Optimal Resource Allocation for ML Model Training and Deployment under Concept Drift

Hasan Burhan Beytur[1], Gustavo de Veciana[1], Haris Vikalo[1], Kevin S Chan[2]
University of Texas at Austin[1] DEVCOM Army Research Laboratory[2]
{hbbeytur, deveciana}@utexas.edu hvikalo@ece.utexas.edu kevin.s.chan.civ@army.mil

***Abstract*—We study how to best allocate resources for training and deployment of Machine Learning (ML) models under concept drift and limited resources. We consider a setting in which a model provider distributes trained models to multiple clients whose devices are capable of running inference but lack resources to retrain those models, and thus the burden of managing them falls upon the provider. We introduce a model-agnostic framework that captures the interaction between resource allocation, concept drift dynamics, and deployment timing. Assuming sudden concept changes, we derive optimal training policies under budget constraints when concept durations have distributions with Decreasing Mean Residual Life, and show that intuitive heuristics are provably suboptimal if they have Increasing Mean Residual Life. We further study model deployment schedules under communication constraints, prove that under mild conditions the associated optimization problem is quasi-convex, and propose a randomized strategy that achieves near-optimal client-side performance. These results provide theoretical foundations and algorithmic principles for designing resilient networked ML model deployments.**

***Index Terms*—Concept Drift, Resource Allocation, Model Training and Deployment, Optimal Control, MLOps**

I. Introduction

The rapid advancement of Machine Learning (ML), particularly in generative models for image synthesis and Large Language Models (LLMs), is transforming the way we carry out both professional and personal tasks [1]. As models grow more complex, their training and inference require increasingly more data and computational power, making their management resource-intensive. While the growing demand for advanced models and the necessary compute power have largely been addressed through centralized cloud infrastructure, there is a notable shift toward localized inference on end-user devices. This shift enhances scalability by reducing reliance on centralized resources and promotes user privacy [2], [3]. However, despite the rise of on-device inference, model training remains largely cloud-based due to its heavy computational and coordination requirements.

In real-world applications, ML model performance often degrades over time due to changes in the underlying data distributions, collectively referred to as *concept drift*. Such drift can result from shifts in the relationship between input features and target variables, evolving class distributions, or the appearance of out-of-distribution data. For example, a rental price predictor might lose accuracy as market conditions evolve; a medical model may suffer from demographic shifts; and an LLM may fail to answer questions about recent events. In all cases, degradation stems from a growing mismatch between training and deployment data [4], [5]. Sustaining ML performance under concept drift requires continuous monitoring, retraining, and redeployment – an operational challenge addressed by the field of ML Operations (MLOps) [6]. These operations add significant overhead to already costly model development. As ML adoption grows and models become more complex, efficient resource allocation within MLOps workflows becomes essential for scalable and sustainable deployment.

We consider a setting where a service provider trains ML models and distributes them to client devices for local inference. While clients can perform inference, they typically lack the computational and data resources required for retraining. Thus, the burden of maintaining model accuracy in the presence of concept drift falls on the provider, who must allocate limited resources to retraining and redeployment as performance deteriorates. We focus on sudden, discrete concept drift, and develop a model-agnostic framework that captures how the model performance is affected by both the amount of allocated training resources and the occurrence of drift events. Since client-side performance depends on timely redeployment, we also study drift-aware deployment strategies that aim to optimize long-term average performance. Our results offer practical guidance for designing resilient, resource-efficient ML systems that adapt to evolving data.

A. Related Works

Extensive empirical studies have investigated trade-offs among compute, model size, and data volume in ML training, and built a foundation for compute-efficient training of large language models [7], [8]. Complementary work on learning curve modeling has enabled more cost-effective training via early stopping and budget-aware learning rate schedules [9], [10]. Inspired by such insights on learning dynamics, we study how training resource allocation and deployment strategies affect model performance under sudden concept drift – an important but underexplored regime in the context of scaling laws and resource-aware training.

Concept drift breaks the assumption of stationary data distributions in machine learning and requires different mitigation techniques depending on its form. Seminal surveys [4], [5] distinguish gradual drift, typically handled via continuous retraining or time-varying optimization frameworks [11], [12], from sudden drift, which is abrupt and often requires change-point detection and model resets [13], [14]. While recent studies analyze trade-offs between retraining cost and model

staleness [15], [16] for specific models or domains, they largely ignore how drift statistics affect the allocation of compute resources and the scheduling of re-trained model deployment —gaps our model-agnostic framework addresses.

Our work lies within the scope of MLOps [6]. While MLOps primarily aims to establish best practices for ML system operation, the increasing complexity and rising operational costs of modern ML pipelines [17] have spurred interest in cost-aware and compute-efficient retraining and deployment strategies [18], [19].

