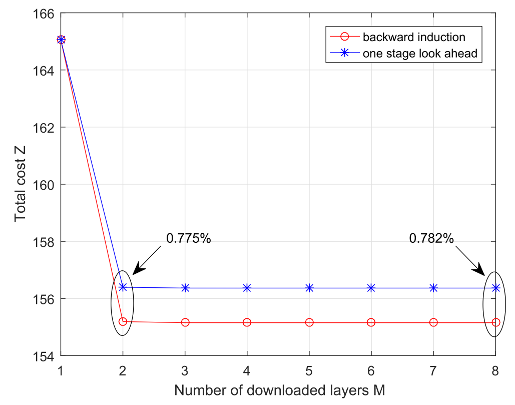

Fig. 10. Performance gap between backward induction and one-stage look-ahead rule when $K = \infty$ in the considered AlexNet.

probability decreases with M for both the autoencoder and AlexNet. It is because the 1-sla stopping rule becomes more myopic when the optimal stopping problem has a larger horizon (i.e., M is large). It can be seen that when $M = 1$, $Pr\{S_1(M,\gamma) = S^*(M,\gamma)\} = 1$, which means that the 1-sla stopping rule $S_1(1,\gamma)$ is equal to the optimal $S^*(1,\gamma)$ in this case. Besides, we observe that the optimality probability is larger than 0.91 for any M with the AlexNet. This confirms the effectiveness of the 1-sla stopping rule. Besides, the optimality probability of the 1-sla stopping rule with the autoencoder is larger than 0.85 when $M \leq 3$. We claim the effectiveness of the 1-sla stopping rule for the autoencoder under some system settings. We will demonstrate this point in Fig. 13 in Section V C. One interesting observation is that the optimality probability of the 1-sla stopping rule for the AlexNet is larger than that for the autoencoder. It is due to the fact that compared with the autoencoder, higher computation workloads of the early convolutional layers in the AlexNet dominate the decision making, which leads to an early stopping with higher probability for the backward-induction based optimal stopping rule. In this case, the myopic 1-sla stopping rule is close to the optimum by accounting for the expected ETC of continuing for just one stage.

B. Performance Evaluation When $K = \infty$

In this subsection, we consider the scenario where $K = \infty$, i.e., the cost of model placement is negligible. In this case, the overall average cost Z is equal to the average inference cost $E_{\gamma}[\eta_{S(M,\gamma)}(\gamma_{S(M,\gamma)})]$.

In Fig. 9 and Fig. 10, we illustrate the total average cost Z versus the number of downloaded layers M when $K = \infty$ and $d = 50$ meters. We observe that for both the autoencoder and AlexNet, Z decreases with the increase of M for both optimal (through backward induction) and 1-sla stopping rules. It is because when more layers are downloaded from the edge server, the WD has more choices of the model splitting points, which improves the average inference performance. In particular, when $M = 1$, the 1-sla stopping rule has the same performance as the optimal stopping rule. In addition, the gap of Z between the optimal and 1-sla stopping rules increases when M grows. For example, Z grows from 8.73% to 12.48% when M increases from 2 to 8 in Fig. 9.

Compared to the autoencoder, the performance gap between the optimal and 1-sla stopping rules in the AlexNet is smaller, e.g., 0.782% when $M = 8$. It is because as illustrated in Fig. 7 and Fig. 8, the optimality probability of the 1-sla stopping rule for the AlexNet is always larger than that for the autoencoder. Moreover, Fig. 10 shows that the inference cost in the AlexNet remains stable as M varies from 2 to 8. It is due to the fact that the heavy computation workloads of the early convolutional layers in the AlexNet dominate the decision making for the model splitting point selection. That is, even though more layers are downloaded, the WD still prefers to stop at an early stage.

C. Performance Evaluation When $K < \infty$

We now consider the general scenario where the DNN parameters need to be updated from time to time, i.e., $K < \infty$.

