Cooperative Convolutional Neural Network Deployment over Mobile Networks

Chia-Chun Hsu[§], Chung-Kai Yang[‡], Jian-Jhih Kuo[†], Wen-Tsuen Chen[‡], and Jang-Ping Sheu[‡]
[§]Dept. of Information Systems and Applications, National Tsing Hua University, Hsinchu, Taiwan
[‡]Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[†]Dept. of Computer Science & Information Engineering, National Chung Cheng University, Chiayi, Taiwan
E-mail: {s107065528,s107062546}@m107.nthu.edu.tw, lajacky@cs.ccu.edu.tw, {wtchen,sheujp}@cs.nthu.edu.tw

***Abstract*—Inference acceleration has drawn much attention to cope with the real-time requirement of artificial intelligence (AI) applications. To this end, model partition for Deep Neural Networks (DNN) has been proposed to utilize the parallel and distributed computing units. However, the previous works focus on the load balancing among servers but may overlook the interplay between the computing and communication. This issue makes the existing approaches less efficient especially in mobile edge networks at which smart devices usually with limited computing capacity have to offload the tasks via limited bandwidth capacity to nearby servers. In this paper, therefore, we innovate a new system and formulate a new optimization problem, CONVENE, to minimize the completion time of inference for the smart devices with one or more antennas. To explore the intrinsic properties, we first study CONVENE with Single Antenna and derive an algorithm termed THREAD-SA to foster the optimum solution. Then, an extension, THREAD, is proposed to subtly utilize multiple antennas to further reduce completion time. Simulation results manifest that our algorithm outperforms others by 100%.**

***Index Terms*—mobile edge networks, convolutional neural network, model partition, model deployment**

I. Introduction

Recently, artificial intelligence (AI) has drawn more attention and made much progress in many applications [1]. The emerging applications benefit from AI models to improve their system accuracy [2]. AI models typically run on either the smart devices or the cloud. However, smart devices usually rely on a simple model (e.g., support vector machine (SVM)) due to their limited capacity, which may incur poor accuracy. By contrast, the more effective models (e.g., convolutional neural network (CNN)), need more plentiful resources to reduce the computing time, and the cloud is the main platform for scientific research or marketing analysis [3]. The paradigm, nonetheless, may be inadequate for the issues as follows in the near future. The first is the considerable number of smart devices connecting the cloud (e.g., 12 billion in 2020 predicted by [4]). Particularly, the computationally-intensive services, like video surveillance and augmented/virtual reality (AR/VR), require real-time image recognition, which may overwhelm the backhaul network and servers if videos are only processed in the cloud. In addition, the progression toward much *deeper* neural network structures[i] may make a single, smart device no longer meet the user requirement. It also prolongs the response time since every input has to visit the overall network.

[i]The model depth expands rapidly in recent years, such as AlexNet with 8 layers, VGGNet with 19 layers, and ResNet with 152 layers in 2012, 2014, and 2015, respectively.

To remedy these issues, we employ the following concepts: 1) Fog/Mobile Edge Computing (FC/MEC). The concept enables the smart device to cooperate with its nearby servers to form a *mobile edge network* [5]–[7] to execute local distributed computing to reduce response time and relieve backhaul overhead. 2) Model partition for CNN and parallel computing. The idea is to partition a model into several *blocks*, and each server computes one of them based on its computing capacity [8]–[10]. Traditionally, each block processes a *disjoint horizontal slice* of an input image, and any two neighboring blocks have to exchange their data to obtain the input for each subsequent layer (see black areas in Fig. 2), which greatly slows down the inference (see Fig. 3). To this end, our system innovates *data-lookahead partition* (see Fig. 4), which allows the server of each block to prefetch and compute the data within other blocks from the smart devices such that every block can be independently executed on different servers in parallel without waiting for other blocks (detail in Section III). 3) Parallel data transfer. A smart device with more antennas can transmit data to more servers in parallel at a time to speed up inference [11].

Nevertheless, not all available servers are suitable to cooperate with others since selecting less powerful servers increases the computing overhead of prefetched data while slightly raises overall computing capacity. Besides, the biased association of servers to the antennas may cause imbalanced data transfer among different antennas, which may idle the servers due to a long data transfer time and cause poor parallel data transfer. In addition, an arbitrary partition for different servers via the same antenna may prolong the *completion time* (i.e., the time to make an inference, including data transfer and computing) due to their different computing capacity. The increasing completion time may deteriorate user experience of time-sensitive applications (e.g., AR/VR requires less than 20 ms latency to avoid motion sickness [12]). However, the *time-aware partition for CNN models* that jointly decides *server selection*, *antenna association*, *model partition*, and *data transfer scheduling* has not been explored to partition a model into several blocks and deploy them with a proper number and place of servers over the mobile edge network to achieve the low completion time.

Therefore, in this paper, we make the first attempt to explore time-aware partition for CNN models. It raises three new research challenges as follows. 1) *Suitable server number*. The more involved servers may enjoy smaller computing time for a single smaller block but have to process more prefetched data, which may induce longer completion time. 2) *Proper*

978-1-7281-5089-5/20/$31.00 ©2020 IEEE

antenna association. Each server has to associate with an antenna to retrieve the data for computing. To avoid overlength waiting of servers for data transfer, it is essential to evenly associate the selected servers to the antennas. 3) *Elastic model partition*. Intuitively, the block size for each server should be proportional to the capacity of servers to reduce completion time. However, such a partition overlooks the influence of *data transfer scheduling*. The servers with an earlier time to start computing tend to process a larger block. Thus, whether, when, and where to compute the partition should also be jointly determined, which leads to a challenge.

