The Dissertation Committee for Hasan Burhan Beytur
certifies that this is the approved version of the following dissertation:

Offloading and Managing Change in Cloud-Assisted Machine Learning Systems

Committee:

Gustavo de Veciana, Supervisor

Haris Vikalo

Sanjay Shakkottai

Aryan Mokhtari

John Hasenbein

Offloading and Managing Change in Cloud-Assisted Machine Learning Systems

by

Hasan Burhan Beytur

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2026

Dedication

To Duygu

Acknowledgments

As someone who has always been motivated and inspired by those who pursue knowledge and mastery in their fields, doctoral studies have been my humble effort to expand my understanding of the world and refine my perspective. It has been a rewarding, yet challenging, journey of self-discovery, growth, and learning, not only intellectually but also personally. This experience would not have been possible without the support, guidance, and companionship of my mentors, friends, and family at every step along the way.

I feel incredibly fortunate to have pursued my Ph.D. under the supervision of Prof. Gustavo de Veciana, whose unwavering support, guidance, and encouragement made this journey truly remarkable. During my time at UT Austin, he spent countless hours providing invaluable feedback on my research, helping me think through research directions, and engaging deeply with all of my projects. Whenever I came to him feeling stuck or uncertain, I left our conversations with a clearer path forward and a renewed sense of motivation. He was not only a guide, but also someone who walked alongside me throughout this journey. His generosity, patience, and dedication have shaped my growth in profound ways, and I am truly grateful for his mentorship. The high standards he sets in both work and life, through his own example, will continue to inspire and guide me for years to come.

I owe a great debt of gratitude to Prof. Haris Vikalo, with whom I had the privilege of working closely during my Ph.D. His guidance, insightful comments, and thoughtful suggestions were instrumental in shaping both the direction and quality of my research. I sincerely appreciate his mentorship and support throughout my doctoral studies.

I would also like to thank my committee members, Prof. Haris Vikalo, Prof. Sanjay Shakkottai, Prof. Aryan Mokhtari, and Prof. John Hasenbein, for their valuable feedback and support. I am also grateful to the WNCG faculty, especially

Prof. Gustavo de Veciana, Prof. Sanjay Shakkottai, Prof. Aryan Mokhtari, and Prof. Jeffrey Andrews, from whom I have learned a great deal through their exceptional courses. I greatly appreciate the safe, supportive, and collaborative environment at WNCG, where I always felt encouraged by my peers and colleagues despite the natural pressures and competitiveness of academic life. I believe this atmosphere was not accidental, but rather the result of the welcoming and supportive environment created by the WNCG faculty and staff. I feel privileged to have been part of such a community.

I would also like to thank all the incredible people I met during my time at UT Austin, with whom I shared this experience as classmates, collaborators, and companions. I began this journey alongside Agrim, with whom I spent countless hours. He has been a wonderful friend and colleague, always there for me and offering more support and companionship than I could have asked for. I am grateful to Agrim for standing by me throughout these years.

I also wish to thank Parikshit, Ian, Saadallah, and Nithin, who became dear friends and colleagues throughout this journey. Having experienced many of the challenges of graduate school before me, they offered generous advice, thoughtful perspective, and encouragement during difficult times. I am grateful to Ahmet Günhan, Hakan, and Oğuzhan, whose companionship gave me a sense of familiarity and belonging far from home.

My sincere thanks go to Shorya, Ahmet Günhan, Seohyeon, Vinay, and Haoran, with whom I spent many hours working on various projects. It was a privilege to collaborate with such talented colleagues and to learn from them. I also thank my friends at WNCG, including Heasung, Sravan, George, Karl, Deland, Jean, Geetha, Juseong, Jianhan, Tianqu, Alperen, Ezgi, Manan, Karthik, and many others, for their friendship and support throughout these years.

I would like to thank Emrehan Aktuğ, who has been my friend since high school, for his companionship during the time our paths overlapped in Austin. I am

especially grateful to my friends from METU, Furkan Karakaya, Halil Ibrahim Uğurlu, and Fatih Uzgur, whose friendship has remained a source of comfort, encouragement, and strength throughout my Ph.D. I also thank Anıl Gürses for the friendship we developed during our internship at Qualcomm, and I hope it will continue for many years to come.

My deepest gratitude goes to my wife, Duygu, who has been by my side through every joy and difficulty of this journey. Through all the challenges we faced together, her unwavering support, love, and generosity have been my greatest source of strength and motivation. Her patience, understanding, and encouragement helped me navigate the many ups and downs along the way. I can never thank her enough for being my best teammate; without her, this journey would not have been possible.

Finally, I am grateful to my parents, Melek Kurtoğlu-Beytur and Mehmet Beytur, for their love and support, which have shaped who I am and encouraged me to pursue my dreams. I am also grateful to my parents-in-law, Leyla Öksünlü and Yahya Öksünlü, for their support and encouragement throughout this journey.

Abstract

Offloading and Managing Change in Cloud-Assisted Machine Learning Systems

Hasan Burhan Beytur, PhD
The University of Texas at Austin, 2026

SUPERVISOR: Gustavo de Veciana

As machine learning systems become more capable, the applications that rely on them are growing in scale and complexity across industries and everyday life. This expanding reliance on machine learning systems has led to rapidly growing demand for inference, making the design of efficient and scalable systems increasingly important. This dissertation investigates resource allocation and decision-making problems that arise in the operation and maintenance of large-scale deployments of machine learning systems. In particular, it focuses on the coordination of client-side and server-side resources to improve inference efficiency, monitor the performance of deployed models, and support model updating under resource constraints.

First, we study congestion-aware inference task offloading in machine learning systems in which lightweight models deployed on clients are supported by a more accurate server-side model. We propose Lyapunov-EXP4, a distributed online learning algorithm that learns offloading policies to maximize system-level accuracy while satisfying a resource utilization constraint in the presence of heterogeneous offloading costs and task distributions across clients. Our analysis provides near-optimal performance guarantees and characterizes the resulting tradeoffs among accuracy, delay,

and fairness across heterogeneous clients.

Second, the dissertation addresses performance monitoring and concept drift detection in large-scale deployments where collecting feedback from clients is costly. In contrast to existing work that focuses on detecting the earliest degradation at any individual client, this part of the dissertation formulates a more operationally relevant problem of detecting significant performance degradations affecting a significant fraction of the client population. To solve this problem, it proposes two complementary algorithms, Greedy Partial Sum-CUSUM, which focuses on client-level performance, and Adaptive Global-CUSUM, which focuses on population-level performance. By analyzing their detection delay and false alarm characteristics, the dissertation identifies the regimes in which each approach is most effective and shows how performance depends on the population size, as well as the magnitude and spread of performance degradation across clients.

Third, the dissertation studies the optimization of cyclic maintenance processes of model training and deployment under concept drift. It addresses when to train, how much to train, and when to deploy updated models in order to maximize the long-term average performance of the system under resource constraints. It formulates training resource allocation as a continuous-time optimal control problem and highlights the relationship between the structure of optimal policies and the statistical properties of the concept drift process. Within the same framework, it studies deployment scheduling from the perspective of client-side performance and develops randomized deployment policies that achieve target deployment rates while maintaining near-optimal performance.

Together, these results provide a theoretical foundation for designing efficient and scalable machine learning systems that jointly leverage client-side and server-side resources. By developing analytical formulations and algorithms with provable guarantees for inference serving, performance monitoring, retraining, and redeployment, this dissertation offers insights for the scalable operation and maintenance of

large-scale deployments of machine learning systems.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Motivation

Over the past several decades, sustained advances in machine learning (ML) have transformed it from a specialized research area into a foundational technology for optimization and automation in modern engineering systems. Early breakthroughs in deep learning for visual modalities [57], followed more recently by transformer-based architectures for language, multimodal reasoning, and generative modeling [101] have accelerated the integration of ML across industries, enabling new classes of applications while also replacing many heuristic and manually designed pipelines with data-driven systems.

As ML capabilities improve and adoption widens, computational demand is no longer solely driven by model training, but also by the need to serve inference demand at scale. For many applications, cloud-based deployment has been the dominant strategy because centralized infrastructure can pool demand across users, host large models, exploit specialized accelerators, and simplify maintenance, performance monitoring, and model updates. However, applications with stringent latency requirements, limited or intermittent connectivity, or privacy-sensitive inputs may not be suitable for cloud-based deployment and instead require inference to be performed closer to where data are generated. Even when inference can be served from the cloud, the rapid growth in demand places increasing pressure on data-center capacity, energy infrastructure, hardware refresh cycles, and operational costs, raising questions about the long-term feasibility and scalability of centralized deployment strategies [90, 1].

In parallel, there has been substantial momentum and market push to make end-user devices more capable of performing ML inference locally. Advancements in power-efficient hardware, especially in low-precision and sparsity-optimized ML accelerators, together with advancements in model compression, quantization and memory-

efficient ML architectures have made on-device inference practical for a growing range of applications [60, 105]. This gives rise to two distinct deployment strategies for ML inference, each with its own advantages and limitations. While on-device inference enables privacy-preserving and low-latency applications, the range of applications and predictive performance are typically constrained by the limited computational resources of individual devices. Alternatively, cloud-based inference can support more computationally intensive and accurate models, but serving inference demand at scale with consistently high performance can introduce latency and require substantial investment in building and operating additional data-center capacity. These feasibility concerns have increased interest in hybrid deployment strategies that can leverage both client-side and server-side resources and combine the advantages of both on-device and cloud-based inference.

1.1.1 Hierarchical Inference

Hierarchical inference [3, 77] has been proposed as an edge computing paradigm for ML inference that jointly leverages client-side and server-side resources to support privacy-preserving and scalable deployment. In a typical hierarchical inference system, each client runs a lightweight model optimized for low-latency inference, while a higher-capacity and more accurate model is hosted on a server. For each inference task, the client first produces a local prediction using a lightweight local model. Depending on its confidence in the local prediction and the application requirements, it either uses the local prediction or offloads the task to the server for a more accurate prediction. By invoking the server only for selected tasks, hierarchical inference trades predictive performance against offloading costs such as communication latency, bandwidth usage, energy consumption, and computational load.

This cooperation between client-side and server-side models introduces new challenges in system design and optimization. The most immediate challenge is to design offloading policies that decide whether tasks should be handled locally at clients or sent to a server, while accounting for heterogeneous task distributions and

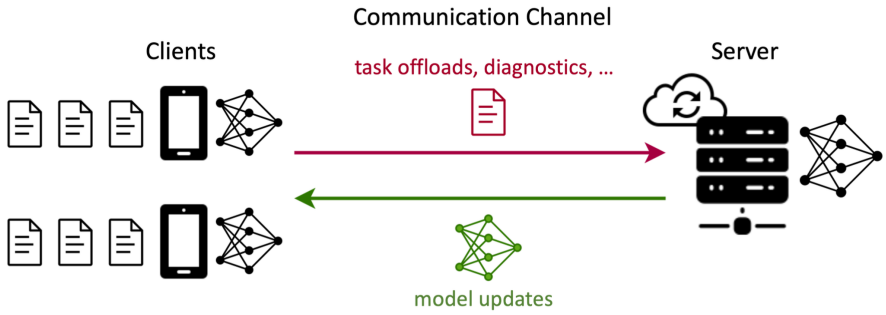


Figure 1.1: Hierarchical inference system with client-side and server-side models.

communication delays across clients. Beyond offloading, hierarchical inference also complicates the maintenance operations that ML systems require to mitigate performance degradation due to distributional shifts. This is particularly important in large-scale deployments, where clients may experience different levels of performance degradation at different times. The system needs to monitor the performance experienced by clients without intruding on their privacy or creating excessive load on the server, and then take appropriate actions to mitigate the degradation, such as retraining and updating the local models under operational constraints. Although these challenges have been studied in related contexts, their treatment in hierarchical inference systems remains relatively limited.

1.2 Scope of the Dissertation

This dissertation studies resource allocation and decision-making problems that arise from coordination between clients and the server in ML inference systems that jointly leverage client-side and server-side resources. From the perspective of an ML service provider responsible for operating and maintaining an inference service for a large population of clients, the dissertation provides a unified treatment of the process, covering not only offloading policies that enable cost-effective hierarchical inference but also the repeated core operations in the Machine Learning Operations (MLOps) cycle [55], including monitoring, training, and deployment, that are neces-

sary to sustain performance over time. The remainder of this section summarizes the main problems studied in the dissertation and the key contributions in each area.

1.2.1 Congestion-Aware Offloading Policies

The core function of a hierarchical inference system is to decide whether a task should be offloaded to the server, and this offloading policy determines how effectively client-side and server-side resources are utilized. Ideally, a client should offload a task only when the local prediction is incorrect, thereby avoiding unnecessary offloading costs due to communication overheads or increased server load. However, by the nature of the problem, clients typically have at best a noisy signal of the correctness of the local prediction, such as a confidence score, which can serve as a proxy for whether the anticipated accuracy gain from server-side inference justifies the offloading cost. The offloading mechanism becomes even more complex in large systems where many clients with heterogeneous task distributions and nontrivial offloading costs share limited communication resources or server compute resources, especially when decisions must be made in a distributed manner without centralized coordination.

Contributions. Chapter 2 studies the design of offloading policies that maximize the system's inference accuracy under a shared resource utilization constraint. The resulting constrained online learning problem is addressed through a proposed Lyapunov-EXP4 (Ly-EXP4) algorithm, which combines Lyapunov drift minimization with the bandits with expert advice framework and is applicable beyond hierarchical inference to a broader class of constrained contextual bandit problems. The chapter provides theoretical guarantees for the convergence of the proposed algorithm and explores the associated accuracy, delay, and fairness tradeoffs in hierarchical inference.

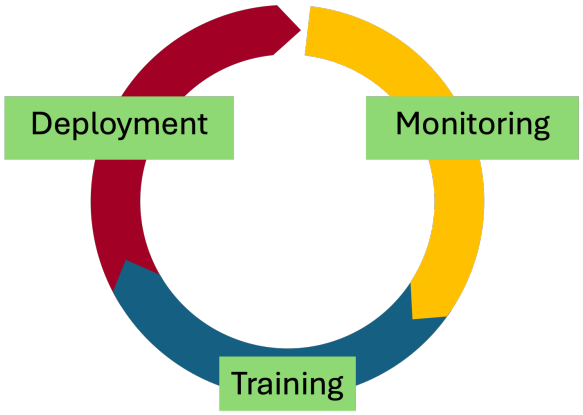


Figure 1.2: The core maintenance operations in an MLOps cycle.

1.2.2 Performance Monitoring in Large-Scale Deployments

When ML models are released into production environments, their performance may differ from what was observed during development because production data may not match the training data or may change over time, a phenomenon known as concept drift [34, 68]. Maintaining reliable performance therefore requires monitoring model behavior on production data and taking corrective action when degradation is detected. This is difficult in large-scale deployments that rely on on-device or hierarchical inference, since performance monitoring typically requires collecting data from clients and obtaining annotations from human experts or a more powerful server-side model. At scale, this feedback collection can create excessive workload and communication overhead, and it may conflict with the privacy motivation for keeping inference close to the clients. Moreover, when clients with heterogeneous data distributions experience asynchronous performance degradation, the relevant operational objective is often not to detect the earliest occurrence of degradation at any client, but rather to detect when degradation has become sufficiently widespread across the population of clients to justify a system-wide intervention.

Contributions. Based on the sequential hypothesis testing framework, Chapter 3 formulates the problem of detecting concept drift that affects a sufficiently large

fraction of clients by a significant amount under partial feedback, where feedback is available sequentially from only one selected client at a time. The resulting composite hypothesis testing problem is addressed through two proposed policies: Greedy Partial Sum-CUSUM (Greedy-PSC), which is based on a generalized likelihood ratio test and uses a deterministic client-sampling policy, yielding an individual-statistics-based approach; and, Adaptive Global-CUSUM (Adaptive-GC), which uses a mixture-based statistic with a uniformly randomized client-sampling policy, yielding a population-statistics-based approach. Chapter 3 analyzes the detection delay and false alarm characteristics of these policies and identifies the operating regimes in which they are most effective, including their scalability and sensitivity to different drift patterns.

1.2.3 Model Training and Deployment Optimization under Concept Drift

Deployed ML models often require periodic retraining and updating because their performance can degrade under shifts in the data distribution, such as when training data become stale and no longer reflect the latest information, or when user queries evolve over time. However, as ML models become larger and more complex, the amount of data and computational resources required for retraining can grow substantially, and retraining and delivering the resulting performance improvements to a large population of clients in a timely manner becomes a costly operation. Therefore, under limited resources and concept drift, the decisions of when and how quickly to train ML models, and the scheduling of model updates to clients, must be made strategically to maximize the long-term performance of the system.

Contributions. From the perspective of an ML service provider responsible for maintaining the performance of models deployed at clients, Chapter 4 studies how to allocate resources for training the model at the server, where resources may correspond to compute capacity in supervised learning or to the data generation rate in online learning, and how to schedule the deployment of refined models to clients in order to maximize long-term average system performance under concept drift. The

training resource allocation problem is formulated as a continuous-time optimal control problem under an average training budget constraint. The analysis shows that the structure of the optimal allocation policy depends on the aging statistics of the concept drift process. In particular, it is determined by whether the expected remaining time until the detection of the next drift event increases or decreases over time.

The deployment scheduling problem focuses on the long-term performance experienced by clients using deployed models. Under a communication budget that limits the deployment rate, the resulting scheduling problem is generally non-convex. Chapter 4 addresses this difficulty through a randomized policy built from deployment schedules that solve a relaxed version of the original problem, allowing the system to meet a target deployment rate while maintaining near-optimal performance. Together, the analysis characterizes how different drift dynamics shape optimal training and deployment policies and provides insights for designing resilient and resource-efficient ML systems.

Together, the three chapters develop a unified perspective on resource allocation and decision-making in large-scale deployments of ML systems. The dissertation studies how such systems can efficiently serve inference requests, monitor performance degradation across many clients, and allocate resources for training and deployment. Across these problems, the proposed methods characterize fundamental tradeoffs among accuracy, latency and operational costs, providing analytical tools for the reliable operation of ML systems at scale.

1.3 Publications

The chapters of this dissertation are based on prior publications and ongoing submissions. Chapter 2 is based on work published at IEEE INFOCOM 2024 [9]. Chapter 3 is based on a manuscript under submission to ACM SIGMETRICS 2027 at the time of this writing. Chapter 4 is based on work published at IEEE ICMLCN

2026 [10] and currently under submission to the ACM Transactions on Modeling and Performance Evaluation of Computing Systems (ToMPECS).

Chapter 2: Optimization of Offloading Policies for Accuracy-Delay Tradeoffs in Hierarchical Inference

2.1 Introduction

Machine learning (ML) applications and services are evolving to enable finding solutions to increasingly more challenging problems by deploying computationally intense models [14, 91]. This presents a challenge to their widespread deployment, especially on platforms with limited power and computational capabilities such as smartphones and IoT devices.

At one extreme, commonly seen in many real-world ML based applications [21], the entire ML model is executed on a server with sufficient computational power allowing devices to draw on minimal local computation. Although this approach relieves the computational burden on the users' devices, it leads to increased communication costs, reduced responsiveness, and may bring up privacy concerns.

At the other extreme, one can run ML based applications solely on users' devices. To facilitate this, there has been significant research on techniques for compressing complex ML models, resulting in what is commonly referred to as tinyML [86]. While this allows running models with small computational and memory footprints, they often come at the cost of sacrificing performance and accuracy as compared to the larger complex ML models typically run at the edge/cloud.

Given this trade-off a possible approach is to partition ML compute between devices and servers while minimizing communication overheads to improve the systems responsiveness– this is sometimes referred to as DNN Partitioning. To benefit from the fact that the output of the intermediate layers of DNN models can be significantly smaller than the input data, after the first couple of layers of a DNN model are computed locally, the rest of the layers can be processed by the server by only receiving the output of the part computed on the user device [50]. This technique can

achieve (1) better responsiveness by reducing the overhead of transmitting data to the server and balancing computation between user and server, and (2) a more secure and private system by keeping the actual input data on the user side. However, DNN partitioning needs to be engineered based on the system parameters, such as the computational power of the user device and the server and communication channel, and may not be as flexible or adaptive to change as one would like.

Between these extremes is Hierarchical Inference, a framework aiming to enable combining execution of ML tasks on users' devices with computational offloading to a server [76, 82, 19, 31]. In particular, users' devices employ a low-complexity ML model (L-ML) while the server hosts a more powerful high-complexity ML model (H-ML). A task is first processed by L-ML on the user's device. If the result requires further investigation or is deemed to have low confidence, the task is offloaded to the server to be processed by H-ML. Unlike DNN Partitioning [50, 46], Hierarchical Inference does not (partially) offload every task to the server, which helps improve the system's responsiveness. Although a hierarchical inference system can be built based on any pair of L-ML and H-ML, or with an ML model allowing early exit [100], it requires an offloading policy based on the output of the L-ML decides if offloading is needed. Furthermore, such decisions might depend on congestion of the communication resources.

This chapter is focused on hierarchical inference systems with multiple user devices and random offloading costs. We propose an online algorithm that learns how to manage offloading in such settings while being oblivious to the statistics of ML models, input task and offloading cost distributions. The proposed algorithm utilizes a multi-armed bandit with expert advice framework [58] to maximize system's inference accuracy while deploying a novel loss structure based on Lyapunov optimization theory to meet a constraint on resource utilization or, more generally, on offloading costs. The algorithm converges to an optimal threshold policy applied to the inference confidence seen on the L-ML model.

2.1.1 Related Work

Hierarchical Inference

Recently, hierarchical inference systems have attracted significant attention from the research community [76, 82, 19, 31]. An important line of this research focuses on designing metrics that capture the confidence of the local inference without knowing the correctness of the result. Examples include [82], where a confidence metric based on the variance of the local classifier's output is proposed, and [19], where the confidence of a prediction is assessed by applying a radial function kernel to the entropy of the local classifier's output. Ultimately, confidence metrics are meant to aid in designing offloading algorithms for hierarchical inference systems such as the one in [31], which deploys multiple local ML models on devices and a more complex/accurate ML model on a server. Given the different accuracy, processing and computation times of the models, the accuracy maximization problem under a time constraint is formulated as an integer linear program. An online learning approach proposed in [76] seeks a threshold policy on the confidence of L-ML that maximizes the accuracy under a fixed offloading cost. Although the approach developed in this chapter has a similar goal as [76], it is distinct as it focuses on multiple users under a shared utilization constraint and the distributed online offloading algorithm.

