Joint Optimization of Computation Offloading, Model Caching and Exit Selection for DNN Inference Tasks

1st Tam Minh Nguyen
School of Electronic Engineering
Soongsil University
Seoul, Republic of Korea
nmtam@soongsil.ac.kr

2nd Myungsik Yoo
School of Electronic Engineering
Soongsil University
Seoul, Republic of Korea
myoo@ssu.ac.kr

***Abstract*—In this work, we solve the joint optimization problem of offloading, model caching, and exit selection decisions to minimize the total system delay and maximize the inference accuracy. We first formulate the problem as a Markov Decision Process (MDP) and then use the Advantage Actor Critic (A2C) algorithm to optimize in a dynamic environment. Simulation results have demonstrated the effectiveness of our proposed method.**

***Index Terms*—Mobile edge computing, deep reinforcement learning, computation offloading, early exit, model caching.**

I. Introduction

With the recent rapid advancement of mobile Artificial Intelligence (AI) services, Deep Neural Network (DNN) inference has become a critical task to be addressed on user devices, but it faces challenges regarding limited resources. To address these limitations, computation offloading for DNN inference tasks has been widely investigated and considered a promising solution [1]. An advanced technique called early exit has also been applied to further reduce the computational burden of DNN inference tasks [2]. However, existing studies focus only on the offloading process and do not consider the model caching problem [2], [3], even though DNN models must be cached to be processed at the edge, and their large size and diversity, combined with limited edge storage, pose significant challenges. Motivated by these challenges, this work addresses the joint problem of model caching, computation offloading, and early-exit selection, aiming to minimize overall system delay while maximizing inference accuracy.

II. System model

A. Scenario

This work considers a DNN inference offloading system with a cloud server, a set of non-overlapped edge servers $\mathcal{N} = \{1, \ldots, N\}$, and multiple user devices $\mathcal{U} = \{1, \ldots, U\}$ distributed under their coverage areas. Edge servers are connected through fiber-optic links and to the cloud center via backhaul links. User devices can communicate with their local edge server over wireless channels. Each edge server $n \in \mathcal{N}$ has computational and storage capacities $f_n^{\max}$ and S_n, respectively, and can serve DNN inference offloading requests from either local or remote devices.

The system operates in a timestep model denoted as $t \in \{1, \ldots, T\}$. At each timestep t, user device u can generate an inference offloading task $\tau_u(t) = \{k_u(t), s_u(t)\}$, where $s_u(t)$ is the task input size and $k_u(t) = m \in \mathcal{M} = \{1, 2, \ldots, M\}$ denotes the type of DNN inference task. The demand for each DNN model type is assumed to follow a Zipf distribution, with each edge server having its own distribution to capture localized popularity. At each timestep t, the model popularity evolves with correlation to the request frequency observed in the current slot and additional randomness to reflect stochastic behavior. Each DNN model of type m is defined as a tuple $\{\mathcal{B}_m, h_m\}$, where $\mathcal{B}_m = \{1, 2, \ldots, |\mathcal{B}_m|\}$ is the set of blocks and h_m is the model size. The inference task can exit after any block $b \in \mathcal{B}_m$, with an associated early-exit branch $\{c_m^b, acc_m^b\}$, where c_m^b is the computation required to process the block and acc_m^b is the accuracy achieved if exiting after block b. We assumed that computing more blocks results in higher inference accuracy.

After tasks are generated, each device u sends its task information to the cloud. The cloud then makes global task offloading decisions $X(t) = \{x_{u,n}(t)\} \in \{0, 1\}$, where $x_{u,n}(t) = 1$ if device u offloads to edge server n, and exit selection $Y(t) = \{y_u(t)\}$, where $y_u(t) \in \mathcal{B}_{k_u(t)}$ denotes the chosen block at which computation exits. After tasks are processed, caching decisions $Z(t) = \{z_{m,n}(t)\} \in \{0, 1\}$ are updated for the next timestep to determine which DNN models should be stored for the next timestep.