To address these challenges, we formulate an optimization problem, **Co**operative C**N**N Deployment o**v**er Mobi**l**e **N**etwork **E**dges (CONVENE), to find the time-aware partition for CNN over the heterogeneous network edge. With the given parameters: 1) CNN model with data size and computing overhead of each layer,[ii] 2) capacity of each server, 3) number of antennas equipped on the smart device, and 4) bandwidth capacity[iii] of all antennas, CONVENE aims to minimize the completion time. To explore the intrinsic properties of CONVENE, we first investigate CONVENE-SA (i.e., CONVENE with **S**ingle **A**ntenna) and introduce an insight notion, *cumulative server law*, to derive the overall computing capacity of servers associated to the same antenna. We then propose an algorithm named THREAD-SA based on *cumulative server law* to obtain the optimal solution, which carefully selects servers to address *suitable server number* and *elastic model partition*. Finally, we extend the notion to design THREAD to achieve *proper antenna association* by subtly balancing the overheads over multiple antennas to further reduce the completion time.

II. Related Work

A. Fog Computing and Edge Computing

Recently, FC and MEC has emerged to meet the requirement (e.g., big data analysis and real-time response) of applications (e.g., AR/VR) in smart devices [5]–[7]. Xu *et al.* dynamically offload the popular services to the resource-limited edge to relieve the backhaul network overhead [13]. Wang *et al.* deploy the VR service proxies on the edge to process data of smart devices and avoid tremendous amount of data exchange among users [14]. Ran *et al.* design an offloading mechanism from smart devices to MEC servers to trade off response time and accuracy [15]. However, none of them emphasizes the model partition for inference acceleration.

B. Inference and Training Acceleration

Many approaches aim at inference acceleration and can be divided into two categories, *structure simplification* and *distributed computing*. For the former, Howard *et al.* and Zhang *et al.* lighten the structures, which can be executed on the resource-limited smart devices [16], [17]. However, the lightweight structures usually have a lower accuracy. For the latter, Teerapittayanon *et al.* and Kang *et al.* partially offload deep learning models to other servers to decrease the inference time [18], [19]. Mao *et al.* and Zhao *et al.* partition the model into *disjoint* blocks and deploy them on a set of servers to reduce the inference time, whereas the intermediate data has to exchange among the servers [8]–[10]. Nevertheless, none of them focuses on the latency caused by data exchange during the inference, which may prolong the completion time.

Distributed computing is often addressed for training acceleration. The distributed training schemes usually partition the model structure or divide the dataset [20]. Teerapittayanon *et al.* partition the model into subset of layers and then distribute them over the devices, edge, and cloud [18]. Wang *et al.* trade off the local and global updates across the smart devices to improve the training effectiveness [21]. Wang *et al.* parallel the parameter computing across different servers to speed up training [22]. Nishio *et al.* design a distributed learning protocol to maximize the total number of selected servers within a time interval [23]. Tran *et al.* decompose decentralized learning optimization problems into convex-structure subproblems, each of which can be efficiently solved efficiently [24]. However, none of them focuses on inference acceleration.

III. System Model

The system considers a mobile edge network, which consists of heterogeneous servers around the resource-limited smart device. To enable the services such as real-time image recognition and AR/VR gaming experience for the owners of smart devices, the smart device (e.g., camera) *continuously* sends the data (e.g., video) to the nearby adequate servers for processing. To accelerate the inference of CNN model with the considerable amount of data, the system partitions the CNN model into multiple *blocks*, each of which is deployed on a selected server to processes a *horizontal* slice of an input image in parallel. Note that the server of a block requires the intermediate results of layer i within other blocks for computing layer $i+1$ (e.g., *black areas* are required to compute block 2 in Fig. 2).

In the past, each block processes a *disjoint horizontal slice* of an input image and the block size is proportional to the computing capacity of corresponding servers. Thus, any two servers that process the neighboring blocks have to exchange their intermediate results during the inference (see Fig. 3) [8]. However, it causes massive data transfer and may seriously violate the completion time requirement (e.g., 20 ms latency for real-time image recognition [12]). To this end, we introduce an intelligent system DeepCo with a novel notion, *data-lookahead partition*, which allows the server of each block to prefetch and compute the demanded data within other blocks such that every server with a different block can independently run in parallel without waiting for other servers (see Fig. 4).

Fig. 1 presents the overview of DeepCo, which consists of a smart device and multiple nearby available servers. The CNN model is pre-trained and stored in the smart device. The coordinator of smart device has to select a set of adequate servers from the available servers and partitions the CNN model into suitable-size blocks for each selected server. To minimize the completion time, the partition has to jointly consider the computing capacity of servers, the bandwidth capacity, and the number of antennas. Thus, the images generated by the

[ii]This paper mainly focuses on inference acceleration for the convolutional layers, and uses the traditional methods to handle the fully-connected layers.

[iii]To explore the intrinsic property of CONVENE, it is reasonable to assume that the bandwidth capacity from the smart device to each server via different antennas is similar since they are usually close to each other.

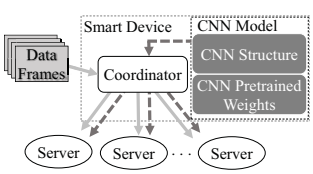


Fig. 1. Overview of DeepCo

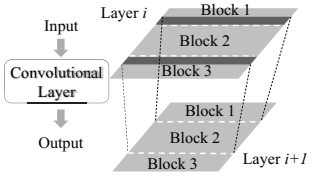


Fig. 2. Data-dependent propagation

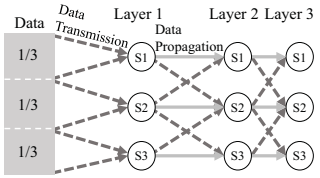


Fig. 3. W/o data-lookahead partition

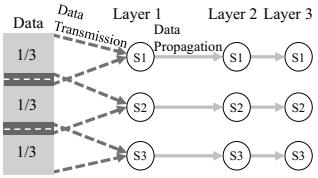


Fig. 4. Data-lookahead partition

sensors on the smart device are partitioned along with the CNN model, and then transferred to the selected servers for distributed computing. Finally, the selected servers complete the computing and then return the results to the smart device.