Lyapunov Optimization

The long-term stochastic optimization problems encountered in various computation offloading problem, are studied using Lyapunov optimization techniques, specifically Drift-plus-Penalty algorithm [81]. The Drift-plus-Penalty formulation is used in [73, 108] to solve offloading problems, where [73] investigates a mobile-edge computing system with energy harvesting devices under a hard execution delay deadline, and [108] formalizes the joint service caching and task offloading problem for minimizing computation latency under a long-term energy consumption constraint. Similarly, in [36, 63] authors present long-term optimization problems as two-step

optimization problems and leverage Drift-plus-Penalty algorithm to solve one of the steps. In all these works, the Lyapunov formulation involves an integer problem and the solution requires knowing the system parameters at each slot.

Constrained Bandit Optimization

In another line of related recent work, Lyapunov optimization techniques have been combined with online learning mechanisms to develop algorithms addressing constrained bandit optimization problems [66, 65, 18, 17, 5, 2]. Specifically, [66, 65], utilize Lyapunov-drift minimization methods for constrained online-dispatching via the UCB algorithm. In [18], a generic constrained bandit problem which incorporates random costs of actions, a knapsack constraint, and a stochastic feasibility constraint was studied; there, a Lyapunov-drift minimization technique was utilized to design a low-complexity bandit algorithm based on UCB. This chapter addresses constrained contextual bandit problems by combining Lyapunov drift minimization techniques with EXP4 algorithm.

2.1.2 Contributions

The contributions of this chapter are summarized as follows.

- For a hierarchical inference system supporting multiple clients, we formulate an accuracy maximization problem under a shared resource utilization constraint and explore the associated accuracy-delay trade-offs. We combine Lyapunov drift minimization techniques and bandits with expert advice framework to propose a low-complexity online offloading algorithm that we refer to as Lyapunov-EXP4 (Ly-EXP4). Ly-EXP4 converges to a near-optimal thresholding policy on the local inference confidence while satisfying the utilization constraint without prior knowledge of the statistics of tasks, confidence metric, and service times across clients. We prove that for finite horizon N and M discretized thresholds, Ly-EXP4 has a regret bound of $O(\sqrt{2N\zeta \log(M)})$ and an optimality-gap of

$O(\frac{1}{V}) + \epsilon_0$, where V is a parameter controlling tradeoff between accuracy and constraint violation. To the best of our knowledge, this chapter is the first to study in the literature discussing a hierarchical inference problem with multiple clients under a utilization constraint.

- We show that Ly-EXP4 can be used as a distributed online offloading algorithm maximizing the total accuracy of a system under a resource utilization constraint. In such settings, each client or a group of clients simultaneously learns an individual thresholding policy.

- Ly-EXP4 is not limited to the hierarchical inference problem; instead, it can be used in a more general class of constrained contextual bandit problems – a potentially valuable contribution in itself.

- We provide extensive simulation results using both real and synthetic ML models and datasets, demonstrating the performance of Ly-EXP4 in terms of convergence, delay and fairness. Additionally, we explore the accuracy-fairness trade-off in settings with multiple thresholds which allow Ly-EXP4 to achieve higher accuracy at the expense of fairness across clients.

2.1.3 Organization

The remainder of the chapter first introduces the system model and problem formulation in Section 2.2, then develops Ly-EXP4 and its multi-threshold extension in Section 2.3, and closes with simulation results in Section 2.4 and concluding remarks in Section 2.5.

2.2 System Model for the Hierarchical Inference

We consider a hierarchical inference system with a set $\mathcal{K}$ of clients $k \in \mathcal{K}$ connected to a server over a shared communication resource, executing classification tasks using a neural network based classifier. The classification tasks that the clients

face are first processed locally using a small ML model with low complexity and low accuracy (L-ML); in addition to the prediction, L-ML quantifies the confidence in its decision. After comparing the confidence with a threshold, the client decides whether to offload the task to a server to be processed with a more complex ML classifier with higher accuracy (H-ML). However, shared communication resources limit the number of tasks that the clients can offload.

We assume that classification tasks arrive to a client according to a Poisson process. Let λ_k denote the arrival rate of tasks to client k, and let $\lambda = \sum_{k \in \mathcal{K}} \lambda_k$. The n^{th} task arriving to the system is received by client K_n at time T_n. Let us denote the true class of task n and the class predicted by the local L-ML by C_n and $\hat{C}_n$, respectively. We assume that the true class of a task received by client k is independent and identically distributed according to an unknown client-specific distribution, i.e., $\boldsymbol{\alpha}_k = (\alpha_k(c) : c \in \mathcal{C})$, where $\alpha_k(c)$ denotes the probability that a task arriving to client k is of class c, and $\mathcal{C}$ is the set of classes. Since true class distributions of tasks differ from one client to another, and the clients deploy potentially different L-ML models, the predicted class distributions generally vary across the clients.

An L-ML model locally processing classification task n provides predicted class $\hat{C}_n$ and confidence measure Z_n. Given C_n and $\hat{C}_n$, Z_n is assumed to be independent and identically distributed on a bounded support Ω_Z. As with C_n and $\hat{C}_n$, we do not assume a specific distribution for Z_n.

Since in our hierarchical inference system the offloading decisions are based on the local inference confidence, the quality of confidence metric is essential. As mentioned in Section 2.1.1, various metrics characterizing confidence of inference exist; essentially, such metrics are potentially erroneous estimates of the confidence. A "good" metric should take on a high value when the classification is correct and a low value when potentially incorrect. Nevertheless, the offloading algorithm introduces in Section 2.3 converges to an optimal threshold policy without prior knowledge of the distribution of Z_n, regardless of the choice of metric.

When L-ML is a neural network used for classification, a simple choice for the confidence metric is the largest output of the last soft-max layer; such a quantity must be at least $1/|\mathcal{C}|$ (when all class outputs are equal), and cannot exceed 1. Without loss of generality, in this chapter, we will use this confidence metric when it is necessary, so $Z_n \in [1/|\mathcal{C}|, 1] = \Omega_Z$.

Next, we define the stationary threshold policy for the hierarchical inference.

Definition 2.1 (Stationary Threshold Policy)**.** *The offloading decision under a threshold policy π_θ with a threshold $\theta \in \Omega_Z$ is given by*

$$I_n^{\pi_\theta} = \mathbb{I}\{Z_n < \theta\}, \quad \theta \in \Omega_Z, \tag{2.1}$$

where $I_n^{\pi_\theta} = 1$ indicates offload of task n.

Offloading a task incurs resource expenditures, e.g., extra load on the communication channel or an operational cost of using the server. The offloading cost of task n is modelled by a random variable X_n; the offloading costs of the tasks received by client k are assumed to be independent and identically distributed according to an unknown client-specific distribution with a bounded support (without a loss of generality, [0,1]) and mean $\mu_k^{-1} = \mathbb{E}[X_n | K_n = k]$. We consider the case where the clients share a pool of resources, e.g., wireless access to a BS/server. The constraint on resource utilization is given by

$$\sum_{k \in \mathcal{K}} \hat{\lambda}_k^{\pi_\theta} \mu_k^{-1} \leq \gamma, \tag{2.2}$$

where $\hat{\lambda}_k^{\pi_\theta}$ is the offloading rate of client k under the stationary threshold policy π_θ. In practical systems with shared resources, the offloading cost can be thought of as the monetary value of using computational resources at the server, where γ can be interpreted as the budget per unit operational time. When it comes to communications, the mean offloading cost μ_k^{-1} can be thought of as the mean service time of client k, with γ being a utilization constraint which indirectly controls the average

delay in the system. The latter interpretation of γ is due to an observation that in a network where a scheduling policy controls how multiple clients share communication resources, utilization can be treated as a proxy for average delay as the two quantities are monotonically related to each other.

Although our model and algorithm can be used in more general settings, in the remainder of this chapter we focus on the accuracy-delay tradeoff in hierarchical inference systems where the task offloading follows a scheduling policy. Since the offloading rates $\hat{\lambda}_k^{\pi_\theta}$ are well-defined under the stationary threshold policy π_θ, we have that

$$\hat{\lambda}_k^{\pi_\theta} = \lim_{N\to\infty} \frac{1}{T_N} \sum_{n\in[N]:K_n=k} \mathbb{E}[I_n^{\pi_\theta}] \tag{2.3}$$

$$= \lim_{N\to\infty} \frac{\lambda}{N} \sum_{n\in[N]:K_n=k} \mathbb{E}[I_n^{\pi_\theta}]. \tag{2.4}$$

In the sequel, we will use (2.4) to formulate the optimization problem in terms of long-term time averages.

Once task n is received by client K_n at time T_n, the client processes the task using its L-ML model. Based on the confidence Z_n of the L-ML prediction, the client decides whether to send the task to the server to be classified by the H-ML model. We assume that the H-ML classifies the tasks perfectly, i.e., has 100% inference accuracy, and that the computation time by the client and the server is negligible compared to the communication time needed to offload the task. In addition, we assume that there is a feedback channel between the server and the clients, which is used to transmit parameters required by the algorithm (e.g., inference results and bandit loss of an offload, formally defined later in Section 2.3.2). Since the transmitted data is relatively small, we assume that the delay on the feedback channel is negligible.

2.2.1 Problem Formulation

Given the shared communication resource and the hierarchical inference mechanism, we want to find an offloading policy that maximizes the overall inference accuracy of the system while satisfying the utilization constraint. By substituting (2.4) in (2.2), we write the inference accuracy maximization problem as the minimization

$$\min_{\theta \in \Omega_Z} \quad \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}[Y_n(1 - I_n^{\pi_\theta})] \tag{2.5}$$

$$\text{s.t.} \quad \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}[I_n^{\pi_\theta} X_n] \leq \frac{\gamma}{\lambda}, \tag{2.6}$$

where $Y_n = \mathbb{I}\{C_n \neq \hat{C}_n\}$, e.g., the indicator of an incorrect local classification. If X_n and Y_n were available at each time step before taking the offloading decision, the constrained optimization problem in (2.5) could be solved using the Lyapunov drift minimization technique referred to as the drift-plus-penalty algorithm [81]. However, due to the structure of the hierarchical inference system, X_n and Y_n are only revealed if task n is offloaded. Since they are observed in hindsight, we need a methodology that allows efficient exploration/exploitation of the system's unknown statistics. To this end, we adopt a constrained bandit framework.

Next, we introduce an algorithm based on Lyapunov-EXP4 that makes offloading decision based on a single threshold. In Section 2.3.3, this algorithm is extended to the multiple thresholds case, enabling its distributed implementation.

2.3 Threshold Offloading Policy using Lyapunov-EXP4 (Ly-EXP4)

As discussed in Chapter 4.5 in [81], optimization problems with time average expectation objective and constraints can be solved in a sequential manner using a so called drift-plus-penalty algorithm. Although (2.5) is a long-term optimization problem with time average expectations, since X_n and Y_n are observed after the

offloading decisions, one cannot use drift-plus-penalty algorithm directly to solve the optimization problem (2.5). Instead, to solve this optimization problem, we propose a computationally efficient online learning algorithm Lyapunov-EXP4 (Ly-EXP4) based on the contextual bandits framework. Ly-EXP4 makes decisions using a loss structure that employs the Lyapunov drift minimization technique, maximizing accuracy while satisfying a long-term average expectation constraint.

In the next subsections, we first introduce virtual queue process capturing the utilization constraint and drift-plus-penalty ratio, and then the Lyapunov-EXP4 algorithm for a constrained bandit problem with expert advice.

2.3.1 Virtual Queue and Drift-plus-Penalty Ratio

In this subsection, we introduce a virtual queue capturing the constraint (2.6) and a drift-plus-penalty ratio motivated by [81], which are then utilized to specify the bandit loss. To start, we define the causal policy space for our problem.

Definition 2.2 (Causal Policy)**.** *Let π be a policy that yields a sequence of offloading decisions $\{I_n^\pi \in \{0,1\} : N \geq n \geq 1\}$. Under π, the history until time n is the filtration*

$$\mathcal{F}_n^\pi = \sigma\left(\{Z_{t+1}, I_t^\pi, X_t I_t^\pi, Y_t I_t^\pi : 1 \leq t \leq n\}\right), \tag{2.7}$$

where $\sigma(\cdot)$ denotes the sigma-field, and X_t and Y_t are observed only if task t is offloaded (as implied by the terms $X_t I_t^\pi$ and $Y_t I_t^\pi$). A policy π is said to be causal if π is non-anticipating, i.e., $\{I_n^\pi = a\} \in \mathcal{F}_{n-1}^\pi$ for all a, n.

2.3.1.1 Virtual queue

We define the virtual queue under casual policy π as

$$Q_{n+1}^\pi = \max[Q_n^\pi + I_n^\pi X_n - \frac{\gamma}{\lambda}, 0], \tag{2.8}$$

where $Q_1^\pi = 0$. The time average expectation constraint (2.6) can be viewed as a mean rate stability constraint of a virtual queue [81] under causal policy π. To see how the

mean rate stability of Q_n^π enforces (2.6), note that by the telescoping argument

$$\frac{Q_{N+1}^\pi}{N} - \frac{Q_1^\pi}{N} \geq \frac{1}{N}\sum_{n=1}^{N} I_n^\pi X_n - \frac{\gamma}{\lambda}. \tag{2.9}$$

Taking expectations and letting $N \to \infty$ one can show that

$$\lim_{N\to\infty} \frac{\mathbb{E}[Q_{N+1}^\pi]}{N} \geq \lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}[I_n^\pi X_n] - \frac{\gamma}{\lambda}. \tag{2.10}$$

If Q_n^π is mean rate stable, i.e., $\lim_{N\to\infty} \frac{\mathbb{E}[Q_{N+1}^\pi]}{N} = 0$, then the constraint (2.6) is satisfied.

2.3.1.2 Drift-plus-penalty ratio

With a Lyapunov function $\mathcal{L}(Q_n) = \frac{1}{2}Q_n^2$, for any causal policy π we have

$$\begin{aligned}
&\mathcal{L}(Q_{n+1}^\pi) - \mathcal{L}(Q_n^\pi) \\
&\quad= \frac{1}{2}\left(\max[Q_n^\pi + I_n^\pi X_n - \frac{\gamma}{\lambda}, 0]\right)^2 - \frac{1}{2}Q_n^{\pi 2} &(2.11)\\
&\quad= \frac{1}{2}((I_n^\pi X_n)^2 + (\frac{\gamma}{\lambda})^2) + Q_n^\pi I_n^\pi X_n - Q_n^\pi \frac{\gamma}{\lambda} &(2.12)\\
&\quad\leq B + Q_n^\pi I_n^\pi X_n - Q_n^\pi \frac{\gamma}{\lambda}, &(2.13)
\end{aligned}$$

where $B = \frac{1}{2}(1 + (\frac{\gamma}{\lambda})^2)$.

Let $\mathbb{E}_n[\cdot] = \mathbb{E}[\cdot|\mathcal{F}_n^\pi]$ denote the conditional expectation given the history up to time n. Denote the Lyapunov drift by $\Delta(Q_n^\pi) = \mathbb{E}_{n-1}[\mathcal{L}(Q_{n+1}^\pi) - \mathcal{L}(Q_n^\pi)]$, where Q_n^π is measurable with respect to $\mathcal{F}_{n-1}^\pi$ for all n (π is causal). At each slot, the drift-plus-penalty algorithm opportunistically minimizes an upper bound on the drift (2.13) summed with the one-slot penalty, which in our setting leads to

$$\begin{aligned}
&\Delta(Q_n^\pi) + V\mathbb{E}_{n-1}[Y_n(1 - I_n^\pi)] \\
\leq& B + Q_n^\pi \mathbb{E}_{n-1}[I_n^\pi X_n] + V\mathbb{E}_{n-1}[Y_n(1 - I_n^\pi)] - Q_n^\pi \frac{\gamma}{\lambda}, &(2.14)
\end{aligned}$$

where $V > 0$ is the parameter controlling the trade-off between penalty and drift. Note that if X_n and Y_n were observed before the offloading decision of task n, i.e., if $X_n, Y_n \in \mathcal{F}_{n-1}^{\pi}$, the one-slot optimization problem the drift-plus-penalty algorithm tackles would become

$$\min_{I_n^{\pi} \in \{0,1\}} \quad Q_n^{\pi} I_n^{\pi} X_n / V + Y_n (1 - I_n^{\pi}). \tag{2.15}$$

The aforementioned bandit algorithm Ly-EXP4, formally presented in the next subsection, deploys (2.15) as the loss at time n and exhibits convergence to a near-optimal solution of (2.5).

2.3.2 Ly-EXP4: A Bandit with Expert Advice

In the bandits with expert advice framework, instead of learning an arm selection policy, the agent learns a way of combining experts' predictions that minimizes the regret. One common algorithm to tackle this problem is the EXP4 [58], which randomly selects arms according to a distribution computed by a soft-max operation on the cumulative loss each expert received.

In this subsection we introduce Ly-EXP4, which is an extension to EXP4 for constrained bandit problems. We show that by using a loss structure based on Lyapunov drift minimizing techniques Ly-EXP4 solves the hierarchical inference problem defined in (2.5).

We formulate hierarchical inference as a multi-arm-bandit problem with expert advice; the offloading decisions are denoted by $a \in \{0, 1\}$, where $a = 0$ corresponds to "not offloading" while $a = 1$ corresponds to "offloading". In addition, we assume that there are M experts, each corresponding to a threshold policy as in (2.1) with a threshold $\theta_m \in \mathcal{M},\ m \in [1, \ldots, M]$, where the set $\mathcal{M}$ is created by discretizing the support of the confidence metric Ω_Z. where $\mathcal{M}$ is a set of discretized thresholds on the support of the confidence metric Ω_Z. Note that discretizing the thresholds induces a discretization error; while we do not assume a specific discretization scheme, we

assume that the optimality gap between the best threshold in $\mathcal{M}$ and the optimal (continuous) solution to (2.5) is negligible.

Assumption 2.3. *Let $A_N(\pi)$ denote the accuracy at time N under policy π, i.e.,*

$$A_N(\pi) = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[Y_n(1-I_n^{\pi})]. \tag{2.16}$$

Given a set of discretized thresholds $\mathcal{M}$, there exists at least one threshold $\theta_m \in \mathcal{M}$ such that

$$\lim_{N\to\infty} A_N(\pi_{\theta_m}) - A_N(\pi_{\theta^*}) \leq \epsilon_0, \tag{2.17}$$

where $\theta^* \in \Omega_Z$ is the optimal threshold for problem (2.5).

A simple discretization scheme would be to select uniform discretization intervals, which ensures $\epsilon_0 \leq \frac{1}{M}$. Note that Assumption 2.3 enforces not only a condition on the discretization scheme but also a smoothness property on the distribution of the confidence metric Z_n.

Next, we define bandit loss based on (2.15); for task n, let $l_{n,a}$ denote the loss corresponding to decision $a \in \{0,1\}$,

$$l_{n,a} = \begin{cases} Y_n & a = 0 \ \ \text{(do not offload)}, \\ \dfrac{Q_n X_n}{V} & a = 1 \ \ \text{(offload)}. \end{cases} \tag{2.18}$$

Note that such a loss is a combination of the minimization objective and the utilization constraint. Based on (2.18), the loss associated with the offloading decisions taken by the algorithm, L_n, and the loss associated with the m^{th} expert, $L_{n,m}$, can be written as

$$L_n = l_{n,I_n}, \qquad L_{n,m} = l_{n,I_{n,m}}, \quad \forall n, m, \tag{2.19}$$

where $I_{n,m} = \mathbb{I}\{\theta_m > Z_n\}$ is the offloading decision of expert m for the task n. Since the virtual queue process Q_n is driven by the decisions Ly-EXP4 makes, expert loss

$L_{n,m}$ depends not only on the expert's own decisions $I_{n,m}$ but also on the previous decisions of Ly-EXP4.

The regret that we want to minimize is defined relative to the best expert in hindsight,

$$R_N = \mathbb{E}\left[\sum_{n=1}^{N} L_n - \min_{m\in\mathcal{M}} \sum_{n=1}^{N} L_{n,m}\right]. \tag{2.20}$$

The regret R_N, loss L_n and expert loss $L_{n,m}$ are all incurred under the Ly-EXP4 policy. However, for notational simplicity, we drop superscript π_{LyEXP4} from these expressions.

Ly-EXP4 acts in this bandit setting similarly to EXP4. Depending on Z_n, each expert incurs a loss due to either incorrect local inference or offloading cost. Based on the experts' cumulative loss, Ly-EXP4 computes normalized weight for each expert via soft-max, where a weight represents "contribution" of an expert to the stochastic offloading decision.

Since X_n and Y_n are observed only if a task is offloaded, experts' losses are not always known. To overcome this problem, we employ an importance-weighted estimator of the experts' loss, $\hat{L}_{n,m}$, as defined below. Let $\hat{S}_{n,m} = \sum_{t=1}^{n} \hat{L}_{t,m}$ denote the cumulative estimated loss for expert m until time n. The normalized weight of expert m at time n is defined as

$$\omega_{n,m} = \frac{\exp\left(-\eta \hat{S}_{n-1,m}\right)}{\sum_{m\in\mathcal{M}} \exp\left(-\eta \hat{S}_{n-1,m}\right)}. \tag{2.21}$$

It follows from (2.21) that an expert with a higher loss will be assigned a smaller weight. At time n, Ly-EXP4 takes a random offloading decision with probability p_n equal to the sum of the weights of the experts advising to offload based on the observed confidence Z_n,

$$p_n = \sum_{m:\theta_m > Z_n} \omega_{n,m}. \tag{2.22}$$

Note that this procedure is effectively as same as sampling an expert m at random from distribution $\boldsymbol{\omega}_n = (\omega_{n,m} : m \in \mathcal{M})$ and following the offloading decision of the

selected expert, where the experts with lower cumulative loss have higher probability of being chosen. Since p_n depends on the losses received until time $(n-1)$, we define the importance-weighted loss estimate as

$$\hat{L}_{n,m} = \begin{cases} 0, & I_n = 0 \\ \dfrac{Y_n}{p_n}, & I_n = 1 \text{ and } I_{n,m} = 0 \\ \dfrac{Q_n X_n}{V p_n}, & I_n = 1 \text{ and } I_{n,m} = 1 \end{cases}, \quad \forall n, m. \tag{2.23}$$

If task n is offloaded, the experts incur a loss according to their own offloading decisions $I_{n,m}$; if task n is not offloaded, experts incur zero loss. The next lemma establishes that $\hat{L}_{n,m}$ is an unbiased estimator of $l_{n,I_{n,m}}$.

