Online Resource Allocation for Edge Intelligence with Colocated Model Retraining and Inference

Huaiguang Cai, Zhi Zhou, Qianyi Huang
School of Computer Science and Engineering, Sun Yat-Sen University, China
Email: caihg3@mail2.sysu.edu.cn, zhouzhi9@mail.sysu.edu.cn, huangqy89@mail.sysu.edu.cn

***Abstract*—With edge intelligence, AI models are increasingly pushed to the edge to serve ubiquitous users. However, due to the drift of model, data, and task, AI model deployed at the edge suffers from degraded accuracy in the inference serving phase. Model retraining handles such drifts by periodically retraining the model with newly arrived data. When colocating model retraining and model inference serving for the same model on resource-limited edge servers, a fundamental challenge arises in balancing the resource allocation for model retraining and inference, aiming to maximize long-term inference accuracy. This problem is particularly difficult due to the underlying mathematical formulation being time-coupled, non-convex, and NP-hard. To address these challenges, we introduce a lightweight and explainable online approximation algorithm, named ORRIC, designed to optimize resource allocation for adaptively balancing the accuracy of model training and inference. The competitive ratio of ORRIC outperforms that of the traditional Inference-Only paradigm, especially when data drift persists for a sufficiently lengthy time. This highlights the advantages and applicable scenarios of colocating model retraining and inference. Notably, ORRIC can be translated into several heuristic algorithms for different resource environments. Experiments conducted in real scenarios validate the effectiveness of ORRIC.**

I. Introduction

Edge intelligence, the marriage of AI and edge computing, promises to provide ubiquitous users with low-latency, energy-efficient, and privacy-protecting machine learning services by processing data in proximity [1]. However, various types of drift reduce the accuracy of machine learning models in practice, and even worse in edge scenarios when computing resources are limited. Specifically, we classify these drifts into three types: 1) *model drift*: the distribution of model parameters is changed after deployment (e.g., model compression). 2) *data drift*: the distribution of features or labels shift over time (e.g., domain adaptation, test-time adaptation [2]). 3) *task drift*: the model may be applied to perform unseen tasks (e.g., fine tuning [3], embodied AI [4]).

Numerous methods have been proposed to alleviate these drifts, including retraining the deployed model [2], [3], [5], [6] or modifying inference results based on certain data distribution assumptions [7], [8]. However, these methods, mainly proposed by researchers from the machine learning community, tend to emphasize accuracy while overlooking resource consumption. Fortunately, in the field of edge computing, significant research such as Ekya [9], RECL [10] and Shoggoth [11], has been proposed to handle drifts by navigating the trade-off between the tasks of model retraining and model inference under the constraints of limited edge resources. Here, we define the scheme of **retraining the model and performing inference simultaneously on new data** as the *model retraining and inference co-location* paradigm.

Nevertheless, the absence of formal modeling and the reliance on heuristic algorithms in these previous works limit our understanding of the model retraining and inference co-location paradigm. Modeling this paradigm not only provides insights into its advantages and application scenarios but also aids in designing more rational and explainable algorithms that may enjoy better performance and theoretical guarantees.

Intuitively, due to limited edge resources, the task of retraining the model on new data and the task of performing inference on new data form a competitive relationship. If there are more retraining resources currently assigned, the current inference accuracy is low and the future accuracy is high; on the contrary, if the retraining resources are currently assigned less, the current inference accuracy is high but the future accuracy is low. Then a central question arises:

How can resources be credibly allocated for model retraining and inference co-location to optimize long-term model performance under various drifts?

To answer this question, our work makes the following contributions:

1) We provide a natural modeling of the model retraining and inference co-location paradigm and demonstrate a corresponding typical and practical system (Section III).
2) We design a lightweight and explainable algorithm ORRIC (Section IV) for the paradigm. The proved competitive ratio of ORRIC is strictly better than that of the traditional Inference-Only paradigm when data drift occurs for a sufficiently lengthy time, implying the advantages and application scenarios of model retraining and inference co-location paradigm (Section V).
3) Our experimental results of ORRIC on CIFAR-10-C validate the effectiveness of model retraining and inference co-location in drift scenarios (Section VI). Our code is available at https://github.com/caihuaiguang/ORRIC.

This work was supported by the National Science Foundation of China under Grant 62172454, the Guangdong Basic and Applied Basic Research Foundation under Grant 2023B1515020120. The corresponding author is Zhi Zhou.

II. Background and Related Works

We motivate our work with prior studies on 1) drifts in machine learning and 2) inference and retraining configuration adapting in edge computing.

A. Drift in Machine Learning

The basic process of machine learning is to collect a large amount of data for a task and then use the data to train a machine learning model. However, in practice, the model, data, and task may change after the deployment of the model. We categorize the inconsistency between the training phase and inference phase as model drift, data drift, and task drift.

1) Model drift: DNN compression [12] is commonly adopted for lower latency and improved energy efficiency [1]. However, the distribution of model parameters is changed [13] after compression, usually leading to a decrease in the accuracy of model. We classify this inconsistency between the model training phase and the inference phase as model drift. Even though model performance on training data remains the same after the compression, the compressed model has less generalization power on unseen data [14], necessitating model retraining [9].

2) Data Drift: This type of drift represents a shift in the distribution of features or labels. Specifically, let X denote the feature vector and y denote the label. We use $P_t(X, y)$ to represent the joint probability density function of X, y at time t. Concept drift [6] occurs when there exists a time t such that $P_t(X, y) \neq P_{t+1}(X, y)$. Label shift [7] occurs when there exists a time t such that $P_t(y) \neq P_{t+1}(y)$ but for all t, $P_t(X|y) = P_{t+1}(X|y)$. In a broader sense, we classify any shift in $P_t(X)$, $P_t(y)$, $P_t(X|y)$, $P_t(y|X)$, $P_t(X, y)$ (such as concept drift, label shift, domain adaptation, or test-time adaptation [2]) as data drift. In real-world scenarios, the proportion of pedestrians, cars, and bicycles may vary throughout the day [9], corresponding to label shift ($P_t(y)$ shift). Another common phenomenon is that, due to variations in angles, weather conditions, lighting, and sensors [9], the appearance of the same class of objects ($P_t(X|y)$) may differ from that during training, while the true label of the object ($P_t(y)$) remains unchanged, corresponding to concept drift.

Several methods, such as the unbiased risk estimator [7] and the online ensemble algorithm [5], can alleviate the negative effect of data drift on model performance without retraining based on assumptions on data distribution, such as label shift [7] or gradual data evolution [8]. However, these methods are heavily dependent on the assumed type of drift and may not be universally applicable. As a result, retraining the model [2] remains a mainstream approach [5].

3) Task drift: This drift encompasses changes in tasks during both the training and inference phases, including meta-learning [15], continual learning [2], transfer learning [5], and fine-tuning [3]. Additionally, embodied AI has gained considerable attention recently, necessitating models to learn through interactions with the real world [4]. All these studies expect the model to perform well on new tasks, and retraining the model is nearly the only viable approach.

Motivated by the extensive attention to drift problems in the machine learning research community and the prevalent use of model retraining, we seek to formulate the model performance under these drifts. Moreover, in contrast to prior works that concentrate solely on model accuracy, our approach considers both model accuracy and the computational cost of model retraining. This makes it more suitable for practical deployment in resource-constrained edge environments.

B. Inference and Training Configuration Adaption

Resources provisioned for edge computing are limited [9], motivating research on reducing resource consumption for model retraining or inference on edge while meeting basic accuracy requirements, known as model inference or retraining configuration adaptation.

Inference Configuration Adaptation refers to adapting the content of the inference request or the model used, such as the frame rate of the input video, the resolution of input images [16], the type of model [17], or the extent of early exiting in model inference [18]. This adaptation, influencing the corresponding output of the model, is typically determined by the available computing, storage, bandwidth resources, and the difficulty level of the input [16].

Training Configuration Adaption refers to adapting the hyperparameters of the training, such as epochs, training data size [19], or the layers performing back-propagation [2]. This adaptation, influencing the model itself, is typically determined by available computing, storage, bandwidth resources, and the performance of the model being used [9].

