WiseAdjuster: Temporal Cost-Aware Adjustment for Collaborative Inference with Dynamic Bandwidth

*Paper 227

***Abstract*—Collaborative inference splits deep neural networks across edge devices to reduce latency, yet its efficacy is heavily compromised by dynamic wireless bandwidth. Existing adaptive methods rely on fixed adjustment strategies (e.g., periodic triggers or static thresholds) that ignore temporal adjustment costs and only leverage instantaneous bandwidth, leading to inefficient decisions where adjustment costs outweigh raw performance gains (performance gains without considering adjustment costs). In this paper, we propose WiseAdjuster, an adaptive adjustment method that employs contextual multi-armed bandits (C-MAB) to dynamically select optimal model-splitting strategies, while incorporating temporal adjustment costs and bandwidth information to avoid inefficient adjustments. Specifically, WiseAdjuster fuses GRU-based bandwidth prediction with the current strategy as inputs to C-MAB, enabling farsighted adjustments that account for temporal costs. To accurately quantify real-world adjustment costs, we further introduce a reward evaluation mechanism tailored to this cost metric. We establish rigorous theoretical bounds for the C-MAB based decision policy, proving that the regret and computational complexity are bounded by $\mathcal{O}\left(\sqrt{mT\log T}\right)$ and $\mathcal{O}(mT)$, respectively. Extensive experiments in heterogeneous edge environments demonstrate that WiseAdjuster outperforms state-of-the-art adaptation methods. WiseAdjuster reduces inference latency by 21.8%–34%, achieving performance improvements 1.2× to 2.3× greater than baselines.**

***Index Terms*—Edge computing, collaborative inference, adaptive adjustment, multi-armed bandit,**

I. Introduction

The widespread deployment of AI in edge environments enables latency-sensitive applications ranging from autonomous driving [1] and fault detection [2] to traffic monitoring [3] and smart factories [4] [5] [6]. Edge-AI is constrained by limited computational resources and data privacy requirements. By splitting DNNs across devices, collaborative inference addresses resource limitations and enables privacy-preserving execution through the exchange of intermediate features [7]. However, this paradigm heavily relies on wireless networks, where performance is significantly degraded by bandwidth fluctuations [8]. Rapid changes in network conditions, driven by factors such as mobile devices in user contention [9] [10], necessitate dynamic adjustment of split strategies to maintain inference stability.

Existing adaptive adjustment methods for collaborative inference follow fixed strategies based on preset rules without considering both adjustment costs and the future network states [11]. For instance, several works [12]–[14] adopt fixed thresholds obtained from statistics such as how frequently adjustments occurred in the past. When the monitored metric exceeds this threshold, the system adjusts the split strategy regardless of the actual bandwidth trend. Other works [15]–[17] rely on periodic adjustment, in which adjustments are conducted at predetermined time intervals (e.g., every few seconds or minutes). Many existing studies [18]–[21] focus on optimizing model accuracy or scheduling frequency, they typically assume that the costs of adjustment are negligible. These fixed adjustment strategies would lead to unnecessary and inefficient adjustments. For example, frequent adjustments can introduce overhead that outweighs the raw performance gains, ultimately reducing the overall efficiency of collaborative inference. It still lacks a comprehensive framework that jointly considers temporal adjustment costs and future network bandwidth, especially in the context of edge collaborative inference.

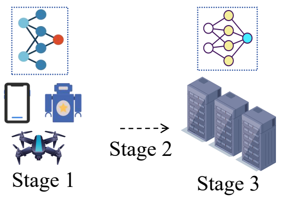
Although some recent works [22] [23] have addressed adjustment costs, they predominantly view this through the lens of energy consumption, aiming to strike a trade-off between latency and energy costs. In contrast, this paper highlights a critical but often overlooked aspect: the temporal costs incurred during the adjustment process itself. In practice, operations such as decision-making, strategy switching, and preparatory impose non-trivial latency, which is exacerbated in resource-constrained edge devices (see Fig. 7(d) to Fig. 10(d) in V). Frequent adjustments may help the system respond to changes. Unfortunately, if the time consumed by the adjustment outweighs the raw performance gain from a better split strategy, the system becomes less efficient than remaining static. Existing adjustment methods based on energy consumption cannot be directly applied to this scenario, because energy and inference latency are measured in different units and cannot be unified into a single objective. This raises the first major challenge: **quantifying and balancing the temporal adjustment costs against the raw performance gain to maximize overall inference efficiency.**

Moreover, compounding the efficiency issue is the temporal mismatch between rapid network fluctuations and system adaptation latency. Even when an adjustment is triggered at an appropriate moment, detection and reconfiguration incur delays, and the network may have changed again by the time the adjustment takes effect. This creates a risk of oscillatory behavior, where the system continuously toggles between configurations without performing useful inference. While network bandwidth prediction techniques have been extensively explored in other domains, a significant research gap remains:

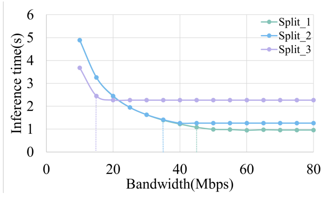
how to effectively leverage anticipated future bandwidth trends to guide collaborative inference decisions. Merely knowing the future bandwidth is insufficient, the system must intelligently map these predictions to optimal model splitting strategies to prevent shortsighted adjustments. This leads to the second challenge: **bridging the gap between bandwidth prediction and decision-making to maintain robust adaptation under fast-varying network conditions.**

This work introduces WiseAdjuster, a novel temporal cost-aware adjustment mechanism, overlooked in most prior collaborative inference studies, which balances temporal adjustment costs with raw performance gains in resource-constrained edge environments. To evaluate the temporal costs of adjustment, we develop a mathematical model with learning mechanism to adapt heterogeneous devices. We employ GRU for bandwidth prediction and C-MAB for decision-making as lightweight tools tailored to edge constraints, but the core mechanism is agnostic to specific predictors and can integrate alternatives like LSTM or transformers. WiseAdjuster employs a GRU-based predictor trained on historical traces to estimate short-term bandwidth trends instead of reacting solely to instantaneous bandwidth measurements. To reduce inefficient adjustments, we design a reward function that balances adjustment costs against raw performance gains. During each adjustment cycle, the system evaluates both performance gains and temporal costs for all candidate split points, and then selects the optimal split point with the highest overall reward via C-MAB. **The main contributions of this work are summarized as follows:**

- We propose a temporal cost-aware adaptive adjustment framework for device-edge collaborative inference. WiseAdjuster adopts a reward evaluation mechanism that jointly considers adjustment costs and raw performance gain. The reward function incorporates a learnable parameter that can adaptively fit real-world reward dynamics. We also establish a solid theoretical foundation validating the effectiveness of WiseAdjuster.
- We integrate a GRU-based prediction model to forecast future bandwidth and use it as contextual input to the dynamic adjustment method, improving WiseAdjuster's proactiveness and stability. Additionally, we design a threshold-guided bandwidth monitor for each model split strategy to ensure that adjustments are triggered only when network fluctuations significantly degrade inference performance, thereby reducing unnecessary overhead.
- We perform extensive empirical evaluations on 4 representative DNN models (AlexNet, VGG16, ResNet50, ViT-large) and heterogeneous edge devices (Raspberry Pi 4B + GPU server) under diverse bandwidth conditions. Experimental results demonstrate that WiseAdjuster outperforms state-of-the-art baselines in improving overall collaborative inference performance and exhibits strong adaptability across varying network conditions.



(a) The process of collaborative inference.



(b) The variation of inference time with bandwidth.

Fig. 1: Performance metrics of models under different methods.