B. Contributions

Motivated by the need to sustain performance of distributed ML systems operating in resource-constrained and non-stationary data settings, we develop a model-agnostic framework capturing the effects of elastic resource allocation and sudden concept drift. Our main contributions are:

1) We propose a model-agnostic analytical framework that characterizes the performance of both client-side and server-side models, capturing the effects of training resource allocation, sudden concept drift, and model deployment policy. The proposed framework enables principal analysis of how these factors influence model performance.

2) We derive optimal training resource allocation policies for concept duration following Decreasing Mean Residual Life (DMRL) distribution, aiming to maximize the long-term performance of server-side models. We further demonstrate the sub-optimality of intuitive allocation heuristics under Increasing Mean Residual Life (IMRL) concept durations.

3) We analyze the deployment scheduling problem under communication constraints and establish quasi-convexity of the objective under mild conditions. We derive necessary optimality conditions and introduce a randomized deployment policy that achieves target deployment rates with theoretical performance guarantees.

Together, these contributions provide theoretical foundations and algorithmic principles for designing resource-aware training and deployment policies for real-world, large-scale ML systems.

II. A Framework for Resource Allocation under Sudden Concept Drift

Consider a system where a resource-rich centralized ML model provider manages model training, monitoring, and model updates to resource-constrained client devices that can only perform local inference and rely on the provider to manage performance degradation due to distributional changes. In this work, we focus specifically on modeling and managing the effects of *sudden concept drift*[1], where shifts in data distribution occur abruptly and lead to immediate drops in model performance.

To address the challenges of accurately evaluating model performance during training, particularly the fluctuations induced by stochastic optimization, our framework adopts an idealized metric, *expected concept loss*, defined as the model's expected loss over the true data distribution associated with the current concept. We assume this *average* loss decreases monotonically with training, with its exact form determined by the model architecture, optimization algorithm, data characteristics, and allocated resources [9]. Under sudden concept drift, the trajectory of expected concept loss varies across concepts due to abrupt distributional changes. Let T_i and Y_i denote the start time of the i^{th} concept and its duration so that $T_{i+1} - T_i = Y_i$, with $\mathbb{E}[Y] < \infty$. We assume that the shape of expected concept loss during concept i is governed by a random function $G_i(\cdot) : \mathbb{R}^+ \to \mathbb{R}^+$, drawn from a countable set $\mathcal{G}$ of convex, decreasing functions. Here, $G_i(0)$ represents the loss immediately following the onset of concept i, and $G_i(t)$ models the loss decay over time under unit resource allocation. The overall concept dynamics are captured by the stochastic process $((Y_i, G_i(\cdot)) : i \in \mathbb{N})$, where the pairs $(Y_i, G_i(\cdot))$ are assumed to be independent and identically distributed (i.i.d).

Assuming training resources[2] can be adjusted dynamically over time, let $e_i(t) : [0, y_i) \to [0, M]$ denote the resource allocation during concept i, where M is the maximum allowable resource level and y_i is a realization of the concept duration Y_i. We assume that training speed scales linearly with $e_i(t)$, and model this effect by applying a horizontal scaling transformation to $G_i(\cdot)$ controlled by the the resource allocation function $e_i(\cdot)$. The server-side expected concept loss at time t, $\mathcal{L}(t)$, reflective of both the stochastic concept drift process $(Y_i, G_i(\cdot))$ and the corresponding resource allocation $e_i(\cdot)$, is then formalized as

$$\mathcal{L}(t) = G_{I(t)}\left(\int_0^{t-T_{I(t)}} e_{I(t)}(\tau)d\tau\right), \tag{1}$$

where $I(t) = \max\{i \in \mathbb{N} : T_i \le t\}$ is the index of the concept active at time t. Since both $G_{I(t)}$ and $I(t)$ are random, $\mathcal{L}(t)$ is itself a stochastic process. While $\mathcal{L}(t)$ represents the expected concept loss of the server-side model under ongoing training, client devices operate on previously deployed versions. Since deployments occur at discrete times, the client model often lags behind the server model. As a result, the expected concept loss experienced by clients, $\hat{\mathcal{L}}(t)$, may differ from $\mathcal{L}(t)$.

Let $D_{i,j}$ denote the time of j^{th} model deployment during concept i, with $D_{i,0} = T_i$. At any time t, client devices run the most recently deployed model version. The expected concept loss experienced by clients at time t, denoted $\hat{\mathcal{L}}(t)$, is thus given by

$$\hat{\mathcal{L}}(t) = G_{I(t)}\left(\int_0^{D_{I(t),J(t)}-T_{I(t)}} e_{I(t)}(\tau)d\tau\right), \tag{2}$$

where $D_{I(t),J(t)} = \max\{D_{i,j} : D_{i,j} \le t\}$ is the latest deployment time prior to time t.