Although some studies such as Ekya [9] and RECL [10] have explored the trade-off between the tasks of model retraining and model inference under the constraints of limited edge resources, they lack the formal modeling of the model retraining and inference co-location paradigm, and the employed algorithms are heuristic. Differing from these works, we aim to gain a deeper understanding of the paradigm and design a more rational and explainable algorithm by formally modeling the paradigm and proposing a theoretically guaranteed algorithm.
Remark: Existing researches on model retraining and inference co-location typically deploy the model on edge [9] or cloud [10]. However, we argue that with hardware upgrades [20] and technological advances [21], model retraining and inference co-location on devices holds promise for enhanced privacy protection, reduced bandwidth usage and personalized AI models. While some existing frameworks support on-device model training, such as MNN [22], nntrainer [23], TensorFlow Lite [24], PyTorch Mobile [25], their documentation is not comprehensive, and some lack regular maintenance. We call for further efforts in this direction.

III. System Model and Problem Formulation for Model Retraining and Inference Co-location

In this section, we present the system model for model retraining and inference colocation in an edge server, and the problem formulation for the dynamical resource allocation to maximize the long-term inference accuracy.

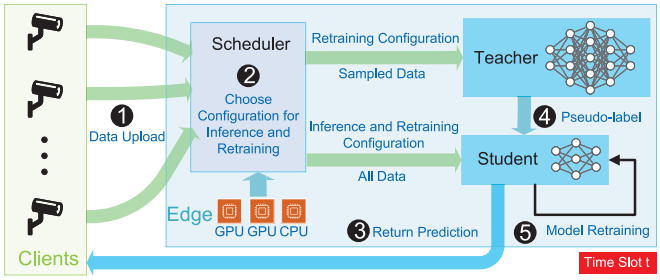


Fig. 1: Model Retraining and Inference Co-location.

A. System Overview

Following the pilot effort of Ekya [9], we adopt a general system architecture as illustrated in Fig. 1 for edge intelligence with colocated model training and inference. In this architecture, an edge server equipped with moderate CPU and GPU resources simultaneously performs model retraining and inference serving, with the new data stream (or inference requests) collected from a set of nearby device clients (e.g., surveillance cameras) running the same AI application (e.g., object detection). Since manual labeling for the online data stream is not feasible at the edge, the labels for the retraining are obtained from a "teacher model" — a highly accurate but expensive model (with deeper architecture and larger size). Since the inference latency of the teacher model cannot meet the stringent latency requirements of mission-critical edge AI applications such as safety surveillance, we only use it for labeling. Instead, for the actual inference serving and model retraining, a "student model" which is less accurate but more responsive and resource-efficient is adopted. Notably, this philosophy of supervising a small student model with a large teacher model has been widely applied in the community of computer vision.

To model the periodic behavior of model retraining, we assume that the system works in a time-slotted fashion. Each time slot $t \in \mathcal{T} \triangleq \{1, 2, \cdots, T\}$ denotes a "retraining window" that designates model retraining once on the newly collected data. Specifically, as shown in Fig. 1, at each time slot t, the group of device clients first ❶ upload their inference requests to the edge server. Here we use $D_{(t)}$ to denote the amount of data uploaded at time slot t. Without loss of generality, we assume that $D_{(t)} \in [D_{\min}, D_{\max}]$ and $D_{\min} > 0$. After receiving the data, the scheduler ❷ determines the configuration for model retraining and inference, based on the data amount ($D_{(t)}$) and the available computational resource at the edge server (denoted as $C_{(t)}$). Afterward, the student model will immediately ❸ return predictions of all inference requests to the clients based on inference configuration from the scheduler. Next, some uniformly random-chosen data according to the retraining configuration will be sent to the teacher model to ❹ get the corresponding high-credit labels (or pseudo-labels). The student model ❺ then updates its weight by retraining the model according to the pseudo-labels and the retraining configuration determined by the scheduler. Then in the next time slot $t+1$, the student model can serve the inference request with retrained model weights, thus improving accuracy.

TABLE I: Notations.

Notation	Description
$C_{(t)}$	the available computational resource in time slot t.
$D_{(t)}$	the amount of uploaded data at the beginning of t.
A_i^T, C_i^T	the profit and resource consumption of i-th retraining configuration.
A_j^I, C_j^I	the profit and resource consumption of j-th inference configuration.
$x_i(t)$	binary variable indicating whether i-th retraining configuration is chosen at time slot t.
$y_j(t)$	binary variable indicating whether j-th inference configuration is chosen at time slot t.

B. Resource Allocation Model

When colocating model retraining and inference serving at the edge server, they may compete for the limited computational resource such as CPU and GPU, especially when the data arrival $D_{(t)}$ bursts. Therefore, the resource allocation to model retraining and inference serving faces a fundamental tradeoff between the retrained model's accuracy and the inference accuracy. Specifically, if we allocate more resource to model retraining to improve its accuracy, the accuracy of the current model inference would diminish due to reduced resource allocation. Vice versa, if we take away resource from model retraining to inference serving, the current inference accuracy would increase but the subsequent inference may decrease due to the reduced accuracy of the retrained model.

The knob to navigate the tradeoff between the retrained model's accuracy and the inference accuracy is the configuration of both model retraining and inference, which controls the resource-accuracy tradeoff of both model retraining and inference. For model retraining, the configuration refers to the hyperparameters of the training, such as the number of epochs, training data size [19], or the layers performing back-propagation [2]. For these hyperparameters, a larger value results in higher accuracy, but also at cost of more resource demand. For model inference, the configuration includes the hyperparameters such as frame rate/resolution of the input video, the compressed variant of the model [17], or the early exit point of the branchy model [18].

1) Retraining configuration adaption: In each time slot t, the scheduler selects one retraining configuration from the set of feasible configurations $\mathcal{M} \triangleq \{1, 2, \cdots, M\}$. This selection is represented by binary variables $x_i(t) \in \{0, 1\}$, where $x_i(t) = 1$ indicates the i-th retraining configuration is selected at time slot t. Formally, this can be expressed as:

$$x_i(t) \in \{0, 1\}, \quad \forall i \in \mathcal{M}, \ \forall t \in \mathcal{T}, \tag{1}$$

$$\sum_{i=1}^{M} x_i(t) = 1, \quad \forall t \in \mathcal{T}. \tag{2}$$

To characterize the resource-accuracy of the i-th retraining configuration, we use C_i^T to denote the resource demand (per data sample, measured by FLOPs or MACs) and A_i^T to aid in modeling the tested accuracy. Given an amount of $D_{(t)}$ data samples at time slot t, the total amount of resource demand of model retraining (including pseudo-labeling) is $D_{(t)} C_i^T$.

2) Inference configuration adaption: similar to the retraining configuration, in each time slot t, the scheduler selects one inference configuration from the set $\mathcal{N} \triangleq \{1, 2, \cdots, N\}$. This selection is represented by binary variables $y_j(t) \in \{0, 1\}$, where $y_j(t) = 1$ indicates that the j-th inference configuration is selected at time slot t. Formally, we have:

$$y_j(t) \in \{0, 1\}, \quad \forall j \in \mathcal{N}, \ \forall t \in \mathcal{T}, \tag{3}$$

$$\sum_{j=1}^{N} y_j(t) = 1, \quad \forall t \in \mathcal{T}. \tag{4}$$

The j-th inference configuration consumes $D_{(t)}C_j^I$ computational resources (measured by FLOPs or MACs). The profit, $D_{(t)}A_j^I$, is the corresponding j-th result of normalizing the model accuracy for all N inference configurations using the maximum value as a reference. Both $D_{(t)}C_j^I$ and $D_{(t)}A_j^I$ are easy to calculate in practice. In our experiments, the MACs and accuracy of different inference configurations on the test dataset, whose distribution is the same as the training dataset, are used to represent $D_{(t)}C_j^I$ and $D_{(t)}A_j^I$.

Let $A_{\min}^T$ and $A_{\max}^T$ denote the minimum and maximum of the set $\{A_i^T \mid \forall i \in \mathcal{M}\}$. Similarly, let $A_{\min}^I$ and $A_{\max}^I$ represent the minimum and maximum of the set $\{A_j^I \mid \forall j \in \mathcal{N}\}$. Then $A_{\min}^T = 0$ and $A_{\min}^I > 0$ according to a natural assumption that if the computational resources are scarce and for the consideration of satisfaction from users, model retraining is not unnecessary compared with model inference.