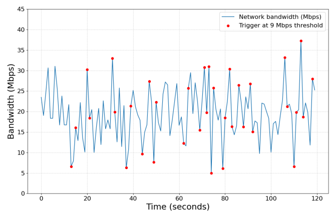
II. BACKGROUND AND MOTIVATION

The DNN model is partitioned into multiple segments and deployed across devices to enable collaborative inference, where data transmission and computation occur concurrently. Each segment corresponds to a distinct stage in the pipeline (e.g., Stage 1, Stage 2, Stage 3, as illustrated in Fig. 1(a)), and the overall throughput is determined by the stage with the longest execution time. As the edge network degrades, the transmission rate of intermediate data progressively declines. When the transmission stage becomes the performance bottleneck, dynamic reconfiguration is necessary to maintain efficient inference.

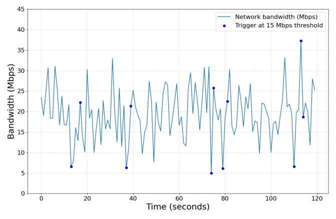
However, during collaborative inference, network fluctuations can significantly impact inference performance. In our experiments, we employ VGG16 consisting of 16 layers and 15 candidate split points for collaborative inference across two devices. It is partitioned at three specific locations (the 2nd, 6th, and 8th split points). These positions correspond to different amounts of intermediate feature data and therefore lead to three distinct communication loads. As the available bandwidth decreases, the inference time (the longest execution time of stages) varies accordingly for each partitioning scheme. The performance differences are reported in Fig. 1(b), where the purple, blue, and green curves represent the three selected split points.

Existing adjustment methods for collaborative inference are predominantly fixed or rely on pre-defined heuristics. Some methods employ periodic updates at fixed time intervals (e.g., every 5 or 10 seconds), regardless of current bandwidth states. Another adjustment method [12] only works when the observed bandwidth deviation exceeds a specified threshold. Upon identifying that the newly selected partitioning strategy differs from the current one, model repartitioning is executed accordingly. These thresholds are typically determined via empirical tuning to trade off between system responsiveness and reconfiguration overhead. To evaluate the effectiveness and limitations of existing approaches, we conducted a series of controlled experiments under representative edge computing scenarios. From these experiments, we derive two key insights that inform the design of our proposed adaptive adjustment framework.

Observation 1. Fixed adjustment methods are often inherently suboptimal in dynamic edge environments due to their inability to adapt to varying external conditions. Consider the network bandwidth fluctuations: a fixed low threshold may lead to frequent reconfiguration under highly volatile conditions, incurring unnecessary switching overhead. Conversely, an overly high threshold may delay essential adjustments, degrading system performance (see Fig. 2). In resource-constrained edge systems, such excessive reconfiguration can introduce non-negligible latency and energy overhead.



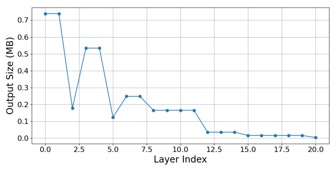
(a) Triggering time under a low threshold.



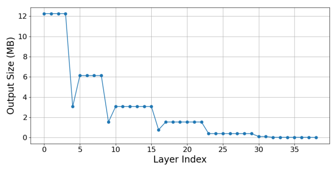
(b) Triggering time under a high threshold.

Fig. 2: Number of adjustments under different bandwidth thresholds.

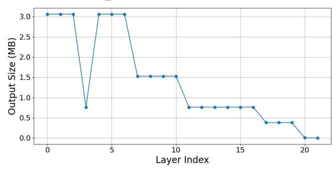
Observation 2. The structural characteristics of neural networks can be exploited to enhance the efficiency of dynamic adjustment. While network bandwidth impacts inference latency by affecting the transmission of intermediate data, its influence is often negligible within a certain fluctuation range. The point at which bandwidth variation begins to degrade collaborative inference performance is jointly determined by the size of intermediate data and the computational capabilities of the involved edge devices. Figure 3 illustrates the layer-wise size distribution of intermediate outputs in a representative deep neural network. As shown, the output size generally decreases with increasing network depth, and remains constant for a given input.



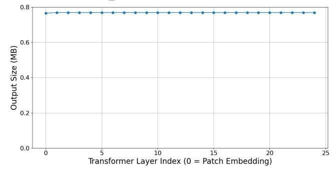
(a) Output size of AlexNet.



(b) Output size of VGG16.



(c) Output size of ResNet50.



(d) Output size of ViT-large.

Fig. 3: Performance metrics of models under different methods.

Motivated by the limitations of existing dynamic adjustment schemes and empirical insights from our experiments, we propose WiseAdjuster, a lightweight and adaptive method for dynamic model adjustment in collaborative inference. Design of WiseAdjuster should be aligned with the system's primary optimization objective. This work focuses on maximizing throughput in collaborative inference. Alternative optimization goals can be accommodated by adjusting the design of the reward function. To avoid frequent and unnecessary adjustments, we design a reward-evaluation mechanism that quantifies the benefit of each adjustment and guides adjustments accordingly.

III. WiseAdjuster design and PROBLEM FORMULATION

WiseAdjuster is an adaptive adjustment method designed for collaborative inference, offering broad applicability. It supports various model partitioning strategies and only requires information about the partition points and the corresponding transmission data sizes. In this paper, we use the model partitioning approach from DADS as an example and formalize the dynamic adjustment process based on the objective of maximizing inference throughput. WiseAdjuster aims to maximize the throughput of the pipeline, which is equivalent to minimizing the maximum per-stage processing time. Moreover, WiseAdjuster can be flexibly adapted to different optimization goals—such as minimizing inference latency [24] or jointly optimizing latency and energy consumption [25]—by replacing the reward function.

A. *WiseAdjuster Design*

GRU-based Bandwidth Prediction. Accurate estimation of future bandwidth variations enables more precise optimization of collaborative inference adjustment. To achieve this, we employ GRU [26] to forecast future network bandwidth over a period. Compared to alternative approaches such as LSTM and Transformer-based models, GRU offers a favorable trade-off between predictive accuracy and computational efficiency, making it particularly suitable for resource-constrained edge devices. Instead of forecasting instantaneous bandwidth values, our model is designed to predict the average bandwidth over time T. This design is guided by two primary considerations: (i) average predictions can be more effectively captured; (ii) average bandwidth provides a more robust characterization of overall transmission quality within a time window. To further enhance predictive accuracy and contextual sensitivity, the model also estimates the trend of bandwidth dynamics—i.e., whether the bandwidth is increasing, decreasing, or remaining stable.

Temporal Cost-aware Adaptive Adjustment. Since our goal is to maximize pipeline throughput, the costs considered in this paper refer to the additional time overhead incurred during dynamic adjustments, rather than the conventional notion of energy consumption. The adjustment costs depend on both hardware and the operating system. It includes model-loading time, scheduling delays, memory interference, and warm-up effects. These factors vary across devices and are difficult to capture with a single analytical model. Moreover, even with reliable bandwidth forecasts, the relationship between predicted bandwidth and actual end-to-end latency remains highly non-linear because of protocol overheads and device heterogeneity. Given these sources of mismatch, we adopt a C-MAB formulation. Instead of relying on an exact analytic model, the bandit learns the gap between the estimated and observed performance. Using predicted bandwidth as context, it continuously refines its decision policy online, avoiding the need for environment-specific manual calibration.

Based on both the observed and forecasted bandwidth conditions, the C-MAB assesses each candidate collaboration strategy by jointly weighing its raw performance gain against the corresponding adjustment costs. Once the optimal strategy is determined, the system initiates a dedicated thread to execute the selected policy. Formulation of C-MAB and its algorithmic solution are presented in Section III-B.