[1]Assuming a detection algorithm that signals change points, our framework applies directly to sudden drift and also extends to gradual drift via the sequence of detected change points as long as detection delay is negligible.

[2]In supervised learning, more compute allows faster model parameter updates, while in online learning, training speed is limited by the data arrival rate, which can sometimes be increased (e.g, by boosting sensor frequency or acquiring more labels via crowdsourcing [20], [21]). Our framework captures both cases under the unified notion of training resource allocation.

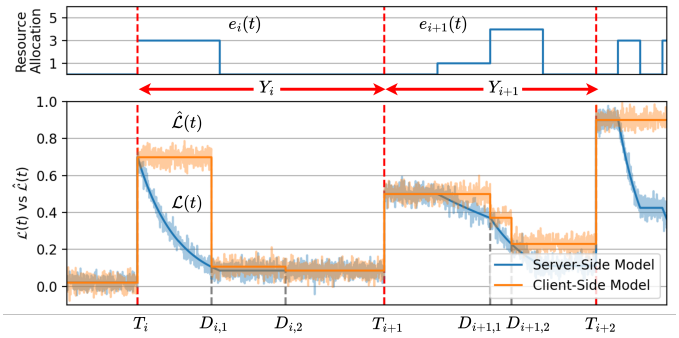

Fig. 1: Top: allocated training resources $e_i(t)$. Bottom: sample paths of expected concept loss at the server $\mathcal{L}(t)$ and clients $\hat{\mathcal{L}}(t)$, as defined in (1) and (2).

In Figure 1 the top plot displays the allocated training resources, which determine training speed, while the bottom plot shows server-side loss (blue), which jumps at concept changes and then decays according to the allocation, and deployed-model loss (orange), which updates to match the server model at deployment times; fluctuations around both curves represent the true loss computed during training.

Next, we investigate two key questions: (1) Given a constraint on total training resources, how should they be allocated over time to mitigate the impact of sudden concept drift and optimize server-side model performance? (2) Under a constraint on deployment update frequency, what characterizes the optimal deployment policy that maximizes client-side performance under concept drift?

III. Optimizing Resource Allocation for Training under Concept Drift

In this section, we investigate resource allocation policies for optimizing the time average ML model performance on the *server-side* under a time average budget constraint in the presence of sudden concept drift, as outlined in the framework introduced in Section II. We discuss ML model deployment and *client-side* performance in Section IV.

Compute budget constraint on training resources. As previously noted, we assume the ML model service provider has access to elastic compute resources, i.e., can dynamically purchase training resources over time. We assume a pay-as-you-go pricing model, where resources are billed at a fixed rate σ_e per unit of time and resource, and the provider has a fixed average cost budget constraint, denoted by B. In particular given the resource allocation functions $e_i(t) : [0, \infty) \to [0, M]$ over the duration of each concept i the time-average budget constraint is given by:

$$\limsup_{t\to\infty} \frac{\sigma_e}{t} \int_{\tau=0}^{t} e_{I(\tau)}(\tau - T_{I(\tau)})d\tau \leq B, \tag{3}$$

where $I(\tau)$ denotes the index of the current concept at time τ, with arrival time $T_{I(\tau)}$.

In this paper we shall consider stationary *static training resource allocation policies*, where the same resource allocation function is applied across all concept periods, independent of their concept-specific characteristics, $g_i(\cdot)$ and y_i.

***Definition** 1 (Static training resource allocation policy):* A set of resource allocation functions $\{e_i(\cdot) : i \in \mathbb{N}\}$ defines a training resource allocation policy. A training resource allocation policy is said to be static – and therefore stationary and causal – if, for all concepts $i \in \mathbb{N}$ and $t \in [0, \infty]$, it holds that $e_i(t) = e(t)$.

Under a static training resource allocation policy denoted by $e(\cdot)$ and i.i.d. concept durations and expected concept loss functions $((Y_i, G_i(\cdot)) : i \in \mathbb{N})$. The limits in (3) exist and by the Renewal-reward Theorem [22, Section 3.4], the server-side time-average expected loss is given by

$$\lim_{t\to\infty} \frac{1}{t} \int_{\tau=0}^{t} \mathcal{L}(\tau)d\tau = \frac{\mathbb{E}[\int_{\tau=0}^{Y} \bar{g}(\int_0^\tau e(u)du)d\tau]}{\mathbb{E}[Y]}, \tag{4}$$

where the left hand side follows by the independence of G_i and Y_i[3], and we define $\bar{g}(\cdot) = \mathbb{E}[G_i(\cdot)|Y_i] = \mathbb{E}[G_i(\cdot)]$ as the expected loss function, which is the average of the expected concept loss across different concepts. Since each $g_i(\cdot)$ is assumed to be convex and decreasing, their expectation $\bar{g}(\cdot)$ also inherits these properties. Similarly, the time-average budget constraint in (3) becomes:

$$\frac{\sigma_e}{\mathbb{E}[Y]} \mathbb{E}[\int_0^Y e(t)dt] \leq B, \tag{5}$$

To determine the optimal static resource allocation policy minimizing the time-average expected loss in (4) while satisfying the budget constraint in (5), we rewrite these using Fubini's Theorem to formulate the following infinite-horizon continuous-time optimal control problem.