3) Computational resources constraint: We use $C_{(t)}$ to denote the available computational resource at time slot t. And we suppose that there is at least one feasible solution to the following inequality, regardless of the value of $D_{(t)}$ and $C_{(t)}$:

$$D_{(t)} \sum_{i=1}^{M} C_i^T x_i(t) + D_{(t)} \sum_{j=1}^{N} C_j^I y_j(t) \leq C_{(t)}, \ \forall t \in \mathcal{T}. \tag{5}$$

C. Long-term Accuracy Model

We model the basic model performance at time slot t as $f\left(\frac{\sum_{\tau=1}^{t-1} D_{(\tau)} \sum_{i=1}^{M} x_i(\tau) A_i^T}{\sum_{\tau=1}^{t-1} D_{(\tau)}}\right)$. Our modeling of model performance under various drifts is based on two key observations: (1) Irrespective of the type of drift, model performance declines to a minimum gradually if the model is not retrained regularly. (2) The increase in model performance resulting from training exhibits a diminishing marginal effect [26].

To incorporate the first observation into our modeling, it is essential to explore the correlation between current testing data and the previous data used to retrain the model. However, precisely determining the relationship between previous data used for retraining and current testing data is usually challenging or maybe compute-intensive. So similar to the maximum entropy principle, we make the following assumption: **every previously used retraining configuration has the same effect on current model performance**. That is where $\frac{\sum_{\tau=1}^{t-1} D_{(\tau)} \sum_{i=1}^{M} x_i(\tau) A_i^T}{\sum_{\tau=1}^{t-1} D_{(\tau)}}$ comes from. Then to align with the second observation, we introduce a function f that maps the average learning extent of all past data to the current model performance to represent the average influence of the drifts on model performance over time. If there are no drifts, wherein the current test data follows the same distribution as the training dataset on which the model has been fully trained, then model retraining has a small influence on model performance and f reduces to a constant function.

In the study of learning curves [26], the expression of the function f can take on various forms, such as power functions like $f(x) = c - ax^{-\alpha}$, exponential functions like $f(x) = \exp(a + \frac{b}{x} + c\log(x))$, logarithmic functions like $f(x) = \log(a\log(x) + b)$, or even a weighted linear combination of these forms. Here, x represents training time, number of iterations, or training dataset size, and $f(x)$ denotes accuracy on the validation set. For a more comprehensive understanding of the potential expressions of the function f, please refer to Figure 1 in [26]. In our study, rather than making assumptions about the exact expression of f, we identify that these expressions share a common property: they are concave and increasing. Consequently, we introduce the following assumption about f:

Assumption 1. *The function $f(x)$ is increasing, concave, and continuously differentiable over the interval $[0, A_{\max}^T]$, and $f(0) > 0$, but its analytical expression is unknown.*

Other assumptions about the relationship between retraining configuration and model performance may be reasonable as well. For instance, if the current model performance is only related to past data within a time window (e.g., in-context learning), then model performance can be modeled as $f\left(\frac{\sum_{\tau=t-w}^{t-1} D_{(\tau)} \sum_{i=1}^{M} x_i(\tau) A_i^T}{\sum_{\tau=1}^{t-1} D_{(\tau)}}\right)$, where w is the window size. And if the current data is more related to nearby data than former data, model performance would be $f\left(\frac{\sum_{\tau=1}^{t-1} D_{(\tau)} \alpha^{t-1-\tau} \sum_{i=1}^{M} x_i(\tau) A_i^T}{\sum_{\tau=1}^{t-1} \alpha^{t-1-\tau} D_{(\tau)}}\right)$, where α is a positive decay factor less than 1. A more complex modeling is that the formula of f may change over time, i.e., $f_t\left(\frac{\sum_{\tau=1}^{t-1} D_{(\tau)} \sum_{i=1}^{M} x_i(\tau) A_i^T}{\sum_{\tau=1}^{t-1} D_{(\tau)}}\right)$. We leave these variants for future research.

In the presence of an unknown analytical formula for the function f, we introduce the following assumptions to facilitate the algorithm design:

Assumption 2. *The value of $f(A_{\max}^T)$ is known. And a positive lower bound of $f'(A_{\max}^T)$, denoted as L, is known.*

In practice, the accuracy of the trained model on the test dataset with the same distribution as the training dataset serves as an estimate for $f(A_{\max}^T)$. This is because this accuracy indicates the model's performance when drifts are absent, effectively representing the highest achievable accuracy ($f(A_{\max}^T)$) under the best retraining and inference configurations in the presence of drifts. The value of $f'(A_{\max}^T)$, reflecting the rate of improvement in model accuracy when the model always uses the best retraining configuration under drifts, can only be determined with prior knowledge of the drifts. In our experiments, we set a small constant (e.g., 0.01) as the value

of L based on the accuracy improvement on the mentioned test dataset between the last two epochs of the training process. For a typical task, the optimal L can be determined empirically by experimenting with the algorithm with various values of L in the real world. Approximating an unknown value (L) is much simpler than approximating an unknown function (f).

Moreover, the inference configuration, viewed as the utilization of the model, also plays a significant role in determining its performance. We assume that **model performance at time slot t is directly proportional to the profit of the inference configuration used at that time**, i.e., $\sum_{j=1}^{N} y_j(t) A_j^I D_{(t)}$. While output accuracy with different inference configurations may vary over time, we argue that A_j^I represents the average utilization degree of the j-th inference configuration on the model and is assumed to be a constant known in advance.

Now, the problem of maximizing long-term average accuracy within the constraints of varying computing resources over time, with decision variables $(x_i(t), y_j(t))$ representing the chosen retraining and inference configurations, is formulated as:

$$\max_{x_i(t),y_j(t)} \sum_{t=1}^{T} f\left(\frac{\sum_{\tau=1}^{t-1} D_{(\tau)} \sum_{i=1}^{M} x_i(\tau) A_i^T}{\sum_{\tau=1}^{t-1} D_{(\tau)}}\right) \sum_{j=1}^{N} y_j(t) A_j^I D_{(t)} \tag{P}$$

$$\text{s.t. Constraints } (1)-(5).$$

D. Existing Approaches

To the best of our knowledge, no online algorithms for a similar problem (P) have been proposed in the literature so far. There are three main difficulties when dealing with it: (1) The objective function is nonconvex-nonconcave, as demonstrated in Theorem 1 with its proof in Appendix A. (2) Decision variables are heavily coupled. (3) The analytical formula for f is commonly unknown in practice.

Theorem 1. *If $f(x)$ is a concave function and defined on $[0, A_{\max}^T]$, then $f(x)y$, defined on $[0, A_{\max}^T] \times [A_{\min}^I, A_{\max}^I]$, is a nonconvex-nonconcave function.*

For the third difficulty, a promising technique is *bandit convex optimization* [27]. The method usually adds random noise to the decision variable and then estimates the gradient information [28] of the function, but these techniques are specifically designed for convex functions and may not be readily applicable to our situation. Moreover, the decision variables of our problem are discrete, and therefore, the technique of adding random noise is not suitable.

The well-known primal-dual [29] method in online algorithms is also not applicable to our problem. Even if we can approximate the analytical expression of f, it is difficult to find the dual function of the original function due to the heavy coupling between decision variables. Moreover, since the problem is nonconvex-nonconcave, there may not be strong duality as in linear programming. Then even if the dual function is found and solved, it may violate the original problem constraints. Review [30] has more details on the time-varying convex optimization algorithms.

IV. Algorithm Design

We follow a three-step procedure to design a lightweight and theoretically guaranteed algorithm: (i) Leverage the concave property of f to move the decision variables $\{x_i(t)\}$ out of f (Lemma 1). Then the problem of interest changes from (P) to (Q). (ii) Decouple the interaction of $\{x_i(t), y_j(t)\}$ by involving a particular regularization term (Lemma 3), with the concerned problem specialized to (D). Then ORRIC is proposed to solve (Dt), the subproblem of (D) in every time slot. (iii) Lemma 2 and Theorem 4 are used to facilitate the proof of the competitive ratio of ORRIC.