Lemma 2.4. *Since $\mathbb{E}_{n-1}[\mathbb{I}\{I_n = 1\}] = p_n$ and p_n is $\mathcal{F}_{n-1}$-measurable, $L_{n,m}$ is an unbiased estimator of $l_{n,I_{n,m}}$, i.e.,*

$$\mathbb{E}_{n-1}[\hat{L}_{n,m}] = l_{n,I_{n,m}}. \tag{2.24}$$

The proof of Lemma 2.4 is given in Appendix 2.A.1.

In contrast to EXP4, the offloading decisions in Ly-EXP4 affect the loss in subsequent steps through the virtual queue Q_n, representing the constraint violation. As a consequence of the adopted loss function, when the resource utilization grows the weights of the experts in favor of offloading become smaller. Similarly, when the resource utilization is low, the algorithm tends to offload more due to decreased offloading loss. Ly-EXP4 is formalized as Algorithm 1. In what follows, we prove finite-horizon regret bounds for Ly-EXP4.

Proposition 2.5 (Ly-EXP4 Regret Analysis)**.** *Given the number of experts M, learning rate $\eta > 0$, and $V > 0$, the regret under Ly-EXP4 satisfies*

$$R_N \leq R_{N,m} \leq \frac{\log(M)}{\eta} + \frac{\eta N}{2}\zeta, \quad \forall m \in \mathcal{M}, \tag{2.25}$$

where $\zeta > 2$. Specifically, for $\eta = \sqrt{\frac{2\log(M)}{N\zeta}}$ the regret is

$$R_N = O\left(\sqrt{2N\zeta \log(M)}\right). \tag{2.26}$$

Algorithm 1 Lyapunov - EXP4 (Ly-EXP4)

Input: N, γ, λ, V, $\mathcal{M}$, η
Initialize $Q_1 = 0$, $\hat{S}_{0,m} = 0, \forall m$
for $n = 1, \cdots, N$ **do**
    Observe Z_n,
    $p_n = \dfrac{\sum\limits_{m:\theta_m > Z_n} e^{-\eta \hat{S}_{n-1,m}}}{\sum\limits_{m \in \mathcal{M}} e^{-\eta \hat{S}_{n-1,m}}}, \quad \forall m$
    Choose the offloading decision $I_n \sim p_n$,
    Compute estimates of expert losses $\hat{L}_{n,m}, \ \forall m$
    $\hat{S}_{n,m} = \hat{S}_{n-1,m} + \hat{L}_{n,m}, \ \forall m$
    $Q_{n+1} = \max[Q_n + I_n X_n - \frac{\gamma}{\lambda}, 0]$

The proof is outlined in Appendix 2.A.2.

Note that Ly-EXP4 converges to the threshold minimizing the regret based on the loss formed as a combination of the objective and the problem constraint. Therefore, the best expert's policy Ly-EXP4 converges is not necessarily the optimal threshold policy for problem (2.5). In the next theorem, we prove Ly-EXP4 is asymptotically near-optimal.

Theorem 2.6 (Optimality of Ly-EXP4)**.** *Let π and π_{θ^*} denote the policy of Ly-EXP4 and the optimal threshold policy, respectively. By Assumption 2.3, the regret bound in Proposition 2.5, and for $B = \frac{1}{2}(1 + (\frac{\gamma}{\lambda})^2)$,*

$$A_N(\pi) - A_N(\pi_{\theta^*}) \leq \frac{B}{V} + \frac{R_{N,m'}}{N} + \epsilon_0, \tag{2.27}$$

where m' is the expert with threshold $\theta_{m'} \in \mathcal{M}$ that yields the best solution to problem (2.5). For $\eta = \sqrt{\frac{2\log(M)}{N\zeta}}$ and as $N \to \infty$, the optimality gap between Ly-EXP4 and the optimal threshold policy is

$$\lim_{N \to \infty} [A_N(\pi) - A_N(\pi_{\theta^*})] = O(\frac{1}{V}) + \epsilon_0. \tag{2.28}$$

The proof is provided in Appendix 2.A.3.

As seen from Proposition 2.5, the regret achieved by Ly-EXP4 differs by only a constant factor from EXP4. This means that by Theorem 2.6, Ly-EXP4 solves constrained bandit problems at a convergence rate similar to that of EXP4. Note that increasing M and/or V will decrease the sub-optimality error, but M increases the complexity of the algorithm while V increases the sensitivity to constraint violation and decreases convergence speed.

2.3.3 Extension to Multiple Thresholds

In Section 2.3.2, we provide a regret and discuss an optimality analysis for the setting where a single instance of Ly-EXP4 is used to find a single threshold policy applied by all clients. However, by treating different groups of clients as different "contexts", one can deploy separate instances of Ly-EXP4 for each group of clients to learn a multi-threshold policy for the offloading problem (2.5). At one extreme, each client can use a separate instance of Ly-EXP4 to find an individual threshold policy, which eventually allows our algorithm to be used in a distributed manner.

In what follows, we show the regret for the multi-threshold case based on the principles outlined in Chapter 18.1 of [58]. Let $g \in \mathcal{G}$ denote the set of disjoint groups of clients, and G_n denote the group task n belongs. By Proposition 2.5, the regret of the Ly-EXP4 instance executed for group $g \in \mathcal{G}$ is

$$R_N^{(g)} \leq \sqrt{2 \sum_{n=1}^{N} \mathbb{I}\{G_n = g\} \zeta \log(M)}, \tag{2.29}$$

where the sum under the square root counts the number of tasks that belong to group $g \in \mathcal{G}$. Since the number of tasks received by the clients in a group is not known in advance, we use the changing learning rate encountered in the anytime version of EXP4,

$$\eta_n^{(g)} = \sqrt{\frac{\log(M)}{\zeta \sum_{t=1}^{n} \mathbb{I}\{G_t = g\}}}. \tag{2.30}$$

Defining R_N as the sum of individual regrets across all groups, the regret for multi-threshold case can be bounded as

$$R_N \leq \sum_{g \in \mathcal{G}} R_N^{(g)} \tag{2.31}$$

$$\leq \sum_{g \in \mathcal{G}} \sqrt{2 \sum_{n=1}^{N} \mathbb{I}\{G_n = g\} \zeta \log(M)} \tag{2.32}$$

$$\leq \sqrt{2N|\mathcal{G}|\zeta \log(M)}, \tag{2.33}$$

where (2.33) is the worst-case regret corresponding to the setting where each group receives same number of tasks.

As seen from (2.33), the regret bound for multiple threshold case scales up with the square root of the number of client groups. As discussed in Section 2.4, although Ly-EXP4 learns a multiple threshold policy slower than a single threshold policy, it achieves significantly better accuracy with a multiple threshold policy by exploiting the heterogeneity in service time and confidence metric distributions across client groups.

2.4 Simulation Results

We conducted extensive simulations running Ly-EXP4 in both single and multiple thresholds settings on real and synthetic models. Specifically, we consider a hierarchical inference system with 4 client groups – with 25 clients each, each experiencing different service times due to various factors including distance to the server and channel quality. For this system, we report results that demonstrate convergence of Ly-EXP4 as well as delay-accuracy and fairness-accuracy tradeoffs.

In the synthetic model, client $k \in \mathcal{K}^{(g)}$ receives tasks according to a Poisson process with rate λ_k, where $\mathcal{K}^{(g)}$ denotes the set of clients in client group $g \in \mathcal{G}$. The true class C_n is assigned to each task according to a client-specific class distribution $\boldsymbol{\alpha}_k$, where the arrival rates λ_k, and the class probabilities $\alpha_k(c),\ c \in \mathcal{C}$ are sampled

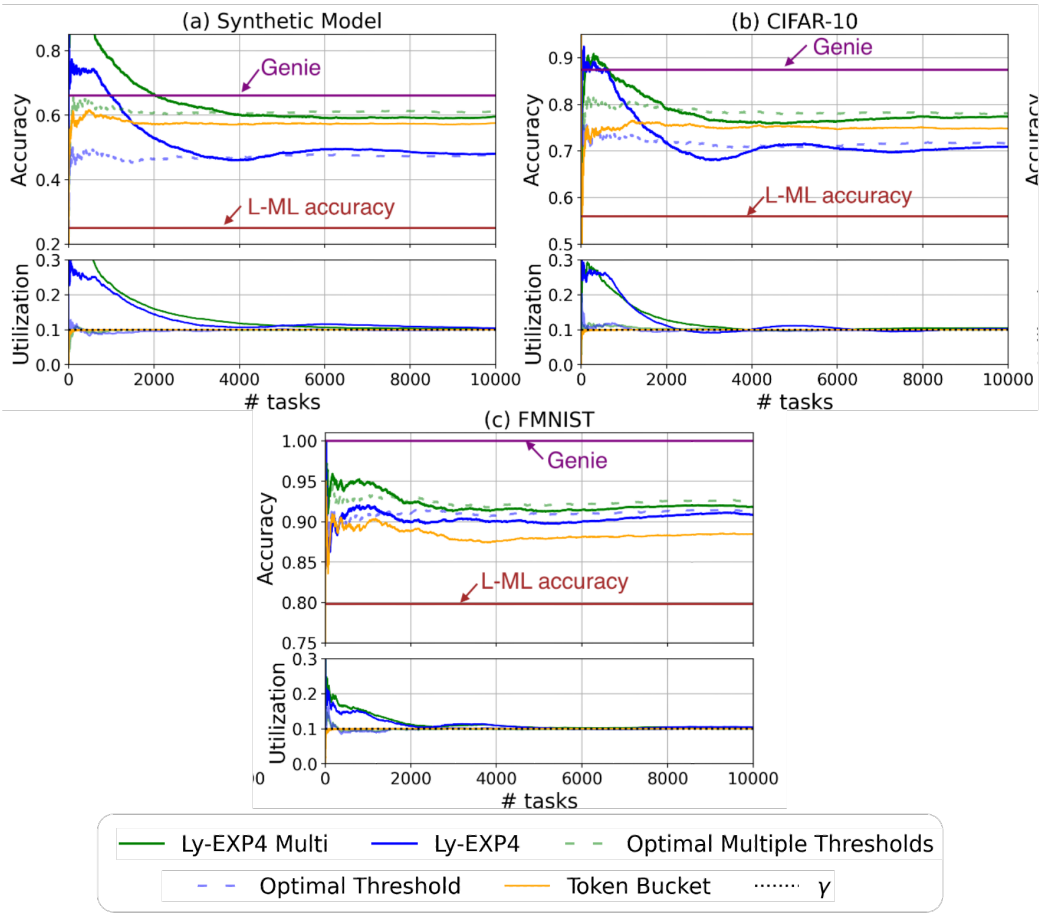


Figure 2.1: Sample accuracy and utilization over tasks for (a) the synthetic model, (b) CIFAR-10, and (c) FMNIST. The dotted horizontal line marks utilization constraint γ. The learning rate η follows Theorem 2.6 and (2.30); in all cases $V = 30$, $\gamma = 0.1$ and $\lambda = 1$.

from a uniform distribution with a support $[0, 1]$ and normalized so that total arrival rate $\lambda = \sum_{g\in\mathcal{G}} \sum_{k\in\mathcal{K}^{(g)}} \lambda_k = 1$ and $\sum_{c\in\mathcal{C}} \alpha_k(c) = 1$. A synthetic classifier model for L-ML is characterized by a confusion matrix, denoted by H, where each element represents the likelihood of the L-ML model predicting class c' given that the true class is c, i.e., $H_{c,c'} = \mathbb{P}(\hat{C}_n = c' | C_n = c)$. The elements of H are generated uniformly at random from $[0, 1]$, and the diagonal elements are adjusted such that the model is typically most confident about the true class. Each row is then normalized so that $\sum_{c'\in\mathcal{C}} H_{c,c'} = 1, \ \forall c \in \mathcal{C}$.

In addition to the above, confidence Z_n is sampled from a distribution depending on the correctness of the inference,

$$Z_n \sim \begin{cases} \text{Uniform}(0.1, 0.9), & \text{if } C_n \neq \hat{C}_n, \\ \text{Uniform}(0.5, 1), & \text{if } C_n = \hat{C}_n. \end{cases} \tag{2.34}$$

In the case of real datasets and models, the tasks are assigned to the clients with probability λ_k/λ regardless of the true class of a task; confidence Z_n is observed as the maximum of the last soft-max layer of the classifier model. We utilize relatively small models that are deployable on IoT devices, including a simple neural network with a single dense layer applied to FMNIST dataset [106], and a LeNet-5 model [59] applied to CIFAR-10 dataset [56].

The clients in a client group experience random service times with group-specific distributions. We assign equally-separated mean service times $[0.2, 0.4, 0.6, 0.8]$ to the client groups. The service times of a client in group g are distributed according to a beta distribution with mean μ_g^{-1} and variance 0.01. The service times for synthetic and real dataset/model cases are generated the same way.

To our knowledge, this chapter presents the first study focusing on the hierarchical inference under utilization constraint. To benchmark the proposed Ly-EXP4 algorithm, we compared it with the optimal threshold policies and a simple token bucket policy. When running Ly-EXP4, confidence support Ω_Z is uniformly discretized using $M = 1000$ thresholds. To characterize upper performance limits, we introduce a genie algorithm: a non-causal offline algorithm with access to the ground truth, capable of offloading the maximum number of incorrectly classified tasks at the lowest cost within the constraint. We also compute the optimal single and multiple threshold policies using an exhaustive search over $[1/C, 1]^{|\mathcal{G}|}$. The aforementioned token bucket policy makes greedy offloading decisions based on the service times of the tasks and the token bucket increments by $\frac{\gamma}{\lambda}$ at each task arrival. It offloads task n if the token bucket has more tokens than the service time X_n. When task n is offloaded,

X_n gets deducted from the token bucket. Therefore, it is aware of the service times before offloading.

2.4.1 Convergence of Ly-EXP4

Figure 2.1 shows the average accuracy and utilization achieved by Ly-EXP4 and the benchmarks on both synthetic and real datasets/models. The service times, arrival rates and utilization constraint parameter γ are kept constant across the simulations; however, inference confidences, L-ML accuracy of the models, and task class distributions $\boldsymbol{\alpha}_k$ depend on the dataset and model used. From these graphs, it is evident that both versions of the Ly-EXP4 algorithm successfully converge to their respective optimal solutions within the specified constraint. Additionally, it is observed that Ly-EXP4 achieves higher accuracy with a slightly slower convergence speed by exploiting heterogeneity across clients while meeting the utilization constraint.

Note that in some scenarios the token bucket algorithm achieves accuracy similar to Ly-EXP4 with multiple thresholds (see Figure 2.1a), while in other scenarios it performs worse than Ly-EXP4 with single threshold (see Figure 2.1c). This inconsistent performance of the token bucket algorithm is due to its greedy policy that prioritizes the clients with small service times. If those clients' L-ML have low local accuracy, token bucket achieves very good performance; otherwise, it offloads many correctly classified task and thus uses the resources inefficiently. Ly-EXP4 avoids this by making decisions based on the inference confidence.

2.4.2 Delay-Accuracy Tradeoff

We further investigate the accuracy-delay trade-off relying on the monotonic relation between utilization and average delay that is typical of scheduling policies. We simulate a network scenario where at most one task from each client can be offloaded using a shared communication channel under a processor sharing scheduling. If a client has an ongoing offload, any new tasks received by the client wait in a client-

specific FCFS queue. If a task is offloaded, the delay includes the waiting time in the client's FCFS queue and the transmission time on the processor sharing channel; if a task is not offloaded, its delay is zero. The communication time of task n depends not only on the random service time X_n but also on other tasks simultaneously being offloaded.

The level of communication between clients and the server depends on whether the algorithm is used in distributed or centralized settings. In a centralized setting the server makes offloading decisions and thus the clients need to send confidence of each task to the server. In a distributed setting, separate instances of the algorithm run on each client and thus the server needs to send the bandit loss and the true class of offloaded tasks to the corresponding clients. Clearly, distributed framework is a more scalable solution.

Ly-EXP4 updates its weights based on the loss it receives after each offload, which is acceptable for scheduling policies where offloads are performed sequentially (e.g., FCFS). However, this poses a problem for scheduling policies allowing simultaneous offloads such as ours, since some offloading decisions may be taken based on outdated parameters. It was shown in [12] that delayed feedback in EXP3 causes an increase in the regret bound which scales with the number of decisions based on outdated parameters. However, the effect of delayed feedback in our settings is negligible since the offloading decisions for the upcoming tasks are made only after the ongoing offload is completed.

For increasing utilization constraint, corresponding average delay and accuracy results are illustrated together in Figure 2.2. One can observe diminishing returns in accuracy with respect to delay. Moreover, the accuracy improvements obtained by utilizing multiple thresholds is more significant when the utilization constraint is low, implying that using multiple thresholds is more beneficial when the resources are scarce and thus more efficient offloading decisions are needed.

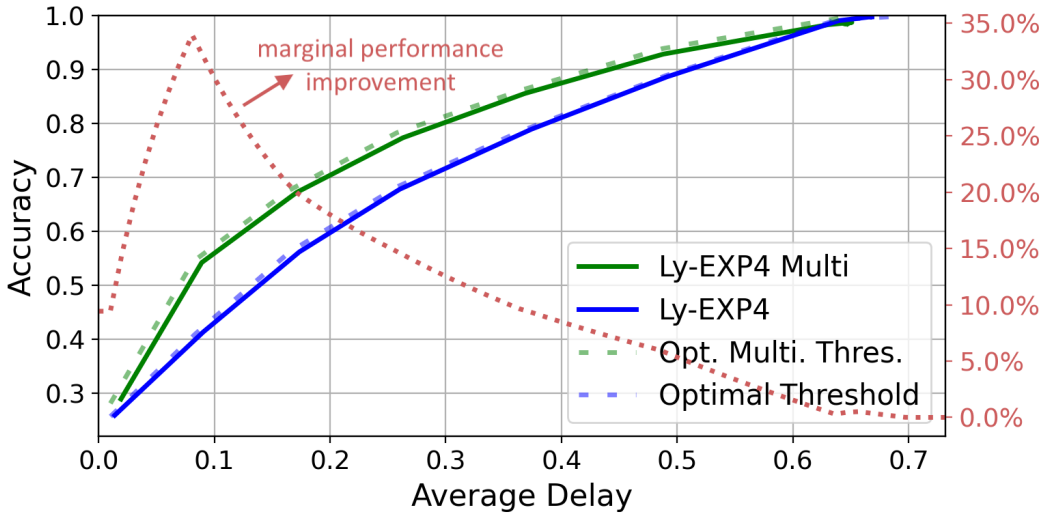


Figure 2.2: Average delay vs. accuracy under increasing utilization constraint for Ly-EXP4 with single and multiple thresholds (solid), together with the corresponding optimal threshold benchmarks (dashed). The red dotted curve is computed as the percentage accuracy gain of the multiple-threshold Ly-EXP4 policy over the single-threshold policy, i.e., $(A_{\text{multi}} - A_{\text{single}})/A_{\text{single}} \times 100\%$.

2.4.3 Fairness-Accuracy Tradeoff

As the results show, Ly-EXP4 with multiple thresholds generally achieves higher accuracy than the single threshold strategy. This is accomplished by exploiting the heterogeneity in service time, and task and class distributions across clients. We complement those results by characterizing fairness as quantified by the Jain's index

$$\mathcal{J}(A_N^{(1)}, A_N^{(2)}, \ldots, A_N^{(K)}) = \frac{(\sum_{k=1}^{K} A_N^{(k)})^2}{K \sum_{k=1}^{K} A_N^{(k)^2}}, \tag{2.35}$$

where $A_N^{(k)} = \frac{1}{N_k} \sum_{n=1}^{N} Y_n(1 - I_n)\mathbb{I}\{K_n = k\}$ is the sample accuracy of client k and N_k is the number of tasks it receives.

Figure 2.3 shows that Ly-EXP4 with a single threshold is almost perfectly fair across clients, which also means that a single instance of Ly-EXP4 is fair to the clients within a group. As the number of client groups grows, one observes an improvement in the overall accuracy at the expense of fairness. This allows flexibility

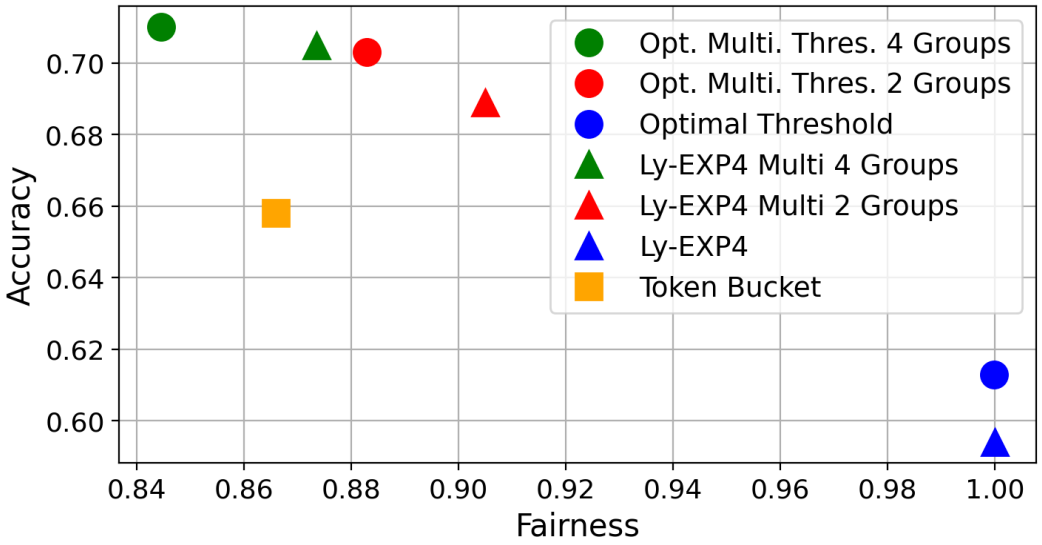


Figure 2.3: Accuracy vs. fairness, measured by Jain's index of the per-client sample accuracies, for single-threshold Ly-EXP4, multiple-threshold Ly-EXP4 with two and four client groups, the corresponding optimal threshold policies, and the token bucket baseline.

when utilizing Ly-EXP4 in different settings. By comparison, the token bucket policy achieves fairness similar to Ly-EXP4 with 4 groups. However, token bucket cannot provide an option of trading fairness for accuracy the way Ly-EXP4 can.

