CoRE-Learning with Look-Ahead and Immediate Resource Allocation

Jing Wang, Xi-Tong Liu, Zhi-Hua Zhou

National Key Laboratory for Novel Software Technology, Nanjing University, China
School of Artificial Intelligence, Nanjing University, China
{wangjing, liuxt, zhouzh}@lamda.nju.edu.cn

Abstract

Machine learning under limited computational resources has gained increasing attention recently. A common yet challenging scenario is managing multiple time-constrained learning tasks with budgeted computational resources, known as *Computational Resource Efficient Learning* (CoRE-Learning). To this end, a recently proposed framework, *Learning with Adaptive Resource Allocation* (LARA), offers a preliminary solution. In this paper, we point out the limitations of LARA, including its reliance on interpolation-based extrapolation methods, the need for a fixed and long exploration phase, and the use of high-frequency re-estimation and reallocation strategies. To address these issues, we propose *Look-ahead and immediate Resource Allocation* (LaiRA). Our approach incorporates an efficient Dynamic Kalman Filtering (DKF) for look-ahead feasibility check with limited data and a weighted online estimator for immediate performance evaluation. For resource allocation, LaiRA constructs an Upper Confidence Bound (UCB) to enable adaptive exploration and introduces an adaptive time-slicing method to reduce task switching costs. Empirical studies validate the effectiveness of our approach.

Introduction

Machine learning (ML) under limited computational resources has gained increasing attention due to its widespread occurrence in real-world model training process, from large language model training (Achiam et al. 2023) to tiny training devices (Lin et al. 2023). Previous efforts have made significant progress in improving computational resource usage efficiency (Dean et al. 2012; Li et al. 2014) and compressing model sizes (Frankle and Carbin 2019; Mirzadeh et al. 2020), primarily for *single* learning tasks. However, a common resource-limited scenario is less explored: *multiple* time-constrained learning tasks competing for insufficient resources (Zhou 2024; Wang et al. 2024).

Such resource-limited scenarios are common in real-world production environments, where learning models need to be regularly retrained to meet specific performance goals within strict deadlines for regular product releases (Wang, Liu, and Shen 2020; Gao et al. 2021; Gu et al. 2023). For instance, recommendation systems require daily fine-tuning with fresh data to ensure real-time performance, while financial risk management models must incorporate the latest market data to provide timely predictions and risk assessments. In such scenarios, developers often face numerous time-sensitive tasks with strict performance requirements but lack sufficient computational resources to handle them all simultaneously. Consequently, they typically adopt a first-come, first-served (FCFS) strategy for resource allocation. However, this approach can lead to inefficiencies: early tasks may continue to occupy resources inefficiently, delaying critical tasks and affecting timely delivery.

To this end, Zhou (2024) proposed Computational Resource Efficient Learning (CoRE-Learning), which is the first learning-theoretic framework that explicitly models the influence of computational resource on learning performance. In addition to its theoretical importance, the CoRE-Learning framework also introduces the *scheduling* of computational resource into learning process, enabling a time-sharing usage of computational facilities. Thus, it can guide the design of *time-sharing* schedules of multiple time-constrained learning tasks with budgeted computational resources. This scheduling process involves real-time evaluation of the learning progress of each task and adaptive resource allocation. In this line of research, a key challenge is that effective allocation relies on accurate evaluation of learning progress, which itself consumes resources. Consequently, it is important to balance accurate evaluation and effective allocation over budgeted resources, similar to the exploration-exploitation dilemma in bandit problems (Lattimore and Szepesvári 2020).

Under the CoRE-Learning paradigm, Wang et al. (2024) made the first attempt by introducing Learning with Adaptive Resource Allocation (LARA) approach. LARA predicts the resource requirements for each task and models allocation as an optimization problem based on these predictions, then employs an exploration-then-exploitation strategy to balance prediction and allocation. However, LARA faces two critical limitations: (i) its heavy reliance on prediction accuracy necessitates an extensive exploration phase, which potentially reduces the time available for effective resource allocation; (ii) to compensate for prediction errors, LARA requires high-frequency re-estimation and reallocation, resulting in excessive task switches, which is impractical and inefficient in the real world.

To tackle the challenges of CoRE-Learning and overcome LARA's limitations, we propose the *Look-ahead and immediate Resource Allocation* (LaiRA) approach. LaiRA combines

look-ahead and immediate performance evaluation to enhance accuracy, reduces task switching frequency through adaptive time slicing, and eliminates the need for a fixed exploration phase with an adaptive exploration strategy. It comprises three key components: (i) *Look-ahead feasibility check*: this component employs our proposed efficient Dynamic Kalman Filtering (DKF) method, effectively identifying infeasible tasks with limited observed loss data; (ii) *Immediate performance evaluation*: this component employs a PD control mechanism to adaptively adjust allocation periods based on immediate prediction accuracy, along with an efficient weighted estimator to track immediate loss convergence progress; and (iii) *Resource allocation with exploration*: this component leverages a UCB-based adaptive exploration strategy to dynamically guide resource allocation decisions. Experimental results demonstrate that LaiRA achieves better resource utilization efficiency while reducing task-switching frequency in the CoRE-Learning paradigm.

Preliminary

In this section, we present CoRE-Learning framework and review recent progress (Wang et al. 2024).

Problem Formulation

The CoRE-Learning framework (Zhou 2024; Wang et al. 2024) considers a task bundle $\{\mathcal{T}_k\}_{k=1}^K$ over time horizon $[T]$, where each task $\mathcal{T}_k$ is characterized by: (i) a time constraint including the beginning time b_k and the user-defined deadline time d_k such that $1 \leq b_k < d_k \leq T$; (ii) the overall computational resource capability, characterized by the data budget N_k which represents the upper limit of the data amount that k-th task can learn within any unit time $t \in [T]$. At each time step t, the system maintains an active task set $A_t = \{k \mid b_k \leq t \leq d_k, k \in [K]\}$ and has to determine a resource allocation ratio $\eta_{k,t}$ for each active task $k \in A_t$, where $\eta_{k,t}N_k$ represents the actual amount of data processed for task k at time t. These allocation ratios must satisfy the resource constraints such that $\forall t \in [T], \forall k \in A_t$, $\eta_{k,t} \geq 0$, and $\sum_{k\in A_t} \eta_{k,t} \leq 1$. At the beginning of next time step $t+1$, the system observes the last round training loss $\ell_k(s)$ for each task, where $s = \sum_{i=b_k}^{t} \eta_{k,i}N_k$ represents the cumulative amount of data processed up to the time t. A task k is considered successful if its loss reaches the target threshold within its time constraint: $\ell_k(\sum_{t=b_k}^{d_k} \eta_{k,t}N_k) \leq \epsilon_k$ and will be removed from the active task set. The objective is to maximize the number of successful tasks within $[T]$. The CoRE-Learning problem can be formulated as follows:

$$\begin{aligned} \max_{\{\eta_{k,t}\}_{k\in[K],t\in[T]}} & \sum_{k\in[K]} \mathbb{I}\left[\ell_k\left(\sum_{t=b_k}^{d_k} \eta_{k,t}N_k\right) \leq \epsilon_k\right] \\ \text{s.t.} \quad & \forall t \in [T], \sum_{k\in A_t} \eta_{k,t} \leq 1, \forall k \in [K], \eta_{k,t} \geq 0. \end{aligned} \tag{1}$$

Here, the loss function ℓ_k is unknown, so real-time observations for each task are needed to update the estimation of ℓ_k before making the scheduling decisions.

Previous Effort

Under the CoRE-Learning framework, Wang et al. (2024) proposed the Learning with Adaptive Resource Allocation (LARA) approach. Since the loss curve ℓ_k is unknown, LARA uses weighted least squares (WLS) to extrapolate the curve and estimate the resources each task needs to succeed. It then applies an adaptive search method to solve the allocation problem based on the estimated curve. To ensure effective allocation, LARA adopts an explore-then-exploit strategy to improve prediction accuracy, combined with high-frequency re-estimation and reallocation to mitigate compounding errors from prediction and allocation.

However, their method faces a significant gap between resource prediction and allocation. The solution to problem (1) relies on accurately estimating the minimum resource requirements for each task, making it highly dependent on prediction accuracy. This dependence necessitates a long exploration period to achieve accurate predictions, but the extended exploration phase can cause some tasks to fail due to insufficient allocation. As a result, even though their allocation method is theoretically optimal for the estimated loss curves, the overall performance is still heavily limited by prediction accuracy. Additionally, high-frequency re-estimation and reallocation lead to extensive task switching and substantial computational costs, which is impractical for real-world scenarios.

Our Approach

In this section, we present the Look-ahead and immediate Resource Allocation (LaiRA) approach for CoRE-Learning, which operates in two phases: a look-ahead feasibility check and an immediate performance evaluation for resource allocation. In the look-ahead phase, LaiRA employs Dynamic Kalman Filtering (DKF) method to identify and eliminate infeasible tasks. In the immediate phase, it utilizes weighted least squares with UCB strategy to guide resource allocation among feasible tasks, and adaptively adjust the allocation period based on immediate prediction accuracy.

Instead of scheduling at every unit time step t, LaiRA operates on time slices indexed by τ. Time slice τ corresponds to the interval $(t_\tau, t_{\tau+1}] \subseteq [T]$, with length $M_\tau \triangleq t_{\tau+1} - t_\tau$ and $t_1 = 0$. For notational simplicity, in the following we index time by slices τ (e.g., $\eta_{k,\tau}$ denotes the allocation to task k in slice τ, and A_τ is the corresponding active set), and we let the cumulative data s have two indices: (i) s_n represents the data volume at the n-th loss observation (reported every B data points, i.e., $s_n = n \cdot B$); (ii) s_τ^k represents the data volume of k-th task at the end of time slice τ.

Look-ahead Feasibility Check

In this part, we introduce our feasibility check method. This process requires performing a look-ahead extrapolation of the loss curve based on a small amount of observed loss and corresponding cumulative data volumes. The goal is to predict whether a task can succeed with all remaining time, thereby identifying tasks that are impossible to complete.