A. Deal with the Target Function of (P)

Lemma 1. *$f(x) \le Lx + g(A_{\max}^T)$, where $g(A_{\max}^T) = f(A_{\max}^T) - LA_{\max}^T$.*

Proof. $f(x) \le f(A_{\max}^T) + f'(A_{\max}^T)(x - A_{\max}^T)$ for the concave property of $f(x)$; Because $x \le A_{\max}^T$ and $f'(A_{\max}^T) \ge L > 0$, then $f'(A_{\max}^T)(x - A_{\max}^T) \le L(x - A_{\max}^T)$, and thus $f(x) \le Lx + f(A_{\max}^T) - LA_{\max}^T \le Lx + g(A_{\max}^T)$. □

We use z_{t-1} to denote $\sum_{\tau=1}^{t-1} D_{(\tau)} \sum_{i=1}^{M} x_i(\tau) A_i^T$, and then transform the objective function using Lemma 1 for $1 \le t \le T$. Then we have $f\left(\frac{z_{t-1}}{\sum_{\tau=1}^{t-1} D_{(\tau)}}\right) \le g(A_{\max}^T) + \frac{Lz_{t-1}}{\sum_{\tau=1}^{t-1} D_{(\tau)}}$. To further deal with z_t, we introduce the positive regularization term $\lambda_t(\sum_{\tau=1}^{t} D_{(\tau)} \sum_{i=1}^{M} x_i(\tau) A_i^T - z_t)$, where $\lambda_t > 0, \forall t$. With this, the new problem (Q) is formulated as follows:

$$\begin{aligned}\max_{x_i(t),y_j(t),z_t} \quad & \sum_{t=1}^{T} g\left(A_{\max}^T\right) \sum_{j=1}^{N} y_j(t) A_j^I D_{(t)} \\ & + \sum_{t=2}^{T} L \frac{z_{t-1}}{\sum_{\tau=1}^{t-1} D_{(\tau)}} \sum_{j=1}^{N} y_j(t) A_j^I D_{(t)} \\ & + \sum_{t=1}^{T-1} \lambda_t \Big(\sum_{\tau=1}^{t} D_{(\tau)} \sum_{i=1}^{M} x_i(\tau) A_i^T - z_t\Big) \qquad \text{(Q)} \\ \text{s.t. } & \text{Constraints } (1)-(5), \\ & z_t \le \sum_{\tau=1}^{t} D_{(\tau)} A_{\max}^T, \quad 1 \le t \le T-1.\end{aligned}$$

Using P^* and Q^* to respectively denote the optimal offline objective function value of the problems (P) and (Q), then the following lemma holds.

Lemma 2. *Suppose $\lambda_t > 0$ for $1 \le t \le T-1$, then $P^* \le Q^*$.*

Proof. We can prove this by demonstrating that the optimal solution to (P) is also a feasible solution for (Q). Let $x_i^*(t)$ and $y_j^*(t)$ be the optimal solutions to (P), and define $z_t = \sum_{\tau=1}^{t} D_{(\tau)} \sum_{i=1}^{M} x_i^*(\tau) A_i^T$. It follows that the optimal solution of (P) satisfies the constraints of (Q), and the objective function value of (P) is less than that of (Q) for the optimal solution of (P) based on the former inequalities. □

We choose a particular realization of λ_t to simplify (Q).

Lemma 3. *when $\lambda_t = L \frac{D_{\min} A_{\min}^I}{t D_{\max}}$, then the corresponding z_t to the optimal solution of (Q) is $\sum_{\tau=1}^{t} D_{(\tau)} A_{\max}^T$*

Proof. Extracting all terms containing z_t in (Q), we have: $\sum_{t=2}^{T} L\frac{z_{t-1}}{\sum_{\tau=1}^{t-1} D_{(\tau)}} \sum_{j=1}^{N} y_j(t) A_j^I D_{(t)} - \sum_{t=1}^{T-1} \lambda_t z_t = \sum_{t=1}^{T-1} \left[L\frac{1}{\sum_{\tau=1}^{t} D_{(\tau)}} \sum_{j=1}^{N} y_j(t+1) A_j^I D_{(t+1)} - \lambda_t \right] z_t$. When $\lambda_t = L\frac{A_{\min}^I D_{\min}}{t D_{\max}}$, the coefficient of z_t is no less than 0, regardless of the values of $y_j(t+1)$ and $D_{(t+1)}$. To maximize (Q), z_t will equal its maximum: $\sum_{\tau=1}^{t} D_{(\tau)} A_{\max}^T$. □

After setting $\lambda_t = L\frac{D_{\min} A_{\min}^I}{t D_{\max}}$ and $z_t = \sum_{\tau=1}^{t} D_{(\tau)} A_{\max}^T$ based on Lemma 3, we obtain a specialized version of (Q), and we also have $P^* \leq D^*$ by Lemma 2:

$$\max_{x_i(t), y_j(t)} \sum_{t=1}^{T-1} \lambda_t \Big(\sum_{\tau=1}^{t} D_{(\tau)} \sum_{i=1}^{M} x_i(\tau) A_i^T - \sum_{\tau=1}^{t} D_{(\tau)} A_{\max}^T \Big) + \sum_{t=1}^{T} g(A_{\max}^T) \sum_{j=1}^{N} y_j(t) A_j^I D_{(t)} + \sum_{t=2}^{T} L A_{\max}^T \sum_{j=1}^{N} y_j(t) A_j^I D_{(t)} \tag{D}$$

$$\mathbf{s.t.} \quad \text{Constraints } (1) - (5)$$

Then we make some equivalent transformations to the target function of (D) and decouple the problem to every time slot:

$$\max_{x_i(t), y_j(t)} V_t \sum_{i=1}^{M} x_i(t) A_i^T + W_t \sum_{j=1}^{N} y_j(t) A_j^I \tag{Dt}$$

$$\mathbf{s.t.} \quad \text{Constraints } (1) - (5) \text{ only at } t \ .$$

where $V_t = L\frac{D_{\min} A_{\min}^I}{D_{\max}} \left(\sum_{\tau=t}^{T-1} \frac{1}{\tau} \right)$, $W_1 = f\left(A_{\max}^T\right) - L A_{\max}^T$ and $W_t = f\left(A_{\max}^T\right), \forall t > 1$.

B. Online Robust Retraining and Inference Co-location

The problem (Dt) can be solved by an exhaustive method with $O(MN)$ complexity. To further speed up the solving, we introduce the following property, as in [16]:

Property 1. *$\forall a, b \in \mathcal{M}, C_a^T > C_b^T \Rightarrow A_a^T > A_b^T$; and $\forall a, b \in \mathcal{N}, C_a^I > C_b^I \Rightarrow A_a^I > A_b^I$.*

It should be noted that there are some infrequent cases where Property 1 is not satisfied, i.e., better performance can be achieved with fewer resources. For instance, when the image is corrupted by Gaussian noise, downsampling may improve model performance, as illustrated in Table III. Similarly, in model training, more iterations may not necessarily lead to better performance [9]. A practical solution [9] to this issue is regularly measuring the resource-performance profiles of different configurations after model deployment.

However, since we have assumed the resource requirements and profits of retraining and inference configurations remain constant throughout the whole time span $\mathcal{T}$ in III-B, configurations that consume more resources yet yield lower profits can be reasonably eliminated before running the algorithm for (Dt), ensuring satisfaction of Property 1. This is because any reasonable algorithm for (Dt) would not choose these configurations when there are better alternatives available—ones with equivalent or lower resource requirements but higher profits.

Based on Property 1, we propose ORRIC (Online Robust Retraining and Inference Co-location), outlined in Algorithm 1. The underlying principle of ORRIC is that the optimal configuration is likely the one about to violate the computational resource constraint. Thus, the optimal configuration can be identified by searching through configurations likely to exceed the computational resource constraint. The proof of the correctness of ORRIC can be found in Appendix C. The complexity of ORRIC is $O(M+N)$: During each iteration of the loop, either $i = i+1$ or $j = j-1$. i increases mostly to $M+1$ and j decreases mostly to 0, the total number of iterations in the loop must be no more than $M+N$.

Algorithm 1 ORRIC

Input: V_t, W_t, $U_t = \frac{C_{(t)}}{D_{(t)}}$ and four ascending lists: $\{A_i^T, i \in \mathcal{M}\}$, $\{A_j^I, j \in \mathcal{N}\}$, $\{C_i^T, i \in \mathcal{M}\}$, $\{C_j^I, j \in \mathcal{N}\}$.
Output: A pair of retraining and inference configurations.

```
Initialization: Set i = 1, j = N, i* = j* = K = 0.
while i ≤ M and j ≥ 1 do
    if C_i^T + C_j^I ≤ U_t then
        if V_t A_i^T + W_t A_j^I > K then
            i* = i; j* = j; K = V_t A_i^T + W_t A_j^I;
        i = i + 1;
    else
        j = j − 1;
return i*, j*;
```

In particular, ORRIC aligns with our intuition about the way to allocate limited computing resources to model retraining and inference to optimize long-term model performance. As depicted in Table II, ORRIC can be regarded as a combination of four heuristic algorithms, transitioning between them based on the duration of time and the availability of computing resources: 1) Knowledge-Distillation: The teacher model imparts knowledge to the student model without considering resource consumption. 2) Inference-Greedy: Prioritize using a higher configuration for inference and utilize the remaining resources for retraining. 3) Focus-Shift: Shift the focus from retraining to inference as time passes. 4) Inference-Only: This algorithm is actually the traditional computing paradigm that deploys a trained model and then performs inference.