2.5 Conclusion

In this chapter, we proposed a low-complexity online offloading algorithm that we refer to as Lyapunov-EXP4 (Ly-EXP4). Ly-EXP4 combines Lyapunov drift minimization techniques and the bandits with expert advice framework to maximize accuracy in a hierarchical system with multiple clients, converging to an optimal threshold policy within the utilization constraint. We proved a sublinear regret bound and near-optimality of Ly-EXP4; the proposed framework and presented analysis may be relevant to a broader class of constrained bandit problems. The convergence of Ly-EXP4 and the delay-accuracy-fairness tradeoffs achievable in hierarchical inference systems are demonstrated via simulations on real and synthetic datasets and models.

Future research directions include settings with multiple servers having varying communication capacities and model accuracy, as well as offloading policies that exploit heterogeneity across task classes in addition to clients. Fairness-performance tradeoffs in such systems provide another potentially interesting direction.

2.A Appendix

2.A.1 Proof of Lemma 2.4

The loss estimate (2.23) can be re-written as

$$\hat{L}_{n,m} = \frac{l_{n,I_{n,m}} \mathbb{I}\{I_n = 1\}}{p_n}, \quad \forall n, m. \tag{2.36}$$

Since $\mathbb{E}_{n-1}[\mathbb{I}\{I_n = 1\}] = p_n$,

$$\mathbb{E}_{n-1}[\hat{L}_{n,m}] = \mathbb{E}_{n-1}\left[\frac{l_{n,I_{n,m}} \mathbb{I}\{I_n = 1\}}{p_n}\right] \tag{2.37}$$

$$= \frac{l_{n,I_{n,m}}}{p_n} \mathbb{E}_{n-1}\left[\mathbb{I}\{I_n = 1\}\right] \tag{2.38}$$

$$= l_{n,I_{n,m}}, \tag{2.39}$$

which shows $\hat{L}_{n,m}$ is indeed an unbiased estimate of $l_{n,I_{n,m}}$.

2.A.2 Proof of Proposition 2.5

Lemma 2.7 (Bounded first and second moment of Q_n)**.** *For $V > 0$ and a lower bound on service times $x_{min} \leq X_n$, $\forall n$, the virtual queue process under Ly-EXP4 policy has bounded first and second moment*

$$\begin{aligned} \mathbb{E}[Q_n] &= O(\tfrac{V}{x_{min}}), \\ \mathbb{E}[Q_n^2] &= O((\tfrac{V}{x_{min}})^2). \end{aligned} \tag{2.40}$$

Proof Sketch. This proof is inspired by [39] and the extensions in [18] which allow us to bound the moment generating function of Q_n. We refer the reader to Lemma 7 in [18] for the intermediate steps of the proof.

From (2.8), for some $l^* > \frac{\gamma}{\lambda}$, $\beta^* < \frac{\gamma}{\lambda}$ and $\epsilon > 0$, we obtain

$$\mathbb{E}_{n-1}[Q_{n+1} - Q_n \mathbb{I}\{Q_n \geq l^*,\ p_n < \beta^*\}] \leq -\epsilon. \tag{2.41}$$

By using the Taylor expansion on $\mathbb{E}_{n-1}[e^{\eta_0(Q_{n+1}-Q_n)}]$, the negative drift (2.41), and due to the fact $|Q_{n+1} - Q_n| < 1$ because X_n is bounded and Y_n is in $(0, 1]$, for

some $\eta_0 < 1$ and $\rho_0 = 1 - \epsilon\eta_0 + \eta_0^2(e-2)$ it holds that

$$\mathbb{E}_{n-1}\left[e^{\eta_0(Q_{n+1}-Q_n)}\mathbb{I}\{Q_n \geq l^*,\ p_n < \beta^*\}\right] \leq \rho_0 < 1, \tag{2.42}$$

$$\mathbb{E}_{n-1}\left[e^{\eta_0(Q_{n+1}-Q_n)}\mathbb{I}\left\{Q_n < l^*\right\}\right] \leq e^{\eta_0}. \tag{2.43}$$

Let $x_{\min}$ denote a lower bound on X_n, i.e., $X_n \geq x_{\min} > 0,\ \forall n$. Defining $l^* = \frac{V}{x_{\min}}$, we argue that $\{Q_n \geq l^*,\ p_n \geq \beta^*\}$ is a low-probability event, i.e.,

$$\mathbb{P}\{Q_n \geq l^*,\ p_n \geq \beta^*\} \leq \delta_0. \tag{2.44}$$

Let L_{n,m_a} be the loss of any expert m with offloading decision $I_{n,m} = a$. The low-probability event argument stems from the fact that by the definition of the expert weights and expert losses, if $Q_n \geq \frac{V}{x_{\min}}$, then $L_{n,m_1} \geq L_{n,m_0}$ for all m, and the weight distribution $\boldsymbol{\omega}_n$ shifts towards smaller values, making the algorithm less likely to offload.

Following the steps in [18], and using (2.42), (2.43) and (2.44), leads us to the inequality

$$\mathbb{E}_0\left[e^{\eta_0 Q_{n+1}}\right] \leq \rho\mathbb{E}_0\left[e^{\eta_0 Q_n}\right] + e^{\eta_0(l^*+1)}, \tag{2.45}$$

where $\rho = \rho_0 + \sqrt{\delta_0(\frac{1}{1-\psi}+1)} < 1$ and $\psi = \frac{1}{4}(e^{3\eta_0} - e^{\eta_0})$. Taking telescoping sum over n yields

$$\mathbb{E}_0\left[e^{\eta_0 Q_n}\right] \leq \rho^n e^{\eta_0 Q_1} + \frac{1-\rho^n}{1-\rho}e^{\eta_0(l^*+1)}. \tag{2.46}$$

Applying the Chernoff bound, we obtain a bound on the cumulative CDF of Q_n,

$$\mathbb{P}\left[Q_n \geq q\right] \leq \rho^n e^{\eta_0 Q_1 - q} + \frac{1-\rho^n}{1-\rho}e^{\eta_0(l^*+1-q)}. \tag{2.47}$$

Since Q_n is non-negative for all n and $Q_1 = 0$, for $\eta_0 = \frac{2}{l^*} < 1$ we obtain a bound on the second moment

$$\mathbb{E}[Q_n^2] = 2\int_0^\infty q\mathbb{P}\left[Q_n \geq q\right] dq \tag{2.48}$$

$$\leq \frac{2}{\eta_0^2}(\rho^n + \frac{1-\rho^n}{1-\rho})e^{\eta_0(l^*+1)} \tag{2.49}$$

$$= O(l^{*2}). \tag{2.50}$$

Setting $l^* = \frac{V}{x_{\min}}$ completes the proof. The proof for the first moment follows similar procedure. □

Proof of the Regret Bound. Let us define the expected regret relative to expert m as

$$R_{N,m} = \mathbb{E}\left[\sum_{n=1}^{N} L_n - \sum_{n=1}^{N} l_{n,I_{n,m}}\right]. \tag{2.51}$$

To prove the claim, we will bound $R_{n,m}$ for all m, including the best expert. Let $\hat{S}_{n,m} = \sum_{t=1}^{n} \hat{L}_{t,m}$ denote the cumulative estimated loss for expert m. For the sake of convenience and brevity, we introduce $p_{n,a} = \mathbb{P}\{I_n = a\}$, $\forall a \in \{0,1\}$, where $p_{n,1} = p_n$ and $p_{n,0} = 1 - p_n$. Let $\hat{L}_{n,m_a}$ be the loss estimate of any expert m with offloading decision $I_{n,m} = a$. From Lemma 2.4,

$$\mathbb{E}[\hat{S}_{n,m}] = \sum_{t=1}^{n} l_{t,I_{t,m}}, \tag{2.52}$$

$$\mathbb{E}_{n-1}[L_n] = \sum_{a\in\{0,1\}} p_{n,a} l_{n,a} = \sum_{a\in\{0,1\}} p_{n,a}\mathbb{E}_{n-1}[\hat{L}_{n,m_a}]. \tag{2.53}$$

By the tower rule for conditional expectations and (2.53),

$$R_{N,m} = \mathbb{E}[\sum_{n=1}^{N}\sum_{a\in\{0,1\}} p_{n,a}\mathbb{E}_{n-1}[\hat{L}_{n,m_a}]] - \mathbb{E}[\hat{S}_{N,m}] \tag{2.54}$$

$$= \mathbb{E}[\sum_{n=1}^{N}\sum_{a\in\{0,1\}} p_{n,a}\hat{L}_{n,m_a}] - \mathbb{E}[\hat{S}_{N,m}] \tag{2.55}$$

$$= \mathbb{E}[\hat{S}_N - \hat{S}_{N,m}]. \tag{2.56}$$

Let us introduce $W_n = \sum_{m=1}^{M} \exp\left(-\eta \hat{S}_{n,m}\right)$ and note that for all experts $\hat{S}_{0,m} = 0$. Due to $\hat{S}_{n,m} \geq 0$, $\forall m, n$,

$$\begin{aligned} \exp\left(-\eta\hat{S}_{N,m}\right) &\leq \sum_{m=1}^{M} \exp\left(-\eta\hat{S}_{N,m}\right) = W_N \\ &= M\prod_{n=1}^{N}\frac{W_n}{W_{n-1}} \end{aligned} \tag{2.57}$$

The ratio in the product above can be written in terms of $p_{n,a}$,

$$\frac{W_n}{W_{n-1}} = \sum_{m=1}^{M} \frac{\exp\left(-\eta \hat{S}_{n-1,m}\right)}{W_{n-1}} \exp\left(-\eta \hat{L}_{n,m}\right) \tag{2.58}$$

$$= \sum_{a\in\{0,1\}} p_{n,a} \exp\left(-\eta \hat{L}_{n,m_a}\right) \tag{2.59}$$

By using the fact that $1 - x \leq \exp(-x) \leq 1 - x + \frac{x^2}{2}$, where the lower bound is valid $\forall x \in \mathbb{R}$ and the upper bound is valid $\forall x \geq 0$,

$$\frac{W_n}{W_{n-1}} \overset{\text{(a)}}{\leq} \sum_{a\in\{0,1\}} p_{n,a}(1 - \eta \hat{L}_{n,m_a} + \frac{\eta^2}{2} \hat{L}^2_{n,m_a}) \tag{2.60}$$

$$\overset{\text{(b)}}{\leq} \exp(-\eta \sum_{a\in\{0,1\}} p_{n,a} \hat{L}_{n,m_a} + \frac{\eta^2}{2} \sum_{a\in\{0,1\}} p_{n,a} \hat{L}^2_{n,m_a}), \tag{2.61}$$

where in (a) the upper bound and in (b) the lower bound of $\exp(-x)$ are used. When we substitute (2.60) in (2.57), we obtain

$$e^{-\eta \hat{S}_{N,m}} \leq M \exp(-\eta \sum_{n=1}^{N} \sum_{a\in\{0,1\}} p_{n,a} \hat{L}_{n,m_a} + \frac{\eta^2}{2} \sum_{t=1}^{N} \sum_{a\in\{0,1\}} p_{n,a} \hat{L}^2_{n,m_a}) \tag{2.62}$$

$$= M \exp(-\eta \hat{S}_N + \frac{\eta^2}{2} \sum_{t=1}^{N} \sum_{a\in\{0,1\}} p_{n,a} \hat{L}^2_{n,m_a}) \tag{2.63}$$

By taking the logarithm of both sides and rearranging the terms, it follows that

$$\hat{S}_N - \hat{S}_{N,m} \leq \frac{log(M)}{\eta} + \frac{\eta}{2} \sum_{n=1}^{N} \sum_{a\in\{0,1\}} p_{n,a} \hat{L}^2_{n,m_a}. \tag{2.64}$$

Taking the expectation gives us a bound on regret relative to any expert,

$$R_{N,m} \leq \frac{log(M)}{\eta} + \frac{\eta}{2} \mathbb{E}[\sum_{n=1}^{N} \sum_{a\in\{0,1\}} p_{n,a} \hat{L}^2_{n,m_a}]. \tag{2.65}$$

Finally, we need to bound the last term on the right side. Since $\mathbb{E}_{n-1}[\hat{L}_{n,m}] = l_{t,I_{n,m}}$,

$$\begin{aligned}
\mathbb{E}[\sum_{n=1}^{N}\sum_{a\in\{0,1\}} p_{n,a}\hat{L}_{n,m_a}^2] &= \mathbb{E}[\sum_{n=1}^{N}\sum_{a\in\{0,1\}} p_{n,a}\mathbb{E}_{n-1}[\hat{L}_{n,m_a}^2]] \\
&= \mathbb{E}[\sum_{n=1}^{N}(p_{n,0}l_{n,0}^2 + p_{n,1}l_{n,1}^2)] \\
&\leq N + \sum_{n=1}^{N}\mathbb{E}[p_{n,1}(\frac{Q_nX_n}{V})^2] \\
&\leq N + \frac{1}{V^2}\sum_{n=1}^{N}\mathbb{E}[Q_n^2] \\
&\leq N\zeta, \qquad (2.66)
\end{aligned}$$

where the last inequality follows from Lemma (2.7) and $\zeta = O(\frac{1}{x_{\min}})+1$ is a constant. Substituting (2.66) in (2.65) completes the proof. □

2.A.3 Proof of Suboptimality of Ly-EXP4

Let π denote the Ly-EXP4 policy. From (2.14),

$$\begin{aligned}
&\Delta(Q_n^\pi) + V\mathbb{E}_{n-1}[Y_n(1-I_n^\pi)] \\
&\qquad \leq B + Q_n^\pi\mathbb{E}_{n-1}[I_n^\pi X_n] + V\mathbb{E}_{n-1}[Y_n(1-I_n^\pi)] - Q_n^\pi\frac{\gamma}{\lambda} \qquad (2.67)\\
&\qquad = B + \mathbb{E}_{n-1}[L_n] - Q_n^\pi\frac{\gamma}{\lambda}, \qquad (2.68)
\end{aligned}$$

where the last equation stems from the definition of L_n. By the telescoping argument and taking the expectation we obtain

$$\begin{aligned}
&\mathbb{E}[\mathcal{L}(Q_N^\pi)] + V\sum_{n=1}^{N}\mathbb{E}[Y_n(1-I_n^\pi)] \\
&\qquad \leq NB + V\mathbb{E}[S_N] - \frac{\gamma}{\lambda}\sum_{n=1}^{N}E\ [Q_n^\pi]. \qquad (2.69)
\end{aligned}$$

Let $\theta_{m'} \in \mathcal{M}$ denote the best discretized threshold for the optimization problem (2.5). For brevity, we denote threshold policy π_θ by (θ). Using the regret bound relative to

$\theta_{m'}$ in (2.69) yields

$$\begin{aligned}
&\mathbb{E}[\mathcal{L}(Q_N^\pi)] + V\sum_{n=1}^{N}\mathbb{E}[Y_n(1-I_n^\pi)] \\
\leq & NB + V\mathbb{E}[S_{N,m'}] + VR_{N,m'} - \frac{\gamma}{\lambda}\sum_{n=1}^{N} E\ [Q_n^\pi] \qquad (2.70)\\
= & NB + \sum_{n=1}^{N}\mathbb{E}[Q_n^\pi]\mathbb{E}[I_n^{(\theta_{m'})}X_n]\ + V\sum_{n=1}^{N}\mathbb{E}[Y_n(1-I_n^{(\theta_{m'})})] \\
& + VR_{N,m'} - \frac{\gamma}{\lambda}\sum_{n=1}^{N} E\ [Q_n^\pi] \qquad (2.71)\\
\leq & NB + V\sum_{n=1}^{N}\mathbb{E}[Y_n(1-I_n^{(\theta_{m'})})] + VR_{N,m'}, \qquad (2.72)
\end{aligned}$$

where in the last inequality we use the fact that $\theta_{m'}$ satisfies the utilization constraint (2.6). Since $\mathcal{L}(Q_N^\pi) \geq 0$ and by Assumption 2.3, rearranging the terms and dividing both sides by NV leads to

$$A_N(\pi) - A_N(\theta^*) \leq \frac{B}{V} + \frac{R_{N,m'}}{N} + \epsilon_0, \qquad (2.73)$$

where $A_N(\pi)$ and $A_N(\theta^*)$ are the average accuracy at time N under Ly-EXP4 and the optimal threshold policy, respectively. According to Proposition 2.5, Ly-EXP4 converges to a threshold policy corresponding to the best expert in the bandit setting. Therefore, the limit $\lim_{N\to\infty} A_N(\pi)$ exists, which completes the proof.

Chapter 3: Population-Level Concept Drift Detection in Large Scale Deployment under Resource-Constrained Feedback

3.1 Introduction

A fundamental challenge in deploying machine learning (ML) models is that the underlying data distribution may change over time, gradually degrading post-deployment performance. This phenomenon, commonly referred to as *concept drift*, has received sustained attention over the past two decades [34, 68, 43] and motivates monitoring mechanisms that can detect performance degradation and initiate corrective action.

Recent advances in power-efficient hardware for on-device inference and computationally efficient ML architectures [93, 42] have enabled ML applications to be deployed across large populations of edge clients. At the same time, large-scale deployments introduce challenges for concept drift detection that are not adequately captured by the commonly studied single-stream settings [112]. When an ML model is deployed across a heterogeneous population, clients may experience performance degradation asynchronously. Consequently, the relevant operational question is often not whether drift has occurred somewhere in the system, but whether degradation has become sufficiently widespread to justify a system-level intervention. Since global model updates, retraining, or redeployment may incur nontrivial operational costs, one may wish to trigger such actions only after a sufficiently large number of clients have experienced significant degradation, unless safety or security considerations require an immediate response.

Moreover, many drift detection methods require labeled production data. Such data can be costly, slow, and difficult to obtain at scale, particularly when annotation is performed by human experts. Model-based annotation methods have therefore

been proposed as a scalable alternative, in which a larger and more reliable ML model that is highly accurate and robust to concept drift is used to generate surrogate labels for newly observed data [95]. This perspective is particularly natural in Hierarchical Inference Systems [9, 61, 78, 4], where clients perform inference using a small local model, often derived from a larger server-side model through techniques such as quantization, distillation, or pruning, while selectively offloading tasks to the server for more accurate predictions. Although such systems are proposed as scalable deployment architectures that trade off accuracy against offloading cost, the same offloading mechanism can also be used to provide feedback for monitoring the client-side models. Regardless of whether feedback is obtained through model-based annotation or human annotation, monitoring remains constrained by the cost of collecting feedback at scale, including the associated communication, computation, and latency overheads.

Together, these considerations make concept drift detection in large populations a challenging problem in which the statistical objective, the data-collection policy, and the detection mechanism are inherently coupled. Thus, a monitoring policy must account for distributional heterogeneity and asynchronous degradation across clients while allocating limited data-collection and annotation resources in a sample-efficient and scalable manner.

In this chapter, we study the problem of detecting concept drift that causes a reduction in accuracy greater than an operational tolerance across a sufficiently large fraction of a heterogeneous population, which we refer to as *fractional change detection*. We consider a setting in which feedback about the performance of a lightweight classifier deployed across clients is collected sequentially, one selected client at a time, by having a large server-side model evaluate the selected clients' local classification tasks. This formulation leads to two complementary classes of monitoring policies: individual-statistics-based detectors, which track client-level evidence, and population-statistics-based detectors, which aggregate observations to infer population-level degradation.

3.1.1 Contributions

Our contributions are summarized as follows:

- Based on the classical change point detection framework, we formulate the *fractional change detection* problem, where the aim is to detect concept drift affecting a significant fraction of a population by a significant amount under limited observability, i.e., when the feedback is available sequentially only for one selected client at a time.

- We propose an individual-statistics-based policy, *Greedy Partial Sum-CUSUM*, which maintains client-specific statistics and uses a greedy client-sampling policy that probes the client closest to the decision boundary at each time step. Our analysis shows that this policy is effective for detecting isolated and sparse changes, but scales poorly for large populations due to the dependence of its average detection delay on the population size.

- We also propose a population-statistics-based policy, *Adaptive Global-CUSUM*, which tracks a single population-level statistic updated using a mixture model under a uniform client-sampling policy. Our analysis shows that this policy is especially suitable for detecting widespread degradations in large-scale systems.

- Finally, we provide simulation results under a range of realistic and synthetic concept drift scenarios, which support the theoretical analysis and validate the applicability of the proposed policies.

3.1.2 Related Work

Hierarchical Inference

Hierarchical inference has been proposed as an edge computing paradigm for machine learning deployments [77] that leverages client-side compute resources to reduce inference workload on datacenters. Recent work has explored selective task

offloading based on the *confidence* of local predictions in order to balance accuracy against offloading costs, including latency, energy, and computation [9, 61, 78, 4]. In contrast, this chapter focuses on exploiting limited offloading capacity for drift detection, a complementary use case in hierarchical inference systems that, to the best of our knowledge, has not been studied previously.

Concept Drift Detection in Multi-stream Settings

Recent work has studied concept drift detection in multi-stream settings across several contexts [49, 112, 72, 16, 114, 111]. Early efforts arose in federated learning [72, 16], where local drift detectors at each client identify drift that may harm the global model. More recent studies consider drift detection as part of adaptation in graph learning and multi-stream mixture of experts models [114, 111]. Although these works exploit inter-stream correlations, they neither provide a principled framework for multi-stream drift detection nor address limited data collection and scalability. The closest related work is [112], but it assumes continuous access to annotated data and full observability of all streams, which limits its applicability to large-scale systems.