B. Communication Model

For simplicity, we assume the wireless channel bandwidth is equally allocated, with each user device u is allocated a bandwidth of B_u. The wireless data rate of user device u can be computed as $R_u = B_u \cdot \log_2\left(1 + \frac{P_u g_u(t)}{\sigma^2}\right)$ where P_u is the transmission power, σ^2 is the noise power, and $g_u(t)$ is the channel gain. The data rate between edge servers over wired connections is denoted as R_{i2i}, and the data rate from an edge server to the cloud via the backhaul is denoted as R_{i2c}.

To perform offloading, devices have to transmit their task input data to the selected edge server and require a corresponding DNN model to process the task. The transmission delay is dependent on the offloading decision and the caching status. When the required DNN model for the inference is cached at the chosen edges:

- If device u offloads its task to its local edge, the transmission delay is computed as:

$$D_u^{trans}(t) = \frac{s_u(t)}{R_u(t)}. \tag{1}$$

- If device u offloads its task to the remote edge u, the transmission delay is computed as:

$$D_u^{trans} = \frac{s_u(t)}{R_u(t)} + \frac{s_u(t)}{R^{i2i}} \cdot h, \tag{2}$$

  where h is the hop count.

However, within the two above cases, if the required DNN model is not cached at the selected edge, it must be downloaded from the cloud server. Since the DNN model size $h_{k_u(t)}$ is much larger than the task input, and the task input transmission and model downloading can occur in parallel, task transmission is negligible. The delay in this case is assumed to be dominated by the model download time:

$$D_u^{trans}(t) = \frac{h_{k_u(t)}}{R_{i2c}}. \tag{3}$$

C. Computation Model

At each timestep t, edge server n divides its total computation capacity equally among all offloaded tasks, $f_n(t) = \frac{f_n^{\max}}{\sum_u x_{u,n}(t)}$. The computation requirement of task $\tau_u(t)$ depends on the exit selection $y_u(t)$ and computed as $c_u(t) = \sum_{b=1}^{y_u(t)} c_{k_u(t)}^b$. Task $\tau_u(t)$ can be offloaded to edge server n with computation delay $D_u^{comp}(t) = \frac{c_u(t)}{f_n(t)}$, and inference accuracy $Acc_u(t) = acc_{k_u(t)}^{y_u(t)}$.

D. Problem formulation

The total delay includes the processing and transmission delays: $D_u^{total}(t) = D_u^{trans}(t) + D_u^{comp}(t)$. The joint problem of caching, task offloading, and exit selection is formulated as an optimization problem to minimize the average of total delay and maximize the average inference accuracy:

$$\min_{X(t),Y(t),Z(t)} \frac{1}{T}\frac{1}{U}\sum_{t=1}^{T}\sum_{u=1}^{U}\left[\alpha D_u^{total}(t) - \beta Acc_u(t)\right] \tag{4}$$

$$\begin{aligned}
\text{s.t.} \quad & \text{(C1): } x_{u,n}(t), z_{m,n}(t) \in \{0,1\} \quad \forall u,n,m \\
& \text{(C2): } y_u(t) \le \left|\mathcal{B}_{k_u(t)}\right|, \quad \forall n,u, \\
& \text{(C3): } \sum_{m\in\mathcal{M}} z_{m,n}(t)\cdot h_m \le S_n, \forall n
\end{aligned}$$

where (C1) specifies that the caching and offloading decisions are binary; (C2) indicates the exit selection decision cannot be more than the number of exits in the model; (C3) states that the total sizes of the cached models cannot exceed the storage of the edge server; α, β are scaling factors.

III. Method

A. Popularity-based Caching Model

For any offloading decision, if the required DNN model is not cached at the selected edge server, the transmission delay increases significantly due to the download of the model from the cloud server (Equation (3)). To mitigate this overhead, each edge server updates its cache based on model request frequencies over a sliding window of W timesteps. Specifically, at each timestep t, the caching decision $Z(t) = \{z_{m,n}(t)\}$ for edge server n is determined by selecting the most frequently requested model types within the past W timesteps, subject to constraint (C3). As a result, the most popular models are proactively cached at each edge server for the next timestep, thereby reducing the chance of cache misses.

B. MDP-based Task Offloading and Exit Selection problem

Given the cache status $Z(t)$, which can be determined as described in Section III-A, we want to make the task offloading and exit selection decision to optimize the objective in Equation (4). The optimization problem in Section II-D is reformulated as and Markov Decision Process (MDP).