When the computational resources are sufficient for the use of the best inference and retraining configuration, ORRIC converts to Knowledge-Distillation because $V_t \geq 0, W_t > 0, \forall t$. When resources are really scarce, e.g., $C_{(t)} = D_{(t)} C_{\min}^I$, ORRIC converts to Inference-Only because $C_{\min}^T = 0$ while $C_{\min}^I > 0$. When resources are limited (but not scarce) and T is large, ORRIC converts to Focus-Shift because $\sum_{\tau=t}^{T-1} \frac{1}{\tau} > \ln(T) - \ln(t)$ and $V_T = 0$. When resources are limited (but not scarce) and T is small, ORRIC converts to Inference-Greedy because W_t is a constant when $t > 1$ while V_t will decrease to 0 with the increasing of t.

The translation relationship between ORRIC and the four heuristic algorithms not only illustrates the rationality of ORRIC but also provides insights into the properties of algorithms designed for the model retraining and inference co-location paradigm. We believe that all rational algorithms for this

TABLE II: ORRIC and Several Heuristic Algorithms.

Resources \ T	Large	Small
Sufficient	Knowledge-Distillation	
Limited	Focus-Shift	Inference-Greedy
Scarce	Inference-Only	

paradigm should similarly translate to these four heuristic algorithms given specific conditions regarding time length and available computing resources, as illustrated in Table II.

Remark: Our algorithm is an open-loop algorithm that does not leverage feedback from the system. We acknowledge that it is possible to calculate the current accuracy of the student model ($f\left(\frac{\sum_{\tau=1}^{t-1} D_{(\tau)} \sum_{i=1}^{M} x_i(\tau) A_i^T}{\sum_{\tau=1}^{t-1} D_{(\tau)}}\right) \sum_{j=1}^{N} y_j(t) A_j^I D_{(t)}$) at every end of time slot t by considering the pseudo-labels output by the teacher model as the ground truth labels, but the relevant mathematical techniques used to incorporate such feedback into the design of algorithms for similar formulas as problem (P) are lacking in the existing literature. We leave the research on the closed-loop algorithm to the problem (P) as future work.

V. Performance Analysis

Definition 1. *For a maximization problem, the competitive ratio (or CR) c of algorithm ALG is defined as $c \leq \frac{ALG(I)}{OPT(I)}$ for every input I, where OPT represents the optimal offline algorithm with complete knowledge of future information.*

Definition 2. *For a maximization problem, the **tight competitive ratio** c of algorithm ALG is also a competitive ratio of algorithm ALG, and there is no $c' > c$ such that for every input I, $c' \leq \frac{ALG(I)}{OPT(I)}$.*

Theorem 2. *The CR of Inference-Only is $\frac{f(0)}{f(A_{\max}^T)}$.*

Proof. Denote $\{x_i^*(t), y_j^*(t)\}$ as the optimal offline solution to (P) and $\{x_i(t), y_j(t)\}$ as the solution given by Inference-Only. Then $P^* \leq \sum_{t=1}^{T} f(A_{\max}^T) \sum_{j=1}^{N} y_j^*(t) A_j^I D_{(t)} \leq \frac{f(A_{\max}^T)}{f(0)} \sum_{t=1}^{T} f(0) \sum_{j=1}^{N} y_j(t) A_j^I D_{(t)} = \frac{f(A_{\max}^T)}{f(0)} P$. □

Theorem 3. *An upper bound of the tight competitive ratio of Inference-Only is $\frac{Tf(0)}{f(0)+(T-1)f(A_{\max}^T)}$.*

Insight: The closer $f(0)$ and $f(A_{\max}^T)$ are, the closer the competitive ratio ($\frac{f(0)}{f(A_{\max}^T)}$) and the upper bound of the tight competitive ratio ($\frac{Tf(0)}{f(0)+(T-1)f(A_{\max}^T)}$) of Inference-Only are to 1. This implies that when the drift is very slight, Inference-Only approaches the optimal offline algorithm. The proof of Theorem 3 is provided in Appendix B.

Theorem 4. *The CR of ORRIC is $\frac{(1+\alpha)f(0)}{f(A_{\max}^T)}$ or $\frac{1}{\frac{f(A_{\max}^T)}{f(0)}-\alpha}$, where $\alpha = \frac{L A_{\max}^T D_{\min}^2 A_{\min}^I}{f(A_{\max}^T) D_{\max}^2 A_{\max}^I}$.*

Proof. The basic idea is that we have $P^* \leq D^*$ based on Lemma 2, and if we prove $D^* \leq \frac{1}{c} P$, then c is the competitive ratio. Details see Appendix D □

Insight: First, similarly to Theorem 2 and 3, if drift is slight, L is close to 0, and then ORRIC reduces to Inference-Only (which is an almost optimal algorithm in this case). Second, ORRIC relies on the precise estimation of the lower bound (L) of the degree of drift ($f'(A_{\max}^T)$). If the degree of drift is large, then L and V_t are large too if the estimation is precise, making the model pay more attention to retraining to get good future performance. When $f'(A_{\max}^T)$ is underestimated too much, the model pays more attention to inference, reducing its future performance due to a lack of retraining, consistent with term L in CR. Third, the term $\frac{D_{\min}^2}{D_{\max}^2}$ suggests that ORRIC performs better with less variability in input data size.

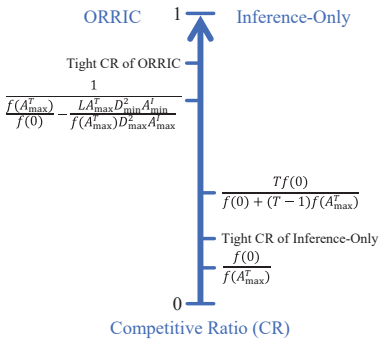


Fig. 2: Competitive Ratio Result.

Corollary 1. *When $T > \frac{f(A_{\max}^T)-f(0)}{\alpha f(0)}$, the tight competitive ratio of ORRIC is strictly better (bigger) than the tight competitive ratio of Inference-Only.*

Proof. Denote c_1, c_2 as the tight competitive ratio of ORRIC and Inference-Only. From Definition 2 and Theorem 4, we have $c_1 \geq \frac{1}{\frac{f(A_{\max}^T)}{f(0)}-\alpha}$. And when $T > \frac{f(A_{\max}^T)-f(0)}{\alpha f(0)}$, we have $\frac{1}{\frac{f(A_{\max}^T)}{f(0)}-\alpha} > \frac{Tf(0)}{f(0)+(T-1)f(A_{\max}^T)}$. Then, based on Theorem 3, we have $\frac{Tf(0)}{f(0)+(T-1)f(A_{\max}^T)} \geq c_2$. Besides, from Definition 2 and Theorem 2, we have $c_2 \geq \frac{f(0)}{f(A_{\max}^T)}$. We summarize our theoretical results in Fig. 2. □

Insight: Due to the rough approximation to the objective function of (P), ORRIC may not fully represent the potentiality of model retraining and inference co-location paradigm. However, the tight competitive ratio of ORRIC still surpasses that of Inference-Only when drift occurs ($L > 0$) for a sufficiently lengthy time ($T > \frac{f(A_{\max}^T)-f(0)}{\alpha f(0)}$). This implies that, in such scenarios, the worst-case performance of the model retraining and inference co-location paradigm is strictly better than that of the traditional Inference-Only paradigm.

VI. Experiments

We conduct the experiments to answer the following question: Can model retraining and inference co-location paradigm alleviate the negative effect of data drift on model performance?

A. Setup

CIFAR-10-C [31], a dataset that is generated by adding 15 common corruptions and 4 extra corruptions to the test dataset of CIFAR-10, is typically used in experiments of out-of-distribution generalization or continual test-time adaptation [32]. We treat these corruptions as imitations of data drift. We

TABLE III: Top-1 Accuracy (%) on CIFAR-10 and CIFAR-10-C.

