Dynamic Compression Ratio Selection for Edge Inference Systems With Hard Deadlines

Xiufeng Huang and Sheng Zhou, *Member, IEEE*

***Abstract*—Implementing machine learning algorithms on the Internet-of-Things (IoT) devices has become essential for emerging applications, such as autonomous driving and environment monitoring. But the limitations of computation capability and energy consumption make it difficult to run complex machine learning algorithms on IoT devices, especially when the latency deadline exists. One solution is to offload the computation-intensive tasks to the edge server. However, the wireless uploading of the raw data is time consuming and may lead to deadline violation. To reduce the communication cost, lossy data compression can be exploited for inference tasks but may bring more erroneous inference results. In this article, we propose a dynamic compression ratio selection scheme for edge inference system with hard deadlines. The key idea is to balance the tradeoff between the communication cost and inference accuracy. By dynamically selecting the optimal compression ratio with the remaining deadline budgets for queued tasks, more tasks can be timely completed with correct inference under limited communication resources. Furthermore, information augmentation that retransmits less compressed data of task with erroneous inference, is proposed to enhance the accuracy performance. While it is often hard to know the correctness of inference, we use uncertainty to estimate the confidence of the inference and based on that, jointly optimize the information augmentation and compression ratio selection. Finally, considering the wireless transmission errors, we further design a retransmission scheme to reduce performance degradation due to packet losses. The simulation results show the performance of the proposed schemes under different deadlines and task arrival rates.**

***Index Terms*—Data compression, edge computing, edge inference, machine learning, Markov decision process (MDP).**

I. Introduction

THE DEPLOYMENT of machine learning algorithms on Internet-of-Things (IoT) devices at the network edge [2], can enable many applications, such as autonomous driving, intelligent manufacturing, environment monitoring, etc. To address the limited processing capabilities of IoT devices, the cloud-based method offloads a massive amount of data generated by IoT devices to cloud servers to perform the calculation. However, offloading data from the network edge to the cloud server can bring serious network congestion and long overall latency. Edge computing is proposed to solve this problem [3], by exploiting edge servers, which are much closer to the user devices, to process tasks. Nevertheless, the computation capability of the edge and wireless communication resources are still limited. For a resource-constrained edge computing system, resource management is crucial. A lightweight online learning algorithm is proposed based on multiarmed bandit theory to make task offloading decisions among multiple candidate servers, in order to minimize the average task offloading delay [4]. On the other hand, some researchers focus on designing resource allocation schemes to minimize the energy consumption or latency by convex optimization [5], [6], Lyapunov optimization [7], reinforcement learning [8], [9], etc.

Conventional edge computing considers general computing tasks, largely overlooking the characteristics of specific computing tasks such as machine learning. More recently, emerging *edge learning* technology [10], [11] focuses on deploying machine learning algorithms at the wireless access network edge. Advanced machine learning algorithms, such as a deep neural network (DNN), are usually computing and storage-intensive tasks. Therefore, the deployment of machine learning algorithms at user devices is difficult due to their limited computation capabilities, storage size, and power. To enable DNN at the network edge, some lightweight models, such as MobileNet [12], ShuffleNet [13], are proposed to save computation and memory usages. In addition, model compression technologies, such as weight pruning [14] and data quantization [15], can also support low latency and energy-constrained edge inference.

Another promising way to solve the deployment problem is jointly utilizing the computation resources of edge servers and user devices. In co-inference schemes with device–server synergy [16], [17], user devices finish part of computing tasks and send the intermediate result to edge servers for the rest of computing. A co-inference scheme should allocate the computation tasks among edge servers and user devices, as well as carefully splitting the neural network to reduce the communication cost of sending the intermediate results. For example, Branchynet [18] focused on neural network splitting in order to provide fast inference while guaranteeing accuracy, and BottleNet++ [19] adds a pair of encoder–decoder on the split point to perform joint source–channel coding. However, these methods fail when the user devices do not have the model, for example when the model is continuously

Manuscript received February 19, 2020; revised April 27, 2020; accepted May 19, 2020. Date of publication May 25, 2020; date of current version September 15, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1800804; in part by the Nature Science Foundation of China under Grant 61871254, Grant 91638204, and Grant 61861136003; and in part by Hitachi Ltd. This article was presented in part at the 2020 IEEE Wireless Communications and Networking Conference. *(Corresponding author: Sheng Zhou.)*

The authors are with the Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: huangxf18@mails.tsinghua.edu.cn; sheng.zhou@tsinghua.edu.cn).

Digital Object Identifier 10.1109/JIOT.2020.2997128

trained on the edge server and broadcasting the model is too costly.

We, therefore, consider an edge learning system where user devices send data to the edge server for inference [1]. In this case, new challenges emerge: the unreliable wireless transmission of raw data, typically with large size, can easily violate the task latency deadlines. To optimize the edge inference performance, in terms of accuracy and task completion latency, we exploit a major characteristic of machine learning: the *information redundancy* in the input data [20]. Removing redundant information can save communication resources, while at the time, it may not significantly affect the inference accuracy. This inspires us to use lossy compression before transmission. An energy-efficient lossy compression algorithm is exploited for IoT devices to perform edge inference [21]. Xu *et al.* [22] investigated saving energy by joint data compression, computation offloading, and resource allocation. However, these works mainly aim at the energy-saving issue, rather than the inference performance (e.g., the inference accuracy, latency, etc.) of the machine learning model. If the compression ratio, defined as the ratio of the size of the raw data to the size of compressed data, is high, the data can be transmitted faster but the inference accuracy will degrade. To keep a certain accuracy, additional retransmissions may occur and inevitably bring excessive latency. If the compression ratio is low, the learning model can perform better at the cost of long communication latency. In short, the selection of the compression ratio should balance the tradeoff between transmission latency and inference accuracy.

Realizing that different tasks have different information redundancy in the raw data, intuitively one should select a higher compression ratio for the tasks with more redundancy. However, calculating the information redundancy before transmission and inference is impractical, and thus we propose an information augmentation scheme. The user devices can first transmit data with a high compression ratio, with only a few communication resources. If the inference result is wrong, the user devices can perform information augmentation by retransmitting the less compressed task. The challenge is that the edge inference system may not be able to determine whether the result is correct. Luckily, *uncertainty* from the machine learning output can be exploited to estimate the confidence of the inference result [23].

Finally, because the wireless channel is unreliable, the inference task can fail due to packet losses. Retransmission [24] can solve this problem but results in extra delay. Due to the extra delay, the task may break the latency deadline and even the transmissions of the following tasks are affected. To this end, the compression ratio selection should jointly consider the packet loss probability, the queue state of waiting tasks, and the inference accuracy with different compression ratios.

In this article, we consider an edge inference system with random arrival of tasks, where tasks are offloaded to the edge server for inference. To maximize the number of tasks with correct inference results subject to hard latency deadlines, we use lossy compression on the raw data and design a dynamic compression ratio selection scheme to balance the tradeoff between transmission latency and the inference accuracy. Furthermore, we propose information augmentation to save more communication resources, and retransmission scheme, trying to avoid the performance degradation due to packet losses. In particular, our contributions include the following.