WiseAdjuster maintains low computational and communication overhead while enabling timely and efficient adaptation in collaborative inference. Instead of constantly predicting and adjusting, WiseAdjuster adopts a bandwidth-aware monitoring mechanism that suppresses unnecessary operations when network conditions do not impact overall performance. When the network reaches the adjustment thresholds, WiseAdjuster activates C-MAB algorithm to evaluate whether adjustments are worthwhile by jointly considering raw performance gain and adjustment costs. WiseAdjuster executes an adjustment only when its expected performance gain exceeds the adjustment costs.

B. Temporal Cost-Aware Adaptive Adjustment Problem Formulation

Consider a scenario involving an edge device and an edge server. DNN M executed can be split into m segments using the method described in the work [14]. The inference time for each segment of M on device and edge server is recorded as $t_{i,j}(i = 1, 2; j = 1, 2, ..., m)$. The available bandwidth monitored within time t in the edge environment is denoted by $B(t)$. Sizes of the data output at each layer's split point are represented by $d_i(i = 1, 2, ..., m)$.

During high load tasks, the inference and transmission can be decoupled in collaborative inference. The throughput of a pipeline can be approximately measured by the maximum value between inference time and transmission time. Therefore, the inference latency T is defined as equation (1):

$$T = \max\left\{\max_i(T_i^c), \max_j(T_i^t)\right\}. \tag{1}$$

T_i^c and T_i^t represent the inference time and transmission time of device i. We assume that the segment points of the DNN allocated across the n_i devices are $(\delta_0, \delta_1, ..., \delta_n)$. Equation (2) ensures that the DNN fragments assigned to each device do not exceed the total range.

$$\delta_i \in Z^+, \delta_0 = 0, \delta_n = m, \delta_i < \delta_{i+1}. \tag{2}$$

Therefore, T_i^c can be represented as

$$T_i^c \approx Y_i(\delta_i, i) = \sum_{j=\delta_{i+1}}^{\delta_i} t_{i,j}. \tag{3}$$

The transmission time depends on the network bandwidth $B(i, j, t)$ and the amount of data transmitted d_i. Network bandwidth is calculated as the average value over a period of time t.

$$T_i^t = \frac{d_i}{B(t)}, \tag{4}$$

Threshold-Guided Bandwidth Monitor periodically tracks variations in network bandwidth in edge environment and decides whether to trigger reward evaluation based on the current bandwidth conditions. We record the maximum inference latency t_i at each candidate split point, where the associated transmission sizes d_i are ordered such that $d_1 > d_2 > \ldots > d_m$.

To determine whether the current split point remains optimal under bandwidth fluctuations, we evaluate the relative dominance of computation versus communication. Reconfiguration is triggered under two conditions: (1) when the bandwidth drops such that the transmission time at the current split point $d_i/B(t)$ exceeds the computation latency at the next split t_{i+1}; (2) when the bandwidth increases and the current computation latency t_i becomes larger than the transmission time at the previous coarser split $d_{i-1}/B(t)$. These conditions ensure that adjustments are only triggered when the system bottleneck shifts. This dynamic thresholding ensures that adjustments are made only when the system's performance bottleneck shifts between computation and communication, thereby minimizing unnecessary strategy switches.

Raw Performance Gain. To assess the anticipated performance improvement over the upcoming period, the forecasted bandwidth conditions must be taken into account. Utilizing the historical inference times of the devices and the predicted bandwidth $\widetilde{B}$, we first define the raw performance gain P as the difference between predicted latency under the current strategy and the present observed latency.

$$P = \widetilde{T} - T, \; \widetilde{T} = max[max(\widetilde{T}_i^c), max(\widetilde{T}_j^t)]. \tag{5}$$

$\widetilde{T}$ represents the inference latency in the future, and the bandwidth B is replaced by $\widetilde{B}$. T is equivalent to Equation (1). Although the inference latency for a single task has been reduced, the adjustment costs remain significant.

Temporal Adjustment Costs. The total adjustment costs are composed of three components: the *decision cost* C_d, *switch cost* C_s and preparatory cost C_p (including both model loading and warm-up). Decision cost C_d denotes the extra inference latency introduced to ongoing tasks when computational resources are partially occupied during the adjustment decision-making process. Switch cost C_s refers to the time of transition from the old strategy to the newly deployed one. Once the switching system and strategy are fixed, both C_d and C_s become relatively stable. Their average values are obtained through multiple empirical measurements, and defined as c.

The preparatory cost C_p represents the time delay introduced during the deployment and loading of a new strategy model, and it can be roughly approximated as a linear function of the network layers being loaded [27]. Formally, C_p is defined as:

$$C_p = \alpha \cdot lat(\delta^t) + \beta \cdot load(\delta^t),$$

where $lat(\delta^t)$ denotes the time of inference and $load(\delta^t)$ represents parameters for loading the model. The values of α and β are computed from the results in several experiments. To more accurately estimate the real-world adjustment costs under varying conditions, we introduce a parameter Φ_θ to approximate C_p. By learning from adjustments, Φ_θ can fit costs more precisely. In summary, the total adjustment costs is given by:

$$Cost = C_d + C_s + C_p = c + \alpha \cdot lat(\delta^t) + \beta \cdot load(\delta^t) + \Phi_\theta \quad (6)$$

The above c, $lat(\delta^t)$, $load(\delta^t)$, and Φ_θ are all functions related to time, so the unit of costs is also time. Here, we only consider the time delay cost of adjustment, and the trade-off between energy consumption and delay can be left as future work.

Adaptive Adjustment Optimization Problem. In every adjustment, the objective is to update the strategy maximizing the overall system throughput. The reward r is defined as jointly considering raw performance gain and adjustment costs in the future time T_f. T_f represents the time in the future for which network bandwidth can be predicted, and it is a parameter that can be adjusted based on the prediction of future network bandwidth. For example, for a DNN, if we predict the bandwidth changes in the edge environment for the next 5s, then T_f here is set to 5s. The units of $Cost$, T, and $\widetilde{T}$ are also replaced with s. The significance of setting the reward function r is to consider the overall benefits of the adjustment strategy over a future period of time.

$$r(\widetilde{B}, \delta^t, \delta^0) = \frac{T_f - Cost}{\widetilde{T}} - \frac{T_f}{T}, \quad (7)$$

Prior to each adjustment, the strategy with the highest expected reward is selected.

The objective of dynamic adjustment is to select the adjustment strategy with the highest reward at each time t. By choosing appropriate model splitting points, the efficiency of collaborative inference can be improved. To enable fast decision-making on edge devices, we adopt lightweight decision-making algorithms to reduce the overall system burden. In addition, the reward function is fitted based on the outcomes of multiple selections, allowing it to better approximate the feedback from real-world environments. We leverage a multi-armed bandit algorithm to dynamically adapt to bandwidth fluctuations and make real-time adjustment decisions in response to environmental changes.

IV. Adaptive Adjustment Problem Solution and Theory Analysis

In the context of edge-based collaborative inference, we model the arm selection as a C-MAB problem. Each arm corresponds to a model split point. The observed reward depends on the predicted network bandwidth B_t, the selected arm δ^t, a parameter Φ_{θ_δ}. Let $\mathcal{A} = \{\delta_1, \delta_2, \ldots, \delta_m\}$ denote the set of arms. At each round t, the system observes the context B_t and selects an arm $\delta^t \in \mathcal{A}$. Fig. 4 illustrates a simplified diagram of neural network inference. The black dashed lines indicate the potential split points of the network, while the red dashed line marks the end of the network, selecting this arm implies that no further layers are assigned to other devices. Although different split strategies may yield different split locations, this does not affect the operation of the dynamic adjustment method.