Non-stationary Loss Model. Wang et al. (2024) models the loss curve using a negative power function and applies a

logarithmic transformation to enable linear extrapolation via regression. Motivated by empirical observations showing that log-transformed loss curves exhibit gradual drift (detailed discussion provided in a longer version), we model the loss curve as a non-stationary process, where the power function parameters evolve with cumulative data. Specifically, for the k-th task $\mathcal{T}_k$, we model the relationship between the loss $\ell_k(s)$ and the cumulative data volume s as $\ell_k(s) = a_s^k s^{-b_s^k}$, where unknown parameters a_s^k and b_s^k gradually change with s. The training process outputs an averaged loss every B data points, and after observing n-th data pairs $\{s_n, \ell_k(s_n)\}$, applying a logarithmic transformation yields a time-varying linear regression model:

$$r_n^k = X_n^\top \theta_n^k + v_n^k, \tag{2}$$

where $r_n^k \triangleq \ln \ell_k(s_n)$, $X_n \triangleq [\ln s_n; 1]$, $\theta_n^k \triangleq [-b_{s_n}^k; \ln a_{s_n}^k]$, and v_n^k is the observation noise which is assumed to follow a Gaussian distribution, i.e., $v_n^k \sim \mathcal{N}(0, R)$.

Second-order Loss Dynamics. While the log-linear model (2) enables extrapolation of the loss curve at each step, it is not enough to capture the dynamics of parameters. Empirical observations (provided in a longer version) show that the slope and intercept of the log-transformed loss curves change slowly with data volume, indicating that parameter changes are smooth enough to allow for first- and second-order modeling. To improve prediction accuracy under limited observations, we further model the evolution of the parameter θ_n^k using a second-order dynamic system. We model the dynamics of θ_n^k with state $\Theta_n^k \triangleq [\theta_n^k; \dot{\theta}_n^k; \ddot{\theta}_n^k] \in \mathbb{R}^6$, where $\dot{\theta}_n^k$ and $\ddot{\theta}_n^k$ represent the first- and second-order changes (i.e., velocity and acceleration) of the parameters over time, respectively. This allows us to capture parameter drift using the following linear dynamic system:

$$\Theta_{n+1}^k = F\Theta_n^k + w_n^k, \quad r_n^k = H_n^\top \Theta_n^k + v_n^k, \tag{3}$$

where the observation matrix $H_n \triangleq [\ln s_n; 1; 0; 0; 0; 0]$ and $F \in \mathbb{R}^{6\times 6}$ is the state transition matrix defined as

$$F \triangleq \begin{bmatrix} I_2 & \Delta_k I_2 & \frac{1}{2}\Delta_k^2 I_2 \\ \mathbf{0} & I_2 & \Delta_k I_2 \\ \mathbf{0} & \mathbf{0} & I_2 \end{bmatrix}. \tag{4}$$

The term Δ_k in (4) represents the step size, which controls the impact of $\dot{\theta}_n^k$ and $\ddot{\theta}_n^k$ to the parameter updates θ_n^k. The process noise $w_n^k \in \mathbb{R}^6$ follows a Gaussian distribution, $w_n^k \sim \mathcal{N}(0, Q)$, where $Q \in \mathbb{R}^{6\times 6}$ is the covariance matrix. The transition matrix F is designed to capture second-order dynamics of the underlying parameter Θ_n^k, following a standard formulation commonly used in state-space models for tracking evolving latent variables (Welch and Bishop 1995).

Dynamic Kalman Filtering. Based on dynamics (3), we propose the Dynamic Kalman filtering (DKF) to estimate the unknown non-stationary parameters Θ_n^k. Given the estimation of state Θ_n^k and covariance $P_n^k \triangleq \mathbb{E}\big[(\Theta_n^k - \widehat{\Theta}_n^k)(\Theta_n^k - \widehat{\Theta}_n^k)^\top\big]$ at step n, we can predict Θ_{n+1}^k and P_{n+1}^k at step $n+1$ based on the state transition (3),

$$\widehat{\Theta}_{n+1|n}^k = F\widehat{\Theta}_n^k, \quad \widehat{P}_{n+1|n}^k = F\widehat{P}_n^k F^\top + Q. \tag{5}$$

Then we will update the prediction (5) based on the observed data $\{r_{n+1}^k, H_{n+1}\}$ as follows,

$$\begin{aligned} \widehat{\Theta}_{n+1}^k &= \widehat{\Theta}_{n+1|n}^k + L_{n+1}^k \left(r_{n+1}^k - H_{n+1}^\top \widehat{\Theta}_{n+1|n}^k\right) \\ \widehat{P}_{n+1}^k &= (I_6 - L_{n+1}^k H_{n+1}^\top)\widehat{P}_{n+1|n}^k, \end{aligned} \tag{6}$$

where L_{n+1}^k is the Kalman gain for minimizing the mean squared error (Welch and Bishop 1995), as shown below:

$$L_{n+1}^k = \frac{\widehat{P}_{n+1|n}^k H_{n+1}}{R + H_{n+1}^\top \widehat{P}_{n+1|n}^k H_{n+1}}. \tag{7}$$

After estimating the underlying parameter using Eq. (6) as $\widehat{\Theta}_n^k$, we further predict whether allocating all remaining time to task k would allow its loss at time d_k to reach ϵ_k. Specifically, at the beginning of time slice τ, if the remaining time is fully allocated to task k, it can process $N_k(d_k - t_\tau)$ data volume. Since the training process typically outputs an averaged loss every B data points, we need to predict $D_k = \lfloor N_k(d_k - t_\tau)/B \rfloor$ steps of loss. To achieve this, we calculate the future parameter state as $\widehat{\Theta}_{n+D_k}^k = F^{D_k}\widehat{\Theta}_n^k$ and the cumulative data processed by that time would be $s_{n+D_k} = s_n + N_k(d_k - t_\tau)$. The estimated loss at that point is then given by:

$$\widehat{\ell}_k(s_{n+D_k}) = \widehat{a}_{s_{n+D_k}}^k \cdot {s_{n+D_k}}^{-\widehat{b}_{s_{n+D_k}}^k}, \tag{8}$$

where $\widehat{a}_{s_{n+D_k}}^k$ and $\widehat{b}_{s_{n+D_k}}^k$ are calculated by the first two dimension of $\widehat{\Theta}_{n+D_k}^k$. Finally at t_τ, we can maintain the feasible task set: $\mathcal{F}_\tau \triangleq \{k \mid \widehat{\ell}_k(s_{n+D_k}) \le \epsilon_k,\, k \in A_\tau\}$.

Immediate Performance Evaluation

After determining the feasible task set $\mathcal{F}_\tau$, we further evaluate the immediate performance of each feasible task to guide resource allocation. Specifically, we extrapolate the loss curve for each task over the upcoming time slice τ by Weighted Least Squares (WLS), where the length M_τ is dynamically adjusted. In particular, for the k-th task $\mathcal{T}_k$, after observing n data pairs $\{X_i, r_i^k\}_{i=1}^n$ and adopting the linear model (2), the estimator $\widehat{\theta}_n^k$ is computed as follows:

$$\widehat{\theta}_n^k = \min_\theta \lambda \|\theta\|_2^2 + \sum_{i=1}^n \gamma^{n-i}(X_i^\top \theta - r_i^k)^2, \tag{9}$$

where $\gamma \in (0, 1)$ is the discounted factor. Problem (9) admits a closed-form solution $\widehat{\theta}_n^k = V_n^{-1}\left(\sum_{i=1}^n \gamma^{n-i} r_i^k X_i\right)$, where $V_n \triangleq \lambda I_2 + \sum_{i=1}^n \gamma^{n-i} X_i X_i^\top$ is the covariance matrix. Moreover, the closed-form solution can be reformulated into an *online* update format (Haykin 2002, Chapter 10.3). This new format processes each data only once, hence eliminating the need to store historical data and significantly enhancing the efficiency of the estimation.

Prediction Error Analysis. For the estimator (9), we provide the following prediction error bound for immediate extrapolation. Notably, unlike latest analysis of WLS (Russac, Vernade, and Cappé 2019; Wang, Zhao, and Zhou 2023),

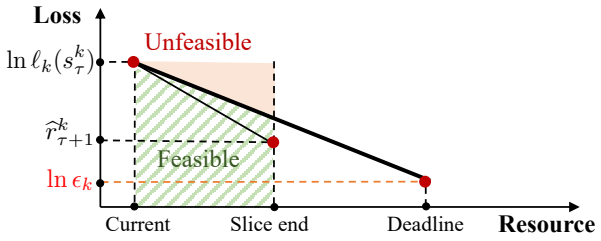


Figure 1: Linearized loss curve.

which focus on interpolation estimation error, we consider the prediction error between the estimation $\widehat{\theta}_n^k$ based on n data points and a future parameter $\theta_{n'}^k$, where $n' > n$. This requires additional analysis of the parameter's future evolution.

Theorem 1. *For any $\gamma \in (0,1)$ and $\delta \in (0,1)$, with probability at least $1-\delta$, the following holds for all $n \geq 1, n' \geq n$*

$$\left\|\widehat{\theta}_n^k - \theta_{n'}^k\right\|_{V_n} \leq L_n\sqrt{2}\sum_{p=1}^{n}\sqrt{\sum_{i=1}^{p}\gamma^{n-i}}\left\|\theta_p^k - \theta_{p+1}^k\right\|_2 + L_n\sqrt{2}\sqrt{\sum_{i=1}^{n}\gamma^{n-i}}\left\|\theta_{n+1}^k - \theta_{n'}^k\right\|_2 + \beta_n,$$

where $\beta_n \triangleq \sqrt{\lambda}S + R\sqrt{2\log\frac{1}{\delta} + 2\log\left(1 + \frac{L_n^2(1-\gamma^{2n})}{2\lambda(1-\gamma^2)}\right)}$, $L_n \triangleq \|X_n\|_2$ and S is the upper bound of unknown parameter such that $\forall k \in [K], n \geq 1, \left\|\theta_n^k\right\|_2 \leq S$.

The proof of Theorem 1 is provided in a longer version. In above prediction error bound, the first term reflects the effects of past parameter drift, controlled by a discount factor γ that downweights older changes. The second term captures the impact of future parameter changes, which cannot be adaptively handled by WLS, leading to degraded performance as the prediction horizon increases. The final term, β_n, accounts for the effect of observation noise v_i^k. Based on this result, we construct a lower confidence bound in the Resource Allocation section to support adaptive exploration.