Model (Resolution)	MACs (M)	Latency (μs)	original	brightness	contrast	defocus blur	elastic transform	fog	frost	gaussian blur	gaussian noise	glass blur	impulse noise	jpeg compression	motion blur	pixelate	saturate	shot noise	snow	spatter	speckle noise	zoom blur	Mean
MobileNetV2 (20*20)	6.35	7.54	44.93	42.60	23.28	40.47	39.25	27.64	39.46	38.41	42.97	40.33	41.35	42.95	35.84	43.02	38.14	43.29	39.29	41.38	43.08	41.69	39.18
MobileNetV2 (24*24)	6.71	8.37	59.38	54.41	28.09	51.26	49.94	37.42	48.49	48.08	55.71	50.18	53.04	57.10	44.07	55.91	50.56	56.77	50.42	55.41	56.78	49.96	50.19
MobileNetV2 (28*28)	7.45	10.15	73.29	67.94	38.33	63.21	62.48	49.68	59.17	59.23	**64.53**	62.21	**60.38**	69.31	53.91	69.67	63.68	**66.57**	61.70	67.33	**66.68**	61.22	61.43
MobileNetV2 (32*32)	7.94	10.51	**79.57**	**76.00**	**47.52**	**71.08**	**71.91**	**62.74**	**62.70**	**67.02**	56.28	**62.90**	57.38	**74.71**	**62.42**	**76.98**	**71.61**	61.98	**65.83**	**71.92**	62.86	**67.78**	**65.87**
ResNet50 (20*20)	65.76	17.41	54.50	49.20	32.26	50.71	49.00	39.31	44.19	48.99	52.23	49.99	49.99	53.04	45.79	53.03	46.06	52.95	45.68	48.92	52.81	52.83	48.26
ResNet50 (24*24)	68.96	19.29	71.95	66.25	40.68	62.58	61.52	50.54	60.49	58.75	68.26	62.61	64.58	69.64	54.60	68.51	62.09	69.10	61.03	64.42	68.98	61.99	61.93
ResNet50 (28*28)	82.01	24.08	79.02	74.19	42.74	66.58	66.79	55.34	66.95	61.60	72.89	68.07	**66.01**	75.72	56.96	75.12	69.01	74.45	69.11	70.41	74.12	64.91	66.89
ResNet50 (32*32)	86.37	24.09	**86.13**	**83.21**	**55.34**	**73.97**	**76.59**	**70.41**	**76.09**	**68.40**	**72.94**	**70.55**	62.42	**82.43**	**66.48**	**82.33**	**78.61**	**76.16**	**76.44**	**75.46**	**75.90**	**70.13**	**73.36**
ORRIC	-	-	79.24	79.06	52.19	72.08	72.35	67.20	70.96	67.51	68.44	64.90	58.99	75.70	64.51	77.23	73.15	69.01	70.46	71.69	69.46	69.69	69.19

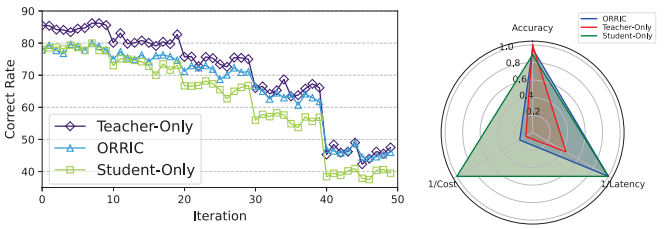
first train MobileNetV2 and ResNet50 on the training set of CIFAR-10, then test them on CIFAR-10-C and the test set of CIFAR-10 (the "original" column in Table III) separately. Specially, we use original images (whose resolution is $32*32$) for training, while resized images (whose resolution may be $32*32$, $28*28$, $24*24$, or $20*20$) are used for testing. We do not use more kinds of lower resolution (e.g., $16*16$) because the predictions given by the trained MobileNetV2 and ResNet50 are random in these cases. We also give the computing resource consumption (measured by MACs, which can be calculated by using a third-party library like PyTorch-OpCounter) and latency (measured on our NVIDIA A40 server) results of MobileNetV2 and ResNet50 on a single image at different resolutions. These varying resolutions represent distinct inference configurations (A_j^I). The retraining configurations (A_i^T) are delineated by the sampling ratios (0, 0.1, 0.2, 0.3, 0.5, 1.0), denoting the portion of uploaded data on the t-th time slot ($D_{(t)}$) utilized for one epoch of model retraining.

B. Results

We compare the following three methods on CIFAR-10-C: Teacher-Only (using ResNet50 for inference and without retraining), Student-Only (using MobileNetV2 for inference and without retraining), and ORRIC (using MobileNetV2 for inference and using ResNet50 to retrain MobileNetV2).

Without loss of generality, we set $D_{(t)} = 1000, \forall t$ for all three methods and $C_{(t)} \sim \mathbb{U}(C_1, C_2), \forall t$ for ORRIC, where $\mathbb{U}(C_1, C_2)$ is a uniform distribution between C_1 (the MACs when MobileNetV2 performs inference on 1000 images whose resolution is $32*32$) and C_2 (the MACs when ResNet50 performs inference on 1000 images whose resolution is $32*32$). To satisfy Property 1, we delete the $32*32$ inference configuration and set $f(A^T_{\max})$ to 0.7329 for the dataset of "gaussian noise", "impulse noise", "shot noise", and "speckle noise". For other datasets, we set $f(A^T_{\max})$ to 0.7957 (see the "original" column). We set L to 0.01 and set $A_i^T = \beta C_i^T$ (where the β is a normalization coefficient, making $\max_i\{A_i^T\} = 1$).

The real-time accuracy of these three methods on the "fog" corruption dataset is shown in Fig. 3 (a). Because each type of corruption dataset in CIFAR-10-C has 5 severity levels, and the first 10,000 images are at severity 1, while the last 10,000 images are at severity 5 [31], the accuracy of all three methods drops suddenly and periodically. However, the curve of ORRIC is almost always higher than the curve of Student-Only, showing the benefit of model retraining. We also give the average accuracy of ORRIC on other corruption datasets in the last row of Table III.



(a) Real-time Accuracy Results Comparison. (b) Accuracy-Cost-Latency Trade-off Comparison.

Fig. 3: Results on the "fog" Corruption Dataset of CIFAR-10-C.

The resource consumption and latency of these three methods on the "fog" corruption dataset can be calculated using the parameters given in Table III, and we report the Accuracy-Cost-Latency trade-off of these three methods while normalizing the maximum value of each axis to 1, see Fig. 3 (b). ORRIC surpasses or equals the Student-Only algorithm in terms of accuracy and latency while utilizing idle available computing resources. ORRIC surpasses the Teacher-Only algorithm in terms of cost and latency while maintaining good accuracy. In general, the model retraining and inference co-location paradigm can utilize idle available resources to improve model accuracy while maintaining low latency, thereby alleviating the negative impact of drift on accuracy.

VII. Conclusion

In this paper, we study the online allocation in the model retraining and inference co-location paradigm. We model the current model performance as a function of past retraining configuration and current inference configuration and then propose a linear complexity online algorithm (named ORRIC). Our competitive analysis implies the advantages and applications of model retraining and inference co-location paradigm over the traditional Inference-Only paradigm. Experiments on the CIFAR-10-C validate the effectiveness of model retraining and inference co-location in drift scenarios.

VIII. Acknowledgment

The authors appreciate the reviewers for their insightful and valuable comments. Discussions with Kongyange Zhao, Tao Ouyang and Qing Ling are gratefully acknowledged.

APPENDIX

A. Proof of Theorem 1

Proof. Supposing (x_1, y_1) and (x_2, y_2) are two points in the domain of $f(x)y$, and denoting $\bar{x} = \alpha x_1 + (1-\alpha)x_2$ and $\bar{y} = \alpha y_1 + (1-\alpha)y_2$, where $0 < \alpha < 1$, then $E = f(\bar{x})\bar{y} - [\alpha f(x_1)y_1 + (1-\alpha)f(x_2)y_2] = \alpha\left[f(\bar{x}) - f(x_1)\right]y_1 + (1-\alpha)\left[f(\bar{x}) - f(x_2)\right]y_2$. If there exist values for $\alpha, x_1, x_2, y_1, y_2$ that make E less than 0, then $f(x)y$ is nonconcave. We can also prove that $f(x)y$ is nonconvex in the same way. Therefore, we can conclude that $f(x)y$ is nonconvex-nonconcave.