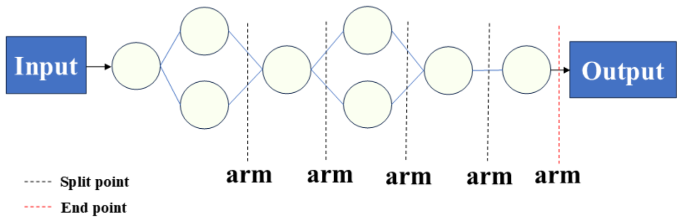


Fig. 4: Arms in contextual multi-arm bandits.

A. Regret Function

The parameters c, α, and β in reward function r were derived through an iterative computational process. Prior to the analytical reasoning phase, parameter c was first determined by conducting multiple system simulations. Subsequently, parameters α and β were calculated for each individual model. Parameter Φ_θ follows a Gaussian distribution.

Each arm is initially assigned a default reward value. During the adjustment process, the system continuously updates these rewards through learning, gradually refining them into more accurate estimates. Initial parameters θ of the MLP Φ_θ are set to zero. To evaluate the performance of dynamic adjustment throughout the inference process, the regret is defined as the cumulative reward loss over multiple adjustments:

$$\text{Regret} = \sum_{t \in T}^{T} r(\widetilde{B}_t, \delta^{t*}, \delta^0) - \sum_{t \in T}^{T} r(\widetilde{B}_t, \delta^t, \delta^0), \quad (8)$$

where δ^t denotes the arm selected at time step t, and δ^{t*} represents the optimal arm under the given context. Both the context information δ^t and $\widetilde{B}_t$ can be obtained in the edge environment.

B. C-MAB Training and Exploration Procedure

We adopt Bayesian approach using Thompson Sampling to estimate Φ_θ for each arm δ^t. Each residual Φ_θ is modeled as a Gaussian variable:

$$\Phi_{\theta_\delta} \sim \mathcal{N}(\mu_{\delta,0}, \sigma^2_{\delta,0}). \quad (9)$$

The detailed training procedure is presented in Algorithm 1.

Algorithm 1 Bayesian Thompson Sampling for Arm Selection

```
1:  Initialize: For each arm $\delta \in \mathcal{A}$, set prior $\Phi_{\theta_\delta} \sim \mathcal{N}(\mu_{\delta,0}, \sigma^2_{\delta,0})$
2:  for each round $t = 1, 2, \ldots$ do
3:      Obtain predicted bandwidth $\widetilde{B}_t$
4:      for each arm $\delta^t \in \mathcal{A}$ do
5:          Sample residual estimate: $\Phi_{\theta_\delta} \sim \mathcal{N}(\mu_{\delta,t}, \sigma^2_{\delta,t})$
6:          Compute reward estimate: $\hat{r}^{(t)}$
7:      end for
8:      Select arm: $\delta^t = \arg\max_{\delta \in \mathcal{A}} \hat{r}^{(t)}_\delta$
9:      Deploy and run the new strategy
10:     Observe true reward $r_t$
11:     Compute residual: $y_t = r_t - \hat{r}^{(t)}_\delta$
12:     Update posterior parameters for arm $\delta^t$:
13:         $\sigma^2_{\delta^t,t+1} = \left(\frac{1}{\sigma^2_{\delta^t,t}} + \frac{1}{\sigma^2}\right)^{-1}$
14:         $\mu_{\delta^t,t+1} = \sigma^2_{\delta^t,t+1}\left(\frac{\mu_{\delta^t,t}}{\sigma^2_{\delta^t,t}} + \frac{y_t}{\sigma^2}\right)$
15: end for
```

C. Theoretical Analysis

Theorem 1. *Bayesian Regret Bound under Additive Noise Assume the number of arms is $|\mathcal{A}| = m$. The residual parameter Φ_{θ_δ} for each arm is 1-dimensional and satisfies $\Phi_{\theta_\delta} \sim \mathcal{N}(\mu_{\delta,t}, \sigma^2_{\delta,t})$. The function $r(\widetilde{B}, \delta^t, \delta^0)$ is bounded in $[0, 1]$. Then the Bayesian regret of Thompson Sampling after T rounds is bounded by:*

$$\mathbb{E}[R(T)] = \mathbb{E}\left[\sum_{t=1}^{T} r_t^* - r_t\right] \le \mathcal{O}\left(\sqrt{mT\log T}\right).$$

Proof. Let the instantaneous regret at round t be defined as:

$$r_t^* - r_t = \max_{\delta\in\mathcal{A}}\left[r(\widetilde{B}_t, \delta) - \Phi_{\theta_\delta}\right] - \left[r(\widetilde{B}_t, \delta^t) - \Phi_{\theta^t_\delta}\right],$$

Due to the shared additive structure, the regret simplifies to:

$$r_t^* - r_t = \max_{\delta\in\mathcal{A}} r(\widetilde{B}_t, \delta) - r(\widetilde{B}_t, \delta^t) + \Phi_{\theta^t_\delta} - \Phi_{\theta_\delta}.$$

Taking expectation over the randomness in Φ_θ and the Thompson Sampling policy, and noting that the additive terms are sub-Gaussian, we can bound the expected cumulative regret using a standard decomposition (cf. [28]):

$$\mathbb{E}[R(T)] \le \sum_{a\in\mathcal{A}} \left(\Delta_a \cdot \mathbb{E}[N_a(T)]\right),$$

where Δ_a is the expected sub-optimality gap:

$$\Delta_a = \mathbb{E}_{\widetilde{B}_t}\left[f(\widetilde{B}_t, \delta^*) - f(\widetilde{B}_t, a)\right],$$

and $N_a(T)$ is the number of times arm a is played. Following the proof technique from Agrawal and Goyal [29]:

$$\mathbb{E}[N_a(T)] \le \mathcal{O}\left(\frac{\log T}{\Delta_a^2}\right).$$

Thus, the total regret is bounded by:

$$\mathbb{E}[R(T)] \le \sum_{a\ne a^*} \frac{\log T}{\Delta_a} \le \mathcal{O}\left(\sqrt{mT\log T}\right),$$

when all Δ_a are bounded away from zero, or applying convexity and Cauchy-Schwarz if Δ_a is unknown.

To extend the analysis to non-stationary bandwidth contexts, assume that the bandwidth changes at most V times, dividing the horizon T into $V+1$ stationary segments with lengths T_v for $v = 1, \ldots, V+1$. Within each segment v, the regret is bounded as

$$O\left(\sqrt{mT_v \log T}\right).$$

Summing over all segments and adding an additional $O(V\log T)$ term to account for change detection overhead (e.g., via windowed statistics), we obtain

$$\mathbb{E}[R(T)] \le \sum_v O\left(\sqrt{mT_v\log T}\right) + O(V\log T).$$

By the Cauchy–Schwarz inequality, which yields

$$\mathbb{E}[R(T)] \le O\left(\sqrt{VmT\log T}\right).$$

□

Theorem 2 (Computational Complexity of Bayesian Thompson Sampling)**.** *Assume that the reward function for each arm δ^t is given by:*

$$r_t(\widetilde{B}_t, \delta^t, \delta^0) = \widetilde{T}_t - T^0 - c - \alpha\cdot lat(\delta^t) - \beta\cdot load(\delta^t) - \Phi_{\theta^t_\delta},$$

where $\Phi_{\theta^t_\delta} \sim \mathcal{N}(\mu_{\delta,t}, \sigma^2_{\delta,t})$ is a learned Gaussian posterior updated via Bayesian inference. Let m be the number of arms (i.e., candidate splitting points), and T the total time steps. Then, the expected total computational complexity of Thompson Sampling over T rounds is $\mathcal{O}(mT)$, and the per-step computational complexity is $\mathcal{O}(m)$.