In Fig. 11, we plot the optimal M^* as a function of the distance d under different model update frequencies K with the autoencoder. It is observed that the optimal M^* increases with the distance d. This is because a larger distance leads to a higher uplink offloading cost for the intermediate feature. In this case, the average inference cost can be improved by downloading more layers to the WD. Besides, we observe that

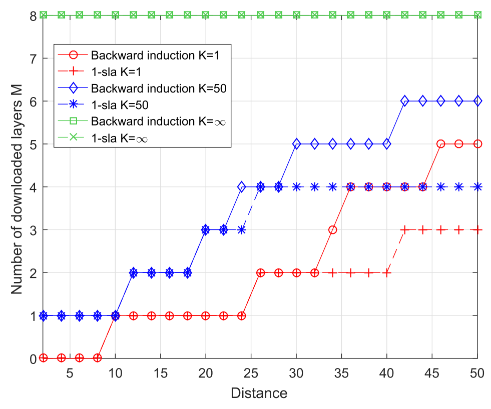

Fig. 11. The number of downloaded layers M versus the distance between the WD and edge server in the considered autoencoder.

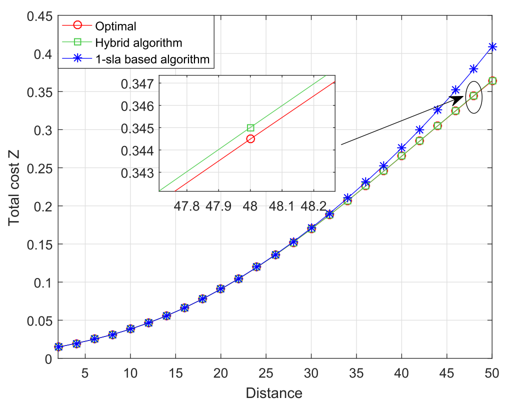

Fig. 13. Total cost versus the distance between the WD and edge server when $K = 50$ in the considered autoencoder.

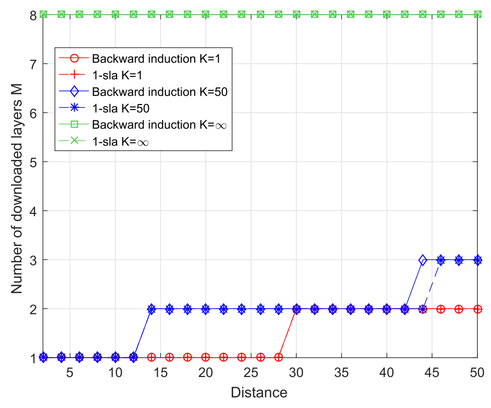

Fig. 12. The number of downloaded layers M versus the distance between the WD and edge server in the considered AlexNet.

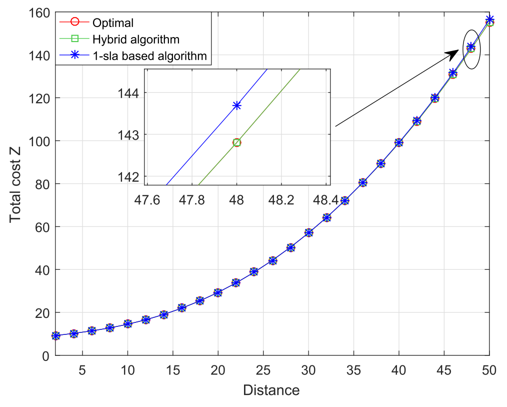

Fig. 14. Total cost versus the distance between the WD and edge server when $K = 50$ in the considered AlexNet.

the optimal M^* increases with the model update frequency parameter K, which means that the BS tends to download more layers to the WD when each downloaded model can be used for more inference requests. In particular, when $K = \infty$, all the layers of the DNN are downloaded, i.e., $M^* = N$. Moreover, given K, the optimal M^* under 1-sla stopping rule is not larger than that under the optimal backward induction rule due to the myopic property of the 1-sla stopping rule.

In Fig. 12, we demonstrate the impact of distance d on the optimal M^* in the AlexNet. Comparing with Fig. 11, we can draw two conclusions from Fig. 12. First, the BS tends to download smaller number of layers to the WD for the AlexNet. This is due to the larger computation workloads and higher model parameter sizes of the convolutional layers in the AlexNet compared with the autoencoder. This leads to higher costs for the local computing and model parameter downloading. Second, the optimal M^* computed based on the 1-sla rule is close to that computed based on the backward induction rule. It is because according to Fig. 7 and 8, the 1-sla stopping rule has higher optimality probability in the AlexNet compared with the autoencoder.