Specifically, DeepCo allows the selected servers to prefetch the data in the other blocks (e.g., black areas for the server processing block 2 in Fig. 2). The prefetched data amount is proportional to the number of convolutional layers (e.g., 152 layers) and the filter size (e.g., 3×3, 5×5) in the CNN model. The height of prefetched data (i.e., black areas in Fig. 2) is $o(M) = 2\sum_{l\in M}\lfloor\frac{f(l)}{2}\rfloor$, where M denotes the CNN model with $|M|$ layers and $f(l)$ denotes the filter height size of layer $l \in M$. Therefore, we have the following definition.

Definition 1. The *prefetched data ratio* of the model M with the input images of height h is $r_o = \frac{o(M)}{h}$.

For instance, if the model M has inputs with 224×224 pixels and five convolutional layers, where the layers are with two 3×3 and three 5×5 filters, respectively, then the prefetched data ratio for a block is $r_o = \frac{2(2\lfloor3/2\rfloor+3\lfloor5/2\rfloor)}{224} \approx 7.142\%$.

IV. Problem Formulation

We formulate the problem as an optimization problem to 1) select a set of adequate servers, 2) associate each server to the proper antennas, and 3) partition the CNN model with the input data into suitable-size blocks, so as to minimize the completion time. Let V denote the set of available servers around the smart device. Let binary variable $x_i \in \{0,1\}$ denote whether server i be selected to compute for the smart device. Due to heterogeneous computing capacity, each selected server may obtain a block with a different size. Let fractional variable $r_i \in [0,1]$ denote the *partition ratio* of overall data amount generated by the smart device. Since some data can be processed by server i only if server i is selected, we obtain the constraint.

$$x_i \geq r_i, \qquad \forall i \in V \tag{1}$$

Moreover, all the blocks must include the overall data, i.e.,

$$\sum_{i\in V} r_i = 1 \tag{2}$$

Let A denote the set of antennas and binary variable $y_{im} \in \{0,1\}$ denote whether server $i \in V$ is associated to antenna $m \in A$. Since every selected server must be associated to a specific antenna, we have the constraint.

$$\sum_{m\in A} y_{im} = x_i, \qquad \forall i \in V \tag{3}$$

Let binary variable $z_{ij} \in \{0,1\}$ denote whether server $i \in V$ is scheduled after server $j \in V \setminus \{i\}$. Two servers must start computing its partition at different time if they are associated to the same antenna since an antenna can only transfer the data to the servers sequentially, and thus

$$z_{ij} + z_{ji} = \sum_{m\in A} y_{im} \cdot y_{jm}, \qquad \forall i,j \in V, i \neq j. \tag{4}$$

Moreover, if server i is scheduled before server j while server j is scheduled before k, then server i is scheduled before k.

$$z_{ij} \cdot z_{jk} \leq z_{ik}, \qquad \forall i,j,k \in V, i \neq j, j \neq k, i \neq k \tag{5}$$

Assume that the data size of an image is s, while the computing capacity[iv] of server i and bandwidth capacity of an antenna are $c(i)$ and b, respectively. Let fractional variable $h_i \geq 0$ be the size of data received by server i, which includes the block assigned to sever i and the prefetched data. Thus, for each server $i \in V$, $h_i = s \cdot (r_i + r_o) \cdot x_i$, where r_o is the *prefetched data ratio* (see Definition 1). Let fractional variables $d_i^p \geq 0$ and $d_i^q \geq 0$ denote the time length for the data transfer and computing of server i. Then, $d_i^p = \frac{h_i}{b}$ and $d_i^q = \frac{h_i}{c(i)}$.

Let fractional variable $d_i \geq 0$ denote the time at which server i starts to receive its data. Therefore, d_i must be after the data transfer of all the servers scheduled before server i, and thus

$$d_i = \sum_{j\in V: i\neq j} z_{ij} \cdot d_j^p, \qquad \forall i \in V \tag{6}$$

We then know that the time points at which server i starts and stops computing are $d_i + d_i^p$ and $d_i + d_i^p + d_i^q$. Let fractional variable T denote the completion time of all servers, and thus

$$T \geq d_i + d_i^p + d_i^q, \qquad \forall i \in V \tag{7}$$

Therefore, we formally define CONVENE as follows.

Definition 2. Given a CNN model M with data size s, a set of antennas A with bandwidth capacity b, and a set of server V with computing capacity $c(i)$ for each server $i \in V$, the **Co**operative **C**NN Deployment o**v**er Mobil**e** **N**etwork **E**dges problem (CONVENE) finds a set of selected servers with *server selection*, *antenna association*, *model partition*, and *data transfer scheduling* such that the above constraints (1)−(7) are satisfied and the completion time is minimized, i.e.,

$$\text{minimize} \quad T \tag{8}$$

In the following, we introduce the novel notions for algorithm design and provide the proof sketch of the hardness.

[iv]The computing overhead per unit of data for an image may not be in the same range of data size. For ease of presentation, we *normalize* the computing overhead to align the range of data size in advance. After normalization, the computing capacity of each server i becomes $c(i) = \hat{c}(i) \cdot \frac{s}{w}$, where w denotes the computing overhead per unit of data for an image and $\hat{c}(i)$ denotes the original computing capacity of server i.