Change Point Detection with Sampling Constraints

The classical *change point detection* problem, formulated within the sequential hypothesis testing framework, has been extended to multi-stream settings under heterogeneous distributions and asynchronous change points, both when the pre- and post-change distributions are known [74, 99, 20, 28] and when the post-change distribution is unknown [107, 98]. More recent work has considered change point detection under sampling constraints [102, 22, 109, 110]. However, these studies focus on detecting the earliest change in the population. The closest related work is [117], which studied the problem of detecting changes affecting a large fraction of nodes in a network, but without sampling constraints. To the best of our knowledge, this chapter is

the first to study fractional change detection with active sampling control. Although we motivate it through concept drift detection in hierarchical inference systems, the formulation applies more broadly.

3.1.3 Organization

The rest of this chapter proceeds as follows. Section 4.2 presents the system model and formulates the fractional change detection problem. Sections 3.3 and 3.4 introduce and compare the proposed monitoring policies. Finally, Section 3.5 evaluates the policies through simulations, and Section 3.6 concludes the chapter.

3.2 System Model and Problem Formulation

We consider a hierarchical inference system with a set of N clients, denoted $\mathcal{N} = \{1, \ldots, N\}$. Each client $i \in \mathcal{N}$ runs a lightweight ML model to classify a stream of incoming tasks locally, and the underlying data distributions may differ across clients. Accordingly, the deployed client models have heterogeneous initial accuracies, i.e., probability of making a correct inference on their local distributions, denoted by $(p_i)_{i\in\mathcal{N}}$, which are assumed to be known at deployment.

We adopt the standard change-detection framework given by a slotted-time model, where concept drift causes the accuracy of client i to undergo degradation at an unknown deterministic time $\tau_i \geq 0$. Specifically, after time τ_i, the accuracy drops by Δ and becomes[1] $q_i = p_i - \Delta$ for each $i \in \mathcal{N}$. While performance may in practice degrade gradually and by varying magnitudes, a standard approach is to test against a small prescribed variation; here, Δ represents the tolerated accuracy loss which should trigger detection.

Definition 3.1 (Population Drift Dynamics)**.** *A population drift is* synchronous *if all*

[1]The analysis of Greedy Partial Sum CUSUM applies to any known (p_i, q_i) pairs, whereas Adaptive Global CUSUM relies on an additive formulation with a common Δ across clients.

affected clients share the same change point, i.e., $\tau_i = \tau$, and asynchronous *otherwise. The drift is said to be* homogeneous *if all affected clients share the same degradation magnitude, i.e., $\Delta_i = p_i - q_i = \Delta$, and* non-homogeneous *otherwise. In this chapter, we model asynchronous change points with homogeneous degradation Δ.*

Conditioned on τ_i, the correctness of task outcomes for client i are modeled as i.i.d. across time. Let $X_t^{(i)} \in \{0, 1\}$ denotes the correctness of the local inference at time t, with $X_t^{(i)} = 1$ indicating a correct classification, and

$$X_t^{(i)} \sim \begin{cases} \mathrm{p}_0^{(i)}(\cdot), & t \leq \tau_i, \\ \mathrm{p}_1^{(i)}(\cdot), & \text{otherwise}, \end{cases} \tag{3.1}$$

where $\mathrm{p}_0^{(i)}$ and $\mathrm{p}_1^{(i)}$ denote Bernoulli PMFs with parameters p_i and q_i, respectively.

In the absence of ground-truth labels, clients cannot directly observe the processes $(X_t^{(i)})_{t\geq 0}$ and are therefore unaware of performance degradations. To enable monitoring of the clients inference accuracies, clients communicate with a server hosting a larger ML model, assumed to be substantially more accurate and more robust to concept drift than the lightweight client-based models. We use the server's inference outputs on offloaded tasks as reference labels for monitoring client performance. Since offloading incurs communication and computation overheads, it is resource-constrained. For simplicity, we model this constraint by allowing at most one task to be offloaded system-wide at each time slot for monitoring purposes.

3.2.1 Fractional Change Detection Problem

In this hierarchical inference system, our objective is to monitor performance and determine when the client-side model should be updated. Since model updates are costly in terms of communication, computation, and possible service disruption, unless there is any urgent problem, isolated degradations affecting only a few clients often do not justify an update. Instead, an update is triggered only when the fraction of significantly underperforming clients exceeds a prescribed threshold.

To formalize this criterion, we define the set of affected clients at time t as

$$\mathcal{A}_t \triangleq \{i \in \mathcal{N} : t > \tau_i\}, \tag{3.2}$$

and let $\rho_t \triangleq |\mathcal{A}_t|/N$ be the corresponding affected fraction. Given a target fraction $\rho_* \in (0, 1)$, or equivalently a target number of affected clients $K_* = \lfloor \rho_* N \rfloor$, we define the first time at which the affected fraction exceeds ρ_* as

$$\nu \triangleq \inf_t \{t : \rho_t > \rho_*\} = \inf_t \{t : |\mathcal{A}_t| > K_*\}, \tag{3.3}$$

and refer to ν as the time *fractional change* happens.

Definition 3.2 (Fractional Change Detection)**.** Fractional change detection *is the problem of identifying the first time at which a prespecified fraction ρ_* of clients in a population see a performance loss of at least Δ.*

Detecting the fractional change point can therefore be formulated as a sequential test of the composite hypotheses

$$\begin{aligned} H_0 &: \rho_t \leq \rho_*, \\ H_1 &: \rho_t > \rho_*, \end{aligned} \tag{3.4}$$

where H_0 denotes the null hypothesis, corresponding to the case in which the affected fraction has not yet exceeded the target level, and H_1 denotes the alternative hypothesis, corresponding to the case in which the affected fraction has exceeded the target level. The test is carried out sequentially until H_0 is rejected.

A monitoring procedure is specified by a policy π, consisting of a (possibly randomized) client sampling rule that selects which client to probe at each time and a stopping rule that declares when a fractional change is deemed to have occurred. Formally, at time t, policy π selects a client index $I_t^\pi \in \mathcal{N}$ for which a task is offloaded, and observes the corresponding binary outcome $Y_t^\pi = X_t^{(I_t^\pi)} \in \{0, 1\}$, i.e., whether the lightweight model makes a mistake based on the large model's output. The information available to the policy up to time $t \geq 1$ is captured by the filtration

$$\mathcal{F}_t^\pi \triangleq \sigma\left(I_1^\pi, Y_1^\pi, \ldots, I_t^\pi, Y_t^\pi\right), \text{ with } \mathcal{F}_0^\pi \text{ trivial.} \tag{3.5}$$

Restricting attention to *causal* policies, the sampling decision I_t^π at time t must be measurable with respect to $\mathcal{F}_{t-1}^\pi$, and the stopping rule T^π must be an $\mathcal{F}_t^\pi$-stopping time, i.e., $\{T^\pi \leq t\} \in \mathcal{F}_t^\pi$ for all t.

We use $\mathbb{P}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}$ and $\mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}$ to denote the probability measure and corresponding expectation governing the client processes, where $\mathcal{A} \triangleq \mathcal{A}_\infty$ is the set of clients that are eventually affected and $\boldsymbol{\tau}(\mathcal{A}) = \{\tau_i : \tau_i < \infty,\ i \in \mathcal{A}\}$ corresponds to their change-point configuration. A realization falls under the alternative hypothesis if the fraction of eventually affected clients exceeds ρ_*; otherwise, it falls under the null hypothesis.

Following Pollak's minimax formulation [84], our goal is to design a causal monitoring procedure π, consisting of a client sampling policy $(I_t^\pi)_{t\geq 0}$ and an associated stopping time T^π, that detects the fractional change point as quickly as possible while controlling false alarms. For a given affected subset $\mathcal{A}$ with $|\mathcal{A}| > K_*$, the Worst-case Average Detection Delay (WADD) is defined as

$$\mathrm{WADD}(T^\pi; \mathcal{A}) \triangleq \sup_{\boldsymbol{\tau}(\mathcal{A})} \mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})} \left[T^\pi - \nu | T^\pi > \nu \right], \tag{3.6}$$

where the supremum is taken over all change-point configurations for a given set $\mathcal{A}$ of affected clients. This is a measure of how responsive the policy π is at detecting change. To measure false alarm performance, we define the Worst-case Average Run Length (WARL) by

$$\mathrm{WARL}(T^\pi) \triangleq \inf_{\mathcal{A}' : |\mathcal{A}'| \leq K_*} \inf_{\boldsymbol{\tau}(\mathcal{A}')} \mathbb{E}^{\mathcal{A}'}_{\boldsymbol{\tau}(\mathcal{A}')} \left[T^\pi \right], \tag{3.7}$$

where the infimum is taken over all affected subsets and associated change-point configurations that fall under the null hypothesis. This is a uniform bound on how quickly a false alarm will take to be observed under policy π. Our objective is to choose a policy that minimizes the worst-case average detection delay for the affected subset $\mathcal{A}$ while ensuring that the worst-case average run length is at least γ:

$$\min_{\pi}\ \mathrm{WADD}(T^\pi; \mathcal{A}) \quad \text{s.t.} \quad \mathrm{WARL}(T^\pi) \geq \gamma. \tag{3.8}$$

Although the optimization problem in (3.8) may not admit an exact solution, the following sections show that uniformly optimal procedures can still be characterized up to a first-order asymptotic approximation.

3.3 Proposed Approaches

Fractional change detection naturally admits two broad classes of monitoring policies. The first class, *Individual Statistics-based Detectors* (ISDs), maintain client-specific statistics to individually detect which clients may have experienced degradation and declare a fractional change once sufficiently many affected clients have been identified. The second class, *Population Statistics-based Detectors* (PSDs), aggregate observations at the system level and performs a single aggregate-based sequential test—typically based on a mixture model—aimed at detecting the fractional change point directly, without individually tracking the affected clients.

These two approaches offer complementary trade-offs. An ISD declares a fractional change after identifying at least $K_* + 1$ affected clients for a given target threshold ρ_*. As a result, when N grows (and hence $K_* = \lfloor \rho_* N \rfloor$ grows), the required number of clients that need to be detected increases, and the overall detection delay typically scales approximately linearly with N, which limits the scalability of the approach for large populations. By contrast, PSDs can achieve detection delays that do not necessarily increase with N, since they leverage pooled evidence from the entire population. However, as we shall see, PSDs generally may not provide client-level change detection, which can be valuable in certain applications (e.g., selectively updating or prioritizing clients).

This chapter proposes two fractional change detection policies: *Greedy Partial Sum-CUSUM*, an ISD policy that leverages client-specific statistics, and *Adaptive Global-CUSUM*, a PSD policy based on an aggregated system-wide statistic. The following section presents the details of these policies.

3.3.1 Greedy Partial Sum-CUSUM

In this section, we introduce the Greedy Partial Sum-CUSUM (G-PSC) policy by formulating the fractional change detection problem as a Generalized Likelihood Ratio (GLR) test conditioned on the affected client subsets and their associated change points under the null and alternative hypotheses. We first derive the test statistic and then specify the sampling rule and stopping rule that together define the G-PSC policy.

Let $Y^{\pi}_{t_0:t_1} \triangleq (Y^{\pi}_{t_0}, \ldots, Y^{\pi}_{t_1})$ denote the sequence of observations collected under policy π over the time interval $[t_0, t_1]$.

The likelihood of $Y^{\pi}_{t_0:t_1}$ under a candidate change-point configuration $\tilde{\boldsymbol{\tau}}(\mathcal{A}) = (\tilde{\tau}_i)_{i\in\mathcal{A}}$ is given by

$$\begin{aligned}\mathbb{P}^{\mathcal{A}}_{\tilde{\boldsymbol{\tau}}(\mathcal{A})}(Y^{\pi}_{t_0:t_1}) &= \prod_{k=t_0}^{t_1} \mathrm{p}_1^{(I^{\pi}_k)}(Y^{\pi}_k)^{\mathbb{I}\{I^{\pi}_k\in\mathcal{A}_k\}} \mathrm{p}_0^{(I^{\pi}_k)}(Y^{\pi}_k)^{\mathbb{I}\{I^{\pi}_k\notin\mathcal{A}_k\}} \\ &= \left(\prod_{k=t_0}^{t_1} \left(\frac{\mathrm{p}_1^{(I^{\pi}_k)}(Y^{\pi}_k)}{\mathrm{p}_0^{(I^{\pi}_k)}(Y^{\pi}_k)}\right)^{\mathbb{I}\{I^{\pi}_k\in\mathcal{A}_k\}}\right)\left(\prod_{k'=t_0}^{t_1} \mathrm{p}_0^{(I^{\pi}_{k'})}(Y^{\pi}_{k'})\right), \qquad (3.9)\end{aligned}$$

where $\mathcal{A}_k$ is defined in (3.2), and the term $\prod_{k'=t_0}^{t_1} \mathrm{p}_0^{(I^{\pi}_{k'})}(Y^{\pi}_{k'})$ does not depend on $\tilde{\boldsymbol{\tau}}(\mathcal{A})$.

The GLR statistic at time t is defined as the log-ratio of the likelihood maximized over all affected subsets and change-point configurations admissible under the alternative hypothesis to the likelihood maximized over all affected subsets and change-point configurations admissible under the null hypothesis, hence,

$$G_t = \log\left(\frac{\max\limits_{\mathcal{A}:|\mathcal{A}|>K_*} \max\limits_{\tilde{\boldsymbol{\tau}}(\mathcal{A})} \mathbb{P}^{\mathcal{A}}_{\tilde{\boldsymbol{\tau}}(\mathcal{A})}(Y^{\pi}_{1:t})}{\max\limits_{\mathcal{A}':|\mathcal{A}'|\le K_*} \max\limits_{\tilde{\boldsymbol{\tau}}(\mathcal{A}')} \mathbb{P}^{\mathcal{A}'}_{\tilde{\boldsymbol{\tau}}(\mathcal{A}')}(Y^{\pi}_{1:t})}\right). \qquad (3.10)$$

As written in (3.10), a naive implementation of the GLR statistic at time t would require the entire sample history, while the maximization in both the numerator and the denominator would entail a computationally expensive search over all possible change-point configurations.

Fortunately, the GLR statistic in (3.10) admits a computationally simple recursive representation. Using (3.9), we obtain

$$G_t = \max_{\substack{\mathcal{A}:\,|\mathcal{A}|>K_*,\\ \tilde{\boldsymbol{\tau}}(\mathcal{A})}} \sum_{k=1}^{t} \mathbb{I}\{k > \tilde{\tau}_{I_k^\pi}\} \log\left(\frac{\mathrm{p}_1^{(I_k^\pi)}(Y_k^\pi)}{\mathrm{p}_0^{(I_k^\pi)}(Y_k^\pi)}\right)$$

$$- \max_{\substack{\mathcal{A}':\,|\mathcal{A}'|\le K_*,\\ \tilde{\boldsymbol{\tau}}(\mathcal{A}')}} \sum_{k=1}^{t} \mathbb{I}\{k > \tilde{\tau}'_{I_k^\pi}\} \log\left(\frac{\mathrm{p}_1^{(I_k^\pi)}(Y_k^\pi)}{\mathrm{p}_0^{(I_k^\pi)}(Y_k^\pi)}\right) \tag{3.11}$$

$$= \max_{\substack{\mathcal{A}:\,|\mathcal{A}|>K_*,\\ \tilde{\boldsymbol{\tau}}(\mathcal{A})}} \sum_{i\in\mathcal{A}} L^{(i)}_{\tilde{\tau}_i+1:t} - \max_{\substack{\mathcal{A}':\,|\mathcal{A}'|\le K_*,\\ \tilde{\boldsymbol{\tau}}(\mathcal{A}')}} \sum_{i\in\mathcal{A}'} L^{(i)}_{\tilde{\tau}'_i+1:t}. \tag{3.12}$$

where the terms $\prod_{k=1}^{t} \mathrm{p}_0^{(I_k^\pi)}(Y_k^\pi)$ coming from (3.9) and in the numerator and denominator of (3.10) cancel to obtain (3.11), and $L^{(i)}_{\tilde{\tau}+1:t}$ denotes the log-likelihood ratio (LLR) accumulated from client i over $[\tilde{\tau}+1, t]$:

$$L^{(i)}_{\tilde{\tau}+1:t} \triangleq \sum_{k=\tilde{\tau}+1}^{t} \mathbb{I}\left(I_k^\pi = i\right) \log \frac{\mathrm{p}_1^{(i)}(Y_k^\pi)}{\mathrm{p}_0^{(i)}(Y_k^\pi)}. \tag{3.13}$$

For client i, we define the CUSUM statistic at time t by

$$W_t^{(i)} \triangleq \max_{\tilde{\tau}_i:\tilde{\tau}_i\le t} L^{(i)}_{\tilde{\tau}_i+1:t}, \tag{3.14}$$

which admits the standard recursion

$$W_t^{(i)} = \max\left[0, W_{t-1}^{(i)} + \mathbb{I}\{I_t^\pi = i\} \log \frac{\mathrm{p}_1^{(i)}(Y_t^\pi)}{\mathrm{p}_0^{(i)}(Y_t^\pi)}\right], \tag{3.15}$$

with initialization $W_0^{(i)} = 0$. We refer to [97, Section 8.2] for the derivation of the CUSUM procedure.

Because the maximization over change points is separable across clients, (3.12) can be rewritten using (3.14) as

$$G_t \triangleq \max_{\mathcal{A}:|\mathcal{A}|>K_*} \sum_{i\in\mathcal{A}} \max_{\tilde{\tau}_i:\tilde{\tau}_i\le t} L^{(i)}_{\tilde{\tau}_i+1:t} - \max_{\mathcal{A}':|\mathcal{A}'|\le K_*} \sum_{i\in\mathcal{A}'} \max_{\tilde{\tau}'_i:\tilde{\tau}'_i\le t} L^{(i)}_{\tilde{\tau}'_i+1:t}$$

$$= \max_{\mathcal{A}:|\mathcal{A}|>K_*} \sum_{i\in\mathcal{A}} W_t^{(i)} - \max_{\mathcal{A}':|\mathcal{A}'|\le K_*} \sum_{i\in\mathcal{A}'} W_t^{(i)}, \tag{3.16}$$

For each time t, let $\sigma_t(\cdot)$ denote a permutation of client indices $1, 2, \ldots, N$ such that

$$W_t^{(\sigma_t(1))} \geq W_t^{(\sigma_t(2))} \geq \cdots \geq W_t^{(\sigma_t(N))} \geq 0, \tag{3.17}$$

where ties are broken arbitrarily. Since $W_t^{(i)} \geq 0$ for all i and t, the first maximization in (3.16) is attained by selecting the set with all clients, whereas the second is attained by selecting the K_* clients with the largest CUSUM values. Therefore, G_t reduces to the partial sum of the CUSUM statistics excluding the K_* clients with the highest CUSUM:

$$G_t = \sum_{j=K_*+1}^{N} W_t^{(\sigma_t(j))}. \tag{3.18}$$

It remains to specify the sampling policy that determines the selected client I_t^{π} at each time step. The partial-sum structure of (3.18) creates an exploration–exploitation trade-off that differs from standard policies for detecting the earliest change in a population. A policy that always samples the client with the largest CUSUM statistic would keep refining evidence for one of the top K_* clients; these statistics, however, are excluded from G_t and therefore do not directly advance the stopping rule. We therefore propose a greedy deterministic sampling rule that selects the client with the largest CUSUM statistic still included in G_t, namely the client ranked $(K_* + 1)$:

$$I_t^{\text{Greedy-PSC}} \triangleq \sigma_{t-1}(K_* + 1). \tag{3.19}$$

This rule is exploitative because it samples the client closest to contributing to the test statistic yet it also induces exploration, as the sampled statistic either rises into the top K_* positions or resets to zero, the ordering and tie-breaking mechanism move the policy to another client. In this way, the policy gradually promotes affected clients toward the leading K_*+1 ranks and eventually settles on a fixed set of affected clients.

Together with the sampling rule in (3.19), the Greedy-PSC stopping rule is

$$T^{\text{G-PSC}}(h) \triangleq \inf_t \{t : G_t \geq h\}, \tag{3.20}$$

Algorithm 2 Greedy Partial Sum-CUSUM (G-PSC)

Input: threshold h, target count K_*, initial accuracies $p_i,\ \forall i \in \mathcal{N}$, tolerance Δ
Initialize: $G \leftarrow 0$, $W^{(i)} \leftarrow 0$ for all $i \in \mathcal{N}$, and an arbitrary ordering σ

while $G < h$ **do**
  Select $I^\pi \leftarrow \sigma(K_* + 1)$ as in (3.19) and observe $Y^\pi = X^{(I^\pi)}$
  Update $W^{(I^\pi)}$ by (3.15), using $q_{I^\pi} = p_{I^\pi} - \Delta$
  if $W^{(I^\pi)} = 0$ **then**
    Move I^π to the end of the ordering σ
  else
    Update σ in descending order of $W^{(i)}$
  Set $G \leftarrow \sum_{j=K_*+1}^{N} W^{(\sigma(j))}$ as in (3.18)

where the threshold h is chosen to satisfy the WARL constraint in (3.7).

Algorithm 2 summarizes the resulting Greedy-PSC policy.

3.3.1.1 Performance Analysis

For the Greedy-PSC policy defined by the sampling rule in (3.19) and the stopping rule in (3.20), the following lemma establishes a structural property that plays a key role in the analysis.

Lemma 3.3. *Under the Greedy-PSC policy, for every time $t \geq 0$, all CUSUM statistics ranked below the $(K_* + 1)$-st position are zero; that is,*

$$W_t^{(\sigma_t(j))} = 0, \quad \text{for } j > K_* + 1,\ t \geq 0. \tag{3.21}$$

Proof of Lemma 3.3. We prove the lemma by induction. At time $t = 0$, the CUSUM statistics of all clients are initialized at zero, so the claim holds trivially. Now assume that the claim holds at time $t-1$. By the definition of Greedy-PSC, only the CUSUM statistic of the sampled client $I_t = \sigma_{t-1}(K_*+1)$ is updated at time t. Hence, all clients ranked below $K_* + 1$ at time $t - 1$ remain at zero at time t. After reordering the CUSUM statistics, every rank below $K_* + 1$ is therefore still occupied by a zero statistic, which proves the claim. □

The next theorem summarizes the main performance guarantees of Greedy-PSC.