Proof. At each time step t, the Thompson Sampling algorithm samples the posterior reward for each arm $\delta \in \{\delta_1, \ldots, \delta_m\}$ by drawing

$$\Phi_{\theta^t_\delta} \sim \mathcal{N}(\mu_{\delta,t}, \sigma^2_{\delta,t}),$$

and computing the expected reward:

$$r(\widetilde{B}, \delta^t, \delta^0) = f(\widetilde{B}_t, \delta) - \Phi_{\theta^t_\delta},$$

the arm with the highest sampled reward is selected, and the posterior for that arm is updated based on the observed feedback.

Each step involves sampling and computing reward for all m arms, and updating one arm, yielding a per-step complexity of $\mathcal{O}(m)$. Over T steps, the total computational complexity is:

$$\mathcal{O}(mT),$$

assuming reward computation and posterior update per arm are constant-time operations. □

The proven convergence efficiency and lightweight split point computation make our algorithm highly applicable to resource-constrained environments.

Proposition 1 (Fisher Information Interpretation of Algorithm 1)**.** *For each arm δ, the residual parameter $\Phi_{\theta_\delta^t}$ follows a Gaussian posterior*

$$\Phi_{\theta_\delta^t} \sim \mathcal{N}(\mu_{\delta,t}, \sigma_{\delta,t}^2),$$

and the observed residual at round t is

$$y_t = \Phi_{\theta_\delta^t} + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma^2).$$

Then, the posterior variance update in Algorithm 1,

$$\sigma_{\delta,t+1}^2 = \left(\frac{1}{\sigma_{\delta,t}^2} + \frac{1}{\sigma^2} \right)^{-1},$$

is equivalent to additive accumulation of Fisher information, i.e.,

$$\mathcal{I}_{\delta,t+1} = \mathcal{I}_{\delta,t} + \frac{1}{\sigma^2},$$

where $\mathcal{I}_{\delta,t} = 1/\sigma_{\delta,t}^2$.

Proof. For a scalar Gaussian random variable, the Fisher information equals the inverse variance. Under Gaussian observation noise with variance σ^2, each observation contributes a Fisher information of $1/\sigma^2$. Applying Bayes' rule, the posterior precision is the sum of prior precision and observation precision, yielding the stated update. $\square$

V. IMPLEMENTATION AND EVALUATION

Fig. 5: Edge devices and a computer in the experiments

We implement and evaluate WiseAdjuster in a real-world edge computing environment (see Fig. 5). This section begins by detailing the experimental implementation and setup, followed by a comprehensive presentation and analysis of the evaluation results.

A. System Implementation and Experimental Setup

Fig. 6 illustrates the system setup used for dynamic adjustment, where WiseAdjuster is executed on a system comprising a Master node and a Worker node. Master node receives the input images and is responsible for orchestrating adaptive adjustments. Step 1: During collaborative inference, the *Bandwidth Monitor*, deployed on the Master node, monitors the communication bandwidth between the Master and Worker node. When the bandwidth exceeds predefined thresholds, it transmits the collected time-series bandwidth data to the *GRU* predictor. Step 2: Upon receiving the time-series bandwidth information, the *GRU* predictor forecasts future bandwidth trends and forwards both the historical bandwidth data and the predicted outcomes to the *C-MAB*. Step 3: Drawing on the provided bandwidth data, the *C-MAB* utilizes a MAB selection mechanism to determine the most bandwidth-efficient model splitting strategy, which is then synchronized to the Worker nodes. Step 4: *Split Model*, implemented on both the Master and Worker nodes, receives the strategy and subsequently splits the DNN model into layers, loading the resulting submodels into new inference threads. Step 5: *Switching lock* facilitates a smooth transition between existing and new splitting strategies by dispatching switching signals to each Worker node and safeguarding the inference task queue with thread locks.

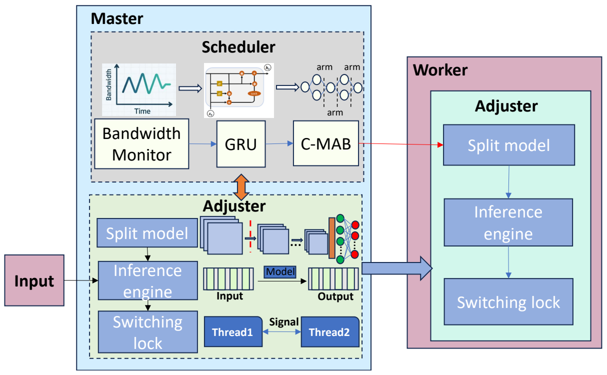


Fig. 6: Architecture overview of adjustment system.

Experimental setup. We construct an edge environment composed of heterogeneous devices and evaluate system performance under diverse network bandwidth conditions.

- **Hardware:** Our experimental testbed comprises four Raspberry Pi devices and one PC. Specifically, the Raspberry Pi 4B units are equipped with a 1.8 GHz quad-core CPU to emulate resource-constrained edge devices. We use a PC equipped with an NVIDIA GeForce RTX 3050 GPU as the edge server. To represent heterogeneous devices with varying computational capabilities, the four Raspberry Pis are configured with different CPU frequencies, and each runs a distinct DNN model.
- **Models:** Four representative deep learning models are evaluated, including *AlexNet* [31], *VGG16* [32], *ResNet* [33] and *ViT-large* [34] which contains a stack of Transformer blocks.
- **Dataset:** The image dataset used is ImageNet [35].
- **Software:** All devices run an inference framework based on PyTorch. We utilize the Linux *tc* and *netem* tools to configure bandwidth-constrained edge network environments. The initial network bandwidth of the edge environment is 20Mbps, and the bandwidth is set based on different conditions (see Table I). A single-layer GRU model with 64 hidden units is trained on data collected from a Raspberry Pi and PC connected to the same Wi-Fi network. Bandwidth traces are recorded at 1-second interval, and the model forecasts average bandwidth over the next 5 steps.

TABLE I: Network conditions used for different models.

	AlexNet	VGG16	ResNet50	ViT-large
Latency (ms)	20	15	10	300
Jitter (ms)	10	10	10	10
Packet Loss (%)	0.5	0.5	0.5	0.5

Evaluation Metrics.

- **Inference time:** We measure the average inference time per image for 100 images, repeating the experiment 5 times and taking the overall average to evaluate the system's throughput
- **Adjustment costs:** Measure the temporal costs including decision cost, switch cost and preparatory cost incurred during adjustments.
- **Algorithm Performance:**
  - **Convergence time:** The number of iterations required for WiseAdjuster to learn and converge to optimal performance under dynamic network conditions.
  - **Predicted bandwidth comparison:** We compare the average predicted bandwidth with the actual value.

Baselines. We evaluate WiseAdjuster (WA) with the following methods based on collaborative inference method [14]. [15] and [12] represent the most recent fixed adjustment strategies selected for comparison

1) No strategy for dynamic adaptation is employed (NO).
2) Periodic adjustment strategy (FT) [15].
3) Fixed bandwidth threshold strategy (FTH) [12].
4) WiseAdjuster without parameter Φ_θ (WA-N).

B. Performance of Different Methods

We conducted experiments under constrained network conditions—sufficient to trigger dynamic adjustment, but not overly severe. Under more challenging network environments, the benefits of dynamic adjustment would be even more pronounced. As shown in Fig. 8(b), FT, FTH, WA-N, and WA reduce inference latency by approximately 17.2%, 24%, 26.9%, and 34%, respectively. Moreover, the effectiveness of adjustment varies across different network conditions and model architectures (see Fig. 7(b) to Fig. 10(b)). These observations indicate that the transmission stage becomes the primary bottleneck in the baseline system. By dynamically adjusting split points, the amount of data transferred is reduced, thereby decreasing the overall transmission latency.