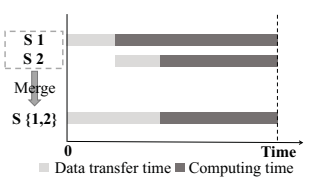


Fig. 5. Cumulative Server Law

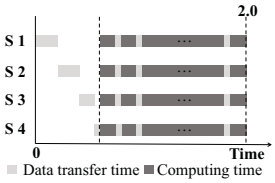


Fig. 6. MoDNN

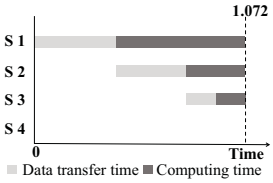


Fig. 7. THREAD-SA

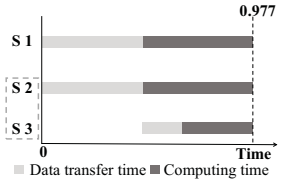


Fig. 8. THREAD

A. Cumulative Server Law

In this subsection, we derive an insightful notion, *cumulative server law*, with a key formula to calculate the overall computing capacity for multiple selected servers associated to the same antenna. It can be envisaged that such servers merge together into a more powerful *virtual server* (see Fig. 5). However, the computing capacity, though, is enhanced as the number of servers increases, the number of prefetched data areas required to transfer also increases. Later in Sectioin V, we will detail how to carefully trade off the two factors.

Theorem 1 (Cumulative Server Law)**.** The overall computing capacity of selected servers denoted by S associated to the same antenna can be regarded as a merged server with the overall computing capacity

$$c(S) = \begin{cases} 0 & \text{if } S = \emptyset \\ \dfrac{\prod_{i\in S}(b + c(i)) - b^{|S|}}{b^{|S|-1}} & \text{otherwise.} \end{cases} \tag{9}$$

Proof. See Appendix A. □

Then, we know that the data overhead of the overall data is $s\sum_{i\in S}(r_i + r_o)$. Therefore, by Theorem 1, we can obtain the completion time $T(S)$ of the selected servers associated to the same antenna, i.e.,

$$T(S) = \Big(\sum_{i\in S}(r_i + r_o)\Big)(\frac{s}{b} + \frac{s}{c(S)}) \tag{10}$$

In particular, as the selected servers only with one antenna have to complete the inference, i.e., $\sum_{i\in S} r_i = 1$, eq. (10) can be further simplified to

$$T(S) = (1 + |S| \cdot r_o)(\frac{s}{b} + \frac{s}{c(S)}) \tag{11}$$

Example 1. This example shows the effect of *server selection*, *antenna association*, *model partition*, and *data transfer scheduling*. Tables I and II summarize the model and network information. First, the traditional method, MoDNN [8], assigns the partition ratios to all the servers based on their computing capacity, so the ratios for them are $\frac{10}{10+10+7+3} \approx 0.333$, 0.333, 0.234, and 0.1, respectively. The data transfer and computing require time $\frac{10}{60} \approx 0.167$ and $\frac{10}{10+10+7+3} \approx 0.333$, respectively. Besides, the data exchange between servers takes time $0.05\times 30 = 1.5$ since the model has 30 convolutional layers and each time of exchange needs time 0.05. Thereby, the overall completion time of MoDNN is roughly $0.167+0.333+1.5 = 2$ (see Fig. 6). By contrast, the optimum solution with a single antenna selects servers 1, 2, and 3 and assigns their partition ratios roughly as 0.519, 0.387, and 0.094, respectively, shown in Table III. The overall computing capacity of servers 1, 2, and 3 is $\frac{(60+10)\times(60+10)\times(60+7)-(60)^3}{(60)^2} \approx 31.194$, and it requires time $(1 + 3 \times 0.4)(\frac{10}{60} + \frac{10}{31.194}) \approx 1.072$ (see Fig. 7 and eq. (11)). The completion time can be reduced by 46%. In addition, the optimum solution further associates servers $\{1\}$, and $\{2, 3\}$ to the two antennas, respectively. It allocates the partition ratios 0.4375, 0.4375, and 0.125 for servers 1, 2, and 3 and takes time 0.977. It can reduce the time by 51% shown in Table IV.

B. Proof Sketch of Hardness

Due to the page limit, we give the proof sketch of hardness. CONVENE is NP-hard since the Product Partition (PP) problem (see Definition 3) can be reduced to the decision version of CONVENE[v] in polynomial time by creating two antennas in A with $b = 1$, n servers in V with computing capacities $c(1) = e_1 - 1, ..., c(n) = e_n - 1$, and model M with $r_o = 0$ (i.e., 1×1 filter) and $s = w = 1$. First, $r_o = 0$ makes the optimum solution must select all servers. Besides, we set the time threshold $\mathcal{T} = \frac{s}{2} \cdot (\frac{1}{b} + \frac{1}{\mathcal{C}}) = \frac{\mathcal{C}+1}{2\mathcal{C}}$, where $\mathcal{C} = \sqrt{\prod_{i\in V}(c(i) + b)} - 1 = \sqrt{\prod_{i\in V} e_i} - 1$, since the more balanced association on the overall capacity computing of the servers for the two antennas indicates the less completion time. Thus, the answer of PP is yes iff the answer of decision version of CONVENE is yes, and then CONVENE is NP-hard.

Definition 3. [25] Given a set of positive integers $U = \{e_1, ..., e_n\}$, the Product Partition Problem asks whether a set $U_1 \subset U$ exists such that $\prod_{e\in U_1} e = \prod_{e\in U\setminus U_1} e$.