1) Lossy compression of raw data before transmission is exploited to save communication resources in a latency guaranteed edge inference system. By modeling the relation between transmission delay and inference accuracy under lossy compression, we formulate the design of dynamic compression ratio selection as an online optimization problem with stochastic task arrivals, in order to balance the tradeoff between transmission delay and inference accuracy, under the hard latency deadlines.
2) To solve the proposed problem of dynamic compression ratio selection, we first design an offline algorithm using dynamic programming (DP) to obtain the performance upper bound. We then suppose that the arrival process of tasks is a Bernoulli process, and propose an online algorithm using the Markov decision process (MDP). Experiments show that online algorithms can perform almost the same as the offline one.
3) We further propose an information augmentation scheme to spend fewer communication resources on the tasks with more information redundancy. Edge devices can transmit data with a high compression ratio at first and retransmit it with a lower compression ratio if the inference result is wrong. However, it is usually difficult to judge whether the result is correct or not, therefore we exploit the concept of uncertainty in machine learning to estimate the confidence of the inference result. The challenge of applying MDP to jointly determine the compression ratio and additional transmissions is that the number of transmissions of each task is a random variable, and thus the number of state transitions related to one task is uncertain. Therefore, we design a new state transition policy of MDP, by making every optional compression ratio correspond to one state transition even if it is not selected, to ensure that every task has the same number of state transitions. Experiments show that information augmentation can bring performance improvement, especially when the arrival rate or the latency requirement is stringent.
4) We address transmission errors in the wireless channel by retransmission for packets that are lost, and the key is to determine whether retransmission is needed and the compression ratio used for retransmission. Experiments show that the performance degradation is less than 2% even when the probability of packet loss reaches 10%, compared with the case of no packet loss.

The remainder of this article is organized as follows. In Section II, we introduce the system model. In Section III, we introduce the proposed transmission schemes in detail, including the offline algorithm, online algorithm, information augmentation scheme, and retransmission scheme. The experimental results are provided in Section IV. This article is concluded in Section V.

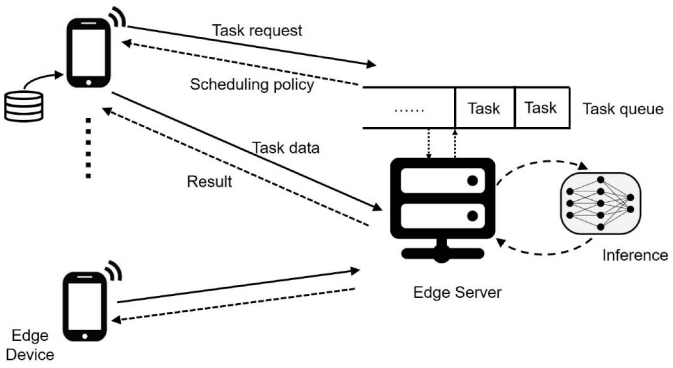


Fig. 1. Edge learning system under consideration.

II. System Model

Fig. 1 shows the edge learning system under consideration, which consists of an edge server (attached to a base station) and several edge devices. At the edge server, there is a well-trained learning model used for processing tasks.

The system is time slotted. There are random task arrivals at every edge device. When a task arrives at a device, the device sends a request to the edge server for performing the calculation, i.e., inference, and waits for being scheduled by the server, in order to transmit the task data. All tasks have the same maximum latency deadline τ, meaning that every task should be completed before τ time slots after arriving at the device, otherwise the task is considered failed. The time used for processing tasks at the server side, and sending requests as well as scheduling decisions is the same for different tasks. Therefore, it can be considered by subtracting the total latency constraint with the corresponding overhead, i.e., with larger time consumption of task processing, sending requests, and scheduling decisions, the system has smaller τ for communications. The task requests from all edge devices form a task queue on the edge server, as all requests are recorded. The arrival process of the task queue is assumed to be a Bernoulli process, meaning that there is at most one task arriving at the edge server in every time slot with probability p. As all tasks have the same priority, the edge server adopts the first-come–first-serve (FCFS) scheduling principle. With FCFS, the tasks from different devices have the same probability to be served, and thus the system can ensure fairness among tasks in the statistical sense. As a result, even if edge devices have different task arrival rates, the edge server will perform inference for the same proportion of tasks from each device.

Transmitting raw data for these tasks is time consuming. Therefore, lossy compression is used to reduce transmission latency. However, the compression of data can degrade the inference performance. To balance the tradeoff between the transmission time and the inference accuracy, the edge server should select the appropriate compression ratio for every task according to the state of the task queue. The inference accuracy can be regarded as the reward of performing inference. In the considered system, the edge server performs inference for the same class of tasks, whose data are collected by the same category of sensors or devices. Therefore, we assume that all tasks have the same raw data size and the reward is a function

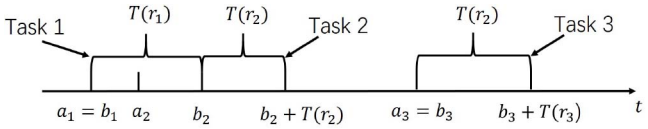


Fig. 2. Example of arriving and transmitting of tasks.

of compression ratio. In the scenarios that tasks have different raw data sizes, we can use the reward as a function of the data size being used, to reflect both the impact from the raw data size and the compression. Nevertheless, adapting the reward function does not change the nature of our proposed algorithm, and thus we stick to the assumption of uniform raw data size. Compression ratio $r \in [1, +\infty)$ is defined as the ratio of the size of the raw data to the size of compressed data, i.e., a larger compression ratio leads to less data size for transmission. The accuracy of the machine learning model with compression ratio r, i.e., the reward function, is $\rho(r) \in [0, 1]$, which can be obtained offline by performing inference on a validation data set and stored as a lookup table at the server side. The edge devices use a fixed rate for transmission and the time slots used for transmission using compression ratio r is $T(r) \in \mathbb{N}$. $\rho(r)$ and $T(r)$ are both decreasing functions. The wireless channel is assumed to be block fading, that is, the fading is independent identically distributed among different slots. In every time slot, the scheduled edge device transmits a packet of data samples, which will be lost with probability p_e. As a result, the transmission of the task using $T(r)$ time slots will fail with probability

$$P_e(T(r)) = 1 - (1 - p_e)^{T(r)} \tag{1}$$

which is also called the packet error ratio (PER).

The objective of the transmission scheme is to maximize the expectation of the number of successfully completed tasks with the deadline τ as shown in (2), considering M tasks, whose arrival time is $a_1 < a_2 < \cdots < a_M$. The ith task is scheduled to transmit data from time slot b_i with compression ratio r_i. Fig. 2 shows an example with three tasks.

The optimization problem is formulated as

$$\max_{r_i} \sum_{i=1}^{M} \rho(r_i)(1 - P_e(T(r_i))) \tag{2}$$

$$\text{s.t. } b_1 = a_1, \tag{3}$$

$$b_{i+1} = \max\{b_i + T(r_i), a_{i+1}\},\ i = 1, 2, \ldots, M-1 \tag{4}$$

$$b_i + T(r_i) \le a_i + \tau,\ i = 1, 2, \ldots, M \tag{5}$$

where (3) and (4) indicate that the $(i+1)$th task will be transmitted when the transmission of the ith task is completed [if the $(i+1)$th task has not arrived, then it will be transmitted when it arrives]. Equation (5) is the latency deadline constraint. Notice that $T(r_i)$ can be equal to 0 and then $\rho(r_i)$ will also be equal to 0 (it means that the task is failed since it cannot be delivered before the deadline), which ensures that (5) can be satisfied for all tasks.