$$\min_{e(\cdot)} \quad J(e) = \int_0^\infty \bar{g}(x(t))\bar{F}_Y(t)dt \tag{6}$$

subject to

$$x'(t) = e(t),\ x(0) = 0,\ 0 \leq e(t) \leq M,\ \forall t \in [0, \infty), \tag{7}$$

$$C(e) = \int_0^\infty e(t)\bar{F}_Y(t)dt \leq B_1,\ \text{for } B_1 = \frac{B \cdot E[Y]}{\sigma_e}, \tag{8}$$

Here, $\bar{F}_Y(t) = \mathbb{P}(Y > t)$ denotes the survival function, and the optimization is over static resource allocation functions $e(t)$ belongs to the space of measurable and bounded functions on $[0, \infty)$ and lies in the admissible set $\mathcal{E} = \{e : e(t) \in [0, M],\ \forall t \geq 0\}$. For any $e \in \mathcal{E}$, we define the state variable $x(t) = \int_0^t e(\tau)d\tau$, which is absolutely continuous.

***Lemma** 1 (Existence of an optimal solution for Problem* (6)*):* For any continuously differentiable, convex, non-increasing $\bar{g} : [0, \infty) \to \mathbb{R}^+$ functional optimization problem (6) is convex and has at least one optimal solution, since the functional $J(e)$ defined by (6) and the feasible set of resource allocations $\mathcal{E}_C = \{e \in \mathcal{E} \mid \int_0^\infty e(t)\bar{F}_Y(t)dt \leq B_1\}$ are convex on the domain $\mathcal{E} = \{e : e(\tau) \in [0, M],\ \forall \tau \geq 0\}$.

A proof of Lemma 1 is provided in [23, A.1].

Further as discussed in more detail in [23, Section 3.1] it follows by Arrow's Sufficiency Theorem [24, Theorem 3.1],

[3]Although we formulate the time-averages under the assumption of independence between G_i and Y_i, a more general case, where the time-average (4) exists with a continuous convex decreasing function $\bar{g} : [0, \infty) \to \mathbb{R}^+$, which is not necessarily the mean of expected concept loss function $\mathbb{E}[G_i(\cdot)]$.

that the necessary conditions provided by Pontryagin's Maximum Principle (PMP) are also sufficient for global optimality in this setting. This in turn allows us to conclude imply that the optimal policy takes the form of a "bang-bang" control, i.e., transitions from one extreme control value to another. In fact as we shall see depending on the characteristics of the distribution of concept durations the optimal policy will have a single switch time. We discuss these characteristics below.

***Definition** 2 (Mean Residual Life (MRL)):* The Mean Residual Life (MRL) function $m_Y(t)$ of a non-negative random variable Y at time t with $\bar{F}_Y(t) > 0$ is defined as the expected remaining lifetime given survival up to time t:

$$m_Y(t) \coloneqq E[Y - t | Y > t] = \frac{\int_t^\infty \bar{F}_Y(u)du}{\bar{F}_Y(t)}. \tag{9}$$

We say that Y has Decreasing/Increasing Mean Residual Life (DMRL/IMRL) if $m_Y(t)$ is non-increasing/non-decreasing.

If concept durations have the DMRL property we shall the optimal resource allocation policy is a *Front-loading policy*, i.e., allocates a maximum amount of resources at the start of each concept duration up to some fixed time–formally stated in the following theorem shown in [23, A.2].

***Theorem** 1 (Optimality of the Front-Loading Policy):* For any convex decreasing expected loss curve $\bar{g} : [0, \infty] \to \mathbb{R}^+$ and any resource budget $B > 0$, if and only if the concept duration Y is DMRL, the optimal resource allocation is a single-switch control given by

$$e^*(t) = \begin{cases} M & t < t^*_{DMRL} \\ 0 & t \geq t^*_{DMRL} \end{cases} \tag{10}$$

where the switching time t^*_{DMRL} is uniquely determined by the budget constraint, $\int_0^{t^*_{DMRL}} \bar{F}_Y(y)dy = \frac{\mathbb{E}[Y]B}{M\sigma_e}$, if $B < M\sigma_e$, and $t^*_{DMRL} = \infty$, otherwise.