Theorem 3.4. *For the fractional change detection problem defined in Definition 3.2, with target fraction $\rho_* \in (0,1)$ and corresponding target count $K_* = \lfloor \rho_* N \rfloor$, under the asynchronous homogeneous population-drift defined in Definition 3.1 with common degradation Δ, where $p_i - \Delta > 0$ for all $i \in \mathcal{N}$, consider the Greedy Partial Sum-CUSUM detector with stopping time $T^{G\text{-}PSC}(h)$ defined by* (3.20) *and greedy sampling policy* (3.19)*. Then, for any threshold $h > 0$, its worst-case average run length (WARL) satisfies*

$$\mathrm{WARL}(T^{G\text{-}PSC}(h)) \;\geq\; e^h. \tag{3.22}$$

Moreover, for any affected subset $\mathcal{A}$ with $|\mathcal{A}| = K_*+1$, the worst-case average detection delay satisfies, as $h \to \infty$,

$$\mathrm{WADD}(T^{G\text{-}PSC}(h);\mathcal{A}) \leq \left(\sum_{i\in\mathcal{A}} \frac{1}{\beta_i}\right) h + \sum_{i\in\mathcal{A}} \frac{1}{1-\alpha_i} \sum_{j\notin\mathcal{A}} \eta_j + O(1), \tag{3.23}$$

where

$$\beta_i = D_{\mathrm{KL}}\left(\mathrm{p}_1^{(i)} \,\middle\|\, \mathrm{p}_0^{(i)}\right), \quad \forall i \in \mathcal{A},$$

and where β_i denotes the Kullback-Leibler (KL) divergence between the post- and pre-change distributions of client i. Here, α_i is the probability that CUSUM statistic of an affected client $i \in \mathcal{A}$, initialized at zero, returns to zero before reaching the threshold h as $h \to \infty$, which satisfies $\Delta \leq 1-\alpha_i \leq \Delta/p_i$ by Lemma 3.8, and η_j is defined in (3.83)*.*

Theorem 3.4 shows that the worst-case average run length grows at least exponentially in the threshold h under the composite null hypothesis, while the worst-case average detection delay grows linearly in h and in the number of affected clients in the asymptotic regime $h \to \infty$, since $|\mathcal{A}| = K_* + 1$ and $(\sum_{i\in\mathcal{A}} \frac{1}{\beta_i}) \propto |\mathcal{A}|$.

The proof of (3.22) and (3.23) hinges on constructing auxiliary statistics that upper- and lower-bound G_t pointwise for all $t \geq 0$ and are then used in domination arguments to obtain the desired inequalities. For the WARL lower bound, we

compare G_t with the sum of the CUSUM statistics of the unaffected clients, which is then controlled by a Shiryaev-Roberts (SR)-type statistic [97, Lemma 8.2.1]. For the WADD upper bound, we bound the number of samples needed for the CUSUM statistics of the affected clients to jointly cross the threshold h, together with the additional samples allocated to unaffected clients during the rare events in which the affected clients' CUSUM statistics return to zero. A proof of the results is given in Appendix 3.A.2.

As expected for an ISD type policy, the leading term in the WADD bound of Greedy PSC (3.23) depends linearly on the number of affected clients, resulting in poor scalability for systems with large client populations. To address this limitation, we next investigate a PSD type policy that offers improved scalability.

3.3.2 Adaptive Global-CUSUM

We propose a PSD approach for fractional change detection based on a mixture model, called *Adaptive Global-CUSUM* (A-GC). Unlike the ISD approach in Section 3.3.1, which relies on a GLR statistic over composite hypotheses, the design of A-GC is based on monitoring a mixture distribution parametrized by the fraction ρ of affected clients and estimates the true affected fraction to resolve the composite hypothesis testing problem.

We begin by formulating the mixture distribution for a given affected fraction parameter and then introduce the estimation of the true affected fraction to construct the Adaptive Global-CUSUM policy.

3.3.2.1 Mixture Model

Let us state a key modeling assumption underlying the mixture formulation.

Assumption 3.5. *For time t, where the affected fraction is ρ_t, each client is affected with the same prior probability, i.e.,*

$$\mathbb{P}(A_t^{(i)} \, ; \rho_t) = \rho_t, \quad \forall i \in \mathcal{N}, \tag{3.24}$$

where $A_t^{(i)}$ denotes the event that client i is affected at time t.[2]

Under Assumption 3.5, the distribution of the observation Y_t^π, after marginalizing over whether the sampled client I_t^π is affected, is a mixture of the pre- and post-change distributions. In particular, under the prior probability of being affected given by ρ_t and conditioned on the selected client index I_t^π, we have the probability of observations as a mixture of pre- and post-change PMFs parametrized by ρ_t, such that

$$\mathbb{P}(Y_t^\pi = y \mid I_t^\pi = i; \rho_t) = \rho_t \mathrm{p}_1^{(i)}(Y_t^\pi = y) + (1-\rho_t)\mathrm{p}_0^{(i)}(Y_t^\pi = y), \quad y \in \{0,1\}. \tag{3.25}$$

Consequently, the likelihood of the sample at time t based on the mixture model can be written as

$$\begin{aligned}
\mathcal{L}\left(Y_t^\pi = y; \rho_t\right) &\triangleq \mathbb{P}\left(Y_t^\pi = y; \rho_t\right) \\
&= \sum_{i \in \mathcal{N}} \mathbb{P}\left(Y_t^\pi = y \mid I_t^\pi = i; \rho_t\right) \cdot \mathbb{P}\left(I_t^\pi = i; \rho_t\right) \\
&= \sum_{i \in \mathcal{N}} \left[\rho_t \mathrm{p}_1^{(i)}(Y_t^\pi = y) + (1-\rho_t)\mathrm{p}_0^{(i)}(Y_t^\pi = y)\right] \cdot \mathbb{P}\left(I_t^\pi = i; \rho_t\right), \quad (3.26)
\end{aligned}$$

where $\mathbb{P}(I_t^\pi = i; \rho_t)$ captures the client sampling probabilities under policy π. To make sure the probability distribution of the observations matches with the likelihood function (3.26) defined based on mixture model, we use a uniform sampling policy, i.e., $\mathbb{P}\left(I_t^\pi = i; \rho_t\right) = 1/N$ for all $i \in \mathcal{N}$, which is shown in the next lemma.

Lemma 3.6. *Under uniform sampling, for any affected subset $\mathcal{A}$ with change point configuration $\boldsymbol{\tau}(\mathcal{A})$ and any $t \geq 1$, the marginal PMF of Y_t^π under $\mathbb{P}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}$ coincides with the mixture likelihood evaluated at the true affected fraction $\rho_t = |\mathcal{A}_t|/N$,*

$$\mathbb{P}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}(Y_t^\pi = y) = \mathcal{L}(Y_t^\pi = y;\, \rho_t), \quad \forall\, y \in \{0,1\}. \tag{3.27}$$

In particular, whenever $\rho_t = \rho$ remains constant over a time interval, the observations $(Y_t^\pi)_{t \geq 0}$ are i.i.d. with PMF $\mathcal{L}(\cdot\,; \rho)$ over that interval.

[2]This assumption corresponds to a symmetry condition, where all client subsets of size $N\rho_t$ are equally likely to be the affected subset.

Although the original formulation involves the composite hypotheses in (3.4), the mixture model suggests an alternative way to build a likelihood-ratio statistic. Under the null hypothesis, the affected fraction satisfies $\rho_t \leq \rho_*$. Among all such null cases, the value $\rho = \rho_*$ is closest to the alternative regime $\rho_t > \rho_*$; hence controlling false alarms when the mixture distribution is evaluated at $\rho = \rho_*$ also provides the relevant conservative reference point for smaller affected fractions. We therefore first consider a simple CUSUM test that compares the mixture distribution at $\rho = \rho_*$ with the mixture distribution at a chosen post-change fraction $\rho_1 > \rho_*$:

$$H_0 : \rho = \rho_*, \tag{3.28}$$

$$H_1 : \rho = \rho_1. \tag{3.29}$$

where ρ_1 is a post-change fraction used to construct the likelihood ratio.

For a fixed anticipated fraction $\rho_1 > \rho_*$, the Global-CUSUM statistic is defined recursively as

$$W_t^{\mathrm{GC}} = \max\left[0, W_{t-1}^{\mathrm{GC}} + L_t(\rho_1, \rho_*)\right], \quad \text{with } W_0^{\mathrm{GC}} = 0, \tag{3.30}$$

where $L_t(\rho_1, \rho_*) = \log \frac{\mathcal{L}(Y_t^{\pi};\rho_1)}{\mathcal{L}(Y_t^{\pi};\rho_*)}$ denotes the log-likelihood ratio at time t. The corresponding stopping rule is

$$T^{\mathrm{GC}}(h) \triangleq \inf_t \{t : W_t^{\mathrm{GC}} \geq h\}. \tag{3.31}$$

3.3.2.2 Adaptive Statistic

Note that the drift of the Global-CUSUM statistic under a fixed post-change fraction $\rho_t > \rho_*$ is maximized when the parameter of the alternative hypothesis matches the underlying distribution, i.e., $\rho_1 = \rho_t$. Moreover, a positive drift is not guaranteed when ρ_t lies between the boundary and the anticipated fraction, i.e., $\rho_* < \rho_t < \rho_1$. This mismatch can degrade detection performance when the post-change fraction is unknown. To mitigate such sensitivity, variants of CUSUM that

adaptively estimate post-change parameters from past observations have been studied (e.g., [67, 109]).

Motivated by this line of work, we propose an adaptive version that replaces ρ_1 with a running estimate $\hat{\rho}_t$:

$$W_t^{\text{A-GC}} = \max\left[0, W_{t-1}^{\text{A-GC}} + L_t(\hat{\rho}_t, \rho_*)\right]. \tag{3.32}$$

Let

$$U(t) \triangleq \sup_k\{k \le t : \ W_k^{\text{A-GC}} = 0\} \tag{3.33}$$

denote the most recent reset time of the statistic. Using observations since $U(t)$, we form the sample-mean estimator

$$\hat{\rho}_t = \left[\frac{1}{t-U(t)} \sum_{k=U(t)}^{t-1} \frac{Y_k^\pi - p_{I_k^\pi}}{q_{I_k^\pi} - p_{I_k^\pi}}\right]_{\rho_*+\epsilon}^{1}, \tag{3.34}$$

where $[x]_l^u \triangleq \max[\min[x, u], l]$ clips the estimate to $[\rho_* + \epsilon, 1]$ and $\epsilon > 0$ is a small constant ensuring the estimator remains strictly above the boundary.

The stopping rule for Adaptive Global-CUSUM is

$$T^{\text{A-GC}}(h) \triangleq \inf_t\{t : W_t^{\text{A-GC}} \ge h\}. \tag{3.35}$$

As in Global-CUSUM, we accompany Adaptive Global-CUSUM with uniform sampling.

Algorithm 3 summarizes the resulting Adaptive Global-CUSUM policy.

Theorem 3.7. *For the fractional change detection problem defined in Definition 3.2, with target fraction $\rho_* \in (0, 1)$, under the asynchronous homogeneous population-drift defined in Definition 3.1 with common degradation Δ, where $p_i - \Delta > 0$ for all $i \in \mathcal{N}$, consider the Adaptive Global-CUSUM detector with stopping time $T^{A\text{-}GC}(h)$ defined in* (3.35) *and i.i.d. uniform sampling policy $\mathbb{P}(I_t^\pi = i) = 1/N$ for all $i \in \mathcal{N}$ and $t \ge 1$. Then, for any threshold $h > 0$, the worst-case average run length (WARL) satisfies*

$$\text{WARL}(T^{A\text{-}GC}(h)) \ \ge \ e^h. \tag{3.36}$$

Algorithm 3 Adaptive Global-CUSUM (A-GC)

Input: threshold h, target fraction ρ_*, initial accuracies $p_i,\ \forall i \in \mathcal{N}$, tolerance Δ, margin ϵ

Initialize: $W^{\text{A-GC}} \leftarrow 0$, $S \leftarrow 0$, $m \leftarrow 0$, and $\hat{\rho} \leftarrow \rho_* + \epsilon$

while $W^{\text{A-GC}} < h$ **do**
  Sample I^π uniformly and observe $Y^\pi = X^{(I^\pi)}$
  Update $W^{\text{A-GC}}$ by (3.32)
  if $W^{\text{A-GC}} = 0$ **then**
    Set $S \leftarrow 0$, $m \leftarrow 0$, $\hat{\rho} \leftarrow \rho_* + \epsilon$
  else
    Update estimation state: $S \leftarrow S + (p_{I^\pi} - Y^\pi)/\Delta$, $m \leftarrow m + 1$
    Set $\hat{\rho} \leftarrow [S/m]^1_{\rho_*+\epsilon}$, as in (3.34)

Moreover, for any affected subset $\mathcal{A}$ with $|\mathcal{A}| = K_*+1$, the worst-case average detection delay satisfies, as $h \to \infty$,

$$\text{WADD}(T^{A\text{-}GC}(h); \mathcal{A}) = \frac{h}{\beta(\rho, \rho_*)}\big(1 + o(1)\big), \tag{3.37}$$

where $\rho = (K_* + 1)/N$ and $\beta(\rho, \rho_*) = D_{\text{KL}}\left(\mathcal{L}\left(\cdot\,; \rho\right) \| \, \mathcal{L}\left(\cdot\,; \rho_*\right)\right)$ denotes the Kullback-Leibler (KL) divergence between the mixture distributions parametrized by the affected fractions ρ and ρ_* under uniform sampling.

As Theorem 3.7 shows, the leading term in the worst-case average detection delay of Adaptive Global-CUSUM is governed by the mixture KL divergence $\beta(\rho, \rho_*)$, rather than by the number of clients that must be individually identified, as in an ISD policy, making it suitable for large-scale settings.

The proof proceeds in two parts. For (3.36), we construct an SR-type statistic that dominates $\exp(W_t^{\text{A-GC}})$ when the null is evaluated at the limiting case $\rho = \rho_*$; its centered version is a supermartingale, so optional stopping gives the e^h run-length lower bound. For (3.37), after ν the sampled observations are i.i.d. from $\mathcal{L}(\cdot; \rho)$. Applying the optional stopping theorem to the oracle log-likelihood sum gives the delay as $(h + x_1 + x_2)/\beta(\rho, \rho_*)$, where x_1 is the crossing overshoot and x_2 is the error from using $\hat{\rho}_t$ instead of ρ. Bounded increments give $x_1 = O(1)$, while a crude $O(h)$

delay bound and an $O(\log h)$ cumulative estimation-error bound imply $x_2 = o(h)$. A full proof is given in Appendix 3.A.3.

3.4 Comparison of Proposed Policies

Theorems 3.4 and 3.7 reveal that the Greedy-PSC and Adaptive Global-CUSUM policies exhibit fundamentally different detection delay characteristics. To facilitate a systematic comparison, we introduce the Oracle Global CUSUM (O-GC), a benchmark policy that has access to the true affected fraction ρ_t and therefore does not require an estimation step.

Furthermore, we consider a homogeneous setting where $p_i = p$ and $q_i = q$ for all i, with a fixed post-change fraction $\rho_t = \rho > \rho_*$ for all $t \geq 1$. Using a second-order Taylor approximation of the KL divergence, and for the stopping rule threshold satisfying the same WARL constraint in (3.8), i.e. $h = \log(\gamma)$ by (3.22) and (3.36), the leading terms of the WADD for Greedy-PSC and Oracle Global CUSUM can be approximated as

$$\begin{aligned} \mathrm{WADD}(T^{\text{G-PSC}}(\log(\gamma))) &\approx \frac{N\rho_* \log(\gamma)}{D_{\mathrm{KL}}\left(\mathrm{Ber}(p-\Delta) \,\|\, \mathrm{Ber}(p)\right)} \\ &\approx N\rho_* \log(\gamma) \frac{2p(1-p)}{\Delta^2}, \end{aligned} \tag{3.38}$$

$$\begin{aligned} \mathrm{WADD}(T^{\text{O-GC}}(\log(\gamma))) &\approx \frac{\log(\gamma)}{D_{\mathrm{KL}}\left(\mathrm{Ber}(p-\rho\Delta) \,\|\, \mathrm{Ber}(p-\rho_*\Delta)\right)} \\ &\approx \log(\gamma) \frac{2p(1-p)}{(\rho-\rho_*)^2\Delta^2}. \end{aligned} \tag{3.39}$$

where $\mathrm{Ber}(p)$ denotes the Bernoulli distribution with parameter p.

According to (3.38) and (3.39), for a fixed target fraction ρ_*, the WADD of Greedy-PSC scales linearly with the population size N and is independent of the true affected fraction ρ, whereas the WADD of Oracle Global CUSUM scales inversely with $(\rho - \rho_*)^2$. Comparing the two approximations yields that Greedy-PSC has a

smaller approximate WADD when

$$N\rho_* < \frac{1}{(\rho - \rho_*)^2}, \tag{3.40}$$

whereas Oracle Global CUSUM has a smaller approximate WADD when

$$N\rho_* > \frac{1}{(\rho - \rho_*)^2}, \tag{3.41}$$

Thus, in the worst-case scenario investigated in Theorem 3.7, where $(\rho - \rho_*) = \frac{1}{N}$, Greedy-PSC achieves a smaller WADD than Oracle Global CUSUM. However, when the true affected fraction ρ is sufficiently larger than ρ_*, or when N is sufficiently large, Oracle Global CUSUM outperforms Greedy-PSC.

To analyze the performance gap between Adaptive Global-CUSUM and Oracle Global-CUSUM, we examine the parameter n_0 in (3.91), which represents the number of samples needed for the estimator $\hat{\rho}_t$ to concentrate sufficiently around the true fraction ρ. By Hoeffding's inequality, n_0 can be approximated as

$$\begin{aligned} n_0 &\approx \frac{1}{2\delta_0^2\Delta^2} \log\left(\frac{2}{\delta_0^2\Delta^2}\right), \\ &\approx \left(\frac{2\bar{L}^2}{\beta(\rho,\rho_*)^2\Delta^2}\right) \log\left(\frac{8\bar{L}^2}{\beta(\rho,\rho_*)^2\Delta^2}\right), \end{aligned} \tag{3.42}$$

where $\bar{L}$ is the Lipschitz constant from property (R2) of Lemma 3.11 for the log-likelihood ratio $L_t(\cdot,\rho_*) = \log \frac{\mathcal{L}(Y_t^\pi;\cdot)}{\mathcal{L}(Y_t^\pi;\rho_*)}$. The parameter δ_0 denotes the concentration radius in the Hoeffding bound (3.91); in the second approximation in (3.42), we use $\delta_0 = \beta(\rho,\rho_*)/(2\bar{L})$. Also, note that, the KL divergence between the mixture distributions parametrized by ρ and ρ_* admits the approximation

$$\beta(\rho,\rho_*) \approx \frac{(\rho-\rho_*)^2\Delta^2}{2p(1-p)}. \tag{3.43}$$

Therefore, the approximations in (3.42) and (3.43) show that the performance gap between Adaptive-GC and Oracle-GC due to estimation error depends strongly on Δ and $(\rho - \rho_*)$, and diminishes as either quantity increases.

In summary, the two proposed policies occupy complementary operating regimes. Greedy-PSC is the preferred choice for detecting surgical changes affecting few clients in small populations, as its client-level tracking provides fine-grained sensitivity in this regime. Adaptive Global-CUSUM, on the other hand, is better suited for large-scale systems and detecting substantial changes affecting a large fraction, since its detection delay depends on the population-level KL divergence rather than on the number of affected clients that must be individually identified. When neither regime clearly dominates, the crude boundaries defined in (3.40) and (3.41) together with the estimation overhead characterized by (3.42) can guide the choice between the two policies for a given deployment.

3.5 Simulation Results

In this section, we evaluate the detection policies under both fully synthetic and data-driven settings. In the synthetic experiments, $(X_t^{(i)})_t$ for each client i is generated from Bernoulli processes with parameter p_i before the client's change point and q_i afterward. In the data-driven experiments, the sample paths are generated using real datasets and trained ML models.

In addition to Greedy Partial Sum-CUSUM (Section 3.3.1), Adaptive Global-CUSUM (Section 3.3.2), and Oracle Global-CUSUM (Section 3.4), we include two additional benchmark policies: Greedy Round-Robin (Greedy-RR) and Score Global-CUSUM (Score-GC).

Greedy Round-Robin (Greedy-RR) is an individual statistics-based detector (ISD) adapted from the Myopic Sampling Policy (MSP) of [110], which was originally proposed for detecting the earliest change in a population under a sampling constraint. Greedy-RR visits clients in a round-robin order and continues sampling the current client until its CUSUM either resets to zero or crosses the threshold h. If the CUSUM resets to zero, the client is returned to the end of the queue and the policy moves to the next client. If the CUSUM crosses h, the client is removed from the queue and the

number of detected clients is incremented by one. Greedy-RR declares a fractional change once it has detected $K_* + 1$ affected clients.

Score Global-CUSUM (Score-GC) is a population statistics-based detector (PSD) based on a CUSUM variant of the Rao score test [13, 29]. Instead of estimating the post-change affected fraction, Score-GC uses the derivative of the mixture log-likelihood in (3.26) with respect to ρ, evaluated at the boundary value $\rho = \rho_*$, as its CUSUM increment:

$$W_t^{\text{S-GC}} = \max\left[0, W_{t-1}^{\text{S-GC}} + S_{\rho_*}(Y_t^\pi)\right], \quad S_{\rho_*}\left(Y_t^\pi\right) = \frac{\partial}{\partial \rho} \log \mathcal{L}\left(Y_t^\pi; \rho\right)\Big|_{\rho=\rho_*}, \tag{3.44}$$

with the corresponding stopping rule $T^{\text{S-GC}}(h) = \inf_t\{t : W_t^{\text{S-GC}} \geq h\}$.

3.5.1 Synthetic Simulations

Figure 3.1 reports the results for the fully synthetic setting. In Figures 3.1a and 3.1b, over a range of threshold values, we plot the average run length (ARL) under the worst-case null $\rho = \rho_* = 0.1$ on the x-axis against the average detection delay (ADD) under the alternative hypotheses $\rho = 0.2$ and $\rho = 0.5$ on the y-axis. The population has $N = 50$ clients, with initial accuracies drawn as $p_i \sim U(0.85, 0.95)$ for all $i \in \mathcal{N}$. Under the alternative hypotheses, affected clients experience a common degradation $\Delta = 0.1$ at time $t = 0$.