Algorithm 1 Offline DP Algorithm

Input:

Number of tasks M;
Maximum waiting time of tasks τ;
Arriving time of tasks a_m;

Output:

Maximum number of successfully completed task $F(m, t)$;
Optimal policy $r(m)$;

set $F(0, t) = 0$ $(1 \le t \le a_M + \tau)$
for $m = 1$ to M **do**
  for $t = a_m + 1$ to $a_m + \tau$ **do**

$$F(m,t) = \max_{a_m \le i \le t-1} \Big(F(m-1,i) + \rho\Big(T^{-1}(t-i)\Big)\Big(1 - P_e(t-i)\Big)\Big)$$

$$G(m,t) = \arg\max_{a_m \le i \le t-1} \Big(F(m-1,i) + \rho\Big(T^{-1}(t-i)\Big)\Big(1 - P_e(t-i)\Big)\Big)$$

  end for
end for
set $t = a_M + \tau$
set $m = M$
while $m > 0$ **do**
  $r(m) = T^{-1}(G(m,t))$
  $t = \min(t - G(m,t), a_{m-1} + \tau)$
  $m = m - 1$
end while

III. Proposed Transmission Schemes

A. Offline Algorithm

An offline algorithm can solve the optimization problem described in Section II if the arrival time of all tasks is known beforehand. The proposed offline algorithm is designed using DP. We use $F(m, t)$, $(1 \le m \le M, 1 \le t \le a_M + \tau)$ to denote the maximum number of successfully *completed* tasks when processing the first m tasks by time t. To get $F(m, t)$, the number of time slots used for transmitting the mth task is denoted by $G(m, t)$. Then, we can get the optimal offline policy, i.e., the selected compression ratio for every task $r(m)$, after calculating $F(m, t)$ and $G(m, t)$ $(1 \le m \le M, 1 \le t \le a_M + \tau)$. The algorithm is shown as Algorithm 1.

B. Online Algorithm

In practice, the arrival times of tasks cannot be known beforehand. Therefore, an online algorithm is needed to select the compression ratios for the tasks in the queue without the knowledge of future arrivals. Assume that the arrival process is known, which is a Bernoulli process with probability p. Then, we can use MDP to solve this problem, by introducing the state, action, reward, and state transition probability of MDP as follows. With MDP, we can map the task queue to the MDP state and use the optimal action based on the state to select the optimal compression ratio for the Head-of-Line (HoL) task.

State: To make the decision on the compression ratio r for the HoL task in the current time slot, the information needed is the arrival time of all tasks waiting in the queue, which can be represented by $s = \{a_1, a_2, \ldots, a_N\}$, where N is the number of tasks in the waiting queue and $a_1 < a_2 < \cdots < a_N$. Due to the deadline constraint, N is at most τ. Note that the difference between the deadline of tasks and current time is sufficient for making decisions, the state is transformed to $s = \{a_1 + \tau - t, a_2 + \tau - t, \ldots, a_N + \tau - t\}$, where τ is the latency constraint of tasks and t is the current time. Because $0 < a_1 + \tau - t < a_2 + \tau - t < \cdots < a_N + \tau - t \le \tau$, the state s can be encoded to a binary number of τ digits

$$s = \sum_{i=1}^{N} 2^{a_i + \tau - t - 1}. \tag{6}$$

In the binary expression of s, the number of 1s is equal to the number of tasks in the queue and the positions of 1s represent the remaining time of tasks (lower digit represents fewer remaining time slots).

Action: The action of the MDP is the optional compression ratio of the HoL task in the queue. The action space is defined as $\mathcal{R}$, which is the set of optional compression ratios of the edge inference system. Furthermore, in the case of using the reward as a function of data size, the system should include the raw data sizes of tasks into the state of MDP, and use the data size, which should be smaller than the raw data size, as the action of MDP, rather than using the compression ratio.

Reward: The reward of taking action $r \in \mathcal{R}$ (selecting compression ratio r) is

$$W(r) = \rho(r)[1 - P_e(T(r))] \tag{7}$$

which is the product of the expected accuracy of the machine learning model with a compression ratio r and the probability of successful transmission.

State Transition Probability: The state transition probability depends on the action and the arrival process of tasks. With action r, state s will transit to state $s' = 2^{\tau - T(r)} i + \lfloor s/2^{T(r)} \rfloor$, $(i = 0, 1, \ldots, 2^{T(r)} - 1)$ with probability

$$\mathbf{P}_{ss'}(r) = p^{B(i)}(1-p)^{T(r) - B(i)} \tag{8}$$

where $B(i)$ is the number of 1s in the binary expression of i.

Generally, one needs to calculate the state transition probability matrix $\mathbf{P}$ and perform value iteration, whose space complexity and time complexity are both $\mathcal{O}(S \times S \times |\mathcal{R}|)$ where $S = 2^{\tau} - 1$ is the number of states. The storage of the state transition probability matrix is unacceptable if τ is large and calculating the state transition probability when performing value iteration is not effective. However, using the similarity of transition probability of different states and the fact that there are many zero elements in the state transition probability matrix, we do not need to calculate and store the state transition probability matrix explicitly, then we can greatly reduce the complexity of the algorithm as follows. First, the value

iteration equation of MDP is

$$V^{k+1}(s) = \min_{r\in\mathcal{R}(s)}\left[W(r) + \sum_{i=1}^{2^{\tau}-1}\mathbf{P}_{si}(r)V^{k}(i)\right] \tag{9}$$

where $V^k(i)$ is the value of state i of the kth iteration and $R(s)$ is the set of optional action of state s. To reduce the complexity, we use the following equation for value iteration:

$$V^{k+1}(s) = \min_{r\in\mathcal{R}(s)}\left[W(r) + \sum_{i=0}^{2^{T(r)}-1}\mathbf{P}'(r,i)V^{k}\left(2^{\tau-T(r)}i + \left\lfloor\frac{s}{2^{T(r)}}\right\rfloor\right)\right]. \tag{10}$$

For (10), the state transition probability is $\mathbf{P}'(r,i) = p^{B(i)}(1-p)^{T(r)-B(i)}$. For efficient value iteration, $\mathbf{P}'$ is calculated and stored before performing value iteration. Therefore, we only need to store $\mathbf{P}'$, with $S\times|\mathcal{R}|$ elements, instead of $\mathbf{P}$, and (10) only sums up $2^{T(r)}-1$ terms when considering action r, instead of $2^{\tau}-1$ terms in (9). With the optimization in (10), the space complexity is now $\mathcal{O}(S\times|\mathcal{R}|)$ and the time complexity is $\mathcal{O}(S\times S\times|\mathcal{R}|\times(1/\tau))$.

It is shown that the complexity of the proposed algorithm highly depends on τ. In Section II, we assume that in every time slot, there is at most one task arriving at the edge server. When the number of devices in the system and the task arrival rate at the edge server become larger, we need to use a shorter time slot to satisfy the assumption, which will result in a larger τ (in terms of the number of time slots). In this case, the edge inference system may fail to make scheduling decisions due to the large complexity. However, to have a reasonable performance of an edge learning system with finite computation capability, the number of active devices should be limited, i.e., the task arrival probability p cannot be too large. In many IoT systems, even if there are many devices, their active probability is still low. In short, as long as the computation load is reasonable, the algorithm can scale with corresponding p and τ.

C. Information Augmentation Scheme

For machine learning algorithms, some tasks can get correct inference result with a very high compression ratio, because of many information redundancies in the raw data. If we spend fewer communication resources on these tasks, leaving more communication resources for tasks that require a lower compression ratio, we can complete more tasks with limited communication resources. Therefore, we propose an information augmentation scheme to find the proper compression ratio for the task and the workflow is shown in Fig. 3.