- State Space:

$$S(t) = \{Z(t), \{s_u(t)\}, \{c_u(t)\}, \{g_u(t)\}, \{f_n(t)\}\}$$

- Action Space:

$$A(t) = \{X(t), Y(t)\}$$

- State Transition Probability:

$$P(S(t+1)|(S(t), A(t))$$

- Reward:

$$R(t) = -\frac{1}{U}\sum_{u=1}^{U}\left[\alpha D_u^{total}(t) - \beta A_u(t)\right]$$

To solve this, we adopt A2C algorithm, which is an on-policy Reinforcement Learning algorithm that can work with either a discrete or continuous action space.

IV. Experimentation

For the experiment, we consider a system consisting of $N = 3$ edge servers, and $U = 30$ users. Each edge server has $f_n^{max} = 20$ GHz computing power and $S_n = 8$ GB storage. There are $M = 15$ DNN task types with model sizes h_m of 300-800 MB and input sizes $s_u(t)$ of 0.5–1 MB, following the Uniform distribution. The total bandwidth B_n of each edge server is 20 MHz. We train the models for 50000 steps at a learning rate of 7×10^{-4} using Adam optimizer and evaluate the results after 500 timesteps.

The evaluation result includes the average delay and the average accuracy. We compare our method with four baselines: Random Decisions, which randomly selects offloading, caching, and exit points within constraints; Random Caching, which randomly selects caching decisions while optimizing offloading and exits accordingly; Local Edge Server, which offloads to each user device's corresponding server with

popularity-based caching and optimizes the exit decisions; and No Early Exit, where the inference exits at the final block, uses popularity-based caching, and optimizes the offloading decisions. The evaluation results of our proposed method, along with other comparison schemes, are shown in TABLE I

TABLE I
COMPARISON RESULT OF DIFFERENT METHODS

Scheme	Delay(ms)	Accuracy
Random Decisions	889.4887	0.7955
Random Caching	750.2834	0.8289
Local Edge Server	507.1049	0.7724
No Early Exit	706.3284	0.8733
Joint optimization	**500.2266**	**0.8591**

As we can observe from the results, the Random Decisions and Random Caching schemes cause the highest delay. This is because the Random Caching scheme does not effectively utilize the edge storage, causing long model downloading time. Compared with other schemes, our proposed method achieves the lowest delay while maintaining the accuracy close to that of the No Early Exit scheme, therefore providing an optimal trade-off between accuracy and delay.

V. CONCLUSION

In this paper, we have proposed a joint optimization framework for computation offloading, model caching, and exit selection for the inference tasks from DNN applications. By leveraging early exit mechanism, our proposed method can effectively reduce the computational burden and the processing delay. Simulation results have demonstrated the effectiveness of our proposed method. In the future, we plan to consider more effective caching methods besides our current caching policy based on popularity.

ACKNOWLEDGMENT

This work was supported in part by Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean government (MSIT) (No. RS-2022-II221015, Development of Candidate Element Technology for Intelligent 6G Mobile Core Network); and in part by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2025-RS-2021-II212046) supervised by the IITP (Institute for Information and Communications Technology Planning and Evaluation).

REFERENCES

[1] Ren, WQ., Qu, YB., Dong, C. et al. A Survey on Collaborative DNN Inference for Edge Intelligence. Mach. Intell. Res. 20, 370–395 (2023). https://doi.org/10.1007/s11633-022-1391-7

[2] Z. Liu et al., "Hastening Stream Offloading of Inference via Multi-Exit DNNs in Mobile Edge Computing," in IEEE Transactions on Mobile Computing, vol. 23, no. 1, pp. 535-548, Jan. 2024, doi: 10.1109/TMC.2022.3218724.

[3] Dong, Fang, Huitian Wang, Dian Shen, et al. "Multi-Exit DNN Inference Acceleration Based on Multi-Dimensional Optimization for Edge Intelligence." IEEE Transactions on Mobile Computing 22, no. 9 (2023): 5389–405. https://doi.org/10.1109/TMC.2022.3172402.