The two ISD policies, Greedy-PSC and Greedy-RR, perform nearly identically in these simulations. When $\rho - \rho_* = 0.1$ (Figure 3.1a), the ISD policies achieve the smallest ADD, while Adaptive-GC exhibits a significant performance gap relative to Oracle-GC. When $\rho - \rho_* = 0.4$ (Figure 3.1b), this gap decreases substantially, and the population-level detectors outperform Greedy-PSC and Greedy-RR. This behavior is consistent with the discussion in Section 3.4: as the separation between ρ and ρ_* increases, the estimation overhead of Adaptive-GC becomes less significant, while the ADD of Oracle-GC also decreases. The same figures also show that Score-GC does not achieve an exponential upper bound on ARL under the worst-case null. At

$\rho = \rho_*$, the Score-GC has zero-mean increments, leading instead to a quadratic upper bound (see [33, Chapter 7.5.3]).

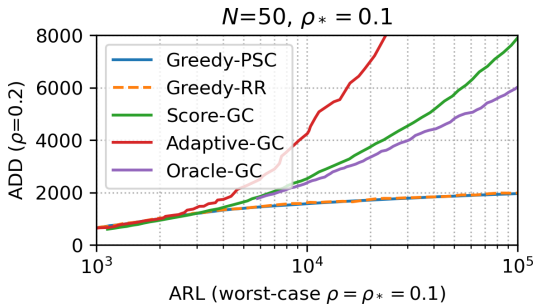
Figures 3.1c and 3.1d examine the dependence on the number of clients. For each value of N, the thresholds are chosen to yield an ARL of 10^5 under the boundary null $\rho = \rho_*$. The resulting ADD is reported for $\rho = 0.2$ and $\rho = 0.5$, respectively. In both regimes, the ADD of the ISD policies grows approximately linearly with N, whereas the ADD of the PSD policies is largely insensitive to N.

Taken together, the four panels in Figure 3.1 support the findings of Section 3.4. PSD policies benefit strongly from a larger affected fraction because the aggregate population-level signal becomes more pronounced, and their ADD is largely independent of the population size in this setting, where the target and affected fractions are fixed as N varies. By contrast, ISD policies must individually identify $K_* + 1$ affected clients, so their delay grows with N. Thus, PSDs are better suited for detecting widespread drifts in large-scale systems, while ISDs are attractive when the drift is sparse and client-level identification is also desired.

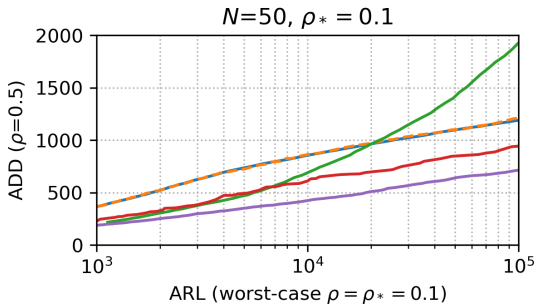
3.5.2 Realistic Simulations

Unlike the synthetic experiments in Section 3.5.1, where the null and alternative hypotheses are simulated in separate controlled runs, the realistic experiments use a single time-evolving drift trajectory. In each experiment, the system remains in the null regime before the drift begins at $t = 10^5$. The right panel of each figure shows the corresponding trajectory of the affected-client fraction, defined as the fraction of clients whose accuracy loss exceeds the prescribed tolerance $\Delta = 0.1$. The left panel reports the mean detection delay of true detections at a 5% false-alarm rate (FAR), together with the interquartile range (IQR), i.e., the range between the 25th and 75th percentiles.

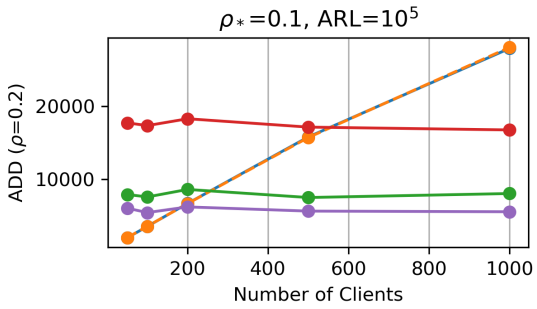
Figure 3.2 considers a network intrusion detection scenario in which lightweight ML models run on router devices to detect attacks in real time. The local model is



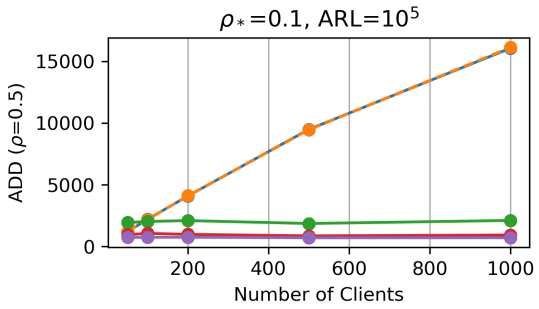
(a) ADD ($\rho = 0.2$) vs ARL ($\rho = \rho_* = 0.1$)



(b) ADD ($\rho = 0.5$) vs ARL ($\rho = \rho_* = 0.1$)



(c) ADD vs number of clients ($\rho = 0.2$, ARL $= 10^5$)



(d) ADD vs number of clients ($\rho = 0.5$, ARL $= 10^5$)

Figure 3.1: Synthetic simulations with $p_i \sim U(0.85, 0.95)$ for all $i \in \mathcal{N}$, target fraction $\rho_* = 0.1$, and degradation magnitude $\Delta = 0.1$. The top panels show ADD as a function of ARL, plotted on a logarithmic scale, under the worst-case null $\rho = \rho_*$ for $N = 50$ clients and affected fractions $\rho = 0.2$ and $\rho = 0.5$. The bottom panels show ADD as a function of the number of clients for the same affected fractions, with thresholds chosen to yield ARL $= 10^5$. All panels compare Greedy-PSC, Greedy-RR, Score-GC, Adaptive-GC, and Oracle-GC.

trained on known attack patterns, and a subset of routers later suffers an accuracy degradation when exposed to new attack techniques. Following [37], we train an MLP with two hidden layers of 100 hidden units each on the CIC-IDS 2017 dataset [89]. The trained model achieves 99.93% test accuracy on the original distribution, representing the performance of unaffected routers. For affected routers, we evaluate the model on a mixture of CIC-IDS 2017 and CSE-CIC-IDS 2018 traffic [89], which produces an accuracy drop of approximately 30% with client-to-client variation.

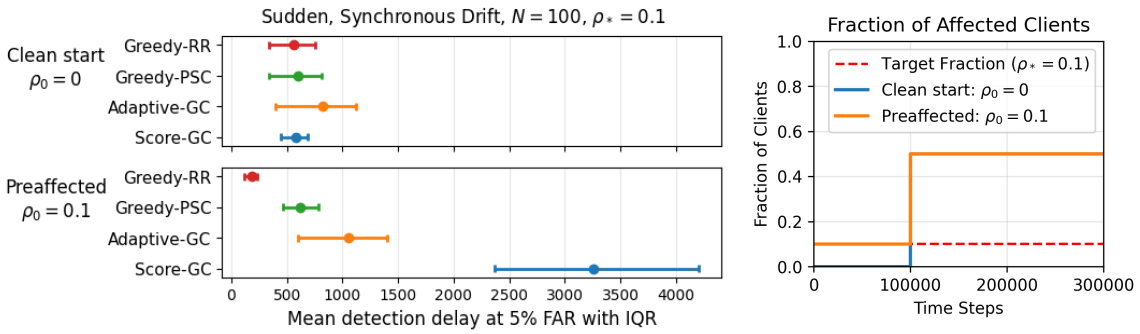


Figure 3.2: Network intrusion detection under sudden synchronous drift for $N = 100$ clients and target fraction $\rho_* = 0.1$. A lightweight MLP trained on CIC-IDS 2017 is evaluated after drift on mixed CIC-IDS 2017 and CSE-CIC-IDS 2018 traffic. The left panel reports the mean detection delay at a 5% false-alarm rate (FAR), with the interquartile range (IQR), for Greedy-RR, Greedy-PSC, Adaptive-GC, and Score-GC under two initial conditions: clean start ($\rho_0 = 0$) and preaffected boundary ($\rho_0 = 0.1$). The right panel shows the affected-client fraction trajectory, where a sudden drift at $t = 10^5$ raises the affected fraction to $\rho_\infty = 0.5$.

We compare two initial conditions in Figure 3.2: a clean start with no initially affected clients, $\rho_0 = 0$, and a preaffected boundary case with $\rho_0 = \rho_* = 0.1$. In both cases, the sudden drift at $t = 10^5$ raises the affected fraction to $\rho_\infty = 0.5$. Under the clean start, Greedy-PSC and Greedy-RR perform similarly. Under the preaffected boundary case, Greedy-RR achieves the smallest delay because it can accumulate and retain client-level evidence for preaffected clients before the fractional change point, so fewer additional clients need to be detected afterward. In contrast, the PSD policies incur larger delays when calibrated to maintain the same FAR under the boundary null. This effect is most pronounced for Score-GC: its score statistic has zero drift at $\rho = \rho_*$, making it more susceptible to false alarms and leading to substantially larger delay and variability.

Figure 3.3 considers a more challenging gradual-drift scenario motivated by camera deployments. We simulate $N = 100$ cameras running a lightweight image classifier, where affected cameras gradually become dirty and therefore experience asynchronous, multistep, and asymmetric accuracy degradation. Similar to Use Case

8 in [35], heterogeneous client streams are generated from STL-10 [25], with client-specific class distributions sampled from a symmetric Dirichlet distribution, and predictions are obtained from MobileNetV3-Small [45]. Drift is induced by six increasing levels of Gaussian blur, with $\sigma \in (0, 5)$, producing average accuracy levels ranging from 93.90% to 56.95%. For each affected camera, the transition times between consecutive blur levels are drawn independently, so both the onset and severity of degradation vary across clients.

In Figure 3.3, we consider two final affected fractions, $\rho_\infty = 0.15$ and $\rho_\infty = 0.5$, with target fraction $\rho_* = 0.1$. To obtain the displayed gradual trajectories, the inter-level durations are sampled from $U(0, 40000)$ for $\rho_\infty = 0.15$ and from $U(0, 70000)$ for $\rho_\infty = 0.5$.

Compared with the sudden-drift setting, all policies incur larger delays because evidence accumulates slowly and unevenly across clients. The PSD policies improve when ρ_∞ increases from 0.15 to 0.5, consistent with their reliance on aggregate population-level evidence: a larger final affected fraction creates a stronger global signal as the drift progresses. In contrast, the ISD policies have larger mean delay and IQR in the $\rho_\infty = 0.5$ case. The slower but more widespread propagation of drift moves the fractional change point earlier, when many affected clients have only weak degradation above Δ. Since ISDs rely on individual client statistics, these weak and gradual degradations cause the corresponding CUSUM statistics to grow more slowly. This increases both the mean detection delay and the variability across runs, especially because degradation speeds differ across clients. Comparing the two ISD policies, Greedy-PSC is more stable than Greedy-RR in both regimes, as reflected by its smaller IQR, because Greedy-RR commits to clients one at a time and therefore produces more variable detection times, whereas Greedy-PSC lets all clients compete in parallel for the top-$(K_* + 1)$ ranks allowing clients with higher degradations to take over.

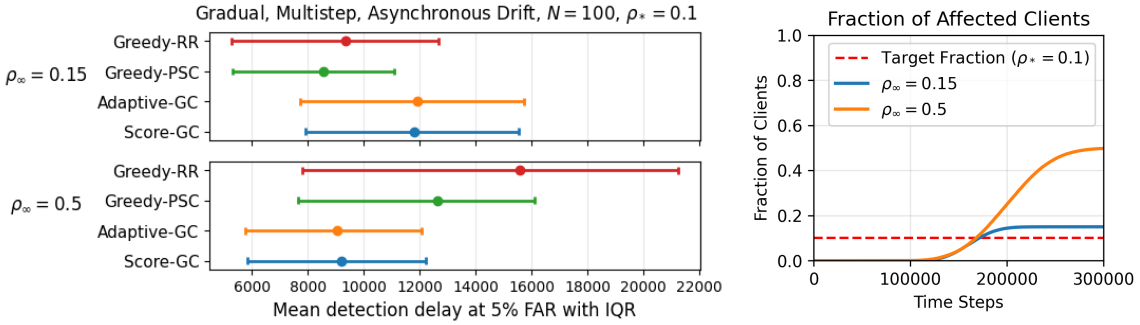


Figure 3.3: Gradual asynchronous drift in a camera-based image classification system with $N = 100$ clients and target fraction $\rho_* = 0.1$. Heterogeneous STL-10 streams are evaluated using MobileNetV3-Small, and drift is induced by client-specific transitions through increasing Gaussian blur levels. The left panel reports the mean detection delay at a 5% false-alarm rate (FAR), with the interquartile range (IQR), for Greedy-RR, Greedy-PSC, Adaptive-GC, and Score-GC under final affected fractions $\rho_\infty = 0.15$ and $\rho_\infty = 0.5$. The right panel shows the corresponding affected-client fraction trajectories.

3.6 Conclusion

This chapter studied concept drift monitoring in hierarchical inference systems, where lightweight client-side models are observed through limited offloaded samples and the goal is to detect when drift has degraded a significant fraction of the population by a significant amount. We formulated this task as fractional change detection, a sequential monitoring problem in which the relevant change point depends not only on individual degradation events, but also on how drift propagates through the population. In particular, detector performance is shaped by the drift propagation speed, the drift style, including synchronous versus asynchronous and symmetric versus asymmetric degradation. These effects are intrinsic to fractional change detection and do not arise in the same form in classical earliest-change formulations.

We developed two complementary detectors for this setting: Greedy Partial Sum-CUSUM, which maintains client-level statistics and can identify affected clients, and Adaptive Global-CUSUM, which uses a mixture model to aggregate population-level evidence. Our theoretical analysis characterizes their false-alarm and delay

behavior, and the simulations demonstrate the resulting trade-offs across synthetic and realistic concept drift scenarios. Overall, the results show that fractional concept drift detection is not merely a variant of multi-stream quickest detection, but a distinct monitoring problem whose performance is shaped by the interaction between sampling, population scale, and drift dynamics.

3.A Appendix

Lemma 3.8 (Return Probability of a CUSUM Excursion)**.** *Let $0 < q < p < 1$ and $\Delta = p - q$. Under the post-change law $\mathbb{P}_1$, let $(X_n)_{n\geq 1}$ be i.i.d. Bernoulli random variables with $\mathbb{P}_1(X_n = 1) = q$, and define the log-likelihood ratio increments*

$$L_n = \begin{cases} a \triangleq \log \frac{1-q}{1-p} > 0, & X_n = 0, \\ -b \triangleq \log \frac{q}{p} < 0, & X_n = 1. \end{cases} \tag{3.45}$$

Let $W_0 = 0$ and $W_n = \max\{0, W_{n-1} + L_n\}$ be the corresponding CUSUM process. For $h > 0$, define the excursion stopping time

$$T(h) = \inf\{n \geq 1 : W_n = 0 \text{ or } W_n \geq h\}, \tag{3.46}$$

and write

$$\alpha(h) \triangleq \mathbb{P}_1(W_{T(h)} = 0). \tag{3.47}$$

Assume that $\alpha = \lim_{h\to\infty} \alpha(h)$ exists. Then

$$\Delta \leq 1 - \alpha \leq \frac{\Delta}{p}. \tag{3.48}$$

Proof. Fix $h > a$. The first step of an excursion from zero is negative with probability q, in which case the CUSUM immediately returns to zero. With probability $1-q$, the first step is positive and the CUSUM moves to a. We therefore analyze the unreflected random walk that starts from a:

$$S_0^{(a)} = a, \qquad S_n^{(a)} = a + \sum_{k=1}^{n} L_k, \quad n \geq 1, \tag{3.49}$$

with stopping time

$$T^{(a)}(h) = \inf\{n \geq 1 : S_n^{(a)} \leq 0 \text{ or } S_n^{(a)} \geq h\}. \tag{3.50}$$

Let $\alpha^{(a)}(h) \triangleq \mathbb{P}_1(S^{(a)}_{T^{(a)}(h)} \leq 0)$.

Under $\mathbb{P}_1$, $\mathbb{E}_1[e^{-L_n}] = 1$, so $(e^{-S_n^{(a)}})_{n\geq 0}$ is a martingale. Moreover, the stopped process is bounded because $S^{(a)}_{T^{(a)}(h)\wedge n} \in [-b, h + a]$. Applying the Optional Stopping Theorem to $T^{(a)}(h)$ in (3.50) gives

$$\mathbb{E}_1[e^{-S^{(a)}_{T^{(a)}(h)}}] = e^{-a}. \tag{3.51}$$

By the law of total expectation and (3.51),

$$\begin{aligned}\mathbb{E}_1[e^{-S^{(a)}_{T^{(a)}(h)}}] = \alpha^{(a)}(h)\,\mathbb{E}_1[e^{-S^{(a)}_{T^{(a)}(h)}} \mid S^{(a)}_{T^{(a)}(h)} \le 0] \\ + (1-\alpha^{(a)}(h))\,\mathbb{E}_1[e^{-S^{(a)}_{T^{(a)}(h)}} \mid S^{(a)}_{T^{(a)}(h)} \ge h] \\ = e^{-a}. \end{aligned} \tag{3.52}$$

The overshoots are bounded by one jump: on the upper exit, $S^{(a)}_{T^{(a)}(h)} \in [h, h+a]$, while on the lower exit, $S^{(a)}_{T^{(a)}(h)} \in [-b, 0]$. Hence

$$\begin{aligned}&\mathbb{E}_1[e^{-S^{(a)}_{T^{(a)}(h)}} \mid S^{(a)}_{T^{(a)}(h)} \ge h] \in [e^{-(h+a)}, e^{-h}], \\ &\mathbb{E}_1[e^{-S^{(a)}_{T^{(a)}(h)}} \mid S^{(a)}_{T^{(a)}(h)} \le 0] \in [1, e^{b}]. \end{aligned} \tag{3.53}$$

Substituting (3.53) into (3.52) yields

$$\frac{e^{-a} - e^{-h}}{e^{b} - e^{-(h+a)}} \le \alpha^{(a)}(h) \le \frac{e^{-a} - e^{-(h+a)}}{1 - e^{-h}}. \tag{3.54}$$

Decomposing according to the first step of the excursion,

$$\alpha(h) = q + (1-q)\alpha^{(a)}(h). \tag{3.55}$$

With $\alpha \triangleq \lim_{h\to\infty} \alpha(h)$, taking $h \to \infty$ in (3.54) and using (3.55) gives

$$q + (1-q)e^{-(a+b)} \le \alpha \le q + (1-q)e^{-a}. \tag{3.56}$$

Since $e^{-a} = (1-p)/(1-q)$ and $e^{-(a+b)} = q(1-p)/(p(1-q))$, this becomes

$$\frac{q}{p} \le \alpha \le 1 - \Delta. \tag{3.57}$$

Equivalently, $\Delta \le 1 - \alpha \le \Delta/p$. □

Lemma 3.9 (Intermittent SR Average Run-Length Bound)**.** *Define a Shiryaev-Roberts (SR)-type statistic [97, Section 7.6] for client i with intermittent samples controlled by a sampling process $(I^{\pi}_t)_t$:*

$$R^{(i)}_t = \begin{cases} R^{(i)}_{t-1}, & I^{\pi}_t \ne i, \\ \left(1 + R^{(i)}_{t-1}\right)\Lambda^{(i)}_t, & I^{\pi}_t = i, \end{cases}, \quad t > 0, \quad R^{(i)}_0 = 0,\ \forall i \tag{3.58}$$

where $\Lambda_t^{(i)}$ *is the intermittent likelihood ratio defined as*

$$\Lambda_t^{(i)} = \mathbb{I}\{I_t^\pi \neq i\} + \mathbb{I}\{I_t^\pi = i\}\frac{\mathrm{p}_1^{(i)}(Y_t^\pi)}{\mathrm{p}_0^{(i)}(Y_t^\pi)}, \tag{3.59}$$

and a stopping time $T^{(i)}(h) = \inf_t \left\{t \geq 1 :\ R_t^{(i)} \geq e^h\right\}$.

Let $\mathcal{A}_*$ *be any affected subset with* $|\mathcal{A}_*| = K_*$*, and suppose client* $i \in \mathcal{A}_*^c$ *satisfies* $\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[\Lambda_t^{(i)} \mid \mathcal{F}_{t-1}] = 1$ *for all* $t \geq 1$*. Then, for all thresholds* h*,*

$$\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[T^{(i)}(h)] \geq e^h. \tag{3.60}$$

Proof. The statistic in (3.58) can be written as

$$R_t^{(i)} = \left(1 + R_{t-1}^{(i)}\right)\Lambda_t^{(i)} - \mathbb{I}\left\{I_t^\pi \neq i\right\}, \quad t \geq 1, \tag{3.61}$$

with $R_0^{(i)} = 0$. Using the recursion and the fact that $\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[\Lambda_t^{(i)} \mid \mathcal{F}_{t-1}] = 1$ for all $t \geq 1$,

$$\begin{aligned}
\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[R_t^{(i)} \mid \mathcal{F}_{t-1}] &= (1 + R_{t-1}^{(i)})\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[\Lambda_t^{(i)} \mid \mathcal{F}_{t-1}] - \mathbb{I}\left\{I_t^\pi \neq i\right\} \\
&= R_{t-1}^{(i)} + \mathbb{I}\left\{I_t^\pi = i\right\} \\
&\leq R_{t-1}^{(i)} + 1,
\end{aligned} \tag{3.62}$$

so $(R_t^{(i)} - t)_{t\geq 0}$ is a $\mathbb{P}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}$-supermartingale.