Adaptive Time Slicing. At the beginning of time slice τ, we need to determine the length M_τ. For $\tau < 4$, we fix $M_\tau = M_1$ with given initial slice length. For $\tau \geq 4$, we use the following adaptive time slicing mechanism. Let s_τ^k denote the cumulative data of task k up to t_τ. We can observe the actual loss reduction for each task during last time slice $\tau - 1$ as $\Delta\ell_{k,\tau-1} = \ell_k(s_{\tau-1}^k) - \ell_k(s_\tau^k)$. We then compute the average prediction error across all tasks for the $\tau - 1$:

$$e_{\tau-1} = \frac{1}{|\mathcal{F}_{\tau-1}|}\sum_{k\in\mathcal{F}_{\tau-1}}\left|\Delta\widehat{\ell}_{k,\tau-1} - \Delta\ell_{k,\tau-1}\right|, \qquad (10)$$

where $\Delta\widehat{\ell}_{k,\tau-1}$ represents the predicted loss reduction for task $\mathcal{T}_k$ based on the immediate prediction (9) at $t_{\tau-1}$. To set M_τ, we first compute a temporary $\widetilde{M}_\tau$ by a proportional-derivative (PD) control mechanism:

$$\widetilde{M}_\tau = M_{\tau-1} + K_p(e_{\tau-1} - e_{\tau-2}) + K_d(e_{\tau-1} - 2e_{\tau-2} + e_{\tau-3}),$$

Algorithm 1: LaiRA

Input: Task bundle $\{\mathcal{T}_k\}_{k=1}^K$, initial slice length M_1, PD parameters K_p, K_d

$A_0 \leftarrow \emptyset, \mathcal{F}_0 \leftarrow \emptyset$
for $\tau = 1, 2, \ldots$ **do**
  Update active set A_τ; set $\mathcal{F}_\tau \leftarrow A_\tau$
  for $k \in A_\tau$ **do**
    Update DKF state $\widehat{\Theta}^k$ by Eq. (6), predict $\widehat{\Theta}_{n+D_k}^k = F^{D_k}\widehat{\Theta}^k$, and $\widehat{\ell}_k(s_{n+D_k})$ by Eq. (8); if $\widehat{\ell}_k(s_{n+D_k}) > \epsilon_k$ then $\mathcal{F}_\tau \leftarrow \mathcal{F}_\tau \setminus \{k\}$
  end for
  $M_\tau \leftarrow M_1$ if $\tau < 4$, else update M_τ by PD control
  for $k \in \mathcal{F}_\tau$ **do**
    Update WLS estimator by Eq. (9), compute ρ_τ^k by Eq. (11) and $\widehat{\rho}_\tau^k$ by Eq. (12)
  end for
  $k_* \leftarrow \arg\max_{k\in\mathcal{F}_\tau}\widehat{\rho}_\tau^k/\rho_\tau^k$; set $\eta_{k_*,\tau} = 1$, $\eta_{k,\tau} = 0$ for $k \neq k_*$; learn $\eta_{k_*,\tau}M_\tau N_{k_*}$ data for task k_*
end for

where K_p and K_d are the proportional and derivative gains, and then let $M_\tau = \max(1, \lfloor\widetilde{M}_\tau\rfloor)$ to ensure that the time slice is at least 1 time unit and is an integer. This control mechanism adapts the time slice length based on prediction accuracy: when predictions are accurate (low $e_{\tau-1}$), the time slice length increases to reduce task switching overhead; when predictions are less accurate (high $e_{\tau-1}$), the time slice length decreases to enable more frequent adjustments to changing conditions.

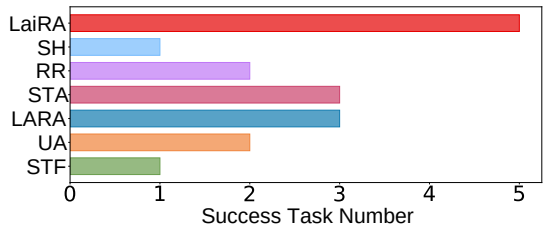
Resource Allocation

After determining the length of time slice M_τ, we allocate this time slice to the task with the highest priority score with UCB. Specifically, we first calculate the minimum required rate of loss reduction for task success. We observe the linearized loss $\ln\ell_k(s_\tau^k)$, as well as the target log-loss $\ln\epsilon_k$. Based on these observations, we define the minimum required rate of loss reduction on the linearized loss curve as a fundamental feasibility criterion for task success:

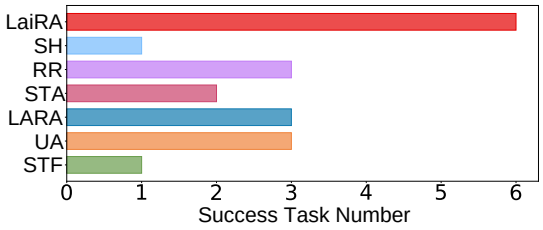
$$\rho_\tau^k = \frac{\ln\ell_k(s_\tau^k) - \ln\epsilon_k}{\ln(s_\tau^k + (d_k - t_\tau)\cdot N_k) - \ln s_\tau^k}, \qquad (11)$$

where the denominator represents the logarithmic increase in resource amount from the current already used s_τ^k to the maximum available resource $s_\tau^k + (d_k - t_\tau)\cdot N_k$.

Next, we use the WLS (9) to evaluate the task's immediate performance over current time slice M_τ. We let $n_\tau^k = s_\tau^k/B$ denote the number of observed loss data pairs for task k, then $\widehat{X}_{\tau+1}^k = [\ln(s_\tau^k + M_\tau\cdot N_k); 1]$ denote the predicted resource amount, and $\widehat{\theta}_{n_\tau^k}^k$ is the estimated parameter. To encourage exploration, we construct the immediate predicted reduction rate with a lower confidence bound (LCB) $\widehat{r}_{\tau+1}^k$ for the loss of each task, which is derived from the noise part of Theorem 1, by using the LCB $\widehat{r}_{\tau+1}^k$, we can construct a heuristic upper



(a) Pure task bundle



(b) Mixed task bundle

Figure 2: The success number of different methods.

confidence bound (UCB) for the predicted reduction rate $\widehat{\rho}_\tau^k$,

$$\begin{aligned} \widehat{\rho}_\tau^k &= \frac{\ln \ell_k(s_\tau^k) - \widehat{r}_{\tau+1}^k}{\ln(s_\tau^k + M_\tau \cdot N_k) - \ln s_\tau^k} \\ \widehat{r}_{\tau+1}^k &= \langle \widehat{X}_{\tau+1}^k, \widehat{\theta}_{n_\tau^k}^k \rangle - \beta_{n_\tau^k} \left\| \widehat{X}_{\tau+1}^k \right\|_{V_{n_\tau^k}^{-1}}. \end{aligned} \tag{12}$$

As shown in Figure 1, if the predicted reduction rate (12) exceeds the required minimum (11), the task is feasible; otherwise, it is unlikely to succeed with the available resources. We then adopt a priority-based strategy: for time slice τ, all resources go to the task with the highest score,

$$\eta_{k,\tau} = \begin{cases} 1 & \text{if } k = \arg\max_{k \in \mathcal{F}_\tau} \widehat{\rho}_\tau^k / \rho_\tau^k, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

A higher predicted reduction rate $\widehat{\rho}_\tau^k$ indicates more efficient resource usage, while a higher required reduction rate ρ_τ^k reflects a more difficult task. Thus, the ratio $\widehat{\rho}_\tau^k / \rho_\tau^k$ captures both efficiency and difficulty, making it a suitable priority score for resource allocation. Notably, this allocation strategy inherently supports adaptive exploration through the use of lower confidence bounds in calculating $\widehat{\rho}_\tau^k$, such that high $\widehat{\rho}_\tau^k$ value indicates either rapid loss reduction or high uncertainty, naturally balancing exploitation and exploration. The complete LaiRA algorithm is shown in Algorithm 1.

Experiments

This section presents experimental results, covering evaluations on pure&mixed task bundle, scalability tests, extrapolation evaluation, and an ablation study of key components.

Pure & Mixed Task Bundle Experiment

In this experiment, we evaluate our algorithms using two task bundles: (i) the pure task bundle, where all tasks use the same dataset; and (ii) the mixed task bundle, where different models are trained on separate datasets. This allows us to validate both the effectiveness and robustness of our algorithms.

Num	Model	d_k	ϵ_k	N_k
1	ViT	1560	1.23	224
2	LSTM	1000	0.23	247
3	CNN	1680	0.41	274
4	ResNet-18	1600	0.38	122
5	ResNet-34	1560	0.28	72
6	VGG	1550	0.44	58
7	EffNet	1590	0.48	90
8	Squiz	1250	1.08	111
9	LeNet	1200	0.27	580
10	DenseNet	1400	0.48	22

Table 1: The setting of pure task bundle.

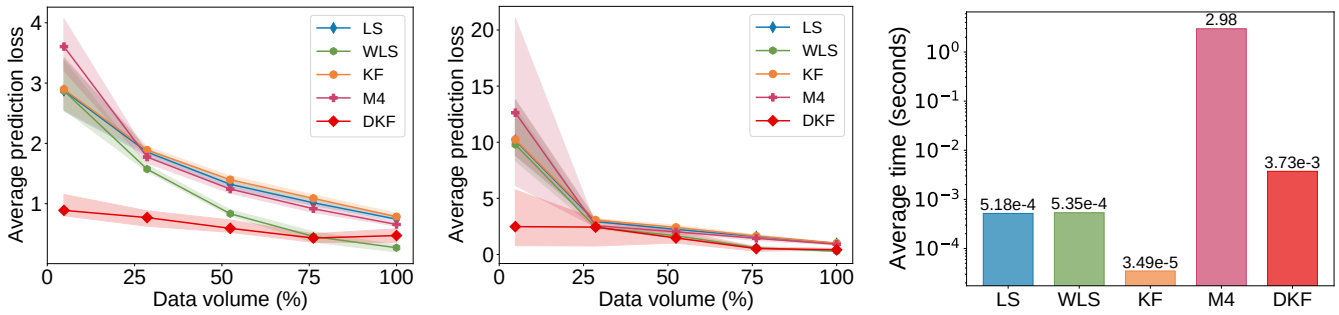
Num	Task	Model	d_k	ϵ_k	N_k
1	CV	ViT	500	0.45	227
2	CV	ResNet-18	975	0.11	113
3	CV	ResNet-34	1250	0.42	71
4	CV	CNN	1400	0.10	328
5	RL	RL_Net	980	0.0012	61
6	NLP	RNN	1360	0.01	578
7	NLP	MLP	1200	0.08	1265
8	NLP	GRU	1380	0.12	826
9	Audio	Audio_TSFM	1120	0.0024	33
10	Audio	LSTM	1120	0.0015	210

Table 2: The setting of mixed task bundle.