Note Theorem 1 states that if Y is DMRL the optimal policy is Front loading, and if the optimal policy is front loading for all budgets then then Y must be DRML. While such policies may appear intuitive, it formally rules out the optimality of commonly used policies like, fixed or periodic training schedules. Moreover, the result implies that if Y is not DMRL then the front-loading policy need not be optimal for all budgets. The next result, shown in [23, A.3] states that if Y is IMRL the optimal policy leads to delayed (i.e., idling) resource allocation at the start of each concept duration.

***Theorem** 2:* For any decreasing expected loss curve $\bar{g} : [0, \infty] \to \mathbb{R}^+$, any random concept duration Y is IMRL, and budget constraint $B < M\sigma_e$, the optimal resource allocation exhibits idling before any allocation,

$$e^*(t) = 0 \quad \text{for} \quad t < t^*_{IMRL}, \tag{11}$$

where $t^*_{IMRL} > 0$ is the time when resource allocation starts.

Additionally, if the expected loss curve is decreasing and linear one can show that the optimal policy Back-loading, i.e, idles and then switches to allocating a maximum. Please refer to [23, Corollary 3.6] for details.

It is important to emphasize that the strength of Theorems 1 and 2 lies in their generality: their conclusions do not depend on the exact functional form of $\bar{g}(\cdot)$, but only on its monotonic and convex properties. As such, they offer broadly applicable guidance for designing training resource allocation strategies based on the characteristics of the training cycles driven by the sudden concept changes.

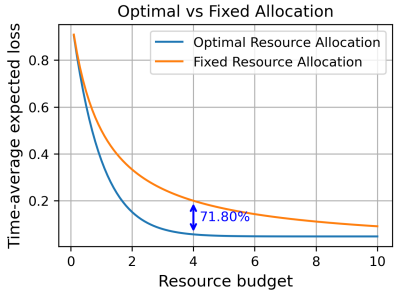

Fig. 2: Comparison between constraint-binding optimal and fixed resource allocation policies. $Y \sim \text{Exp}(1)$, $\bar{g}(t) = e^{-t}$, $\sigma_e = 1$, $M = 20$.

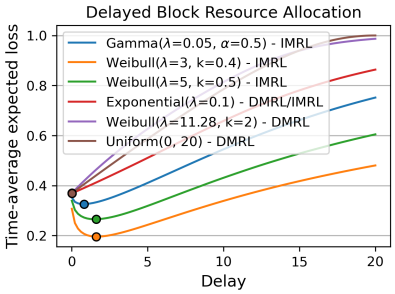

Fig. 3: Block resource allocation with varying delays under different concept duration distributions. $\bar{g}(t) = e^{-t}$, $B = 0.1, \sigma_e = 1, M = 20$.

In Figure 2, for exponentially distributed Y_i and $\bar{g}(t) = e^{-t}$, we compare the time-average expected loss achieved under two policies: the optimal resource allocation (specifically, the front-loading policy derived from Theorem 1) and a fixed resource allocation meeting the budget constraint exactly, i.e., $e_{\text{fixed}}(t) = B/\sigma_e$, $\forall t$. The results demonstrate that, for the same resource budget, the optimal policy reduces the time-average expected loss by up to 71.80% compared to the fixed allocation policy. As the budget B approaches the upper limit $M\sigma_e$, the switching time t^*_{DMRL} increases, and the performance gap between the two policies narrows.

Figure 3 studies delayed block resource allocation under concept duration distributions with different aging properties and using $\bar{g}(t) = e^{-t}$. In this policy, resources are withheld until a delay z, then applied at full capacity M until the budget is exhausted, i.e., $e_{\text{block}}(t) = M$ for $z \leq t < T(z)$ and 0 otherwise, where $T(z)$ satisfies $\int_z^{T(z)} \bar{F}_Y(t)\, dt = \frac{B\mathbb{E}[Y]}{M\sigma_e}$.

Consistent with Theorems 1 and 2, the results show that IMRL distributions benefit from a limited delay in resource allocation, whereas DMRL distributions perform worse with any delay. Overall, the aging properties of concept durations strongly influence optimal training resource allocation, and tailoring training to these characteristics improves performance under budget constraints.

IV. Deployment Optimization under Concept Drift

As discussed in Section II, the performance experienced by clients depends on the version of the model that has

been deployed to them. In this section, we investigate the deployment scheduling problem, aiming to optimize the long-term performance of client-side models under a constraint on the deployment rate, e.g., a proxy for communication costs.