We first assume that there is a method to judge whether the result is correct or not and the packet loss probability $p_e = 0$. As shown in Fig. 3, with the proposed information augmentation scheme, the edge server can first ask the edge devices to transmit data with a high compression ratio (not necessarily the highest one, and the compression ratio for the first attempt is also subject to our optimization). If the result is wrong, the edge server can decide to ask the device to transmit the task data with a lower compression ratio, according to the state of the task queue. In short, the information augmentation scheme needs to exploit the state of the task queue to decide the compression ratio of the first transmission attempt and later on augmentation transmissions if needed.

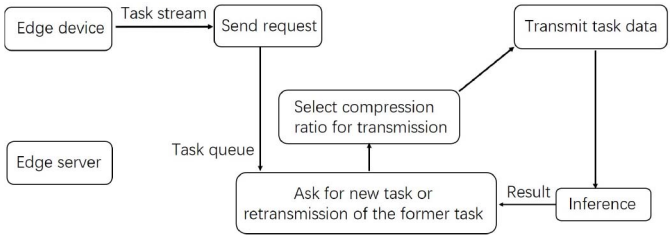


Fig. 3. Workflow of the information augmentation scheme.

We also use MDP to solve the decision-making problem in the proposed information augmentation scheme. In the last section, we use state $s = \{a_1, a_2, \ldots, a_N\}$ for MDP for the transmission scheme without information augmentation and the action space R is the set of optional compression ratios. Notice that one state transition corresponds to one transmission and the objective of MDP is to maximize the average reward per state transition. However, for the information augmentation scheme, the times of transmission of different tasks may be different. If one state transition of MDP still corresponds to one transmission over the wireless channel, different tasks will have different weights of the reward, and thus we cannot directly apply the MDP formulation from the last section.

To ensure that different tasks have the same times of state transitions, we consider a new state space and new state transition formulation. For every task, the edge server will virtually *consider* all optional compression ratios from high to low for its transmission (if the task can be completed with a high compression ratio, the transmission with lower compression is not needed). Every state transition corresponds to one time of *virtual consideration*. There are only two actions for MDP, transmission (or retransmission) with the current compression ratio or no transmission. If the action is no transmission, the edge server will consider the next compression ratio. If the action is to transmit, then the task will be transmitted with the current compression ratio under consideration. With the inference result and the state of the task queue, the edge server will continue to consider the next compression ratio for possible information augmentation, even if they are not selected. In this way, every task will have $|R|$ times of state transitions. The state, action, reward, and state transition probability of the MDP of the proposed information augmentation scheme is shown as follows.

State: The key here is that we put the compression ratio under consideration into the state, and now the state is $s = \{a_1, a_2, \ldots, a_N, r_L, r, f\}$, where r_L is the compression ratio for the last transmission for the current task (if the task has not been transmitted, $r_L = +\infty$), r is the compression ratio considered for the current transmission, and $f \in \{0, 1\}$ is the

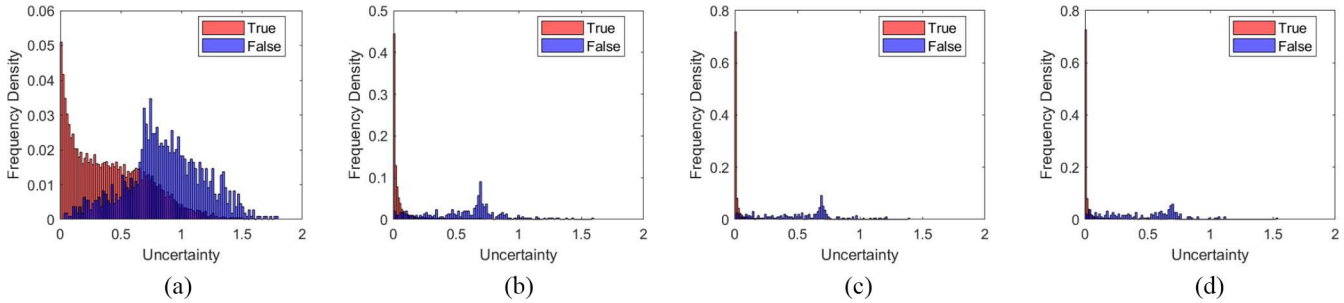


Fig. 4. Histogram of uncertainty. (a) 4×4. (b) 7×7. (c) 14×14. (d) 21×21.

correctness of the result of the last transmission (1 represents correct and 0 represents wrong). For simplicity, as in the last section, the state can be rewritten as

$$s = \left\{ \sum_{i=1}^{N} 2^{a_i+\tau-t-1}, r_L, r, f \right\} = \{a, r_L, r, f\} \tag{11}$$

where $a = \sum_{i=1}^{N} 2^{a_i+\tau-t-1}$ represents the state of the task queue.

Action: Because the compression ratio under consideration is formulated into states, there are only two actions for MDP, transmit or not (denoted by r_I, $r_I = 1$ represents transmission and $r_I = 0$ represents no transmission). If the result of the last transmission is correct ($f = 1$) or the rest latency budget is not enough for additional transmission, there is only one choice of no transmission.

Reward: For the state transition from $s = \{a, r_L, r, f\}$ to $s' = \{a', r'_L, r', f'\}$ with action r_I, the reward is

$$W_I(s, s', r_I) = r_I f'. \tag{12}$$

State Transition Probability: The state transition probability depends on the arrival process of tasks and the accuracy of inference conditioned on the inference correctness of the last transmission. If the action is no transmission, then the state $s = \{a, r_L, r, f\}$ will transit to state $s' = \{a, r_L, r', f\}$ with probability 1, where r' is the next considered compression ratio after r. If the action is to transmit, then $s = \{a, r_L, r, f\}$ will transit to state $s' = \{\lfloor a/2^{T(r)} \rfloor + i \times 2^{\tau - T(r)}, r, r', f\}$ ($i \in [0, 2^{T(r)} - 1] \cap \mathbb{N}$) and the state transition probability is

$$\mathbf{P}(s, s') = p^{B(i)}(1-p)^{T(r)-B(i)} \times \mathbf{P}_a(r, f | r_L). \tag{13}$$

$\mathbf{P}_a(r, 0|r_L)$ is the probability that the inference result is wrong with compression ratio r conditioned on that the last transmission was with compression ratio r_L and the inference result was wrong (otherwise, there is no need to transmit again). $\mathbf{P}_a(r, 1|r_L)$ is the probability that the inference result is correct with compression ratio r conditioned on that the last transmission was with compression ratio r_L. $\mathbf{P}_a$ is obtained by performing inference on the validation data set with optional compression ratios.

The number of states is $S_R = 2(2^\tau - 1)|R|^2$ and the size of action space is 2. The memory usage is the simplified state transition matrix mentioned in the last section and $\mathbf{P}_a$. As a result, the space complexity is $\mathcal{O}(S_R)$ and the time complexity is $\mathcal{O}(S_R \times S \times (1/\tau))$.