Setting. The pure task bundle experiment focuses different model training tasks on the CIFAR-10 dataset (Krizhevsky, Hinton et al. 2009), while the mixed task bundle experiment covers a range of diverse machine learning tasks. In each experiment, 10 models are trained simultaneously, beginning at time zero. Detailed configurations including model type, deadline d_k, success threshold ϵ_k, and data budget N_k, are provided in Table 1 for the pure task bundle and Table 2 for the mixed task bundle. More details about the task and parameter settings are provided in a longer version.

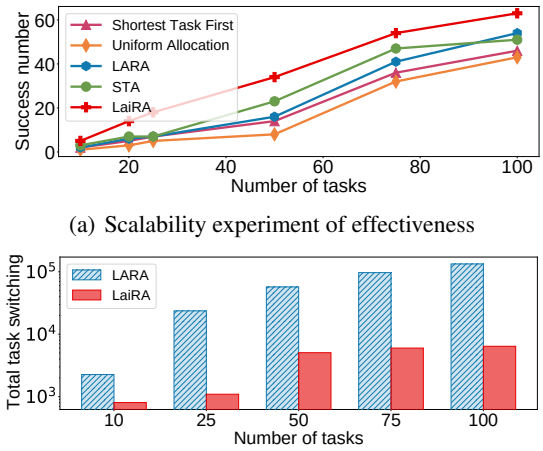
We compare our LaiRA method with several classical scheduling strategies and recent advancements: (a) Uniform Allocation (UA), which evenly distributes resources among all active tasks; (b) Shortest Task First (STF), which prioritizes tasks with the closest deadlines; (c) LARA (Wang et al. 2024), an adaptive resource scheduling strategy; (d) Short Term Allocation (STA), a variant of LaiRA without look-ahead feasibility check; (e) Round Robin (RR), which allocates resources to tasks in a round-robin manner; and (f) Successive Halving (SH) (Karnin, Koren, and Somekh 2013), a classical hyper-parameter optimization strategy that allocates resources to the most promising tasks.

Results. Figure 2(a) shows the number of successful tasks in the pure task bundle experiment. STF over-allocates to infeasible short tasks $(2, 9)$, while UA wastes resources on infeasible long tasks $(5, 6)$, leading to poor performance. LARA identifies hard tasks but spends excessive time on the exploration phase. STA explores adaptively but still wastes effort on infeasible ones, completing only three. RR completes two tasks but lacks adaptiveness. Although SH also considers scheduling based on observed loss curves of multiple training tasks, its goal is to select a single best model,



(a) Pure: prediction error (b) Mixed: prediction error (c) Mixed: time cost

Figure 3: Effectiveness and efficiency experiments of loss curve extrapolation over 5 runs.



(a) Scalability experiment of effectiveness

(b) Scalability experiment of efficiency

Figure 4: Scalability experiments.

rather than maximizing overall success. LaiRA quickly filters out infeasible tasks and focuses on feasible ones, achieving the best result. Figure 2(b) reports results for the mixed task bundle. STF over-commits to one short task; UA finishes three under tight deadlines. LARA completes only three tasks and again wastes early opportunities $(9, 10)$. STA finishes only two after misallocating to infeasible ones $(2, 6, 7)$. RR remains stable at three. SH still completes only one due to its mismatch with the scheduling goal. LaiRA completes six tasks which is the best overall. We provide allocation process in a longer version to validate our explanation.

Loss Curve Extrapolation Evaluation

In this part, we evaluate the performance of our extrapolation methods on pure&mixed task bundles.

Setting. We compare our Dynamic Kalman Filter (DKF) with four baseline extrapolation methods: (i) Least Squares (LS), (ii) Weighted Least Squares (WLS), (iii) standard Kalman Filter (KF) without transition modeling, and (iv) the $\mathcal{M}_4$ estimator (Alabdulmohsin, Neyshabur, and Zhai 2022). Evaluation is based on two metrics: (a) average prediction error $|\widehat{\ell}_k - \ell_k|/\ell_k$, where $\widehat{\ell}_k$ is the predicted loss and ℓ_k the actual loss; and (b) computational efficiency, measured by average time per prediction. We predict the 500th point on each task's loss curve, using 1% to 20% of the data as input.

Results. Figure 3(a) shows that for the pure task bundle, prediction error decreases as more data becomes available. DKF consistently outperforms other methods, especially with limited data. When data exceeds 20%, WLS performs well, but DKF and WLS still outperform LS and KF. A similar trend appears in Figure 3(b) for the mixed task bundle, with DKF leading under data-scarce conditions. Figure 3(c) compares computational efficiency. DKF is about 8× slower than other one-pass baselines due to matrix operations, but still about 800× faster than offline $\mathcal{M}_4$ estimator. This shows DKF offers a acceptable trade-off between accuracy and efficiency.

Scalability Experiment

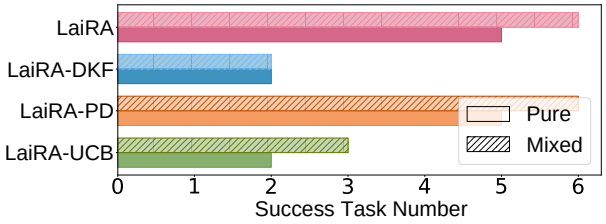
We conducted a scalability experiment to evaluate our approach as the number of tasks increases.

Setting. In this experiment, we evaluate both the performance of our methods and the task switches number ranging from 10 to 100 tasks. Each task involves training an LSTM on CIFAR-10, with varying deadlines and success thresholds. Task bundles follow a $2:6:2$ ratio of short-, mid-, and long-term tasks to reflect different urgency levels. To increase difficulty, we tighten deadline gaps among mid-term tasks and randomly select at least 10% of them to have harder ϵ_k.

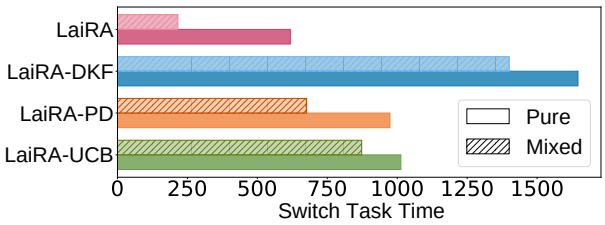
Results. Figure 4(a) shows that, as the number of tasks increases, LaiRA consistently outperforms all other methods. UA and STF perform the worst, while LARA and STA benefit from exploration, but differ in adaptability. LARA lags slightly due to its fixed exploration phase, while STA performs better with adaptive exploration. LaiRA achieves the best results by using a look-ahead feasibility check to exclude infeasible tasks early, enabling efficient allocation to feasible ones. Figure 4(b) compares task-switching counts between LaiRA and LARA, showing that LaiRA requires significantly fewer switches, confirming its better performance with lower switching overhead.

Ablation Study

In this experiment, we evaluate the effectiveness of the key components in LaiRA: the Look-Ahead Feasibility check based on DKF, the adaptive time slicing strategy based on PD, and the adaptive exploration strategy based on UCB.



(a) Success task number



(b) Switch task times

Figure 5: Ablation study on different components of LaiRA.

Setting. We compare LaiRA with three simplified variants: (i) LaiRA-UCB: removes the UCB strategy and directly allocates based on the predicted loss; (ii) LaiRA-PD: disables adaptive time slicing by using a fixed time slice interval; (iii) LaiRA-DKF: disables the long-term feasibility check.

Results. We evaluate LaiRA and its three simplified variants under both pure and mixed task bundles. Figure 5(a) shows that removing UCB or the DKF reduces performance, while removing the adaptive time slicing does not. This is because the adaptive time slicing mainly reduces task switching without affecting estimation accuracy or decision-making. Figure 5(b) shows that removing UCB, PD or the DKF increases the number of switches. Without UCB, direct allocation based on predictions causes frequent updates of priority queue, whereas UCB stabilizes the priority queue, we provide more details to validate this in a longer version. Removing the DKF eliminates the initial uniform allocation, leading to higher uncertainty and more frequent switching due to UCB's exploration strategy. These results demonstrate that each LaiRA component is essential for achieving strong performance and efficient task switching.

Related Topics

In this section, we review topics related to resource allocation in machine learning and discuss their differences with the CoRE-Learning framework studied in this work.

Machine Learning Clusters. Research in ML clusters has explored resource scheduling across multiple tasks, focusing on minimizing job completion time (JCT) (Zhang et al. 2017; Peng et al. 2018; Li et al. 2023) or guaranteeing deadlines (Wang, Liu, and Shen 2020; Gao et al. 2021; Gu et al. 2023). While sharing similar goals, our problem differs fundamentally in three aspects. First, JCT-focused methods rely on pre-trained models to predict loss curves, while we address unknown tasks requiring online learning progress evaluation. Second, deadline-focused methods define success by training iterations, whereas we target loss thresholds. Third, we explicitly account for the cost of progress evaluation itself, introducing an exploration-exploitation trade-off absent in prior work. These distinctions make existing cluster scheduling methods unsuitable for our setting. See (Ye et al. 2024) for a comprehensive survey.

Distributed Machine Learning. Distributed ML (Dean et al. 2012; Li et al. 2014) accelerate single large-scale model training through parallelization strategies (e.g., data/model parallelism) and load balancing, typically assuming sufficient resources. In contrast, our work addresses multiple tasks competing for limited resources, requiring real-time progress evaluation and strategic prioritization to maximize overall success rather than individual task efficiency.

Tiny Machine Learning. Tiny/Edge ML (Lin et al. 2023) manages learning on resource-constrained devices. Recent work (Wang et al. 2020a,b; Zhou et al. 2021) has studied concurrent task execution under power constraints. However, these approaches assume access to pre-trained models or predetermined learning trajectories, eliminating the need for online progress evaluation. Our work tackles the harder problem of simultaneous learning and allocation without such priors, necessitating real-time prediction under computational constraints and an explicit exploration-exploitation balance.

Conclusion and Future Work

This paper focuses on the Computational Resource Efficient Learning (CoRE-Learning) problem, which involves managing multiple time-constrained tasks under limited computational resources. We identify key limitations in the existing LARA algorithm. First, LARA uses WLS for long-term loss curve extrapolation, which relies on a long exploration phase to ensure accuracy shortens the resource allocation phase, causing some tasks to fail. Additionally, LARA's resource allocation requires frequent re-estimations and reallocations, leading to high task-switching costs. To address these issues, we propose LaiRA, a novel approach with two phases. In the look-ahead phase, LaiRA uses a non-stationary model and Kalman Filtering to predict loss curves and account for future parameter changes. In the immediate phase, LaiRA employs WLS with a UCB-based adaptive exploration strategy for dynamic resource allocation and an adaptive time-slicing strategy to minimize switching costs. Through experiments, we demonstrate LaiRA's effectiveness in managing multiple tasks under resource constraints.