To understand the dynamics governing deployment decisions, we analyze deployment policies in the setting where the server-side model is trained using a fixed allocation of training resources. In line with the approach taken in Section III, we restrict our attention to static deployment schedulers, defined as follows:

***Definition** 3 (Static Deployment Scheduler):* Let $D_{i,j} - T_i$ denote the deployment time offset relative to concept arrival time T_i, where $D_{i,j}$ is the j-th deployment time during concept i. A deployment scheduler is said to be static–and thus stationary and causal–if, for all concepts $i \in \mathbb{N}$,

$$D_{i,j} - T_i = \delta_j, \text{ almost surely} \quad \forall i, j \in \mathbb{N},$$

where $\delta_0 = 0 < \delta_1 < \cdots$ so the set $\{\delta_j : j \in \mathbb{N}\}$ defines a static deployment scheduler.

Let $N_i = \max\{j : \delta_j < Y_i\}$ be a random variable denoting the number of deployments occurring during concept i under a static deployment scheduler. Assuming a fixed unit training resource allocation for the server-side model, i.e., $e_i(t) = 1, \ \forall t \geq 0$, and $\bar{g}(\cdot) = \mathbb{E}[G_i(\cdot)]$, then $N \sim N_i$ are i.i.d., and the optimization problem minimizing the time-average expected loss as seen by clients subject to a time-average deployment rate constraint is given by:

$$\min_{\{\Delta_j\}_{j=1}^{\infty}} \quad \sum_{j=0}^{\infty} \bar{g}(\delta_j) \int_{\delta_j}^{\delta_{j+1}} \bar{F}_Y(t)dt \tag{12}$$

$$\text{s.t.} \quad \Delta_j = \delta_j - \delta_{j-1} \geq 0, \ \forall j \in \mathbb{N}^+ \quad \delta_0 = 0, \tag{13}$$

$$E[Y]^{-1} \sum_{j=1}^{\infty} \bar{F}_Y(\delta_j) \leq r_D, \tag{14}$$

where r_D denotes the maximum allowable deployment rate and $\{\Delta_j\}_{j=1}^{\infty}$ denotes the set of inter-deployment durations. The detailed derivation of Equation (12) and (14) is provided in [23, Eq. (24)-(34)]

Note the optimization Problem (12) is not necessarily convex. However, under suitable conditions, we can guarantee that the problem is quasi-convex, as stated in the following lemma.

***Lemma** 2 (Quasi-convexity of Problem* (12)*):* If $\bar{g}$ is a non-negative, convex, decreasing function and $\bar{F}_Y$ is a strictly decreasing function, then the objective function in (12) is quasi-convex in $\{\Delta_j \geq 0\}_{j \in \mathbb{N}}$. Moreover, if $\bar{F}_Y$ is also convex, then Problem (12) is quasi-convex.

A detailed proof for Lemma 2 is provided in [23, A.5]. As discussed in the proof of Lemma 2, the KKT conditions are sufficient for global optimality when the feasible set defined by the deployment rate constraint (14) is convex. However, even under this condition, a numerical method is generally required to obtain the optimal solution.

Using the KKT conditions, for a Lagrange multiplier $\nu > 0$, the optimal deployment scheduler $\{\delta_k^*\}_{k=0}^{\infty}$ must satisfy:

$$\bar{g}'(\delta_k^*) \int_{\delta_k^*}^{\delta_{k+1}^*} \bar{F}_Y(t)dt = \bar{F}_Y(\delta_k^*)[\bar{g}(\delta_k^*) - \bar{g}(\delta_{k-1}^*)] + \nu f_Y(\delta_k^*).$$

A numerical approach such as the bisection method can be employed to solve for the optimal deployment times, typically starting from the last deployment time and proceeding backward, assuming an artificial limit on the number of deployments. Motivated by the structure of this solution, we propose a randomized policy based on a modified version of the original problem (12), which can be applied even when the concept duration distribution has a non-convex survival function. The modified problem, replacing the deployment rate constraint with a fixed, deterministic number of deployments, denoted by N_D, is formulated as:

$$\min_{\{\Delta_j\}_{j=1}^{N_D}} \quad \sum_{j=0}^{N_D-1} \bar{g}(\delta_j) \int_{\delta_j}^{\delta_{j+1}} \bar{F}_Y(t)dt + \bar{g}(\delta_{N_D}) \int_{\delta_{N_D}}^{\infty} \bar{F}_Y(t)dt \tag{15}$$

$$\text{s.t.} \quad \Delta_j = \delta_j - \delta_{j-1} \geq 0, \ \forall j \in [1, N_D] \quad \delta_0 = 0. \tag{16}$$

Since the rate constraint (14) is removed, Lemma 2 ensures that Problem (15) remains quasi-convex.