If $f(x)$ is twice-differentiable, the indefiniteness of the Hessian matrix of $f(x)y$ can prove the theorem too. □

B. Proof of Theorem 3

Proof. We show this fact using proof by contradiction. Suppose a situation where computing resources are sufficient and $D_{(t)} = D$ on every time slot, then $P^* = f(0)A^I_{\max}D_{(1)} + \sum_{t=2}^{T} f(A^T_{\max})A^I_{\max}D_{(t)} = (f(0) + \sum_{t=2}^{T} f(A^T_{\max}))A^I_{\max}D$, while $P = \sum_{t=1}^{T} f(0)A^I_{\max}D_{(t)} = \left(\sum_{t=1}^{T} f(0)\right)A^I_{\max}D$, then we have: $\frac{P}{P^*} = \frac{\sum_{t=1}^{T} f(0)}{f(0)+\sum_{t=2}^{T} f(A^T_{\max})} = \frac{Tf(0)}{f(0)+(T-1)f(A^T_{\max})}$. Suppose the tight competitive ratio of the Inference-Only algorithm (denoted as $\bar{c}$) is strictly bigger than $\frac{Tf(0)}{f(0)+(T-1)f(A^T_{\max})}$, which means for any input, $P \geq \bar{c}P^* > \frac{Tf(0)}{f(0)+(T-1)f(A^T_{\max})}P^*$, but we have found an input that makes $P = \frac{Tf(0)}{f(0)+(T-1)f(A^T_{\max})}P^*$, the contradiction has arisen. Then we show an upper bound of the tight competitive ratio $\bar{c}$ of the Inference-Only algorithm is $\frac{Tf(0)}{f(0)+(T-1)f(A^T_{\max})}$. □

C. Proof of the Correctness of ORRIC

Proof. Let's assume that the optimal configuration indices for retraining and inference in Dt are a and b, so $C^T_a + C^I_b \leq U$. We only need to demonstrate that ORRIC must have explored the pair $(a, b, VA^T_a + WA^I_b)$. ORRIC terminates when either $i > M$ or $j < 1$. Let's consider the scenario where it terminates with $j < 1$ (the case for terminating with $i > M$ is similar). In this case, j will decrease from N to 0. When j reaches b, let's assume that $i = a_1$ at this moment.

First case: If $a_1 \leq a$, then $C^T_{a_1} + C^I_b \leq U$. According to the algorithm, i will start increasing from a_1 until $C^T_i + C^I_b > U$ or until $i > M$, whichever happens first. At this point, $i > a$, so $(a, b, VA^T_a + WA^I_b)$ must have been explored by ORRIC.

Second case: If $a_1 > a$, then the previous iteration is $(a_1, b+1)$ (where $C^T_{a_1} + C^I_{b+1} > C^T_a + C^I_{b+1} > U$). And the former iteration of it won't be $(a_1 - 1, b+1)$ (since $C^T_{a_1-1} + C^I_{b+1} \geq C^T_a + C^I_{b+1} > U$, so $(a_1 - 1, b)$ is next to $(a_1 - 1, b+1)$). Therefore, the next pairs are $(a_1, b+2)$, $(a_1, b+3)$, and so on until (a_1, N) is reached. At this point, the pair before must be $(a_1 - 1, N)$, and $(a_1 - 2, N)$, and so on until $a_1 - k$ is found such that $C^T_{a_1-k} + C^I_N < U$. In this case, (a, N) must be present in these iterations. However, according to the algorithm, the next iteration from (a, N) is $(a, N-1)$, not $(a+1, N)$. Therefore, this case is not possible. □

D. Proof of Theorem 4

Proof.

$$
\begin{aligned}
D^* = & \underbrace{f\left(A^T_{\max}\right)\sum_{j=1}^{N} y_j(1)A^I_j D_{(1)}}_{A_1} \underbrace{- LA^T_{\max}\sum_{j=1}^{N} y_j(1)A^I_j D_{(1)}}_{B_1} \\
& + \underbrace{\sum_{t=2}^{T} f\left(A^T_{\max}\right)\sum_{j=1}^{N} y_j(t)A^I_j D_{(t)}}_{A_2} \\
& + \underbrace{\sum_{t=1}^{T-1} L\frac{D_{\min}A^I_{\min}}{D_{\max}}\frac{1}{t}\left(\sum_{\tau=1}^{t} D_{(\tau)}\sum_{i=1}^{M} x_i(\tau)A^T_i\right)}_{C} \\
& \underbrace{- \sum_{t=1}^{T-1} L\frac{D_{\min}A^I_{\min}}{D_{\max}}\frac{1}{t}\sum_{\tau=1}^{t} D_{(\tau)}A^T_{\max}}_{B_2}
\end{aligned}
$$

For term B_1, we have $B_1 \leq -\frac{LA^T_{\max}D^2_{\min}A^I_{\min}}{D_{\max}}$ by $\sum_{j=1}^{N} y_j(1)A^I_j D_{(1)} \geq D_{\min}A^I_{\min} \geq D_{\min}A^I_{\min}\frac{D_{\min}}{D_{\max}}$. For term B_2, we have $B_2 \leq -(T-1)\frac{LA^T_{\max}D^2_{\min}A^I_{\min}}{D_{\max}}$ due to $\sum_{\tau=1}^{t} D_{(\tau)} \geq tD_{\min}$. Finally $B_1 + B_2 \leq -T\frac{LA^T_{\max}D^2_{\min}A^I_{\min}}{D_{\max}}$.

For term C, since the increasing and concave property of f (Assumption 1), $\frac{\sum_{\tau=1}^{t} D_{(\tau)}\sum_{i=1}^{M} x_i(\tau)A^T_i}{\sum_{\tau=1}^{t} D_{(\tau)}} \leq A^T_{\max}$ by the definition of $A^T_{\max}$ and Assumption 2, we have the following fact: $L \leq f'\left(A^T_{\max}\right) \leq f'\left(\frac{\sum_{\tau=1}^{t} D_{(\tau)}\sum_{i=1}^{M} x_i(\tau)A^T_i}{\sum_{\tau=1}^{t} D_{(\tau)}}\right)$. Combining this fact and $\frac{D_{\min}A^I_{\min}}{D_{\max}}\frac{1}{t} < \frac{\sum_{j=1}^{N} y_j(t+1)A^I_j D_{(t+1)}}{\sum_{\tau=1}^{t} D_{(\tau)}}$, we get: $C \leq \sum_{t=1}^{T-1}\left(f(0) + f'\left(\frac{\sum_{\tau=1}^{t} D_{(\tau)}\sum_{i=1}^{M} x_i(\tau)A^T_i}{\sum_{\tau=1}^{t} D_{(\tau)}}\right)\frac{\sum_{\tau=1}^{t} D_{(\tau)}\sum_{i=1}^{M} x_i(\tau)A^T_i}{\sum_{\tau=1}^{t} D_{(\tau)}}\right)\sum_{j=1}^{N} y_j(t+1)A^I_j D_{(t+1)} - \sum_{t=1}^{T-1} f(0)\sum_{j=1}^{N} y_j(t+1)A^I_j D_{(t+1)}$.

Since the assumed concave property of f (Assumption 1), we have $f(0) \leq f(x) + (0-x)f'(x)$, i.e., $f(0) + xf'(x) \leq f(x)$. Then, $C \leq \sum_{t=1}^{T-1} f\left(\frac{\sum_{\tau=1}^{t} D_{(\tau)}\sum_{i=1}^{M} x_i(\tau)A^T_i}{\sum_{\tau=1}^{t} D_{(\tau)}}\right)\sum_{j=1}^{N} y_j(t+1)A^I_j D_{(t+1)} - \sum_{t=1}^{T-1} f(0)\sum_{j=1}^{N} y_j(t+1)A^I_j D_{(t+1)} = P - \sum_{t=0}^{T-1} f(0)\sum_{j=1}^{N} y_j(t+1)A^I_j D_{(t+1)}$.