In LaiRA, we propose a novel non-stationary loss curve extrapolation model that incorporates future parameter changes, leveraging our proposed Dynamic Kalman Filtering method for more accurate loss curve prediction. In the adaptive exploration component of LaiRA, the current UCB construction relies only on the degree of sampling, without fully accounting for the potential impact of parameter changes. Future research could focus on improving the construction of confidence bounds by explicitly incorporating parameter drift components. These advancements could lead to more robust and efficient resource allocation strategies in CoRE-Learning.

Acknowledgments

This research was supported by National Science and Technology Major Project (2022ZD0114800) and Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. GPT-4 Technical Report. *ArXiv preprint*, arxiv:2303.08774.

Alabdulmohsin, I. M.; Neyshabur, B.; and Zhai, X. 2022. Revisiting Neural Scaling Laws in Language and Vision. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, 22300–22312.

Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Le, Q. V.; Mao, M. Z.; Ranzato, M.; Senior, A. W.; Tucker, P. A.; Yang, K.; and Ng, A. Y. 2012. Large Scale Distributed Deep Networks. In *Advances in Neural Information Processing Systems 25 (NIPS)*, 1232–1240.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*.

Frankle, J.; and Carbin, M. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.

Gao, W.; Ye, Z.; Sun, P.; Wen, Y.; and Zhang, T. 2021. Chronus: A Novel Deadline-aware Scheduler for Deep Learning Training Jobs. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC)*, 609–623.

Gu, D.; Zhao, Y.; Zhong, Y.; Xiong, Y.; Han, Z.; Cheng, P.; Yang, F.; Huang, G.; Jin, X.; and Liu, X. 2023. ElasticFlow: An Elastic Serverless Training Platform for Distributed Deep Learning. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 266–280.

Haykin, S. S. 2002. *Adaptive Filter Theory*. Pearson Education India.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the 29th (IEEE) Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q. 2017. Densely Connected Convolutional Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4700–4708.

Karnin, Z. S.; Koren, T.; and Somekh, O. 2013. Almost Optimal Exploration in Multi-Armed Bandits. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 1238–1246.

Krizhevsky, A.; Hinton, G.; et al. 2009. *Learning Multiple Layers of Features from Tiny Images*. Toronto, ON, Canada.

Lattimore, T.; and Szepesvári, C. 2020. *Bandit Algorithms*. Cambridge University Press.

Li, J.; Xu, H.; Zhu, Y.; Liu, Z.; Guo, C.; and Wang, C. 2023. Lyra: Elastic Scheduling for Deep Learning Clusters. In *Proceedings of the 18th European Conference on Computer Systems (EuroSys)*, 835–850.

Li, M.; Andersen, D. G.; Park, J. W.; Smola, A. J.; Ahmed, A.; Josifovski, V.; Long, J.; Shekita, E. J.; and Su, B. 2014. Scaling Distributed Machine Learning with the Parameter Server. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 583–598.

Lin, J.; Zhu, L.; Chen, W.-M.; Wang, W.-C.; and Han, S. 2023. Tiny Machine Learning: Progress and Futures. *IEEE Circuits and Systems Magazine*, 23(3): 8–34.

Mirzadeh, S.; Farajtabar, M.; Li, A.; Levine, N.; Matsukawa, A.; and Ghasemzadeh, H. 2020. Improved Knowledge Distillation via Teacher Assistant. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 5191–5198.

Peng, Y.; Bao, Y.; Chen, Y.; Wu, C.; and Guo, C. 2018. Optimus: An efficient dynamic resource scheduler for deep learning clusters. In *Proceedings of the 13th European Conference on Computer Systems (EuroSys)*, 1–14.

Russac, Y.; Vernade, C.; and Cappé, O. 2019. Weighted Linear Bandits for Non-Stationary Environments. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 12040–12049.

Wang, H.; Liu, Z.; and Shen, H. 2020. Job scheduling for large-scale machine learning clusters. In *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 108–120.

Wang, J.; Yu, M.; Zhao, P.; and Zhou, Z.-H. 2024. Learning with Adaptive Resource Allocation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 52099–52116.

Wang, J.; Zhao, P.; and Zhou, Z.-H. 2023. Revisiting Weighted Strategy for Non-stationary Parametric Bandits. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 7913–7942.

Wang, P. 2020. vit-pytorch: Vision Transformer - Pytorch Implementation. https://github.com/lucidrains/vit-pytorch.

Wang, S.; Wang, R.; Hao, Q.; Wu, Y.; and Poor, H. V. 2020a. Learning Centric Power Allocation for Edge Intelligence. In *Proceedings of the IEEE International Conference on Communications (ICC)*, 1–6.

Wang, S.; Wu, Y.; Xia, M.; Wang, R.; and Poor, H. V. 2020b. Machine Intelligence at the Edge With Learning Centric Power Allocation. *IEEE Transactions on Wireless Communications*, 19(11): 7293–7308.

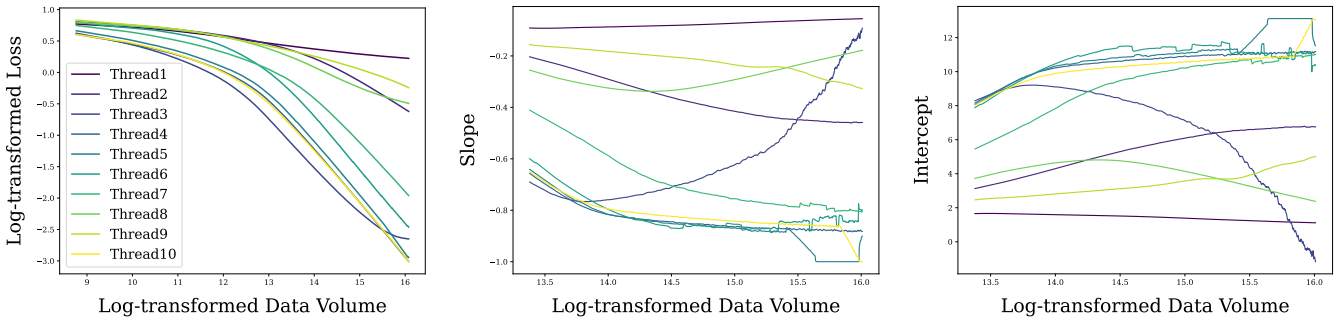
Welch, G.; and Bishop, G. 1995. An Introduction to the Kalman Filter.

Ye, Z.; Gao, W.; Hu, Q.; Sun, P.; Wang, X.; Luo, Y.; Zhang, T.; and Wen, Y. 2024. Deep Learning Workload Scheduling in GPU Datacenters: A Survey. *ACM Computing Surveys*, 56(6): 146:1–146:38.

Zhang, H.; Stafman, L.; Or, A.; and Freedman, M. J. 2017. SLAQ: Quality-driven scheduling for distributed machine learning. In *Proceedings of the 8th Symposium on Cloud Computing (SoCC)*, 390–404.

Zhou, L.; Hong, Y.; Wang, S.; Han, R.; Li, D.; Wang, R.; and Hao, Q. 2021. Learning Centric Wireless Resource Allocation for Edge Computing: Algorithm and Experiment. *IEEE Transactions on Vehicular Technology*, 70(1): 1035–1040.

Zhou, Z.-H. 2024. Learnability with time-sharing computational resource concerns. *National Science Review*, 11(10): nwae204.



(a) Log-transformed Loss Curve (b) Slope vs Log (Data Volume) (c) Intercept vs Log (Data Volume)

Figure 6: Log-transformed loss curves, slope, and intercept across tasks in Table 1.

Empirical Observations on Non-stationary Loss Dynamics

In this section, we provide detailed empirical observations that motivate our non-stationary loss model and second-order dynamic system design in Section .

Observation on Non-stationary Loss Curves

Prior work (Wang et al. 2024) models the loss curve using a negative power function and applies a logarithmic transformation to enable linear extrapolation via regression, assuming the log-transformed loss curves are strictly linear. However, our empirical observations reveal a different picture.

As shown in Figure 6(a), the log-transformed loss curves are not strictly linear but exhibit gradual drift over the course of training. This phenomenon suggests that the underlying parameters of the power function vary during training, rather than remaining constant as assumed in previous work. This observation motivates us to model the loss curve as a non-stationary process, where the power function parameters a_s^k and b_s^k evolve with cumulative data s, capturing the *non-stationary nature* of the loss curve.

Observation on Parameter Dynamics

To further understand the dynamics of these evolving parameters, we examine how the slope and intercept of the log-transformed loss curves change with data volume. Figure 6(b) shows the evolution of the slope (corresponding to parameter $-b_s^k$) across different tasks, while Figure 6(c) shows the evolution of the intercept (corresponding to parameter $\ln a_s^k$).

From these figures, we can observe that both the slope and intercept change slowly and smoothly with data volume, rather than remaining constant or changing abruptly. These findings indicate that the parameter changes are smooth enough to allow for first- and second-order modeling. Specifically, the gradual nature of these changes suggests that modeling the velocity ($\dot{\theta}_n^k$) and acceleration ($\ddot{\theta}_n^k$) of parameter evolution can significantly improve prediction accuracy, especially when limited observations are available.

These empirical observations form the foundation for our Dynamic Kalman Filtering (DKF) approach, which employs a second-order dynamic system to capture the smooth evolution of loss curve parameters and enables accurate look-ahead feasibility checks with limited data.