V. Algorithm Design

Two traditional methods can be applied to solve CONVENE, which 1) deploy the whole CNN model to one server and 2) partition the model into multiple *disjoint* blocks with the computing overhead according to the computing capacity of servers and then distribute them to the near servers. However, the former may cause an enormous burden for a single server if the CNN contains numerous convolution layers (e.g., 152-layer ResNet) while the latter requires massive data exchange between two servers that compute the neighboring blocks. Besides, none of them optimizes antenna association. Hence, these methods may degrade real-time user experience.

To resolve CONVENE-SA, we first design an algorithm, **T**ime-Leng**th**-Awa**re** Coopera**t**or Selection and Mo**d**el Partition Algorithm with **S**ingle **A**ntenna (THREAD-SA), which carefully addresses *suitable server number* and *elastic model partition* to find the optimum solution. THREAD-SA is then extended to achieve *proper antenna association* (i.e., THREAD) to further reduce the inference completion time for CONVENE.

[v]Given a time threshold $\mathcal{T}$, the decision version of CONVENE asks whether a solution can complete the inference no later than $\mathcal{T}$.

TABLE I
MODEL INFORMATION

s	w	$\|M\|$	r_o
10	10	30	40%

TABLE II
NETWORK INFORMATION

i	1	2	3	4
$c(i)$	10	10	7	3
b	60	60	60	60

TABLE III
SELECTED SERVERS WITH DIFFERENT r_i

i	1	2	3
r_i	0.519	0.387	0.094

TABLE IV
ANTENNA ASSOCIATION

Antenna	1	2
2^{nd} iteration	{1}	{2}
3^{rd} iteration	{1}	{2,3}

TABLE V
SELECTED SERVERS WITH DIFFERENT r_i

Server	1	2	3	4
2^{nd} iteration	0.50	0.50	-	-
3^{rd} iteration	0.4375	0.4375	0.125	-

A. THREAD-SA

Based on Theorem 1, THREAD-SA maximizes the overall computing capacity of selected servers while ensuring that the total size of data included in the blocks and prefetched data areas do not overwhelm the selected servers. To this end, THREAD-SA iteratively selects the server with the maximum computing capacity to enhance the overall computing capacity, and calculates the overall computing capacity by merging the selected servers together as a *virtual server* (see Fig. 5). Meanwhile, to avoid the much data overwhelming the selected servers, it evaluates the overall computing overhead at each time of server selection as well as the completion time. Therefore, it repeats server selection until the overall completion time stops decreasing and then returns the last solution with the earliest completion time. Finally, it balances the partition ratios for all selected servers such that the antenna *continuously* transfers the data to the selected servers in sequence, each server receiving its required data then starts the computation, and all the severs complete their computing at the same time. Overall, THREAD-SA consists two phases: 1) Server Selection Phase (SSP) and 2) Transfer Scheduling Phase (TSP). SSP decides the set of servers to cooperate to make an inference while TSP determines the scheduling of data transfer and computing for each selected server.

1) Server Selection Phase (SSP): Let $\mathcal{S}_t$ denote the set of selected servers after t servers are selected and thus $\mathcal{S}_0 = \emptyset$ initially as $t = 0$. For each iteration $t \geq 1$, SSP selects the server i^t with the maximum computing capacity, i.e., $i^t = \arg\max_{i \in V \setminus \mathcal{S}_{t-1}} \{c(i)\}$, and then $\mathcal{S}_t = \mathcal{S}_{t-1} \cup \{i^t\}$. Then, it computes the overall computing capacity $c(\mathcal{S}_t)$ by eq. (9) and the new completion time $T(\mathcal{S}_t)$ by eq. (11). If $T(\mathcal{S}_{t-1}) > T(\mathcal{S}_t)$, SSP continues the next iteration; otherwise, it outputs $\mathcal{S}_{t-1}$ (i.e., the selected servers) as the input of the next phase.

Example 2. Following Example 1, this example demonstrates THREAD-SA. SSP first selects the server with the maximum capacity, i.e., server 1, with data transfer time $(10 \times 1)/60 \approx 0.167$ and computing time $(10 \times 1)/10 = 1$, and thus the completion time is $0.167 + 1 = 1.167$. Next, it adds server 2 with computing capacity 10. The overall computing capacity of two servers is $\frac{(60+10)\times(60+10)-(60)^2}{60} \approx 21.667$, and then the completion time becomes $(1 + 2 \times 0.4)(\frac{10}{60} + \frac{10}{21.667}) \approx 1.131$. Since $1.131 < 1.167$, it continues server selection. The operation stops at the fourth iteration since the completion time increases from 1.072 to 1.161. It then returns the third solution.

2) Transfer Scheduling Phase (TSP): Given the servers selected in SSP denoted by $\mathcal{S}$, let $\mathcal{D}_t$ represent the set of scheduled servers after t servers are scheduled and thus $\mathcal{D}_t = \emptyset$ as $t = 0$. Let τ^t denote the start time of data transfer for the t^{th} server in the scheduling and $T(\mathcal{S})$ denote the completion time of all servers, and hence $\tau^1 = 0$. For each iteration t, TSP randomly picks a server i^t in $\mathcal{S} \setminus \mathcal{D}_{t-1}$, and then $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{i^t\}$. All selected servers including i^t finish their computing at time $T(\mathcal{S})$ (see eq. (11)), i.e., $\tau^t + (r_{i^t} + r_o)(\frac{s}{b} + \frac{s}{c(i)}) = T(\mathcal{S})$. Thus, the partition ratio of server i^t is

$$r_{i^t} = \frac{T(\mathcal{S}) - \tau^t}{(\frac{s}{b} + \frac{s}{c(i)})} - r_o, \tag{12}$$