D. Information Augmentation With Uncertainty

In the last section, we assume that there is a method to judge whether the inference result is correct or not. However, this assumption is invalid in many scenarios. To solve this problem, we further introduce uncertainty to estimate the confidence of the inference result. The uncertainty $\mathcal{U}$ of the output of the learning model is defined in the following equation:

$$\mathcal{U} = -\sum_{i=1}^{n} X_i \log X_i \tag{14}$$

where $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ is the normalized output of the learning model ($\sum_{i=1}^{n} X_i = 1$) and n is number of elements of the model output. The lower uncertainty $\mathcal{U}$ indicates a higher probability of correct inference [23]. For example, the relation between correctness and uncertainty of data samples in MNIST [25], a handwritten digit images data set, is shown in Fig. 4. It shows the histogram of the uncertainty of correct results and wrong results. Different subfigures correspond to different resolutions. It is shown that the true results have lower uncertainty and the wrong results have higher uncertainty. Therefore, it is reasonable to exploit uncertainty for estimating the confidence of the inference results.

The problem of selecting the compression ratio for tasks in the information augmentation scheme with uncertainty is also solved via MDP.

State: Because the correctness of the inference result is unknown, the uncertainty of the inference result should be added into the state for decision making, and thus the state of the MDP is changed to $s = \{a, r_L, r, \mathcal{U}\}$, where $\mathcal{U}$ is the uncertainty of the result of the last transmission of the HoL task. For the practical implementation of MDP, uncertainty $\mathcal{U}$ should be quantized. We use uniform quantization for uncertainty.

Action: Still, there are two actions for the MDP, transmit or not transmit, denoted by r_U.

Reward: If the action is no transmission, the reward is 0. If the action is to transmit, the reward of state transition from state $s = \{a, r_L, r, \mathcal{U}\}$ to state $s' = \{a', r, r', \mathcal{U}'\}$ is

$$W_U(s, s') = \rho(r | r_L, \mathcal{U}, \mathcal{U}') - \rho(r_L | \mathcal{U}). \tag{15}$$

Here, $\rho(r|r_L, \mathcal{U}, \mathcal{U}')$ is the accuracy of the learning model with compression ratio r, conditioned on the compression ratio of the last transmission r_L, the uncertainty of the result of the last transmission $\mathcal{U}$, and the uncertainty of the newest result $\mathcal{U}$'. $\rho(r_L|\mathcal{U})$ is the accuracy of the learning model with compression ratio r_L, conditioned on the uncertainty of the result

$\mathcal{U}$. $\rho(r|r_L, \mathcal{U}, \mathcal{U}')$ and $\rho(r_L|\mathcal{U})$ can be obtained by performing inference on the validation data set with optional compression ratios.

State Transition Probability: The state transition probability depends on the arrival process of tasks and the probability distribution of uncertainty $\mathcal{U}$ of learning model. If the action is no transmission, the state $s = \{a, r_L, r, \mathcal{U}\}$ will transit to state $s' = \{a, r_L, r', \mathcal{U}\}$ with probability 1, where r' is the next considered compression ratio after r. If the action is to transmit, state $s = \{a, r_L, r, \mathcal{U}\}$ will transit to state $s' = \{\lfloor a/2^{T(r)} \rfloor + i \times 2^{\tau - T(u)}, r, r', \mathcal{U}'\}$ $(i \in [0, 2^{T(r)} - 1] \cap \mathbb{N})$ and the state transition probability is

$$\mathbf{P}(s, s') = p^{B(i)}(1-p)^{T(r)-B(i)} \times \mathbf{P}_U(r, \mathcal{U}'|r_L, \mathcal{U}) \quad (16)$$

where $\mathbf{P}_U(r, \mathcal{U}'|r_L, \mathcal{U})$ is the probability that the model output has uncertainty $\mathcal{U}'$ with compression ratio r conditioned on that the model output of the last transmission with compression ratio r_L has uncertainty $\mathcal{U}$.

The number of states is $S_{UR} = (2^\tau - 1)U|R|^2$ and the size of action space is 2, where U is the number of the quantization level of $\mathcal{U}$. The memory usage is the simplified state transition matrix mentioned in the last section and $\mathbf{P}_U$. As a result, the space complexity is $\mathcal{O}(S_{UR})$ and the time complexity is $\mathcal{O}(S_{UR} \times S \times (1/\tau))$.

E. Packet Loss-Aware Retransmission Scheme

The transmission of data samples may fail due to unreliable wireless channels. If the transmission fails, the edge device can retransmit the data before the deadline. Then, we need to design a scheme to decide possible retransmissions of the data samples according to the state of the task queue. When the transmission fails, the edge server can keep this task in the task queue and update the queue state. The edge server can still use the online algorithm designed in Section III-B to select a compression ratio for the transmission of this task according to the new queue state. This original retransmission scheme is regarded as the baseline of the retransmission scheme.

However, the above algorithm does not consider the packet error and retransmissions. Therefore, we introduce PER and retransmissions into the state transition of MDP to design an evolutionary retransmission scheme. Here, the design of MDP meets the same problem as the information augmentation, where the transmission of different tasks should have the same times of state transition in MDP. In the edge learning system we consider, one time of transmission of data samples needs at least one time slot, so the edge device can transmit the data sample of one task at most τ times in this system due to the hard deadline τ. Therefore, we use the same method as information augmentation to design MDP to ensure that different tasks have the same number of state transitions. For every task, the edge server virtually considers τ times of transmission, including whether transmitting and the compression for transmission.

State: The state of MDP is $s = \{a, r_L, \tau_s\}$, where a is the queue state, r_L is the compression ratio of the last successful transmission (no packet error), and τ_s is the number of considered transmissions.

Action: The action of MDP is the compression ratio for transmission or no transmission for the HoL task.

Reward: For state $s = \{a, r_L, \tau_s\}$, if the action is to transmit with compression ratio r and the transmission succeeds, the reward is

$$W_R(s, r) = \rho(r) - \rho(r_L) \quad (17)$$

otherwise, the reward is 0.

State Transition Probability: The state transition probability depends on the arrival process of tasks, the accuracy of learning model, and the PER. With action r, the state $s = \{a, r_L, \tau_s\}$ will transit to state $s' = \{\lfloor a/2^{T(r)} \rfloor + i \times 2^{\tau - T(r)}, r, \tau_s + 1\}$ if transmission succeeds, or state $s'' = \{\lfloor a/2^{T(r)} \rfloor + i \times 2^{\tau - T(r)}, r_L, \tau_s + 1\}$ if transmission fails. The state transition probability is

$$\mathbf{P}(s, s') = p^{B(i)}(1-p)^{T(r)-B(i)} \times (1 - P_e(T(r))) \quad (18)$$

and

$$\mathbf{P}(s, s'') = p^{B(i)}(1-p)^{T(r)-B(i)} \times P_e(T(r)). \quad (19)$$

For this proposed algorithm, the space complexity is $\mathcal{O}(S \times |R| \times \tau)$ and the time complexity is $\mathcal{O}(S \times S \times |R|)$.

IV. Experiment

We use two data sets, namely, MNIST and cifar10 [26] to evaluate our proposed algorithms. The experiments are based on simulations and the performance of the proposed scheme is evaluated by the proportion of successfully completed tasks.

A. MNIST Data Set

1) Experiment Setup: MNIST is the handwritten digit images data set and the task of the edge learning system is the number recognition of an image of the MNIST testing data set. In MNIST, there are 60 000 images in the training data set and 10 000 images in the testing data set. Each image in MNIST has 28×28 pixels and corresponds to an integer number from 0 to 9. The training data set is used for training a machine learning model deployed on the edge server. A task is successfully completed only when it is completed before the deadline *and* its inference result is correct.

The machine learning model deployed on the edge server is a multilayer perceptron (MLP) [27] with one hidden layer and 700 hidden units. The output of the model is vector $\boldsymbol{X}$ with ten elements and $\sum_{i=1}^{10} X_i = 1$. The ith element of $\boldsymbol{X}$ represents the probability that the input image belongs to class i (corresponding to the number $i - 1$).