Prediction Error of Weighted Least Square

Proof of Theorem 1

Proof. Based on the close-form of estimator (9), the prediction error of WLS algorithm can be decomposed as

$$\begin{aligned}
\widehat{\theta}_n^k - \theta_{n'}^k &= V_n^{-1}\left(\sum_{i=1}^{n}\gamma^{n-i} r_i^k X_i\right) - \theta_{n'}^k \\
&= V_n^{-1}\left(\sum_{i=1}^{n}\gamma^{n-i}\left(X_i^\top \theta_i^k + v_i^k\right) X_i\right) - V_n^{-1}\left(\lambda I_2 + \sum_{i=1}^{n}\gamma^{n-i} X_i X_i^\top\right)\theta_{n'}^k \\
&= V_n^{-1}\left(\sum_{i=1}^{n}\gamma^{n-i} X_i X_i^\top \theta_i^k + \sum_{i=1}^{n}\gamma^{n-i} v_i^k X_i\right) - V_n^{-1}\left(\lambda I_2 + \sum_{i=1}^{n}\gamma^{n-i} X_i X_i^\top\right)\theta_{n'}^k \\
&= V_n^{-1}\left(\sum_{i=1}^{n}\gamma^{n-i} X_i X_i^\top \left(\theta_i^k - \theta_{n'}^k\right)\right) + V_n^{-1}\left(\sum_{i=1}^{n}\gamma^{n-i} v_i^k X_i - \lambda\theta_{n'}^k\right).
\end{aligned} \tag{14}$$

Therefore, we know that

$$\left\|\widehat{\theta}_n^k - \theta_{n'}^k\right\|_{V_n} \leq \underbrace{\left\|\sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top \left(\theta_i^k - \theta_{n'}^k\right)\right\|_{V_n^{-1}}}_{\text{TERM (1)}} + \underbrace{\left\|\sum_{i=1}^{n} \gamma^{n-i} v_i^k X_i - \lambda \theta_{n'}^k\right\|_{V_n^{-1}}}_{\text{TERM (2)}}, \tag{15}$$

where TERM (1) represents the impact of parameter variation, and TERM (2) represents the impact of observation noise v_i^k.

Analysis of TERM (1). We first separate the interpolation and extrapolation components of parameter variation, as follows,

$$\begin{aligned}
\left\|\sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top \left(\theta_i^k - \theta_{n'}^k\right)\right\|_{V_n^{-1}} &= \left\|\sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top \left(\theta_i^k - \theta_{n+1}^k + \theta_{n+1}^k - \theta_{n'}^k\right)\right\|_{V_n^{-1}} \\
&\leq \underbrace{\left\|\sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top \left(\theta_i^k - \theta_{n+1}^k\right)\right\|_{V_n^{-1}}}_{\text{interpolation part}} + \underbrace{\left\|\sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top \left(\theta_{n+1}^k - \theta_{n'}^k\right)\right\|_{V_n^{-1}}}_{\text{extrapolation part}},
\end{aligned}$$

where the interpolation part represents parameter variations before round t, and the extrapolation part represents parameter variations after round t. For the interpolation part, we have

$$\begin{aligned}
\left\|\sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top \left(\theta_i^k - \theta_{n+1}^k\right)\right\|_{V_n^{-1}} &= \left\|\sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top \left(\theta_i^k - \theta_{n+1}^k\right)\right\|_{V_n^{-1}} \\
&= \left\|\sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top \sum_{p=i}^{n} \left(\theta_p^k - \theta_{p+1}^k\right)\right\|_{V_n^{-1}} \\
&= \left\|\sum_{p=1}^{n} \sum_{i=1}^{p} \gamma^{n-i} X_i X_i^\top \left(\theta_p^k - \theta_{p+1}^k\right)\right\|_{V_n^{-1}} \\
&\leq \sum_{p=1}^{n} \left\|\sum_{i=1}^{p} \gamma^{n-i} X_i \left\|X_i\right\|_2 \left\|\theta_p^k - \theta_{p+1}^k\right\|_2\right\|_{V_n^{-1}} \\
&\leq L_n \sum_{p=1}^{n} \sum_{i=1}^{p} \gamma^{n-i} \left\|X_i\right\|_{V_n^{-1}} \left\|\theta_p^k - \theta_{p+1}^k\right\|_2,
\end{aligned}$$

where the last inequality comes from the fact that s_n is the cumulative data volume and is monotonically increasing in n, and $X_n = [\ln s_n; 1]$, we have $\left\|X_i\right\|_2 \leq \left\|X_n\right\|_2$ for all $i \leq n$. Hence we can take $L_n = \left\|X_n\right\|_2$ as a valid upper bound in the self-normalized inequality. Term $\sum_{i=1}^{p} \gamma^{n-i} \left\|X_i\right\|_{V_n^{-1}}$ can further derive an expression about discounted factor γ as follows,

$$\sum_{i=1}^{p} \gamma^{n-i} \left\|X_i\right\|_{V_n^{-1}} \leq \sqrt{\sum_{i=1}^{p} \gamma^{n-i}} \sqrt{\sum_{i=1}^{p} \gamma^{n-i} \left\|X_i\right\|_{V_n^{-1}}^2} \leq \sqrt{2} \sqrt{\sum_{i=1}^{p} \gamma^{n-i}}, \tag{16}$$

where the first inequality holds by the Cauchy-Schwarz inequality and the last inequality comes from that

$$\begin{aligned}
\sum_{i=1}^{p} \gamma^{n-i} \left\|X_i\right\|_{V_n^{-1}}^2 &= \sum_{i=1}^{p} \gamma^{n-i} \mathrm{Tr}(X_i^\top V_n^{-1} X_i) = \mathrm{Tr}\left(V_n^{-1} \sum_{i=1}^{p} \gamma^{n-i} X_i X_i^\top\right) \\
&\leq \mathrm{Tr}\left(V_n^{-1} \sum_{i=1}^{p} \gamma^{n-i} X_i X_i^\top\right) + \mathrm{Tr}\left(V_n^{-1} \sum_{i=p+1}^{n} \gamma^{n-i} X_i X_i^\top\right) + \mathrm{Tr}\left(V_n^{-1} \lambda \sum_{i=1}^{2} \mathbf{e}_i \mathbf{e}_i^\top\right) \\
&= \mathrm{Tr}(I_2) = 2.
\end{aligned}$$

For the extrapolation part, we have

$$\left\|\sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top \left(\theta_{n+1}^k - \theta_{n'}^k\right)\right\|_{V_n^{-1}} \leq \left\|\sum_{i=1}^{n} \gamma^{n-i} X_i \left\|X_i\right\|_2 \left\|\theta_{n+1}^k - \theta_{n'}^k\right\|_2\right\|_{V_n^{-1}}$$

$$\leq L_n \sum_{i=1}^{n} \gamma^{n-i} \|X_i\|_{V_n^{-1}} \left\|\theta_{n+1}^k - \theta_{n'}^k\right\|_2$$

$$\leq L_n \sqrt{2} \sqrt{\sum_{i=1}^{n} \gamma^{n-i}} \left\|\theta_{n+1}^k - \theta_{n'}^k\right\|_2,$$

where the last inequality is the same as the derivation of Eq. (16). Thus for TERM (1), we have

$$\left\|\sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top \left(\theta_i^k - \theta_{n'}^k\right)\right\|_{V_n^{-1}} \leq L_n \sqrt{2} \sum_{p=1}^{n} \sqrt{\sum_{i=1}^{p} \gamma^{n-i}} \|\theta_p^k - \theta_{p+1}^k\|_2 + L_n \sqrt{2} \sqrt{\sum_{i=1}^{n} \gamma^{n-i}} \left\|\theta_{n+1}^k - \theta_{n'}^k\right\|_2. \tag{17}$$

Analysis of TERM (2). Let $\widetilde{V}_n \triangleq \lambda I_2 + \sum_{i=1}^{n} \gamma^{2(n-i)} X_i X_i^\top$,

$$\left\|\sum_{i=1}^{n} \gamma^{n-i} v_i^k X_i - \lambda \theta_{n'}^k\right\|_{V_n^{-1}} \leq \left\|\sum_{i=1}^{n} \gamma^{n-i} v_i^k X_i\right\|_{V_n^{-1}} + \left\|\lambda \theta_{n'}^k\right\|_{V_n^{-1}} \leq \left\|\sum_{i=1}^{n} \gamma^{n-i} v_i^k X_i\right\|_{\widetilde{V}_n^{-1}} + \sqrt{\lambda} S.$$

Recall that $V_n = \lambda I_2 + \sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top$, so the last inequality comes from

$$V_n = \lambda I_2 + \sum_{i=1}^{n} \gamma^{n-i} X_i X_i^\top \succeq \lambda I_2 + \sum_{i=1}^{n} \gamma^{2(n-i)} X_i X_i^\top = \widetilde{V}_n.$$

We emphasize that the $\widetilde{V}_n$ is introduced into analysis *only*, which is actually *not* required in our algorithmic implementation. From the weighted version maximal deviation inequality (Russac, Vernade, and Cappé 2019, Theorem 1), restated in Theorem 2, we can get the bound for the first term $\| \sum_{i=1}^{n} \gamma^{n-i} v_i^k X_i \|_{\widetilde{V}_n^{-1}}$ as below by just let $w_i = \gamma^{n-i}, \mu_n = \lambda, d = 2$,

$$\begin{aligned} \left\|\sum_{i=1}^{n} \gamma^{n-i} v_i^k X_i\right\|_{\widetilde{V}_n^{-1}} &\leq R \sqrt{2 \log \frac{1}{\delta} + 2 \log \left(1 + \frac{L_n^2 \sum_{i=1}^{n} \gamma^{2(n-i)}}{2\lambda}\right)} \\ &\leq R \sqrt{2 \log \frac{1}{\delta} + 2 \log \left(1 + \frac{L_n^2 (1 - \gamma^{2n})}{2\lambda (1 - \gamma^2)}\right)}, \end{aligned} \tag{18}$$

Based on the inequality (15), inequality (17) and inequality (18), we have for any $\gamma \in (0, 1)$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$, the following holds for all $n \geq 1, n' \geq n$,

$$\left\|\widehat{\theta}_n^k - \theta_{n'}^k\right\|_{V_n} \leq L_n \sqrt{2} \sum_{p=1}^{n} \sqrt{\sum_{i=1}^{p} \gamma^{n-i}} \left\|\theta_p^k - \theta_{p+1}^k\right\|_2 + L_n \sqrt{2} \sqrt{\sum_{i=1}^{n} \gamma^{n-i}} \left\|\theta_{n+1}^k - \theta_{n'}^k\right\|_2 + \beta_n, \tag{19}$$

where $\beta_n \triangleq \sqrt{\lambda} S + R \sqrt{2 \log \frac{1}{\delta} + 2 \log \left(1 + \frac{L_n^2 (1 - \gamma^{2n})}{2\lambda (1 - \gamma^2)}\right)}$ is the confidence radius. Hence we complete the proof. □

Useful Lemmas

Theorem 2 (Weighted Version Self-Normalized Bound for Vector-Valued Martingales (Russac, Vernade, and Cappé 2019, Theorem 1))**.** *Let $\{\mathcal{F}_t\}_{t=0}^{\infty}$ be a filtration, $\{\eta_t\}_{t=0}^{\infty}$ be a real-valued stochastic process such that η_t is $\mathcal{F}_t$-measurable and η_t is conditionally R-sub-Gaussian for some $R \geq 0$, such that*

$$\forall \lambda \in \mathbb{R}, \mathbb{E}\left[\exp(\lambda \eta_t) \mid X_{1:t}, \eta_{1:t-1}\right] \leq \exp\left(\frac{\lambda^2 R^2}{2}\right).$$

Let $\{X_t\}_{t=1}^{\infty}$ be an $\mathbb{R}^d$-valued stochastic process such that X_t is $\mathcal{F}_{t-1}$-measurable. For any $t \geq 0$, define

$$\widetilde{V}_t = \mu_t I_d + \sum_{s=1}^{t} w_s^2 X_s X_s^\intercal, \qquad S_t = \sum_{s=1}^{t} w_s \eta_s X_s.$$

where $\forall s \geq 0, t \geq 0, w_s, \mu_t > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, we have

$$\forall t \geq 0, \|S_t\|_{\widetilde{V}_t^{-1}} \leq R \sqrt{2 \log \frac{1}{\delta} + d \log \left(1 + \frac{L^2 \sum_{s=1}^{t} w_s^2}{d \mu_t}\right)}.$$

Experimental Supplement

In this section, we provide supplementary information for our experiments. First we include further details about the implementation and parameter settings of training tasks and task bundle from Table 1 and Table 2. Then we describe the allocation process for both pure and mixed task bundles. Third, we compare the priority based on prediction results with the priority based on UCB. Finally, we present a sensitivity analysis for the parameter Δ_k in the state transition model (3) and the discount factor γ in the weighted least squares (9).