Based on all of the above analysis, we have: $P^* \leq D^* = B_1 + B_2 + C + A_1 + A_2 \leq -T\frac{LA^T_{\max}D^2_{\min}A^I_{\min}}{D_{\max}} + P - \sum_{t=0}^{T-1} f(0)\sum_{j=1}^{N} y_j(t+1)A^I_j D_{(t+1)} + \sum_{t=1}^{T} f(A^T_{\max})\sum_{j=1}^{N} y_j(t)A^I_j D_{(t)} = -T\frac{LA^T_{\max}D^2_{\min}A^I_{\min}}{D_{\max}} + P + \frac{f(A^T_{\max})-f(0)}{f(0)}\sum_{t=1}^{T} f(0)\sum_{j=1}^{N} y_j(t)A^I_j D_{(t)} \leq -T\frac{LA^T_{\max}D^2_{\min}A^I_{\min}}{D_{\max}} + P + \frac{f(A^T_{\max})-f(0)}{f(0)}P = \frac{f(A^T_{\max})}{f(0)}P - \alpha Tf(A^T_{\max})A^I_{\max}D_{\max}$, where $\alpha = \frac{LA^T_{\max}D^2_{\min}A^I_{\min}}{f(A^T_{\max})D^2_{\max}A^I_{\max}}$. Further from the fact that $P \leq P^* \leq Tf(A^T_{\max})A^I_{\max}D_{\max}$, we have: 1) $P^* \leq \frac{f(A^T_{\max})}{f(0)}P - \alpha P^*$, 2) $P^* \leq \frac{f(A^T_{\max})}{f(0)}P - \alpha P$. Then we prove that the competitive ratio of ORRIC is $\frac{(1+\alpha)f(0)}{f(A^T_{\max})}$ or $\frac{1}{\frac{f(A^T_{\max})}{f(0)} - \alpha}$, where $\alpha = \frac{LA^T_{\max}D^2_{\min}A^I_{\min}}{f(A^T_{\max})D^2_{\max}A^I_{\max}}$. □

REFERENCES

[1] S. Lin, Z. Zhou, Z. Zhang, X. Chen, and J. Zhang, *Edge Intelligence in the Making: Optimization, Deep Learning, and Applications*, ser. Synthesis Lectures on Learning, Networks, and Algorithms. Morgan & Claypool Publishers, 2020.

[2] S. Niu, J. Wu, Y. Zhang, Y. Chen, S. Zheng, P. Zhao, and M. Tan, "Efficient test-time model adaptation without forgetting," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, vol. 162. PMLR, 2022, pp. 16 888–16 905.

[3] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[4] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan, "A survey of embodied AI: from simulators to research tasks," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 6, no. 2, pp. 230–244, 2022.

[5] J. Liang, R. He, and T. Tan, "A comprehensive survey on test-time adaptation under distribution shifts," *CoRR*, vol. abs/2303.15361, 2023.

[6] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, 2019.

[7] R. Wu, C. Guo, Y. Su, and K. Q. Weinberger, "Online adaptation to label distribution shift," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021, pp. 11 340–11 351.

[8] R. Fakoor, J. Mueller, Z. C. Lipton, P. Chaudhari, and A. J. Smola, "Data drift correction via time-varying importance weight estimator," *CoRR*, vol. abs/2210.01422, 2022.

[9] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *19th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2022, Renton, WA, USA, April 4-6, 2022*. USENIX Association, 2022, pp. 119–135.

[10] M. Khani, G. Ananthanarayanan, K. Hsieh, J. Jiang, R. Netravali, Y. Shu, M. Alizadeh, and V. Bahl, "RECL: responsive resource-efficient continuous learning for video analytics," in *20th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2023, Boston, MA, April 17-19, 2023*. USENIX Association, 2023, pp. 917–932.

[11] L. Wang, K. Lu, N. Zhang, X. Qu, J. Wang, J. Wan, G. Li, and J. Xiao, "Shoggoth: Towards efficient edge-cloud collaborative real-time video inference via adaptive online learning," in *60th ACM/IEEE Design Automation Conference, DAC 2023, San Francisco, CA, USA, July 9-13, 2023*. IEEE, 2023, pp. 1–6.

[12] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016.

[13] R. Dong, Z. Tan, M. Wu, L. Zhang, and K. Ma, "Finding the task-optimal low-bit sub-distribution in deep neural networks," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, vol. 162. PMLR, 2022, pp. 5343–5359.

[14] L. Jia, Z. Zhou, F. Xu, and H. Jin, "Cost-efficient continuous edge learning for artificial intelligence of things," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7325–7337, 2022.

[15] C. Zhao, F. Mi, X. Wu, K. Jiang, L. Khan, and F. Chen, "Adaptive fairness-aware online meta-learning for changing environments," in *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, 2022, pp. 2565–2575.

[16] P. Yang, F. Lyu, W. Wu, N. Zhang, L. Yu, and X. S. Shen, "Edge coordinated query configuration for low-latency and accurate video analytics," *IEEE Trans. Ind. Informatics*, vol. 16, no. 7, pp. 4855–4864, 2020.

[17] K. Zhao, Z. Zhou, X. Chen, R. Zhou, X. Zhang, S. Yu, and D. Wu, "Edgeadaptor: Online configuration adaption, model selection and resource provisioning for edge dnn inference serving at scale," *IEEE Transactions on Mobile Computing*, pp. 1–16, 2022.

[18] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: on-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 1, pp. 447–457, 2020.

[19] M. K. Shirkoohi, P. Hamadanian, A. Nasr-Esfahany, and M. Alizadeh, "Real-time video inference on edge devices via adaptive model streaming," in *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 4552–4562.

[20] S. G. Patil, P. Jain, P. Dutta, I. Stoica, and J. Gonzalez, "POET: training neural networks on tiny devices with integrated rematerialization and paging," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, vol. 162. PMLR, 2022, pp. 17 573–17 583.

[21] D. Xu, M. Xu, Q. Wang, S. Wang, Y. Ma, K. Huang, G. Huang, X. Jin, and X. Liu, "Mandheling: mixed-precision on-device DNN training with DSP offloading," in *ACM MobiCom '22: The 28th Annual International Conference on Mobile Computing and Networking, Sydney, NSW, Australia, October 17 - 21, 2022*. ACM, 2022, pp. 214–227.

[22] C. Lv, C. Niu, R. Gu, X. Jiang, Z. Wang, B. Liu, Z. Wu, Q. Yao, C. Huang, P. Huang, T. Huang, H. Shu, J. Song, B. Zou, P. Lan, G. Xu, F. Wu, S. Tang, F. Wu, and G. Chen, "Walle: An End-to-End, General-Purpose, and Large-Scale production system for Device-Cloud collaborative machine learning," in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. Carlsbad, CA: USENIX Association, Jul. 2022, pp. 249–265.

[23] J. J. Moon, P. Kapoor, J. Lee, M. Ham, and H. S. Lee, "Nntrainer: Lightweight on-device training framework," *CoRR*, vol. abs/2206.04688, 2022.

[24] M. Abadi, "Tensorflow lite," 2023, https://www.tensorflow.org/lite [Accessed: (Jul. 28, 2023)].

[25] A. Paszke, "Pytorch mobile," 2023, https://pytorch.org/mobile/ [Accessed: (Jul. 28, 2023)].

[26] T. Domhan, J. T. Springenberg, and F. Hutter, "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. AAAI Press, 2015, pp. 3460–3468.

[27] T. Lattimore and A. György, "Improved regret for zeroth-order stochastic convex bandits," in *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, ser. Proceedings of Machine Learning Research, vol. 134. PMLR, 2021, pp. 2938–2964.

[28] P. Zhao, G. Wang, L. Zhang, and Z. Zhou, "Bandit convex optimization in non-stationary environments," *J. Mach. Learn. Res.*, vol. 22, pp. 125:1–125:45, 2021.

[29] A. Gupta, R. Krishnaswamy, and K. Pruhs, "Online primal-dual for non-linear optimization with applications to speed scaling," in *Approximation and Online Algorithms - 10th International Workshop, WAOA 2012, Ljubljana, Slovenia, September 13-14, 2012, Revised Selected Papers*, vol. 7846. Springer, 2012, pp. 173–186.

[30] A. Simonetto, E. Dall'Anese, S. Paternain, G. Leus, and G. B. Giannakis, "Time-varying convex optimization: Time-structured algorithms and applications," *Proc. IEEE*, vol. 108, no. 11, pp. 2032–2048, 2020.

[31] D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[32] Q. Wang, O. Fink, L. Van Gool, and D. Dai, "Continual test-time domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 7201–7211.