and the start time of data transfer for the next server is

$$\tau^{t+1} = \tau^t + (r_{i^t} + r_o) \cdot \frac{s}{b} \tag{13}$$

The scheduling is completed when $\mathcal{S} \setminus \mathcal{D}_t = \emptyset$. Note that we can prove that THREAD-SA obtains the optimum solution for CONVENE-SA as follows. Let $\mathcal{O}$ and $\mathcal{A}$ respectively denote the optimum and our solution. Any server in $S_1 = \mathcal{A} \setminus \mathcal{O}$ must be no weaker than any server in $S_2 = \mathcal{O} \setminus \mathcal{A}$ since A only selects powerful servers sequentially. Thus, we can exchange the servers in S_2 of $\mathcal{O}$ with those in S_1 to generate new $\mathcal{O}$ until 1) new $\mathcal{O}$ includes S_1 or 2) S_2 is exhausted, and the exchange does not increases the completion time. It is obvious that $|S_1| = |S_2|$; otherwise, the new $\mathcal{O}$ is not the optimum since selecting fewer servers than $\mathcal{A}$ or more servers in $V \setminus \mathcal{A}$ will increase the completion time. Thus, $\mathcal{A}$ must be the optimum.

Example 3. Following Example 2, this example derives the partition ratio for each server selected at iteration t. TSP first sets r_1 as $\frac{1.072-0}{(\frac{10}{60}+\frac{10}{10})} - 0.4 \approx 0.519$ and thus the next server starts to receive the data at time $\tau^1 = 0 + (0.519 + 0.4) \times \frac{10}{60} \approx 0.153$. Similarly, the remaining partition ratios are shown in Table III.

Time Complexity. The time complexity of THREAD-SA is $O(|V|)$. Due to page limit, the detailed analysis is omitted.

B. THREAD

THREAD extends THREAD-SA to optimize the utilization of multiple antennas to achieve *proper antenna association*. It aims to balance the computing overhead and computing capacity of selected servers associated to different antennas to well utilize the multiple antennas and reduce the completion time. To this end, similar to THREAD-SA, it iteratively selects the server with the maximum computing capacity, while associating the server to the antenna whose servers have the weakest overall computing capacity. Meanwhile, it carefully decides the partition ratio for the servers of each antenna such that all selected servers stop computing at the same time. Also, it examines the change of completion time. It repeats the operations until the overall completion time stops decreasing and returns the last solution with the earliest completion time.

Explicitly, let $\mathcal{G}_m^t$ be the set of servers associated to antenna $m \in A$ after t servers are selected and thus $\mathcal{G}_m^0 = \emptyset$ as $t = 0$. Similarly to SSP, for iteration $t \geq 1$, THREAD selects the server i^t with the maximum computing capacity, and associates server i^t to the antenna m^t whose servers have the weakest

overall computing capacity, i.e., $m^t = \arg\min_{m \in A}\{c(\mathcal{G}_m^{t-1})\}$. Then, $\mathcal{G}_{m^t}^t = \mathcal{G}_{m^t}^{t-1} \cup \{i^t\}$. The other antennas rather than m^t do not change the associated servers, i.e., $\mathcal{G}_m^t = \mathcal{G}_m^{t-1}$, where $m \in A \backslash \{m^t\}$. Then THREAD computes the overall computing capacity of associated servers for each antenna, i.e., $c(\mathcal{G}_m^{t-1} \cup \{m^t\})$, by eq. (9). Recall that all the completion time for each selected server is identical. Thereafter, it calculates the system of equations as follows to derive 1) the completion time $T(\mathcal{G}_m^t)$ and 2) the partition ratio g_m^t for each antenna $m \in A$.

$$\begin{cases} \sum_{m \in A} g_m^t = \sum_{m \in A} \sum_{i \in \mathcal{G}_m^t} r_i = 1 \\ T(\mathcal{G}_{m_1}^t) = T(\mathcal{G}_{m_2}^t), \qquad \forall m_1, m_2 \in A \end{cases} \tag{14}$$

The server selection with antenna association stops when $T(\mathcal{G}_m^t) \geq T(\mathcal{G}_m^{t-1})$, and then THREAD computes and returns the partition ratio r_i for every server i by TSP with eq. (10).

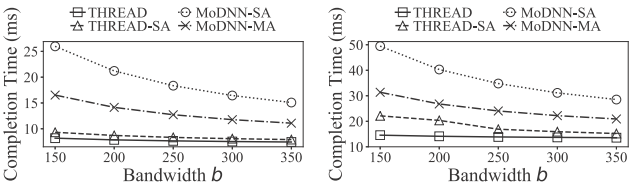
Example 4. Following Example 1, THREAD is depicted in the example. It selects two servers and associates them to different antennas in the first two iterations as shown in Table IV. Then, it solves the system of equations (see eq. (14)): 1) $\frac{10\mathcal{G}_1^1+4}{60} + \frac{10\mathcal{G}_1^1+4}{10} = \frac{10\mathcal{G}_2^1+4}{60} + \frac{10\mathcal{G}_2^1+4}{10}$, 2) $\mathcal{G}_1^1 + \mathcal{G}_2^1 = 1$, and obtains the partition ratios for the antennas, 0.5 and 0.5, shown in Table V. The overall completion time equals $T(\mathcal{G}_2^2) = \frac{10\times 0.5+4}{60} + \frac{10\times 0.5+4}{10} = 1.05$. At the 3^{rd} iteration, THREAD selects server 3 and associates it to any antenna since the two antennas have the identical overall computing capacity. Let's say antenna 2. Also, by eq. (14), THREAD assigns the partition ratios 0.4375 and $0.4375 + 0.125 = 0.5625$ for the two antennas. The overall computing capacity of antenna 2 is $c(\mathcal{G}_2^3) = \frac{(60+10)\times(60+7)-(60)^2}{60} \approx 18.167$, and thus the completion time becomes $T(\mathcal{G}_2^3) = \frac{10\times 0.5625+2\times 4}{60} + \frac{10\times 0.5625+2\times 4}{18.167} \approx 0.977$. The operation continues since $T(\mathcal{G}_2^3) = 0.977 < 1.05 = T(\mathcal{G}_2^2)$. Finally, THREAD returns r_i for each server i by TSP at 4^{th} iteration since $T(\mathcal{G}_2^4) = 1.075 > 0.977 = T(\mathcal{G}_2^3)$.