***Theorem** 3 (Optimal Solution for Problem* (15)*):* Let $\bar{g}$ be a non-negative, convex, decreasing function, and let Y be a random variable with $\bar{F}_Y(t) > 0$ for all $t \in [0, \infty)$. Then the optimal deployment scheduler $\{\delta_k^*\}_{k=0}^{N_D}$ satisfies:

$$\frac{\bar{g}(\delta_{k-1}^*) - \bar{g}(\delta_k^*)}{-\bar{g}'(\delta_k^*)} = \begin{cases} m_Y(d_{N_D}^*), & k = N_D \\ \frac{\int_{\delta_k^*}^{\delta_{k+1}^*} \bar{F}_Y(t)dt}{\bar{F}_Y(\delta_k^*)}, & k \in [1, N_D - 1] \end{cases}. \tag{17}$$

Furthermore, the effective deployment rate achieved by the optimal scheduler $r_e(\{\delta_k^*\}_{k=0}^{N_D}) = \sum_{j=1}^{N_D} \bar{F}_Y(\delta_j)$ is monotonically increasing in number of deployments N_D.

The proof of Theorem 3 follows from the quasi-convexity of Problem (15) and the monotonicity of $\bar{F}_Y(t)$. Details are provided in [23, A.6]. In a special case–discussed in [23, Corollary 4.4]–where concept durations are exponentially distributed and the expected loss curve decays exponentially, Theorem 3 yields a closed-form, chain-structured solution, which can be determined iteratively. We now introduce a randomized deployment scheduling policy that, at each concept, uses two optimal policies for Problem (15) with consecutive number of deployment, $\{\delta_k^*\}_{k=0}^{N_D}$ and $\{\delta_k^*\}_{k=0}^{N_D+1}$, at random with probability γ and $(1-\gamma)$. Such randomized policy achieves an effective deployment rate equals to the convex combinations of $r_e(\{\delta_k^*\}_{k=0}^{N_D})$ and $r_e(\{\delta_k^*\}_{k=0}^{N_D+1})$, which can be matched with the deployment constraint. By leveraging optimal deployment schedulers obtained from Problem (15) with different deterministic deployment counts, the randomized deployment scheduler provides a practical solution to the constrained deployment optimization problem (12). It ensures full utilization of the allowed deployment budget, and does not require the convexity of the survival function. Our simulation results further demonstrate that the performance of the randomized scheduler closely approximates that of the optimal (non-randomized) policy, making it an effective choice for deployment under concept drift. The details of the randomized scheduler is discussed in [23, Theorem 4.5].

Figure 4 presents the performance gains achievable through the optimal deployment policies, compared to a periodic

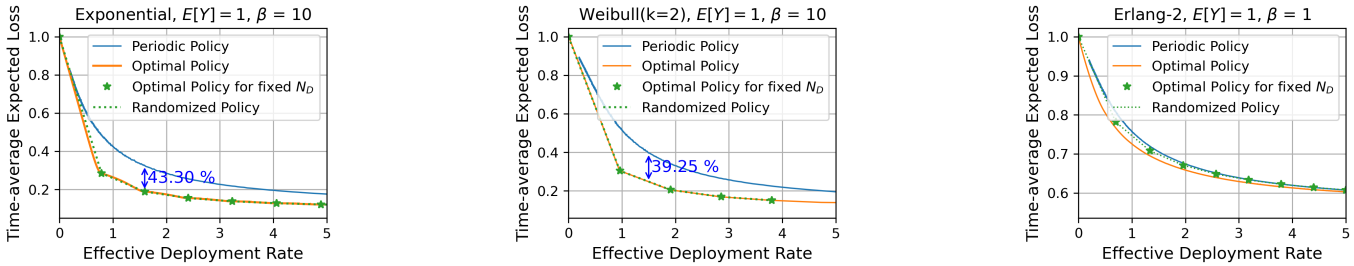

Fig. 4: Comparison between periodic, optimal and randomized deployment policies, under different concept duration distributions with $\mathbb{E}[Y] = 1$ and exponentially decaying expected loss, $\bar{g}(t) = e^{-\beta t}$.

deployment baseline, and demonstrates the near-optimality of the randomized deployment policy, across different concept duration distributions–specifically, Exponential, Weibull($k = 2$), and Erlang-2. Except the Exponential case–where a closed-form solution is available–the optimal deployment policies are computed numerically. The results for the Exponential and Weibull($k = 2$) distributions show that, under the same effective deployment rate, the optimal policy reduces the client-side time-average expected loss by up to 43.30% and 39.25%, respectively, compared to the periodic policy. Moreover, in these settings, the randomized policy, which operates on the convex hull of optimal solutions for different fixed deployment counts N_D, achieves performance that is nearly indistinguishable from the optimal policy. In contrast, the results for the Erlang-2 distribution show a narrower performance gap between the optimal and periodic policies. This is attributed both to the distributional characteristics of Erlang-2, which limit the deployment policy's leverage, and to the use of a small decay rate β in the loss function, which causes the expected penalty to decline more slowly. In addition to that, under these conditions, the randomized policy no longer matches the performance of the optimal policy, providing a concrete example where the optimal policy strictly outperforms the randomized alternative.