The compression algorithm of images in the experiment is downsampling. The optional resolution of images for transmission include 4×4, 7×7, and 14×14, for which the compression ratio r is 49, 16, and 4. The time of one time slot is normalized to be the time used for transmitting an image with a resolution 4×4. Hence, the number of time slots used for transmitting data with these optional resolutions are 1, 3, and 10. We use the training data set to train the model used for performing inference task and obtain the accuracy of the model with respect to different compression ratios, i.e., $\rho(r)$, being 0.89, 0.97, and 0.98, for $r = 49$, 16, and 4, respectively. The data used for simulations are from the test data set.

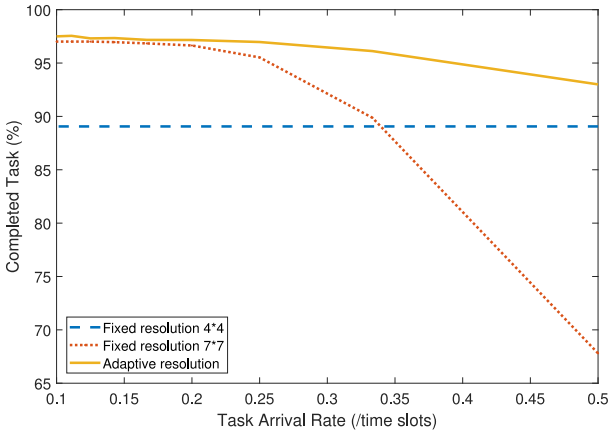


Fig. 5. Performance of the proposed offline algorithm with dynamic compression ratio selection under different arrival rates (MNIST data set).

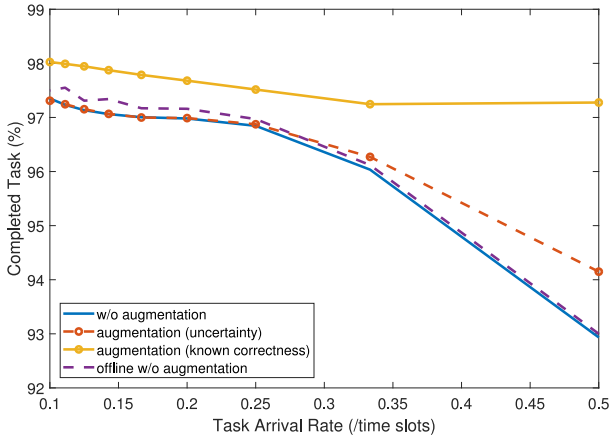


Fig. 6. Performance of the proposed online algorithms with dynamic compression ratio selection under different arrival rates (MNIST data set).

2) Experimental Result: First, we provide performance versus different task arrival rates. We first suppose that there is no packet loss, which means $p_e = 0$. The latency requirement is $\tau = 12$ slots. The number of quantization levels of $\mathcal{U}$ of the information augmentation scheme is 10. To show the gain of dynamic compression ratio selection, we compare the offline algorithm with the algorithm with a fixed resolution. Fig. 5 shows that the dynamic compression ratio selection can substantially increase the number of completed tasks. The performance of the offline algorithm is the upper bound of the online algorithm, which can be used to evaluate the performance of the proposed online algorithm. The performance of the online algorithm and information augmentation using MDP is shown in Fig. 6. It is shown that the online algorithm has almost the same performance as the offline algorithm. The information augmentation brings improvement, especially when the arrival rate is high. Specifically, when the task arrival rate is 0.5 per time slot, 97.5% of all tasks can be completed, with 4.5% improvement compared with the algorithm without information augmentation. The proposed information augmentation scheme is also more robust to the changes in the arrival rate. The performance of the information augmentation scheme with known inference correctness can be regarded as the upper bound of the information augmentation scheme. Using uncertainty for confidence estimation, the information augmentation scheme can still perform better than the online algorithm without information augmentation. Note that when the arrival rate is low, the improvement of the information augmentation is marginal. It is because tasks can be transmitted with a low compression ratio for the first transmission when the arrival rate is low and the information augmentation is unnecessary for many tasks.

Then, we provide performance with different latency requirements. The arrival rate is set to $p = 0.11$. $p_e = 0$. Fig. 7 shows the gain of the offline algorithm with dynamic compression ratio selection, compared with the algorithm with the fixed resolution. Fig. 8 compares the performance of the offline algorithm, online algorithm, and information augmentation scheme. The gaps between the offline algorithm and online algorithm become small when τ becomes larger. The

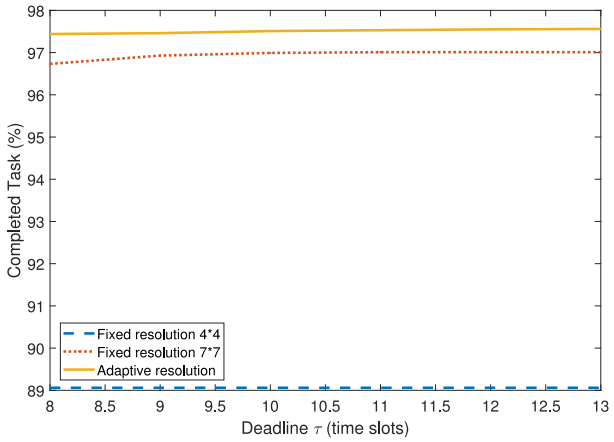


Fig. 7. Performance of the proposed offline algorithm with dynamic compression ratio selection under different deadlines (MNIST data set).

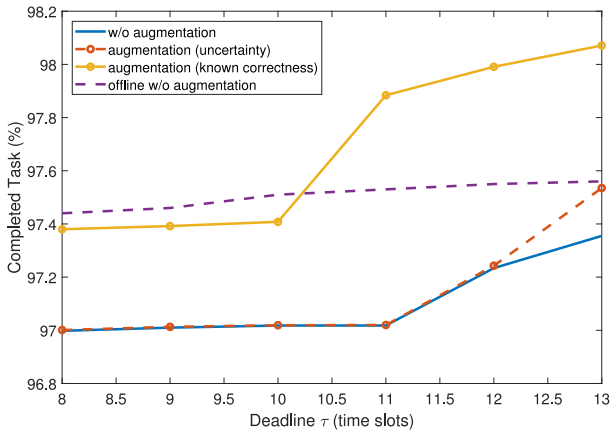


Fig. 8. Performance of the proposed online algorithms with dynamic compression ratio selection under different deadlines (MNIST data set).

information augmentation scheme brings more improvement with a larger τ. It is because that the information augmentation scheme can reduce the average communication cost but results in extra latency of some tasks, leading to possible task failures. With larger τ, fewer tasks fail and the advantage of the

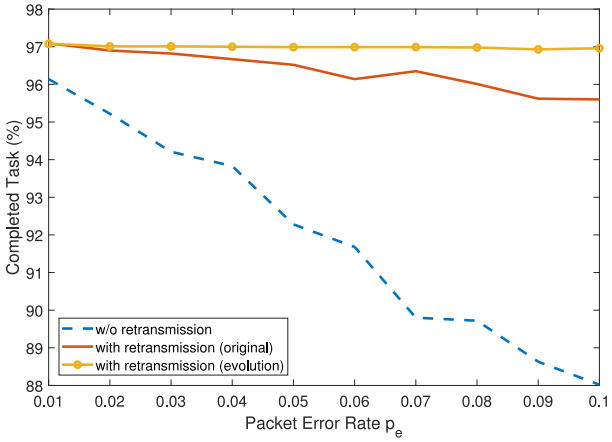


Fig. 9. Performance of the proposed online retransmission schemes without information augmentation (MNIST data set).