Time Complexity. The time complexity of THREAD is $O(|V||A|^3)$. Due to page limit, the detailed analysis is omitted.

VI. Evaluation

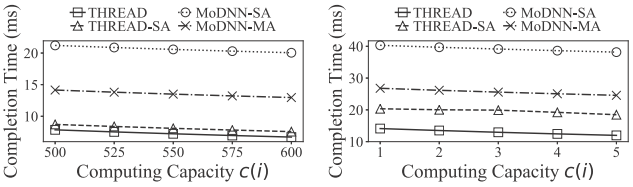
We conduct simulations to show the performance of our methods. In the simulation, the considered mobile edge network consists of a smart device and twenty servers with different computing capacity, where servers also group a network. We assume that the wireless transmission from the smart device to each server capacity is similar. We set $c(i) = 500 + 25\beta$ GFLOPS and $b = 150 + 50\delta$ Mbps, where β and δ are random in $[0, 5]$.[vi] A series of image recognition models, e.g., ResNet-101, ResNet-152 [2], are adopted in the simulation, and the *prefetched data ratio* is defined based on ResNet accordingly. Simulations with different CNN models show that THREAD is adaptable and efficient. Note that the download for the trained CNN model from the smart device to servers is ignored since THREAD only executes the download once. We use images with size $s \approx 150$ Kb and consider the 16-ms and 8-ms latency requirements for 60 FPS and 120 FPS throughput [12]. We compare THREAD-SA and THREAD with MoDNN [8] with single antenna and multiple antennas (i.e., MoDNN-SA and MoDNN-MA[vii]). The performance metric is completion time, and each result is averaged over 500 samples.

[vi]The settings are reasonable since 1) the current general GPU can process $2 \sim 4$ TFLOPS and 2) for upcoming 5G, the bandwidth capacity is expected to be $1 \sim 10$ Gbps. Therefore, the settings do not exhaust the resources of nearby servers to meet the application requirement.



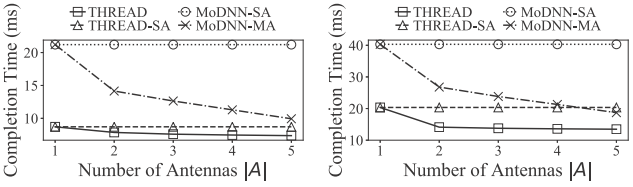
(a) ResNet-101 (b) ResNet-152

Fig. 9. Effect of available bandwidth capacity on completion time.



(a) ResNet-101 (b) ResNet-152

Fig. 10. Effect of available computing capacity on completion time.



(a) ResNet-101 (b) ResNet-152

Fig. 11. Effect of antenna number on completion time.

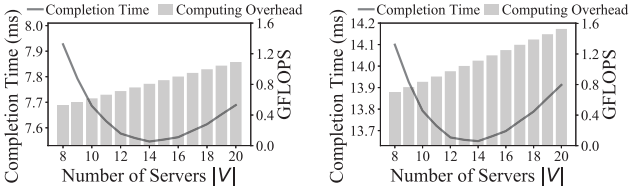
A. The Effectiveness of Bandwidth and Computing capacity

Figs. 9 and 10 show the overall completion time with different settings of bandwidth and computing capacity. Generally, as b, $c(i)$, and $|A|$ increase, the completion time decreases. Compared with MoDNN-SA, MoDNN-MA leverages multiple antennas to efficiently decrease the completion time. However, neither of them addresses data transfer scheduling for model partition. Besides, THREAD-SA and THREAD benefit from *data-lookahead partition* of DeepCo and *cumulative server law* to allow the computing and data transfer to be executed in parallel and precisely calculate the overall computing capacity of servers so as to balance their computing overhead. Overall, THREAD-SA and THREAD outperform other methods by 100% and meet the 16-ms and 8-ms latency requirement.

B. The Effectiveness of Number of Antennas and Servers

Fig. 11 shows the effectiveness of multiple antennas. Generally, the more antennas can reduce the more data transfer time. The completion time of MoDNN descends drastically as the

[vii]We associate the servers to different antennas for the more balanced computing overheads among the antennas for MoDNN.



(a) ResNet-101 (b) ResNet-152

Fig. 12. Effect of selected server number on completion time.

number of antennas increases since the waiting for the data transfer period of all the servers (see Fig. 6) can be relieved. Contrarily, THREAD hardly decreases since it allows data transfer and computing in parallel. We also observe that it does not tend to use all servers (see Fig. 12). It is because the more servers have higher overall computing capacity but require to transfer and compute more data due to data-lookahead partition, which may result in a longer completion time.

VII. Conclusions

In this paper, we design a new system with a novel notion, *data-lookahead partition*, and formulate a new optimization problem, CONVENE-SA, which aims to minimize the total completion time for smart device with heterogeneous resources over one or multiple antennas. To explore the intrinsic properties, we first study CONVENE to address *suitable server number* and *elastic model partition* to derive the optimum solution. Then, an extension, THREAD, is proposed to achieve *proper antenna association* over multiple antennas to properly balance the completion time over multiple antennas. Simulation results show that our algorithm outperforms others by 100%.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016.