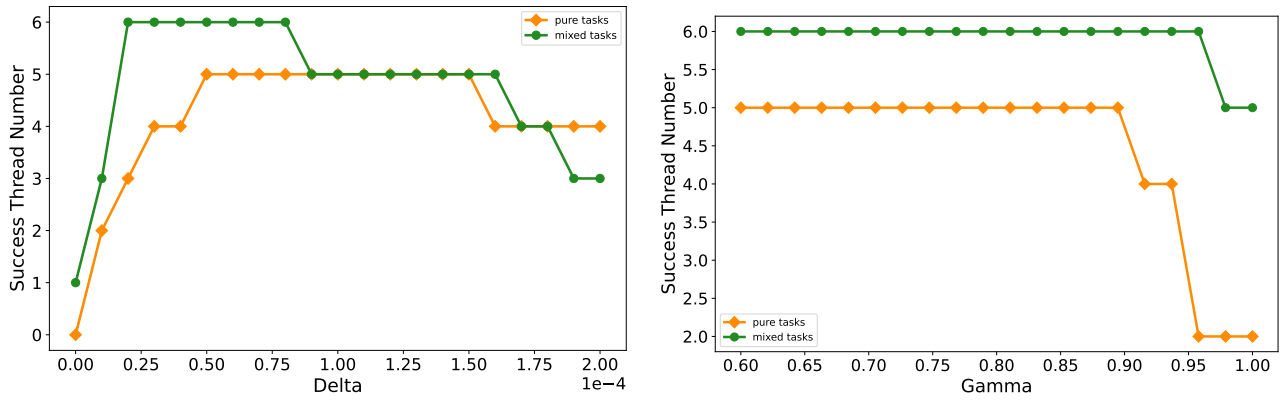
Detailed Settings for Task Bundle Generation

Training task generation. In generating training tasks, we select various types of machine learning tasks, including CV, RL, NLP and Audio tasks, as shown in Table 1 and Table 2. For Table 1, task 1 uses a Vision Transformer (ViT) (Dosovitskiy et al. 2021), implemented as a Simple ViT model (Wang 2020). Task 2 employs a Long Short-Term Memory (LSTM) network, structured with two LSTM layers followed by three fully connected layers. Task 3 is a Convolutional Neural Network (CNN), built with three convolutional layers, one pooling layer, and two fully connected layers. Tasks 4 and 5 use ResNet-18 and ResNet-34, respectively (He et al. 2016). Task 6 employs a VGGNet, featuring sequential convolutional blocks with batch normalization and two large dense layers. Task 7 utilizes EfficientNet, designed with MBConv blocks and scalable coefficients. Task 8 is SqueezeNet, constructed with Fire modules and adaptive pooling for lightweight computation. Task 9 uses LeNet, a classic architecture with convolutional and sigmoid-activated dense layers. Task 10 uses a Dense Convolutional Network (DenseNet) (Huang et al. 2017).

For Table 2, Task 1 uses ViT, tasks 2 and 3 use ResNet-18, ResNet-34 respectively, and task 4 uses CNN, all of which follow the implementation described in Table 1. Task 5 employs RL_Net, a custom CNN with five convolutional layers and three dense layers for reinforcement learning tasks. Task 6 is a Recurrent Neural Network (RNN) with a single recurrent layer and a linear output layer. Task 7 implements a Multi-Layer Perceptron (MLP) model with two fully connected layers. Task 8 employs a Gated Recurrent Unit (GRU) model with a single GRU layer and a linear classifier. Task 9 uses a Transformer model with a single encoder layer and mean pooling for sequence processing. Task 10 is an LSTM model with a single LSTM layer and a linear output layer, designed for sequential data.

Task bundle generation. In generating task bundles based on these training tasks, we designed the parameters in Table 1 and Table 2 to make the task bundles challenging and unsolvable by classical scheduling strategies. Specifically, N_k represents the maximum amount of data that can be processed per second when task k is allocated all resources, measured and calculated at the start of the experiment. Each data unit corresponds to a batch of 64 data points. For the deadline d_k and success threshold ϵ_k, tasks were initially designed to be completed in a short priority sequence. Later, some tasks were modified to include a mix of short, challenging tasks (with small d_k and small ϵ_k) and longer, simpler tasks. This design ensures that the task bundle cannot be solved by simple uniform allocation or by prioritizing shorter tasks.

Parameter Sensitivity



(a) Linear accelerated motion model

(b) Weighted Least Square Estimation

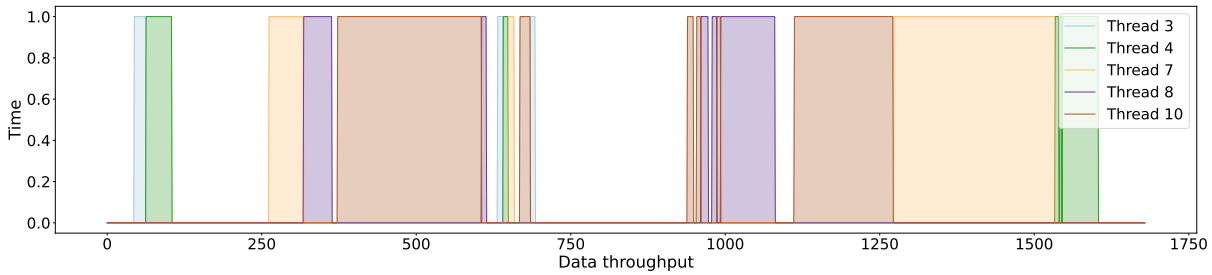
Figure 7: Sensitivity test

DKF Step Size Δ_k. In this experiment, we investigate the sensitivity of the step size Δ_k in Eq. (4), which reflects the extent of linear accelerated motion modeled. When $\Delta_k = 0$, the state parameters follow a random walk and the model in Eq. (3) will recover to the linear model Eq. (2). We evaluate how different values of Δ_k affect the number of successful tasks of the algorithm. *Result:* We test the number of successful tasks under different Δ_k values, both in the pure tasks scenario and the mixed tasks scenario, as shown in Fig. 7(a). In both scenarios, a marked decline is observed at extreme Δ_k values, and the results remain stable within a specific range of Δ_k values.

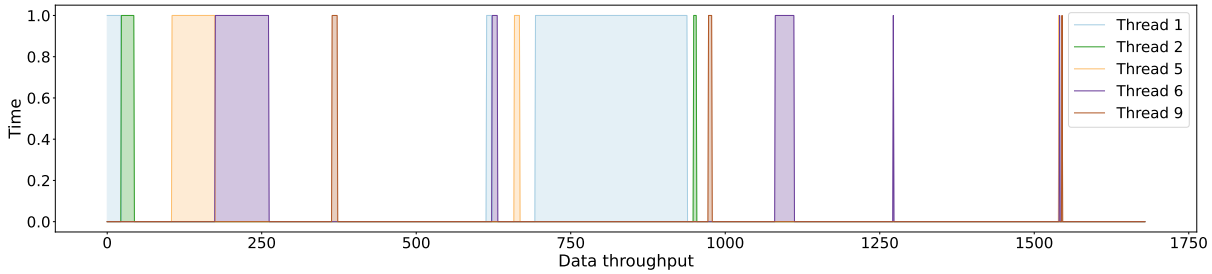
WLS Discount Factor γ. In this experiment, we investigate the sensitivity of WLS discount factor γ in Eq. (9), which reflects the prediction weight of historical observed data. We evaluate how different values of γ affect the number of successful tasks of the algorithm. *Result:* We test the number of successful tasks under different γ values, both in the pure tasks scenario and the mixed tasks scenario, as shown in Fig. 7(b). In both scenarios, a marked decline is observed at large γ values, indicating that for short-term prediction, overweighting older historical data is detrimental, supporting the non-stationary assumption. Since WLS only predicts the training loss within time slice M_τ (usually a relatively small value), experimental results remain stable even under a relatively small γ value.

Allocation process

In this experiment, we demonstrate the execution process of LaiRA method in the pure and mixed task bundle scenario, focusing on the data throughput $\eta_{k,t}$ (defined in Problem Formulation section).



(a) Pure feasible tasks data throughput



(b) Pure infeasible tasks data throughput

Figure 8: Resource allocation process for pure task bundle

Pure task bundle. Figures 8 illustrate the allocation trajectories of $\eta_{k,t}$ for both feasible and infeasible tasks. In particular, Figure 8(b) demonstrates that LaiRA is able to accurately identify infeasible tasks, such as those assigned to Tasks 2, 5, 6, and 9, shortly after the initial exploration phase. Once these tasks are recognized as infeasible, LaiRA promptly reduces their allocated computing resources to a minimum.