Among the evaluated methods, WiseAdjuster achieves a 1.42 $\times$ to 2.6 $\times$ greater reduction in inference latency compared to FT and FTH, demonstrating its superior effectiveness in dynamic collaborative inference. The superiority of WA stems from its ability to jointly consider both adjustment costs and performance gain, effectively avoiding unnecessary adjustments. In contrast, FT performs adjustments at fixed time intervals regardless of real-time network conditions, potentially disrupting the execution of ongoing tasks by reassigning computation resources. FTH triggers adjustments only when the measured bandwidth falls below a predefined threshold, making decisions based solely on instantaneous fluctuations and neglecting the broader network dynamics. Neither FT nor FTH explicitly balances the trade-off between raw performance gain and temporal adjustment costs, thereby limiting their practical effectiveness.

Both WA-N and WA further enhance the decision process by incorporating predicted network trends. A reward function is formulated to evaluate each adjustment option by jointly considering raw performance gain and adjustment costs, including the possibility of retaining the current configuration. However, the temporal adjustment costs in the reward function vary with the device's operating state. In WiseAdjuster, we integrate a learning mechanism using a contextual multi-armed bandit framework to adapt to these changes. This allows the strategy to iteratively refine its decision policy and better approximate the real temporal adjustment costs in dynamic network environments. Therefore, the performance of WA is superior to that of WA-N.

C. Prediction Ablation

Predicting network bandwidth with high accuracy remains a challenging problem. In this work, the GRU-based predictor is able to approximate the general trend of future bandwidth, as illustrated in the Fig. 7(a) to Fig. 10(a). However, its predictions may fail under unexpected or abrupt changes in network conditions.

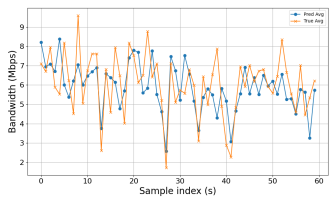
To assess the effectiveness under different prediction scenarios, we conduct experiments on VGG16 inference and record the inference latency for computing 100 images setting $T = 5$. WA-oracle is basically an ideal setup where we know the exact bandwidth for the coming time period T right from the start. WA-GRU is the approach proposed in this paper. WA-average takes the mean bandwidth from past data to represent the future bandwidth. WA-noise pulls in the bandwidth prediction straight from equation (10) as its main input.

$$\widetilde{B}_t = \mathrm{GRU}(B(t)) + \mathcal{N}(0, \sigma^2). \tag{10}$$

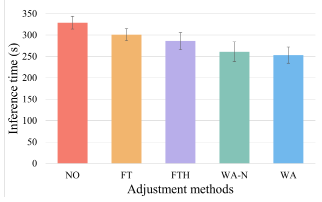
TABLE II: Comparison of WiseAdjuster under different bandwidth prediction methods.

Method	**Lat (ms)**	**Adj-Num**	**Vs. Oracle**
WA-oracle	221	9	1
WA-GRU	262	16	0.84
WA-avg	293	24	0.75
WA-noisy (σ=10)	288	26	0.77
WA-noisy (σ=20)	305	31	0.74

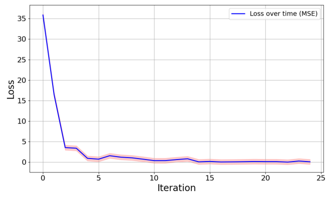
We found that even with some deviation in the prediction results (see TableII), WiseAdjuster still maintains a certain level of effectiveness. This is because the reward parameter Φ_θ can be updated after each adjustment, allowing the estimated rewards to gradually approach their true values. However, this updating process becomes slower under prediction errors.



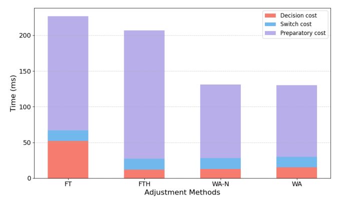
(a) Bandwidth prediction.



(b) Inference time.

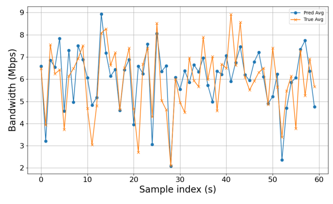


(c) C-MAB loss.

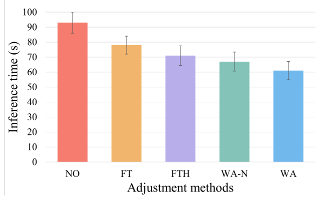


(d) Temporal adjustment costs.

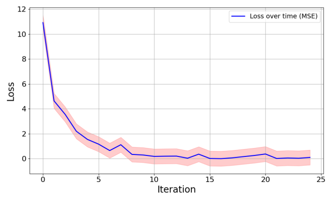
Fig. 7: Performance metrics of VGG16 under different methods.



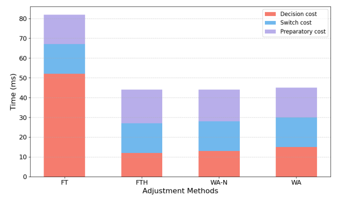
(a) Bandwidth prediction.



(b) Inference time.

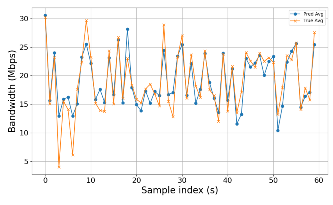


(c) C-MAB loss

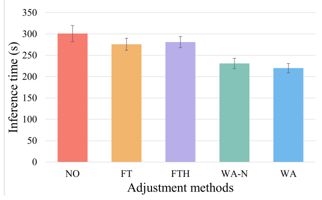


(d) Temporal adjustment costs.

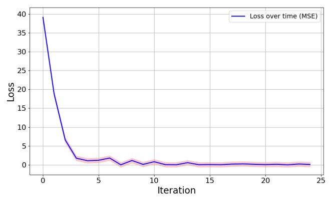
Fig. 8: Performance metrics of AlexNet under different methods.



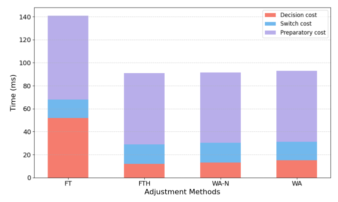
(a) Bandwidth prediction.



(b) Inference time.

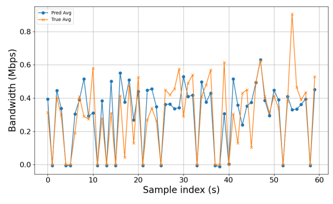


(c) C-MAB loss.

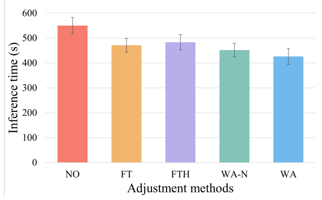


(d) Temporal adjustment costs.

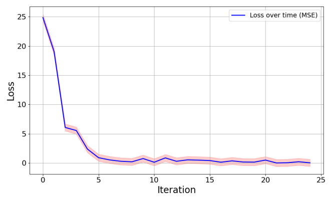
Fig. 9: Performance metrics of ResNet50 under different methods.



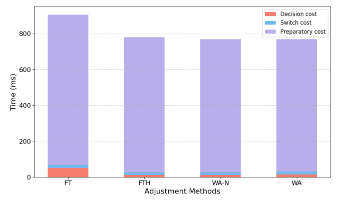
(a) Bandwidth prediction.