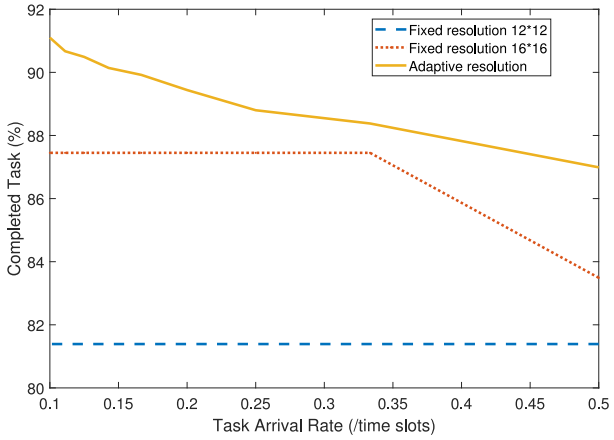


Fig. 10. Performance of the proposed offline algorithm with dynamic compression ratio selection under different arrival rates (cifar10 data set).

information augmentation scheme, which reduces the average communication cost, becomes more significant.

Considering the packet loss in the edge learning system, we compare the performance of the retransmission scheme under different PER. Fig. 9 shows the performance of three schemes (without information augmentation), including scheme without retransmission, the baseline of retransmission scheme (original MDP without considering the packet loss), and the retransmission scheme using MDP that considers packet loss and retransmission. The performance of the scheme without retransmission shows that the packet loss brings severe performance degradation. The retransmission scheme using MDP that considers retransmission when training is much more robust to different PER and improves the performance for over 1% as compared to the baseline when $p_e = 0.1$.

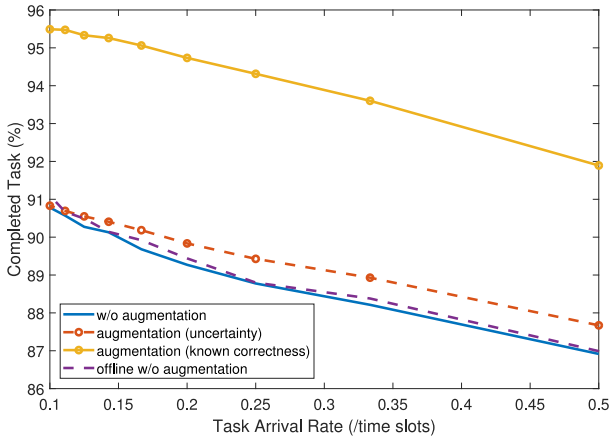


Fig. 11. Performance of the proposed online algorithms with dynamic compression ratio selection under different arrival rates (cifar10 data set).

B. Cifar10 Data Set

1) Experimental Setup: Cifar10 is an image data set with ten classes (cat, dog, etc.) and the task of the edge learning system is image recognition. In cifar10, there are 50 000 images in the training data set and 10 000 images in the testing data set. Each image in cifar10 has 32×32 pixels.

The machine learning model deployed on the edge server is mobilenet-v2 with 3.4M parameters [28]. The output of the model is vector $\boldsymbol{X}$ with ten elements and $\sum_{i=1}^{10} X_i = 1$. The ith element of $\boldsymbol{X}$ represents the probability that the input image belongs to class i.

The compression algorithm of images in the experiment is downsampling. The optional resolutions of images for transmission include 12×12, 16×16, and 32×32, for which the compression ratio r is 7, 4, and 1. The time of one time slot is normalized to be the time used for transmitting an image with resolution 12×12. Hence, the number of time slots used for transmitting data with these optional resolutions are 1, 2, and 8. We use the training data set to train the model and obtain the accuracy of the model by performing inference on the training data set, i.e., $\rho(r) = 0.81$, 0.87, and 0.92, for compression ratio $r = 7$, 4, and 1, respectively. The data used for simulations are from the test data set.

2) Experimental Results: First, we provide the performance versus different task arrival rates and assume that there is no packet loss ($p_e = 0$). The latency requirement is $\tau = 12$ slots. The number of quantization levels of $\mathcal{U}$ of the information augmentation scheme is also 10. The performance of the offline algorithm of dynamic compression ratio selection is compared with the algorithm that transmits data with fixed resolutions in Fig. 10. For the cifar10 data set, the edge learning system can still substantially increase the number of completed tasks with dynamic compression ratio selection. Then, we compare the performance of the offline algorithm, online algorithm, and information augmentation in Fig. 11. It is shown that the online algorithm has almost the same performance as the offline algorithm (the gap is less than 0.2%). The information augmentation (with known correctness) brings about 5% improvement compared with the online algorithm without information augmentation. Using uncertainty for confidence estimation, the improvement resulted from information augmentation becomes less but still brings 1% improvement when the arrival rate is 0.5. The gain of information augmentation becomes marginal when the arrival rate is lower, which is the same as the result of experiments on MNIST.

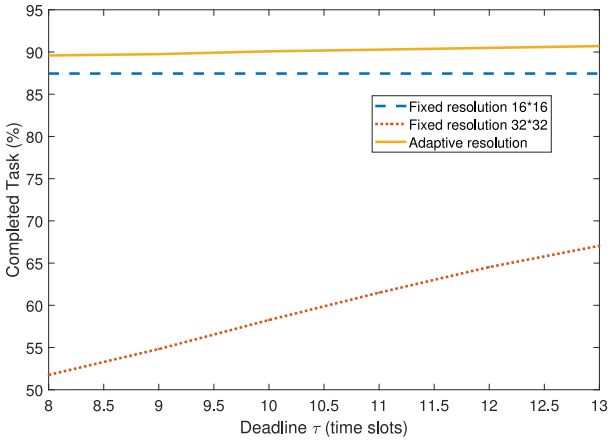


Fig. 12. Performance of the proposed offline algorithm with dynamic compression ratio selection under different deadlines (cifar10 data set).

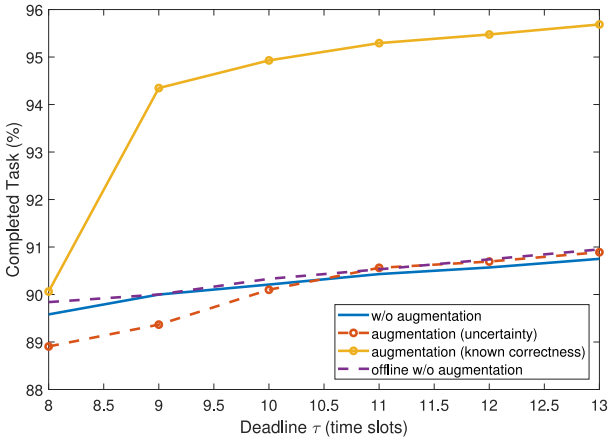


Fig. 13. Performance of the proposed online algorithms with dynamic compression ratio selection under different deadlines (cifar10 data set).