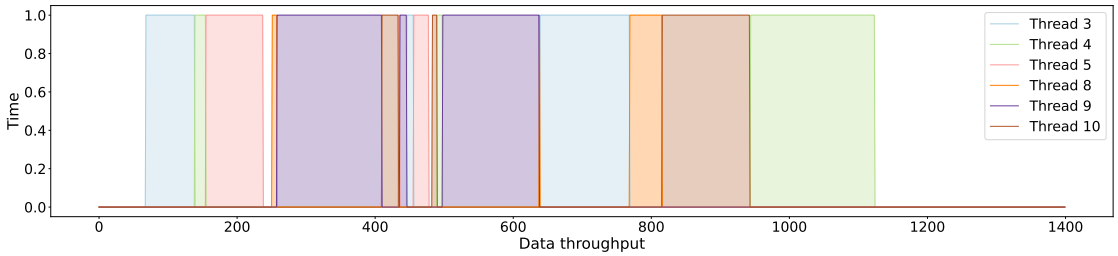
Mixed task bundle. Figure 9 presents the allocation trajectories of $\eta_{k,t}$ for both feasible and infeasible tasks. As illustrated in Figure 9(b), LaiRA is able to promptly identify infeasible tasks, specifically those handled by Tasks 1, 2, 6, and 7, shortly after the initial exploration phase. Once these tasks are recognized as infeasible, LaiRA allocates only minimal computing resources to them for the remainder of the process.

Tracking the Switch Count of LaiRA and LaiRA without UCB

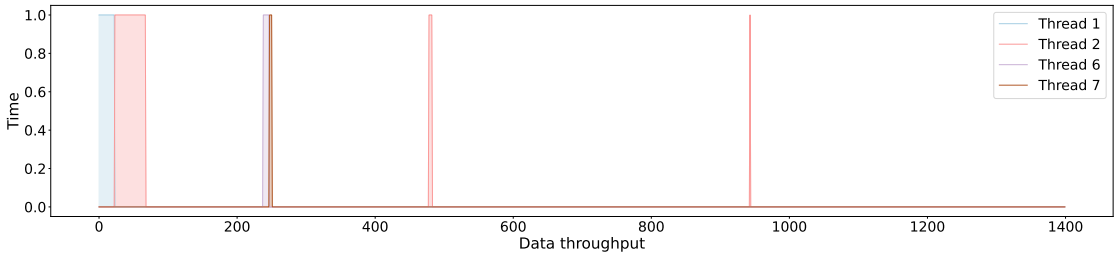
In this section, we provide the real-time priority score changes of LaiRA and LaiRA without UCB, to verify that LaiRA is more stable than LaiRA without UCB, and support our claim in Ablation Study section.

Settings: Specifically, LaiRA using the score in (12) and LaiRA without UCB using a variant score from (12), such that directly using $(\ln \ell_k(s_\tau^k) - \langle \widehat{X}_{\tau+1}^k, \widehat{\theta}_{n_\tau^k}^k \rangle)/(\ln(s_\tau^k + M_\tau \cdot N_k) - \ln(s_\tau^k))$ as the score and remove the term with β_n.

Results: Figure 10 and 11 illustrate the real-time priority score dynamics of LaiRA and LaiRA without UCB under pure and mixed task bundle scenarios, respectively. As shown in Figure 10(a), the priority computed by LaiRA remains highly stable over time, with very few changes in the highest-priority task. In contrast, Figure 10(b) demonstrates that LaiRA without UCB results in highly unstable priority estimations, frequent switches in the top-priority task occur, especially during the early exploration phase. This difference is further quantified in Figure 10(c), which compares the number of switches in the top-priority task between the two methods. LaiRA without UCB exhibits significantly more switches than LaiRA, indicating that the absence

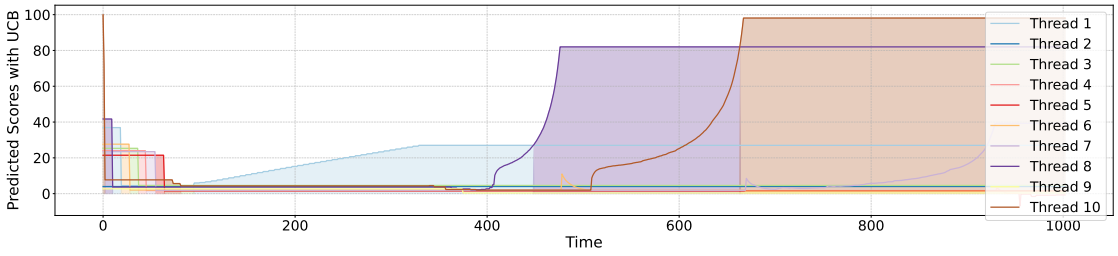


(a) Mixed feasible tasks data throughput

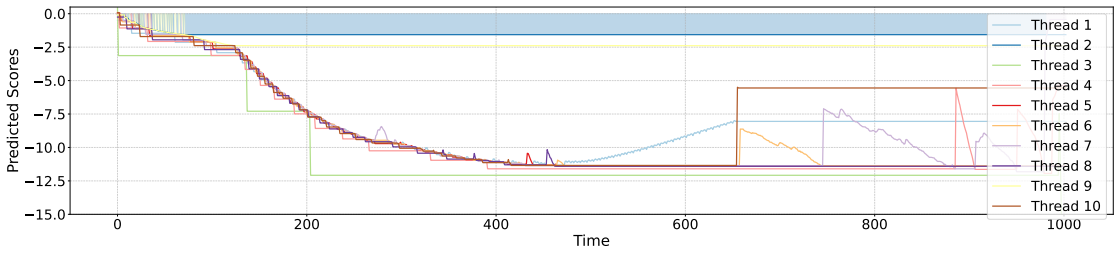


(b) Mixed infeasible tasks data throughput

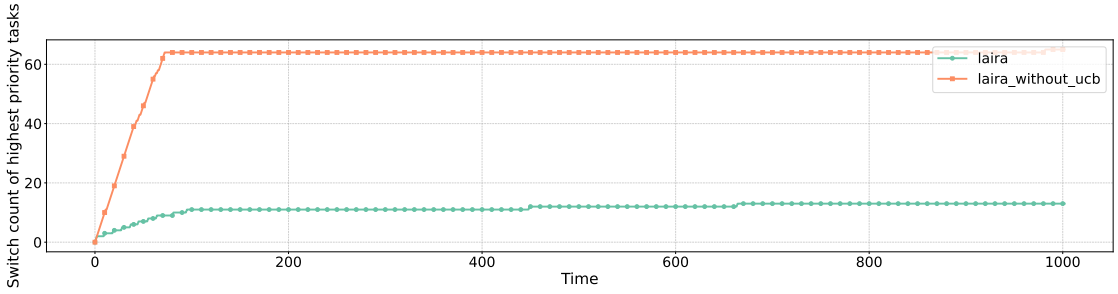
Figure 9: Resource allocation process for mixed task bundle.



(a) Predicted scores with UCB

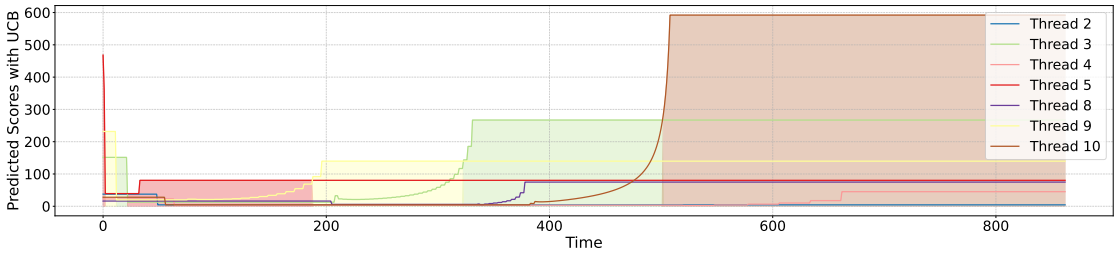


(b) Predicted scores without UCB

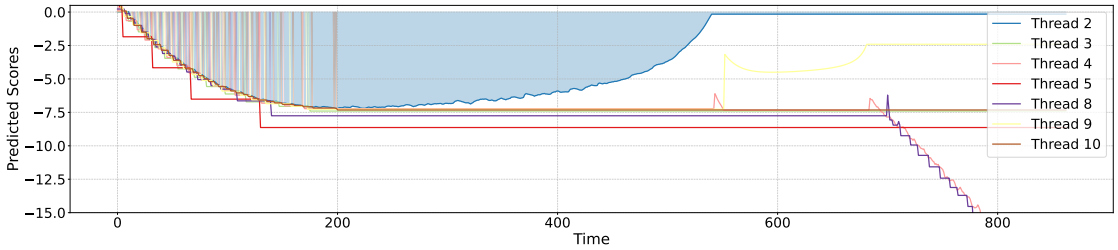


(c) Switch count of highest priority task

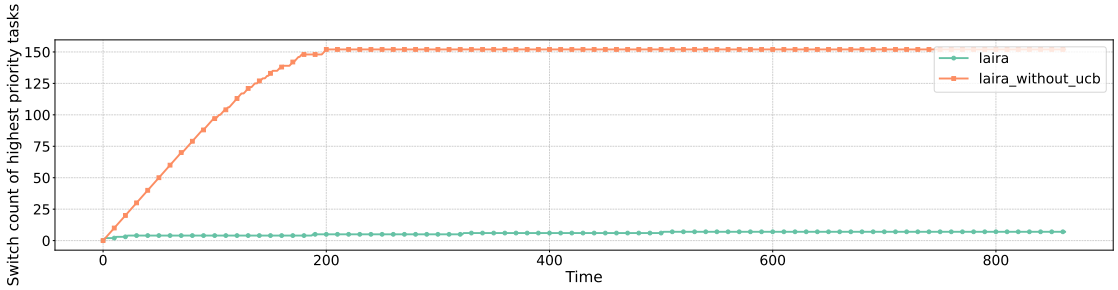
Figure 10: UCB comparison for pure task bundle



(a) Predicted scores with UCB



(b) Predicted scores without UCB



(c) Switch count of highest priority task

Figure 11: UCB comparison for mixed task bundle

of UCB leads to less consistent decision-making during scheduling. Figure 11 shows similar results for the mixed task bundle scenario. Note that the priority scores for LaiRA without UCB become negative in both pure and mixed task bundle scenarios. This occurs because the true linearized loss curve is concave, bending downward over time as shown in Figure 6(a), and the prediction based on historical data is linear and thus lies above the actual curve, leading to consistently higher predicted losses and consequently negative priority scores. However, this does not affect the final scheduling performance, as the algorithm relies on the relative ordering of the scores rather than their absolute values. Overall, these results validate our explanation in Ablation Study section, confirming that UCB not only improves performance but also has the ability to reduce the switching cost.