In Fig. 13 and Fig. 14, we plot the total average cost Z when the distance d varies and $K = 50$. Fig. 13 shows that the hybrid algorithm outperforms the 1-sla stopping rule based scheme and performs very closely to the optimal solution. Jointly considering the close-to-optimal performance and the complexity of the hybrid algorithm, it is a preferred choice for practical implementation. One interesting observation is that when the distance is small (e.g., below 30 meters for the autoencoder), the 1-sla rule based algorithm is also close to the optimum. This is because a smaller distance leads to a smaller optimal M^* as illustrated in Fig. 11 and 12. Under a smaller M^*, the 1-sla stopping rule has a higher optimality probability. Likewise, for the AlexNet in Fig. 14, the 1-sla stopping rule based scheme and the hybrid algorithm achieve close-to-optimal performances even when d is large.

D. Complexity of the Proposed Algorithms

At last, we compare the computational complexity among the proposed algorithms under different DNN structures, where the distance $d = 50$ meters and the model update frequency parameter $K = 50$. Specifically, we further consider the VGG 16 model, where the configuration follows [27]. As shown in Fig. 15, the 1-sla based and hybrid algorithms require shorter runtime than the backward induction based algorithm. In particular, for the VGG 16, around 35.82% and 30.46% lower average runtime is achieved by the 1-sla based and hybrid algorithms, respectively. Therefore, the hybrid algorithm can achieve the near-optimal performance at the price of a slight increase of the computational complexity

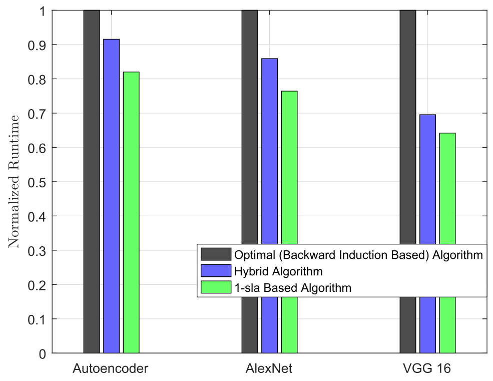

Fig. 15. Comparisons of normalized computation time under different algorithms.

compared with the 1-sla based algorithm. This demonstrates the effectiveness of the low-complexity algorithms.

VI. Conclusion

This paper has studied the joint optimization of model placement and online model splitting strategy to minimize the expected energy and time cost of device-edge co-inference. Given the model placement decision, we have formulated the problem of online model splitting as an optimal stopping problem with finite horizon. Then, we have proposed an online algorithm based on backward induction to find the optimal stopping rule (i.e., the optimal model splitting strategy). To simplify the analysis, we have proposed a 1-sla stopping rule for model splitting, based on which an efficient algorithm is proposed to optimize the model placement decision. Under a specific DNN structure, we have derived the closed-form expressions for the 1-sla stopping rule based model placement. We have further proposed a hybrid algorithm to balance between the solution optimality and the computational complexity. Simulation results have validated the benefits of jointly considering the model placement and splitting under various DNN structures.

References

[1] 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," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[2] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3039–3071, 4th Quart., 2019.

[3] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 160–167.

[4] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.

[5] N. D. Lane *et al.*, "DeepX: A software accelerator for low-power deep learning inference on mobile devices," in *Proc. 15th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2016, pp. 1–12.

[6] S. Liu, Y. Lin, Z. Zhou, K. Nan, H. Liu, and J. Du, "On-demand deep model compression for mobile devices: A usage-driven model selection framework," in *Proc. 16th Annu. Int. Conf. Mobile Syst., Appl., Services*, 2018, pp. 389–400.

[7] S. Bi, L. Huang, and Y.-J.-A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4947–4963, Jul. 2020.

[8] J. Yan, S. Bi, L. Duan, and Y.-J.-A. Zhang, "Pricing-driven service caching and task offloading in mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4495–4512, Jul. 2021.

[9] Z. Lin, S. Bi, and Y.-J. Angela Zhang, "Optimizing AI service placement and resource allocation in mobile edge intelligence systems," 2020, *arXiv:2011.05708*.