Then, we provide performance with different latency requirements. The arrival rate is $p = 0.11$ and $p_e = 0$. Fig. 12 shows the gain of the offline algorithm dynamic compression ratio selection, compared with the fixed resolution counterpart. Fig. 13 compares the performance of the offline algorithm, online algorithm, and information augmentation scheme. The performance of the online algorithm is almost the same as the offline algorithm. The information augmentation scheme brings more improvement with a larger τ, which is the same as the result shown by the experiments on MNIST. When τ is small, the information augmentation scheme using uncertainty for confidence estimation performs worse than the online algorithm without information augmentation. It is because the result of correctness estimation using uncertainty may be wrong, which brings performance degradation. When τ is small, the performance degradation of uncertainty-based confidence estimation is more significant than the improvement of information augmentation, which makes the information augmentation scheme worse than the online scheme without information augmentation.

Finally, we do experiments to compare the performance of the retransmission scheme with different PERs. Fig. 14 shows the performance of three schemes: 1) scheme without retransmission; 2) the baseline of the retransmission scheme (original MDP); and 3) the retransmission scheme using MDP that considers packet loss and retransmissions. Both retransmission schemes bring notable improvement compared with the scheme without retransmission. The retransmission scheme using MDP that considers retransmission for training is much more robust to the change of PER and performs better than the baseline with about 4% improvement when $p_e = 0.1$.

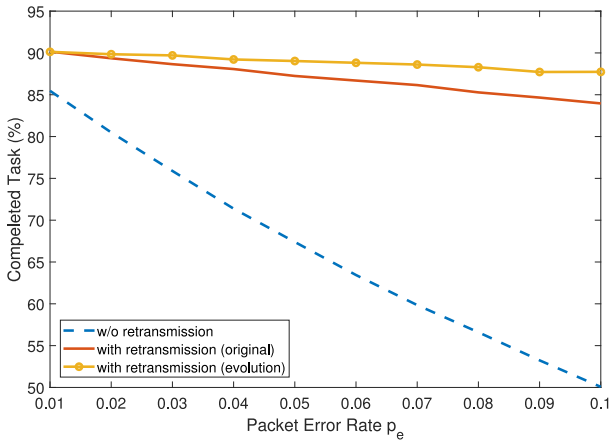


Fig. 14. Performance of the proposed online retransmission schemes without information augmentation (cifar10 data set).

V. Conclusion

In this article, we have exploited lossy compression to reduce the transmission latency for an edge inference system with hard deadlines. MDP is utilized to find the optimal dynamic compression ratio selection algorithm, so as to balance the tradeoff between the transmission latency and inference accuracy. Experiments show that dynamic compression ratio selection helps the edge inference system complete more tasks under the latency deadline constraint and the proposed online algorithm can get almost the same performance as the offline upper bound. Furthermore, we propose an information augmentation scheme to reduce communication costs. The information augmentation scheme is more robust to the change of the arrival rate than the one without information augmentation and its performance becomes better when the latency requirement τ is larger. Uncertainty-based confidence estimation for the inference result enables the system to use the information augmentation scheme when the correctness of inference result unknown, and about 2% more tasks can be completed when the arrival rate is high, compared with the online algorithm without information augmentation. Finally, the proposed retransmission scheme can further address the unreliable wireless channel, the performance is robust even when the PER becomes larger in experiments. To extend this article in IoT systems with a large amount of edge devices, where multiple edge servers should be deployed, the algorithms coordinating task allocation among edge servers are worth studying. The joint inference scheduling and model training is also a promising future direction for many IoT systems.

REFERENCES

[1] X. Huang and S. Zhou, "Latency guaranteed edge inference via dynamic compression ratio selection," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2020, pp. 1–6.

[2] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[3] A. Yousefpour *et al.*, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Architect.*, vol. 98, pp. 289–330, Sep. 2019.

[4] Y. Sun *et al.*, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.

[5] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[6] X. He, H. Xing, Y. Chen, and A. Nallanathan, "Energy-efficient mobile-edge computation offloading for applications with shared data," 2018. [Online]. Available: arXiv:1809.00966.

[7] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.

[8] X. Liu, Z. Qin, and Y. Gao, "Resource allocation for edge computing in IoT networks via reinforcement learning," 2019. [Online]. Available: arXiv:1903.01856.

[9] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," 2018. [Online]. Available: arxiv:1804.00514.

[10] 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.

[11] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," 2018. [Online]. Available: arXiv:1812.02858.

[12] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: arXiv:1704.04861.

[13] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

[14] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. Neural Inf. Process. Syst.*, Dec. 2015, pp. 1135–1143.

[15] Y. H. Oh *et al.*, "A portable, automatic data qantizer for deep neural networks," in *Proc. Int. Conf. Parallel Architect. Compilation Techn.*, Nov. 2018, p. 17.

[16] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. Workshop Mobile Edge Commun.*, Aug. 2018, pp. 31–36.

[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," in *Proc. IEEE INFOCOM Workshop*, May 2019, pp. 1–6.

[18] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. Int. Conf. Pattern Recognit.*, Dec. 2016, pp. 2464–2469.

[19] J. Shao and J. Zhang, "BottleNet++: An end-to-end approach for feature compression in device-edge co-inference systems," 2019. [Online]. Available: arXiv:1910.14315.

[20] J. M. Zurada, A. Malinowski, and S. Usui, "Perturbation method for deleting redundant inputs of perceptron networks," *Neurocomputing*, vol. 14, no. 2, pp. 177–193, Feb. 1997.

[21] J. Azar, A. Makhoul, M. Barhamgi, and R. Couturier, "An energy efficient IoT data compression approach for edge machine learning," *Future Gener. Comput. Syst.*, vol. 96, pp. 168–175, Jul. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18331716

[22] D. Xu, Q. Li, and H. Zhu, "Energy-saving computation offloading by joint data compression and resource allocation for mobile-edge computing," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 704–707, Apr. 2019.

[23] C. Finlay and A. M. Oberman, "Empirical confidence estimates for classification by deep neural networks," 2019. [Online]. Available: arXiv:1903.09215.

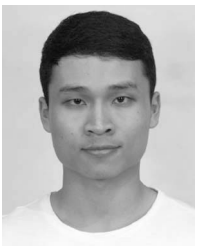
[24] S. Lin, D. J. Costello, and M. J. Miller, "Automatic-repeat-request error-control schemes," *IEEE Commun. Mag.*, vol. 22, no. 12, pp. 5–17, Dec. 1984.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[26] A. Krizhevsky, *Learning Multiple Layers of Features From Tiny Images*, Univ. Toronto, Toronto, ON, Canada, May 2012.

[27] S. Osowski and D. D. Nghia, "Neural networks for classification of 2-D patterns," in *Proc. Int. Conf. Signal Process.*, vol. 3, Aug. 2000, pp. 1568–1571.

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," 2018. [Online]. Available: arXiv:1801.04381.



Xiufeng Huang received the B.E. degree from the Electronic Engineering Department, Tsinghua University, Beijing, China, in 2018, where he is currently pursuing the Ph.D. degree with Niulab.

His research focuses on wireless communication resource allocation for edge learning.

Sheng Zhou (Member, IEEE) received the B.E. and Ph.D. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2005 and 2011, respectively.

From January to June 2010, he was a visiting student with the Wireless System Lab, Department of Electrical Engineering, Stanford University, Stanford, CA, USA. From November 2014 to January 2015, he was a Visiting Researcher with the Central Research Laboratory, Hitachi Ltd., Tokyo, Japan. He is currently an Associate Professor with the Department of Electronic Engineering, Tsinghua University. His research interests include cross-layer design for multiple antenna systems, mobile-edge computing, vehicular networks, and green wireless communications.

Dr. Zhou received the IEEE ComSoc Asia–Pacific Board Outstanding Young Researcher Award in 2017.