V. Conclusion

We present a formal framework for studying training resource allocation and model deployment in distributed ML systems operating under persistent, discrete concept drift. In scenarios where training resources are elastic, i.e., available on demand but incurring computational and communication costs, a provider will want to optimize when to allocate resources and deploy updated models. To that end, we have shown that optimal resource allocation and model deployment policies depend on the characteristics of the distribution of concept durations and loss functions. As ML systems become increasingly ubiquitous, the monitoring, retraining and redeployment cycle will constitute a growing share of operational costs. Our findings lay the groundwork for optimizing this cycle in non-stationary settings through principled, cost-efficient training and deployment strategies.

VI. Acknowledgement

This paper has been supported in part by an DOD Award W911NF-24-2-0185 and NSF Award 2148224 and is supported in part by funds from OUSD R&E, NIST, and industry partners as specified in the Resilient & Intelligent NextG Systems (RINGS) program.

References

[1] H. Hussein, M. Gordon *et al.*, "ChatGPT's Impact Across Sectors: A Systematic Review of Key Themes and Challenges," *BDCC*, Mar. 2025.
[2] X. Zhou, L. Chen *et al.*, "TinyLLaVA: A lightweight vision–language assistant," in *Proceedings of the IEEE/CVF*, 2024, pp. 9876–9885.
[3] D. Chen, S. Yang *et al.*, "What role do small models play in a world of giants?" in *ICLR*, 2024.
[4] M. Han, Z. Chen *et al.*, "A survey of active and passive concept drift handling methods," *IEEE TKDE*, vol. 38, no. 4, pp. 1492–1535, 2022.
[5] J. Lu, A. Liu *et al.*, "Learning under concept drift: A review," *IEEE TKDE*, vol. 31, no. 12, pp. 2346–2363, 2019.
[6] D. Kreuzberger, N. Kühl *et al.*, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," *IEEE Access*, 2023.
[7] J. Kaplan, S. McCandlish *et al.*, "Scaling laws for neural language models," *arXiv:2001.08361*, 2020.
[8] J. Hoffmann, S. Borgeaud *et al.*, "Training compute-optimal large language models," *arXiv:2203.15556*, 2022.
[9] T. Viering and M. Loog, "The shape of learning curves: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, p. 7799–7819, Jun. 2023.
[10] M. Li, E. Yumer *et al.*, "Budgeted training: Rethinking deep neural network training under resource constraints," *arXiv:1905.04753*, 2019.
[11] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," in *ICDM*, 2007, pp. 123–132.
[12] A. Mokhtari, S. Shahrampour *et al.*, "Online optimization in dynamic environments: Improved regret rates for strongly convex problems," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016.
[13] A. Bifet and R. Gavaldà, "Learning from Time-Changing Data with Adaptive Windowing," in *Proceedings of SDM*, Apr. 2007.
[14] M. Budka and B. Gabrys, "Change-point detection in evolving data streams using ensembles of classifiers," *IEEE Signal Processing Letters*, vol. 25, no. 9, pp. 1353–1357, 2018.
[15] A. Mahadevan and M. Mathioudakis, "Cost-effective retraining of machine learning models," *arXiv:2310.04216*, 2023.
[16] R. Florence, S. Leo *et al.*, "When to retrain a machine learning model," *arXiv:2505.14903*, 2025.
[17] B. Cottier, R. Rahman *et al.*, "The rising costs of training frontier ai models," *arXiv:2405.21015*, 2025.
[18] Y. Shen, G. Chen *et al.*, "Cost-aware dynamic cloud workflow scheduling using self-attention and evolutionary reinforcement learning," in *ICSOC*. Springer, 2024, pp. 3–18.
[19] S. Wang, A. Pi *et al.*, "Elastic parameter server: Accelerating ml training with scalable resource scheduling," *IEEE TPDS*, vol. 33, 2021.
[20] W. Huang, J. Zhan *et al.*, "Context-aware adaptive sampling for intelligent data acquisition systems using dqn," *arXiv:2504.09344*, 2025.
[21] A. Mandlekar, Y. Zhu *et al.*, "Roboturk: A crowdsourcing platform for robotic skill learning through imitation," in *CoRL*, 2018, pp. 879–893.
[22] R. G. Gallager, *Discrete Stochastic Processes*. Springer US, 1996.
[23] H. Beytur, G. de Veciana *et al.*, "Optimal resource allocation for ml model training and deployment under concept drift," *arXiv:2512.12816*, 2025.
[24] S. P. Sethi, *Optimal Control Theory: Applications to Management Science and Economics*. Springer International Publishing, 2021.