(b) Inference time.



(c) C-MAB loss.



(d) Temporal adjustment costs.

Fig. 10: Performance metrics of ViT-large under different methods.

D. *Evaluation of Adjustment Dynamics and Costs*

Fig. 7(c) to 10(c) illustrate the iteration of parameter Φ_θ for a specific arm during dynamic adjustment, where the loss is measured using mean squared error. When the device is in a stable status, the parameter converges within approximately five updates. This demonstrates that the system can autonomously learn the temporal adjustment costs under diverse runtime conditions, without requiring manual calibration.

Fig. 7(d) to Fig. 10(d) present the adjustment costs incurred by WiseAdjuster. Across different model configurations, decision cost and switch cost remain approximately constant, due to their dependence on inter-thread signaling, which is largely invariant to model architecture. In contrast, the preparatory cost scales with model size. In light-load scenarios, this cost constitutes around 10%–20% of a single inference time, whereas under high-load conditions, it may approach the full inference latency. These findings suggest that, in resource-constrained environments, the dynamic adjustment costs must be carefully accounted for to prevent the preparatory cost.

E. *Comparison Under Different Loads*

The temporal adjustment costs vary under different device loads. In our system, new strategies are deployed by launching new threads, enabling uninterrupted inference. Under light load (for example, when multiple CPU cores are idle), deploying a new strategy has minimal impact on ongoing inference tasks. However, under heavy load (i.e., running multiple inference tasks at the same time), the performance of collaborative inference is significantly affected.

Table III shows the performance of different DNN models under heavy load. We record the adjustment count (AdjCnt) of the model splitting strategy under different loads. The FT and FTH methods do not account for adjustment costs and thus adopt the same strategy regardless of system states. As a result, each adjustment accumulates temporal adjustment costs, with even greater buildup under heavy load. In contrast, WiseAdjuster explicitly accounts for these costs during adjustments and avoids repeated updates when the costs become excessive, leading to more performance gains in heavy-load scenarios.

TABLE III: Temporal adjustment costs of different models under heavy loads.

Model	Adjustment cost(ms)	AdjCnt (heavy load)	AdjCnt (light load)
AlexNet	90.8	19	42
VGG16	1131	5	13
ResNet50	317	8	17
ViT-large	10923	4	7

VI. Related Work

Recent advances in edge computing have led to various approaches aimed at improving inference efficiency on resource-constrained devices. Model compression techniques [36], such as quantization and pruning, are widely employed to reduce model size and computational cost, making deep neural networks more suitable for deployment on edge devices. Early-exit [37] mechanisms enable inference to terminate at intermediate layers under limited bandwidth, thereby reducing transmission latency. However, such mechanisms inherently involve a trade-off between latency and accuracy, as premature exits may degrade prediction quality.

To further alleviate computation bottlenecks, collaborative inference has been proposed, where DNN models are partitioned and distributed across multiple edge devices or edge servers [38]–[40]. These approaches effectively exploit distributed computational resources, yet many fail to account for fluctuating network conditions, making transmission performance highly sensitive to environmental dynamics. Dynamic adaptation in collaborative inference remains a challenging problem. Most existing methods rely on rule-based strategies. For example, the work [12] triggers adjustment when bandwidth fluctuates beyond a pre-defined threshold, while [15] adopt periodic reconfigurations based on fixed time intervals. However, these methods typically overlook the time overhead incurred by each adjustment and the potential fluctuations in future bandwidth, leading to inefficient adjustment decisions. EdgeTimer [43] further introduced topology-aware scheduling policies that adapt to underlying network characteristics to improve end-to-end performance.

To enable more intelligent adaptation, recent research has explored learning-based algorithms, particularly Multi-Armed Bandits (MAB) and Deep Reinforcement Learning (DRL), to optimize offloading decisions in dynamic environments [41] [42]. While these methods primarily focus on the effectiveness of decisions, without considering the temporal costs involved in the dynamic adjustment decision process. Leveraging historical traces, some works have employed learning-based predictors to forecast bandwidth trends over multiple time steps [44] [45]. However, simply stacking a bandwidth predictor onto a decision algorithm is insufficient. The critical challenge lies in deeply integrating the predicted future context into the decision-making process to explicitly weigh the long-term performance gain against the immediate adjustment costs.

VII. Conclusion

Collaborative inference has attracted increasing attention in resource-constrained edge computing environments. To address the challenge of unstable performance in collaborative inference under dynamic bandwidth, we propose WiseAdjuster, a temporal cost-aware adaptive adjustment framework. WiseAdjuster incorporates GRU-predicted bandwidth information as context and employs a MAB framework to dynamically select optimal model-splitting strategies. We integrate learnable parameters into the reward function and adopt Thompson sampling to estimate the true reward distribution. Extensive real-world edge deployment experiments demonstrate that WiseAdjuster surpasses state-of-the-art dynamic adaptation algorithms.

While this paper focuses on edge–cloud collaboration, the proposed temporal cost-aware dynamic adjustment mechanism can be extended to more complex collaborative inference scenarios involving multiple heterogeneous devices. We defer this direction to future work, as multi-device environments introduce additional challenges such as device heterogeneity and combinatorial strategy search. Nevertheless, the core idea of jointly considering future bandwidth trends and adjustment costs provides a promising foundation for adaptive control in broader collaborative systems.

References

[1] B. Liao, S. Chen, H. Yin, B. Jiang, C. Wang, S. Yan, et al., "DiffusionDrive: Truncated Diffusion Model for End-to-End Autonomous Driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Portland, OR, USA, IEEE, pp. 12037–12047, 2025.

[2] M. Zhao and O. Fink, "DyEdgeGAT: Dynamic Edge via Graph Attention for Early Fault Detection in IIoT Systems," *IEEE Internet of Things Journal*, vol. 11, no. 13, pp. 22950–22965, Dec. 2024.

[3] S. Yuan, Y. Liu, S. Guo, J. Li, H. Chen, C. Wu, and Y. Yang, "Efficient Online Computing Offloading for Budget-Constrained Cloud-Edge Collaborative Video Streaming Systems," *IEEE Transactions on Cloud Computing*, vol.13, no.1, pp.273–287, 2025.

[4] R. Xu, S. Razavi, and R. Zheng, "Edge video analytics: A survey on applications, systems and enabling techniques," *IEEE Communications Surveys & Tutorials*, vol.25, no.4, pp.2951–2982, 2023.

[5] H. Li and L. Duan, "Theory of Mixture-of-Experts for Mobile Edge Computing," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, IEEE, pp.1–10, 2025.

[6] X. Shen, W. Ma, J. Liu, C. Yang, R. Ding, Q. Wang, H. Ding, W. Niu, Y. Wang, P. Zhao, J. Lin, and J. Gu, "QuartDepth: Post-Training Quantization for Real-Time Depth Estimation on the Edge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 11448–11460, 2025.

[7] N. Chen, S. Zhang, S. Zhang, Y. Yan, Y. Chen, and S. Lu, "Resmap: Exploiting sparse residual feature map for accelerating cross-edge video analytics," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, IEEE, pp. 1–10, 2023.

[8] L. Jiang, S. D. Fu, Y. Zhu, and B. Li, “Janus: Collaborative Vision Transformer Under Dynamic Network Environment,” in *Proc. IEEE Conference on Computer Communications (INFOCOM), London, United Kingdom, pp. 1–10, May 2025.

[9] T. Kim, J. Zuo, X. Zhang, and C. Joe-Wong, “Edge-MSL: Split Learning on the Mobile Edge via Multi-Armed Bandits,” in *Proc. IEEE Conference on Computer Communications (INFOCOM), pp. 391–400, 2024.