[3] J. Wang *et al.*, "Deep learning towards mobile applications," in *IEEE ICDCS*, 2018.

[4] (2019) Cisco visual networking index: Global mobile data traffic forecast update. White paper.

[5] C. Marquez *et al.*, "How should I slice my network? A multi-service empirical evaluation of resource sharing efficiency," in *ACM MOBICOM*, 2018.

[6] P. Rost *et al.*, "Mobile network architecture evolution toward 5G," *IEEE Commun. Mag.*, vol. 54, pp. 84–91, 2016.

[7] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, pp. 810–819, 2017.

[8] J. Mao *et al.*, "MoDNN: Local distributed mobile computing system for deep neural network," in *DATE*, 2017.

[9] J. Mao *et al.*, "MeDNN: A distributed mobile system with enhanced partition and deployment for large-scale dnns," in *IEEE/ACM ICCAD*, 2017.

[10] 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, pp. 2348–2359, 2018.

[11] Y. Ban *et al.*, "4G/5G multiple antennas for future multi-mode smartphone applications," *IEEE Access*, vol. 4, pp. 2981–2988, 2016.

[12] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE/ACM ISMAR*, 2007.

[13] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *IEEE INFOCOM*, 2018.

[14] L. Wang *et al.*, "Service entity placement for social virtual reality applications in edge computing," in *IEEE INFOCOM*, 2018.

[15] X. Ran *et al.*, "DeepDecision: A mobile deep learning framework for edge video analytics," in *IEEE INFOCOM*, 2018.

[16] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *IEEE/CVF CVPR*, 2018.

[17] A. G. H. *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[18] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *IEEE ICDCS*, 2017, pp. 328–339.

[19] Y. K. *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *ACM ASPLOS*, 2017.

[20] J. M. *et al.*, "AdaLearner: An adaptive distributed mobile learning system for neural networks," in *IEEE/ACM ICCAD*, 2017.

[21] S. W. *et al.*, "When edge meets learning: Adaptive control for resource constrained distributed machine learning," in *IEEE INFOCOM*, 2018.

[22] S. Wang *et al.*, "Addressing skewness in iterative ML jobs with parameter partition," in *IEEE INFOCOM*, 2019.

[23] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE ICC*, 2019.

[24] N. H. Tran *et al.*, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM*, 2019.

[25] C. Ng, M. Barketau, T. Cheng, and M. Y. Kovalyov, "Product partition and related problems of scheduling and systems reliability: Computational complexity and approximation," *Eur. J. Oper. Res.*, vol. 207, pp. 601 – 604, 2010.

Appendix A
Proof of Theorem 1

Prove it by induction. Let $\hat{s}$ be the total size of data assigned to the servers S and $\hat{r}_i$ be the partition ratio of data size $\hat{s}$ assigned to server $i \in S$. The claim is trivial for $S = \emptyset$. For $|S| = 1$, the claim holds since $c(S) = \frac{b+c(1)-b^1}{b^0} = c(1)$. For $|S| = 2$, assume that T denote the completion time. By the claim, the overall computing capacity $c(S) = \frac{(b+c(1))(b+c(2))-b^2}{b}$, and then the completion time is $T = (1+2r_o)(\frac{\hat{s}}{b} + \frac{\hat{s}}{c(S)})$. On the other hand, for the first server, the completion time is $T = \hat{s}(\frac{\hat{r}_1+r_o}{b}) + \hat{s}(\frac{\hat{r}_1+r_o}{c(1)})$ while for the second server, the completion time is $T = \hat{s}(\frac{1+2r_o}{b}) + \hat{s}(\frac{\hat{r}_2+r_o}{c(2)})$. Thus, $\hat{r}_1 = \frac{T-r_o(\frac{\hat{s}}{b}+\frac{\hat{s}}{c(1)})}{\frac{\hat{s}}{b}+\frac{\hat{s}}{c(1)}} = \frac{\frac{T}{(\frac{1}{b}+\frac{1}{c(1)})} - \hat{s}\cdot r_o}{\hat{s}}$ and $\hat{r}_2 = \frac{c(2)(T-\frac{s+2\hat{s}\cdot r_o}{b})-\hat{s}\cdot r_o}{\hat{s}}$. Since $\hat{r}_1 + \hat{r}_2 = 1$, $T = \frac{(\hat{s}+2r_o\hat{s})(1+\frac{c(2)}{b})}{\frac{b\cdot c(1)}{b+c(1)}+c(2)} = \frac{(\hat{s}+2r_o\hat{s})(b+c(1))(b+c(2))}{b[(b+c(1))(b+c(2)))-b^2]} = (1+2r_o)(\frac{\hat{s}}{b} + \frac{\hat{s}}{c(S)})$. Hence, two servers seems merged as a single server, and the claim also holds. Assume that the claim holds as $|S| = n$. For $S \cup \{j\}$, by induction hypothesis, the overall computing capacity of the previous n servers can merge as a server k with computing capacity $c(S)$. We can obtain

$$
\begin{aligned}
c(S \cup \{j\}) = c(\{k\} \cup \{j\}) &= \frac{(b + c(S))(b + c(j)) - b^2}{b^1} \\
&= \frac{\left(b + \frac{\prod_{i \in c(S)}(b+c(i)) - b^{|S|}}{b^{|S|-1}}\right)(b + c(j)) - b^2}{b} \\
&= \frac{\left(\frac{\prod_{i \in c(S) \cup \{j\}}(b+c(i))}{b^{|S|-1}}\right) - b^2}{b} \\
&= \frac{\prod_{i \in c(S) \cup \{j\}}(b + c(i)) - b^{|S|+1}}{b^{|S|}}
\end{aligned}
$$

Therefore, the claim still holds and the theorem follows.