[10] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garcia, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5419–5427.

[11] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 2019, pp. 1–16.

[12] 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.

[13] 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.

[14] 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.

[15] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 1, pp. 615–629, 2017.

[16] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained Internet-of-Things platforms," in *Proc. 15th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Nov. 2018, pp. 1–6.

[17] 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," 2019, *arXiv:1903.03472*.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[19] T. S. Ferguson. *Optimal Stopping and Applications*. Accessed: 2008. [Online]. Available: https://www.math.ucla.edu/~tom/Stopping/Contents.html

[20] Y. J. Zhang, "Multi-round contention in wireless LANs with multipacket reception," *IEEE Trans. Wireless Commun.*, vol. 9, no. 4, pp. 1503–1513, Apr. 2010.

[21] Q. Gu, Y. Jian, G. Wang, R. Fan, H. Jiang, and Z. Zhong, "Mobile edge computing via wireless power transfer over multiple fading blocks: An optimal stopping approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10348–10361, Sep. 2020.

[22] J. Jia, Q. Zhang, and X. Shen, "HC-MAC: A hardware-constrained cognitive MAC for efficient spectrum management," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 1, pp. 106–117, Jan. 2008.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[24] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 235–250, Jan. 2020.

[25] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. IEEE 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[26] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter, "The multilayer perceptron as an approximation to a Bayes optimal discriminant function," *IEEE Trans. Neural Netw.*, vol. 1, no. 4, pp. 296–298, Dec. 1990.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–14.

Jia Yan (Member, IEEE) received the B.Eng. degree from the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China, in 2017, and the Ph.D. degree in information engineering from The Chinese University of Hong Kong in 2021. He is currently a Postdoctoral Associate with the Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities, MN, USA. His research interests focus on optimization and machine learning techniques in wireless communications and networking, particularly in mobile edge computing and edge intelligence.

Suzhi Bi (Senior Member, IEEE) received the Ph.D. degree in information engineering from The Chinese University of Hong Kong in 2013. From 2013 to 2015, he was a Postdoctoral Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore. Since 2015, he has been with the College of Electronics and Information Engineering, Shenzhen University, China, where he is currently an Associate Professor. His research interests mainly involve in the optimizations in wireless information and power transfer, mobile computing, and wireless sensing. He has received the 2019 IEEE ComSoc Asia–Pacific Outstanding Young Researcher Award. He was a co-recipient of the 2021 IEEE ComSoc Asia–Pacific Outstanding Paper Award and the Conference Best Paper Awards of IEEE SmartGridComm 2013 and IEEE/CIC ICCC 2021. He is an Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE WIRELESS COMMUNICATIONS LETTERS.

Ying-Jun Angela Zhang (Fellow, IEEE) received the Ph.D. degree from the Department of Electrical and Electronic Engineering, The Hong Kong University of Science and Technology.

She joined the Department of Information Engineering, The Chinese University of Hong Kong, in 2005, where she is currently a Professor. Her research interests focus on optimization and learning in wireless communication systems.

Prof. Zhang is a Member-at-Large of the IEEE ComSoc Board of Governors. She is a co-recipient of the 2021 and 2014 IEEE ComSoc Asia–Pacific Outstanding Paper Awards, the 2013 IEEE SmartGridComm Best Paper Award, and the 2011 IEEE Marconi Prize Paper Award on Wireless Communications. As the only winner from engineering science, she has won the Hong Kong Young Scientist Award 2006, conferred by the Hong Kong Institute of Science. She has served on the organizing committees for many top conferences, such as IEEE GLOBECOM, ICC, VTC, and SmartGridComm. She was the Founding Chair of the IEEE ComSoc Technical Committee of Smart Grid Communications. She is an Associate Editor-in-Chief of IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY and a member of the Steering Committees of IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE WIRELESS COMMUNICATIONS LETTERS, and IEEE SmartgridComm Conference. Previously, she has served as the Chair for the Executive Editor Committee of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and many years for the editorial boards of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON COMMUNICATIONS, *Security and Communication Networks* journal (Wiley), IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS special issues, IEEE INTERNET OF THINGS JOURNAL special issues, and *IEEE Communications Magazine* special issues.