[10] Y. Chen, H. Wang, J. Liu, X. Jia, J. Zhao and H. Qiu, ”Partitioning or Not? Hierarchical Task Offloading Optimization in Collaborative Satellite Edge Computing Networks,” 2025 IEEE 45th International Conference on Distributed Computing Systems (ICDCS), Glasgow, United Kingdom, 2025, pp. 725-735.

[11] F. Golpayegani, N. Chen, N. Afraz, E. Gyamff, A. Malekjafarian, D. Schäfer, and C. Krupitzer, “Adaptation in edge computing: A review on design principles and research challenges,” ACM Trans. Auton. Adapt. Syst, vol. 19, no. 3, Art. no. 19:1–19:43, Nov. 2024.

[12] Y. Li, Z. Liu, Z. Kou, et al., “Real-Time Adaptive Partition and Resource Allocation for Multi-User End-Cloud Inference Collaboration in Mobile Environment,” *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 13076–13094, Dec. 2024.

[13] B. Dai, J. Niu, T. Ren, et al., “Toward Mobility-Aware Computation Offloading and Resource Allocation in End–Edge–Cloud Orchestrated Computing,” *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 19450–19462, Sep. 2022.

[14] C. Hu, W. Bao, D. Wang, and F. Liu, “Dynamic Adaptive DNN Surgery for Inference Acceleration on the Edge,” in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, IEEE, pp. 1423–1431, 2019.

[15] X. Wang, J. Ye, and J. C. Lui, “Decentralized Task Offloading in Edge Computing: A Multi-User Multi-Armed Bandit Approach,” in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, IEEE, 2022.

[16] X. Ma, A. Zhou, S. Zhang, and S. Wang, “Cooperative Service Caching and Workload Scheduling in Mobile Edge Computing,” in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, IEEE, 2020.

[17] T. Kim, J. Zuo, X. Zhang, et al., “Edge-MSL: Split Learning on the Mobile Edge via Multi-Armed Bandits,” in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, IEEE, pp. 391–400, 2024.

[18] Z. Zhao, K. M. Barijough, and A. Gerstlauer, “DeepThings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2348–2359, Nov. 2018, doi: 10.1109/TCAD.2018.2858384.

[19] S. Zhang, S. Zhang, Z. Qian, J. Wu, Y. Jin, and S. Lu, “Deepslicing: Cooperative and adaptive CNN inference with low latency,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 9, pp. 2175–2187, 2021.

[20] M. Zhang, X. Shen, J. Cao, Z. Cui and S. Jiang, ”EdgeShard: Efficient LLM Inference via Collaborative Edge Computing,” in IEEE Internet of Things Journal, vol. 12, no. 10, pp. 13119-13131, 15 May15, 2025.

[21] Z. Xu, M. Zhao, Q. Li, S. Liu, F. Wu and G. Chen, ”Distributed DNN-based Video Analytics with Adaptive Multi-Device Collaboration,” 2025 IEEE 45th International Conference on Distributed Computing Systems (ICDCS), Glasgow, United Kingdom, 2025, pp. 221-231.

[22] A. Younis, S. Maheshwari and D. Pompili, ”Energy-Latency Computation Offloading and Approximate Computing in Mobile-Edge Computing Networks,” in IEEE Transactions on Network and Service Management, vol. 21, no. 3, pp. 3401-3415, June 2024.

[23] H. Zou, J. Guo, J. Zeng, Y. Li, J. Cao and T. Wang, ”Fine-Grained Service Lifetime Optimization for Energy-Constrained Edge-Edge Collaboration,” 2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS), Jersey City, NJ, USA, 2024, pp. 565-576.

[24] C. Hu and B. Li, “Distributed Inference with Deep Learning Models Across Heterogeneous Edge Devices,” in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, IEEE, pp. 330–339, 2022.

[25] S. Li, N. Zhang, R. Jiang, et al., “Joint Task Offloading and Resource Allocation in Mobile Edge Computing with Energy Harvesting,” *Journal of Cloud Computing*, vol. 11, no. 1, 2022.

[26] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk et al., “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[27] D. Trihinas, P. Michael, and M. Symeonides, “Evaluating DL model scaling trade-offs during inference via an empirical benchmark analysis,” Future Internet, vol. 16, no. 12, p. 468, 2024.

[28] S. Agrawal and N. Goyal, “Further Optimal Regret Bounds for Thompson Sampling,” in *Proc. 16th International Conference on Artificial Intelligence and Statistics (AISTATS), Scottsdale, AZ, USA, pp. 99–107, Apr. 2013.

[29] S. Agrawal and N. Goyal, “Thompson Sampling for Contextual Bandits with Linear Payoffs,” in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, Atlanta, GA, USA, pp. 127–135, 2013.

[30] Y. -Z. J. Chen, D. S. Menasché and D. Towsley, ”On Collaboration in Distributed Parameter Estimation With Resource Constraints,” in IEEE Transactions on Network and Service Management, vol. 22, no. 1, pp. 151-167, Feb. 2025.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, pp. 1097–1105, 2012.

[32] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[33] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, IEEE, pp. 770–778, 2016.

[34] A. Dosovitskiy et al., “An Image Is Worth 16×16 Words: Transformers for Image Recognition at Scale,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, ”ImageNet: A large-scale hierarchical image database,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Miami, FL, USA, pp. 248–255, 2009.

[36] B. Jacob, S. Kligys, B. Chen, et al., “Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 2704–2713, 2018.

[37] Y. Chen, Z. Niu, M. Roveri, and G. Casale, “CEED: Collaborative Early Exit Neural Network Inference at the Edge,” in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, IEEE, pp. 1–10, 2025.

[38] J. Mao, X. Chen, K. W. Nixon, C. Krieger, and Y. Chen, “Modnn: Local Distributed Mobile Computing System for Deep Neural Network,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, pp. 1396–1401, 2017.

[39] Y. Hu, Y. Zhang, J. Wang, and L. Chen, “PipeEdge: Pipeline Parallelism for Large-Scale Model Inference on Heterogeneous Edge Devices,” in *Proceedings of the 25th Euromicro Conference on Digital System Design (DSD)*, Maspalomas, Spain, IEEE, pp. 298–307, 2022.

[40] X. Yang, Z. Xu, Q. Qi, H. Sun, J. Liao, and S. Guo, “Pico: Pipeline Inference Framework for Versatile CNNs on Diverse Mobile Devices,” *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 2712–2730, 2023.

[41] Slivkins A, Sankararaman K A, Foster D J. Contextual bandits with packing and covering constraints: A modular lagrangian approach via regression[C]//The Thirty Sixth Annual Conference on Learning Theory. PMLR, 2023: 4633-4656.

[42] Fan Y, Ge J, Zhang S, et al. Decentralized scheduling for concurrent tasks in mobile edge computing via deep reinforcement learning[J]. IEEE Transactions on Mobile Computing, 2023, 23(4): 2765-2779.

[43] Y. Hao, S. Yang, F. Li, et al., “EdgeTimer: Adaptive Multi-Timescale Scheduling in Mobile Edge Computing with Deep Reinforcement Learning,” in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, IEEE, pp. 671–680, 2024.

[44] L. Mei, R. Hu, H. Cao, et al., “Realtime Mobile Bandwidth Prediction Using LSTM Neural Network and Bayesian Fusion,” *Computer Networks*, vol. 182, Art. no. 107515, 2020.

[45] R. Kablaoui, I. Ahmad, S. Abed, et al., “Network Traffic Prediction by Learning Time Series as Images,” *Engineering Science and Technology, an International Journal*, vol. 55, Art. no. 101754, 2024.