For each $m \in \mathbb{N}$, define the bounded stopping time $T_m^{(i)} = T^{(i)}(h) \wedge m$. By the Optional Stopping Theorem for supermartingales,

$$\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[T_m^{(i)}] \geq \mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}\left[R^{(i)}_{T_m^{(i)}}\right].$$

On the event $\{T^{(i)}(h) \leq m\}$, we have $R^{(i)}_{T_m^{(i)}} = R^{(i)}_{T^{(i)}(h)} \geq e^h$ by the definition of $T^{(i)}(h)$, and since $R^{(i)}_{T_m^{(i)}} \geq 0$ always. We have,

$$\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}\left[R^{(i)}_{T^{(i)}(h)\wedge m}\right] \geq e^h \mathbb{P}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}\left(T^{(i)}(h) \leq m\right).$$

Letting $m \to \infty$ and applying the Monotone Convergence Theorem yields $\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[T^{(i)}_m] \to \mathbb{E}_\infty[T^{(i)}(h)]$ and suppose $\mathbb{E}^{\mathcal{A}^*}_\infty\left[T^{(i)}(h)\right] < \infty$ (otherwise the statement of the lemma is trivial), $\mathbb{P}^{\mathcal{A}}_\infty(T^{(i)}(h) \le m) \to 1$, and therefore

$$\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[T^{(i)}(h)] \ge \mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}\left[R^{(i)}_{T^{(i)}(h)}\right] \ge e^h,$$

which completes the proof. □

3.A.1 Proof of Lemma 3.6

Proof. Fix $t \ge 1$ and $y \in \{0,1\}$. Under uniform sampling, $\mathbb{P}(I^\pi_t = i) = 1/N$ for every $i \in \mathcal{N}$, so marginalizing over the sampled client gives

$$\begin{aligned}\mathbb{P}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}(Y^\pi_t = y) =& \frac{1}{N}\sum_{i\in\mathcal{N}}\left[\mathbb{I}\{i \in \mathcal{A}_t\}\,\mathrm{p}^{(i)}_1(y) + \mathbb{I}\{i \notin \mathcal{A}_t\}\,\mathrm{p}^{(i)}_0(y)\right] \\ =& \frac{1}{N}\sum_{i\in\mathcal{N}}\mathrm{p}^{(i)}_0(y) + \frac{1}{N}\sum_{i\in\mathcal{A}_t}\left[\mathrm{p}^{(i)}_1(y) - \mathrm{p}^{(i)}_0(y)\right]. \end{aligned} \tag{3.63}$$

Since $q_i = p_i - \Delta$ for every client $i \in \mathcal{A}_t$, the difference $\delta(y) \triangleq \mathrm{p}^{(i)}_1(y) - \mathrm{p}^{(i)}_0(y)$ does not depend on i. Therefore, we have $\sum_{i\in\mathcal{A}_t}[\mathrm{p}^{(i)}_1(y) - \mathrm{p}^{(i)}_0(y)] = |\mathcal{A}_t|\,\delta(y)$, and (3.63) becomes

$$\mathbb{P}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}(Y^\pi_t = y) = \frac{1}{N}\sum_{i\in\mathcal{N}}\mathrm{p}^{(i)}_0(y) + \rho_t\,\delta(y). \tag{3.64}$$

On the other hand, the mixture likelihood (3.26) under uniform sampling evaluates to

$$\begin{aligned}\mathcal{L}(y;\,\rho_t) &= \frac{1}{N}\sum_{i\in\mathcal{N}}\left[\rho_t\,\mathrm{p}^{(i)}_1(y) + (1-\rho_t)\,\mathrm{p}^{(i)}_0(y)\right] \\ &= \frac{1}{N}\sum_{i\in\mathcal{N}}\mathrm{p}^{(i)}_0(y) + \rho_t\cdot\frac{1}{N}\sum_{i\in\mathcal{N}}\left[\mathrm{p}^{(i)}_1(y) - \mathrm{p}^{(i)}_0(y)\right] \\ &= \frac{1}{N}\sum_{i\in\mathcal{N}}\mathrm{p}^{(i)}_0(y) + \rho_t\,\delta(y), \end{aligned} \tag{3.65}$$

which coincides with (3.64). The i.i.d. property when ρ_t is constant follows because, under uniform sampling, the indices $(I^\pi_t)_{t\ge1}$ are i.i.d. uniform on $\mathcal{N}$ and Y^π_t is condi-

tionally independent of all other observations given I_t^π and the cardinality of the set of affected clients. □

3.A.2 Proof of Theorem 3.4

We first prove the lower bound in (3.22).

Proof of the lower bound on WARL. The proof compares G_t with an auxiliary Shiryaev-Roberts-type statistic built from the unaffected clients and then shows that the centered auxiliary process is a supermartingale, which yields the desired lower bound via the arguments used in Lemma 3.9.

By the definition of WARL in (3.7), the least favorable null configuration occurs at the boundary, namely when exactly K_* clients are affected. Thus, let $\mathcal{A}_*$ be any fixed affected subset with $|\mathcal{A}_*| = K_*$ and corresponding change-point vector $\boldsymbol{\tau}(\mathcal{A}_*)$. Then, by the definition of the test statistic in (3.18) and Lemma 3.3, and since the highest CUSUM among $N - K_*$ unaffected clients is at least ranked $(K_* + 1)$, for any time t we have

$$G_t = W_t^{(\sigma_t(K_*+1))} \leq \max_{i \in \mathcal{A}_*^c} W_t^{(i)}, \tag{3.66}$$

where $\mathcal{A}_*^c$ denotes the set of unaffected clients.

As introduced in (3.58), define *intermittent Shiryaev-Roberts (SR)* statistics $R_t^{(i)}$ for each unaffected client $i \in \mathcal{A}_*^c$ with intermittent samples $\Lambda_t^{(i)}$, where

$$\Lambda_t^{(i)} = \mathbb{I}\{I_t^\pi \neq i\} + \mathbb{I}\{I_t^\pi = i\} \frac{\mathrm{p}_1^{(i)}(Y_t^\pi)}{\mathrm{p}_0^{(i)}(Y_t^\pi)}, \tag{3.67}$$

and let $R_t^\Sigma = \sum_{i \in \mathcal{A}_*^c} R_t^{(i)}$ denote the sum of intermittent SR statistics of unaffected clients.

We first compare each unaffected client's CUSUM with its corresponding intermittent SR statistic. Under $\{I_t^\pi \neq i\}$, both $R_t^{(i)} = R_{t-1}^{(i)}$ and $W_t^{(i)} = W_{t-1}^{(i)}$ remain unchanged. For the set of sampling times of client i, $S_{k:t}^{(i)} = \{s \in \{k, \ldots, t\} : I_s^\pi = i\}$,

we have

$$\begin{aligned}\exp\bigl(W_t^{(i)}\bigr) &= \max_{\ell \in S_{1:t}^{(i)}} \prod_{k \in S_{\ell:t}^{(i)}} \Lambda_k^{(i)} \\ &\leq \sum_{\ell \in S_{1:t}^{(i)}} \prod_{k \in S_{\ell:t}^{(i)}} \Lambda_k^{(i)} \\ &= R_t^{(i)}. \end{aligned} \tag{3.68}$$

where the inequality follows because the summands are nonnegative, and the equalities follow from the definitions of the CUSUM and intermittent SR statistics. Therefore, for all $t \geq 0$,

$$\begin{aligned}\max_{i \in \mathcal{A}_*^c} \exp(W_t^{(i)}) &\leq \sum_{i \in \mathcal{A}_*^c} \exp(W_t^{(i)}) \\ &\leq \sum_{i \in \mathcal{A}_*^c} R_t^{(i)} \\ &= R_t^{\Sigma}. \end{aligned} \tag{3.69}$$

Combining (3.69) with (3.66), we have

$$\{G_t \geq h\} \subseteq \left\{ \max_{i \in \mathcal{A}_*^c} W_t^{(i)} \geq h \right\} Z \subseteq \left\{ R_t^{\Sigma} \geq e^h \right\}. \tag{3.70}$$

Therefore, by (3.66) and (3.69), we have

$$\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[T^{\Sigma}(h)] \leq \mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[T^{\text{G-PSC}}(h)], \tag{3.71}$$

where $T^{\Sigma}(h) = \inf\{t \geq 1 : R_t^{\Sigma} \geq e^h\}$ and $T^{\text{G-PSC}}(h)$ is as defined in (3.20).

Under $\mathbb{P}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}$, since I_t^{π} is $\mathcal{F}_{t-1}$-measurable by (3.19), for each $t \geq 1$ and $i \in \mathcal{A}_*^c$, conditioning on $\mathcal{F}_{t-1}$ yields

$$\begin{aligned}\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}\left[\Lambda_t^{(i)} \,\middle|\, \mathcal{F}_{t-1}\right] &= \mathbb{I}\{I_t^{\pi} \neq i\} \\ &\quad + \mathbb{I}\{I_t^{\pi} = i\} \mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}\left[\frac{\mathrm{p}_1^{(i)}(Y_t^{\pi})}{\mathrm{p}_0^{(i)}(Y_t^{\pi})} \,\middle|\, \mathcal{F}_{t-1}, I_t^{\pi} = i\right] \\ &= \mathbb{I}\{I_t^{\pi} \neq i\} + \mathbb{I}\{I_t^{\pi} = i\} \\ &= 1. \end{aligned} \tag{3.72}$$

Here we use that, for $i \in \mathcal{A}_*^c$, the observation Y_t^π is distributed according to $\mathrm{p}_0^{(i)}$ whenever $I_t^\pi = i$, and hence conditioned on the filtration, $\Lambda_t^{(i)}$ has mean one for each unaffected client $i \in \mathcal{A}_*^c$. Therefore, by (3.72) and (3.62),

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\tau}(\mathcal{A}_*)}^{\mathcal{A}_*}\left[R_t^\Sigma \,\middle|\, \mathcal{F}_{t-1}\right] &= \sum_{i\in\mathcal{A}_*^c} \mathbb{E}_{\boldsymbol{\tau}(\mathcal{A}_*)}^{\mathcal{A}_*}\left[R_t^{(i)} \,\middle|\, \mathcal{F}_{t-1}\right] \\
&= \sum_{i\in\mathcal{A}_*^c} \left(R_{t-1}^{(i)} + \mathbb{I}\{I_t^\pi = i\}\right) \\
&= R_{t-1}^\Sigma + \mathbb{I}\{I_t^\pi \in \mathcal{A}_*^c\} \\
&\le R_{t-1}^\Sigma + 1
\end{aligned} \tag{3.73}$$

Thus, $(R_t^\Sigma - t)_{t\ge 0}$ is an $\mathcal{F}_t$-supermartingale under $\mathbb{P}_{\boldsymbol{\tau}(\mathcal{A}_*)}^{\mathcal{A}_*}$, and applying the same optional-stopping argument as in Lemma 3.9, together with (3.71), yields, for any $\mathcal{A}_*$ of size K_* and any $\boldsymbol{\tau}(\mathcal{A}_*)$,

$$\mathbb{E}_{\boldsymbol{\tau}(\mathcal{A}_*)}^{\mathcal{A}_*}[T^{\text{G-PSC}}(h)] \ge \mathbb{E}_{\boldsymbol{\tau}(\mathcal{A}_*)}^{\mathcal{A}_*}[T^\Sigma(h)] \ge e^h, \tag{3.74}$$

which completes the proof.

Next, we analyze the upper bound in (3.23).

Proof of the upper-bound on WADD. The proof is obtained by first showing that the stopping event dominates the case where the CUSUM statistics of all affected clients in $\mathcal{A}$ exceed the threshold, and bounding the number of samples used on unaffected clients based on the dynamics of the greedy sampling policy.

Let $\mathcal{A}$ be any fixed set of affected clients with $|\mathcal{A}| = K_* + 1$. Since $\nu = \max_{i\in\mathcal{A}} \tau_i$, any client that changes before ν may have already accumulated nonnegative CUSUM evidence by the fractional change point. Such accumulated evidence can only decrease the remaining time needed for all affected CUSUM statistics to reach level h. Thus, for the purpose of an upper bound, it suffices to consider the least favorable initialization at the fractional change point, in which all affected CUSUM statistics start from zero. By time homogeneity, this case can be represented by setting $\tau_i = 0$

for all $i \in \mathcal{A}$. Therefore,

$$\sup_{\tau_i: i\in\mathcal{A}} \mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}\left[T^{\text{G-PSC}}(h) - \nu \middle| T^{\text{G-PSC}}(h) > \nu\right] \leq \mathbb{E}^{\mathcal{A}}_{\mathbf{0}}\left[T^{\text{G-PSC}}(h)\right], \tag{3.75}$$

where $\mathbb{E}^{\mathcal{A}}_{\mathbf{0}}$ denotes expectation when $\boldsymbol{\tau}(\mathcal{A}) = \mathbf{0}$.

By Lemma 3.3 and the fact that $|A| = K_* + 1$, for any $t \geq 0$, the test statistic satisfies

$$\begin{aligned} G_t &= \sum_{j=K_*+1}^{N} W_t^{(\sigma_t(j))} \\ &= W_t^{(\sigma_t(K_*+1))} \\ &\geq \min_{i\in\mathcal{A}} W_t^{(i)}. \end{aligned} \tag{3.76}$$

Define a stopping time when CUSUM statistics of all affected clients exceed the threshold h, i.e., $T'(h) = \inf_t\{t \geq 1 : W_t^{(i)} \geq h,\ \forall i \in \mathcal{A}\}$. At this stopping time $T'(h)$, every client $i \in \mathcal{A}$ satisfies $W^{(i)}_{T'(h)} \geq h$. By (3.76), at time $T'(h)$, the stopping condition of Greedy-PSC is also satisfied, i.e., $G_{T'(h)} \geq h$. Since $T^{\text{G-PSC}}(h)$ is the first time the condition is met,

$$T^{\text{G-PSC}}(h) \leq T'(h), \tag{3.77}$$

thus it suffices to bound $\mathbb{E}^{\mathcal{A}}_{\mathbf{0}}[T'(h)]$. We decompose the total time up to $T'(h)$ as

$$T'(h) = C_a(T'(h)) + C_u(T'(h)), \tag{3.78}$$

where $C_a(T'(h)) = \sum_{t=1}^{T'(h)} \mathbb{I}\{I_t^{\text{G-PSC}} \in \mathcal{A}\}$ counts the samples from affected clients and $C_u(T'(h)) = \sum_{t=1}^{T'(h)} \mathbb{I}\{I_t^{\text{G-PSC}} \notin \mathcal{A}\}$ counts the samples from unaffected clients. Next, we will bound $\mathbb{E}^{\mathcal{A}}_{\mathbf{0}}[C_a(T'(h))]$ and $\mathbb{E}^{\mathcal{A}}_{\mathbf{0}}[C_u(T'(h))]$.

Remark 3.10. *Under the Greedy-PSC policy, once the CUSUM of client $i \in \mathcal{N}$ exceeds the threshold h at some t, i.e., $W_t^{(i)} \geq h$, client i is not sampled again before the stopping time.*

Let $N_t^{(i)} = \sum_{k=1}^{t} \mathbb{I}\{I_k^{\text{G-PSC}} = i\}$ denote the number of times client i is sampled by the time t, and let $\widetilde{W}_n^{(i)}$ denote the CUSUM of client i after n samples, where $\widetilde{W}_{N_t^{(i)}}^{(i)} = W_t^{(i)}$. Define $T^{(i)}(h) = \inf\{n \geq 1 : \widetilde{W}_n^{(i)} \geq h\}$.

For an affected client $i \in \mathcal{A}$, the LLR increments of the sampled CUSUM process $(\widetilde{W}_n^{(i)})_{n\geq 0}$ are i.i.d. under the post-change distribution with positive mean

$$\mathbb{E}_{\mathbf{0}}^{\mathcal{A}}\left[\log \frac{\mathrm{p}_1^{(i)}(X_t^{(i)})}{\mathrm{p}_0^{(i)}(X_t^{(i)})}\right] = D_{\mathrm{KL}}(\mathrm{p}_1^{(\mathrm{i})} \,||\, \mathrm{p}_0^{(\mathrm{i})}) \triangleq \beta_i > 0, \quad \forall i \in \mathcal{A}, \quad t > 0.$$

Thus, as $h \to \infty$, by Wald's Identity, we have

$$\mathbb{E}_{\mathbf{0}}^{\mathcal{A}}[T^{(i)}(h)] = \frac{h}{\beta_i} + O(1), \tag{3.79}$$

where the $O(1)$ term accounts for the expected overshoot. By Remark 3.10, no affected client is sampled again after its CUSUM first reaches h before $T'(h)$, so the total number of affected-client samples up to $T'(h)$ is the sum of these per-client hitting times:

$$\begin{aligned}\mathbb{E}_{\mathbf{0}}^{\mathcal{A}}\left[C_a(T'(h))\right] &= \sum_{i\in\mathcal{A}} \mathbb{E}_{\mathbf{0}}^{\mathcal{A}}\left[T^{(i)}(h)\right] \\ &\leq \left(\sum_{i\in\mathcal{A}} \frac{1}{\beta_i}\right) h + O(1).\end{aligned} \tag{3.80}$$

To bound the expected number of samples from unaffected clients, we use the fact that the greedy rule continues sampling the same client until its CUSUM either exceeds the CUSUM of the K_*-th client or returns to zero. Thus, before an affected client's CUSUM reaches the threshold h, the Greedy-PSC policy samples that client intermittently. From the perspective of $(\widetilde{W}_n^{(i)})_{n\geq 0}$, the round-robin tie-breaking rule lets us bound the samples spent on unaffected clients after an affected-client excursion returns to zero by one full round-robin excursion cycle over all unaffected clients.

For each affected client $i \in \mathcal{A}$, let α_i denote the limiting probability that an excursion of its sampled post-change CUSUM process returns to zero. By Lemma 3.8,

applied with $p = p_i$ and $q = q_i = p_i - \Delta$,

$$\Delta \leq 1 - \alpha_i \leq \frac{\Delta}{p_i}. \tag{3.81}$$

Each return of $(\widetilde{W}_n^{(i)})_{n\geq 0}$ to zero is a renewal event, and each new excursion independently either crosses the threshold h or returns to zero. Hence, as $h \to \infty$, the number of returning excursions before the first excursion that crosses h, denoted by $M_i \in \{0, 1, 2, \dots\}$, is geometric with success probability $1 - \alpha_i$. Thus,

$$\mathbb{E}[M_i] = \frac{\alpha_i}{1 - \alpha_i}. \tag{3.82}$$

For each unaffected client $j \in \mathcal{A}^c$, let η_j denote the expected number of samples collected from client j during an excursion that returns to zero. Under the pre-change law of client j, the LLR increments have negative mean

$$\mathbb{E}_{\mathbf{0}}^{\mathcal{A}}\left[\log \frac{\mathrm{p}_1^{(j)}(X_t^{(j)})}{\mathrm{p}_0^{(j)}(X_t^{(j)})}\right] = -D_{\mathrm{KL}}(\mathrm{p}_0^{(\mathrm{j})} \,||\, \mathrm{p}_1^{(\mathrm{j})}) \quad \forall j \in \mathcal{A}^c.$$

We decompose the excursion according to its first sample. If the first increment is negative, which occurs with probability p_j, the CUSUM immediately resets to zero and the excursion length is one. If the first increment is positive, which occurs with probability $1 - p_j$, the CUSUM starts from $\log \frac{1-q_j}{1-p_j}$ and then evolves as a random walk with negative drift until it returns to zero. Applying Wald's Identity to this return time, and upper-bounding the undershoot below zero by the largest negative jump $\log \frac{p_j}{q_j}$, yields

$$\begin{aligned}\eta_j &\leq 1 + (1 - p_j) \cdot \frac{\log \frac{p_j(1-q_j)}{q_j(1-p_j)}}{D_{\mathrm{KL}}(\mathrm{p}_0^{(\mathrm{j})} \,||\, \mathrm{p}_1^{(\mathrm{j})})} \\ &= \frac{\log \frac{p_j}{q_j}}{D_{\mathrm{KL}}(\mathrm{p}_0^{(\mathrm{j})} \,||\, \mathrm{p}_1^{(\mathrm{j})})}.\end{aligned} \tag{3.83}$$

The last equality follows from expanding $D_{\mathrm{KL}}(\mathrm{p}_0^{(\mathrm{j})} \,||\, \mathrm{p}_1^{(\mathrm{j})}) = p_j \log \frac{p_j}{q_j} + (1-p_j) \log \frac{1-p_j}{1-q_j}$.

Combining (3.82) and (3.83), we obtain

$$\begin{aligned}\mathbb{E}_{\mathbf{0}}^{\mathcal{A}}\left[C_u(T'(h))\right] &\leq \sum_{i\in\mathcal{A}} \mathbb{E}[M_i+1] \sum_{j\in\mathcal{A}^c} \eta_j \\ &= \sum_{i\in\mathcal{A}} \frac{1}{1-\alpha_i} \sum_{j\in\mathcal{A}^c} \eta_j \end{aligned} \tag{3.84}$$

3.A.3 Proof of Theorem 3.7

We first prove (3.36).

Proof of the lower bound on WARL

Let $\mathcal{A}_*$ be any fixed affected subset of size $K_* = \lfloor \rho_* N \rfloor$, corresponding change-point vector $\boldsymbol{\tau}(\mathcal{A}_*)$, and $(\hat{\rho}_t)_{t\geq 1}$ be any sequence such that $\hat{\rho}_t$ is $\mathcal{F}_{t-1}$-measurable. Define a new stopping time based on a Shiryaev-Roberts (SR)-type statistic:

$$T_\perp(h) = \inf\left\{ t \geq 1 :\ R_t^\perp = \sum_{\ell=1}^{t} \prod_{k=\ell}^{t} \hat{\Lambda}_k \geq e^h \right\},$$

where $\hat{\Lambda}_k = \frac{\mathcal{L}(Y_k^\pi;\hat{\rho}_k)}{\mathcal{L}(Y_k^\pi;\rho_*)}$. Moreover, for all t,

$$\begin{aligned} R_t^\perp &= \sum_{\ell=1}^{t} \prod_{k=\ell}^{t} \hat{\Lambda}_k \\ &\geq \max_{1\leq \ell \leq t} \prod_{k=\ell}^{t} \hat{\Lambda}_k \\ &= \exp\bigl(W_t^{\text{A-GC}}\bigr), \end{aligned}$$

and therefore $\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[T_\perp(h)] \leq \mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[T^{\text{A-GC}}(h)]$ for every threshold h, where $T^{\text{A-GC}}(h)$ is defined in (3.35).

Under $\mathbb{P}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}$ and uniform sampling, Y_t^π has likelihood $\mathcal{L}(Y_t^\pi;\rho_t)$, and since $\rho_t \leq \rho_* < \hat{\rho}_t$ for all $t \geq 1$, $\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[\hat{\Lambda}_t \mid \mathcal{F}_{t-1}] \leq 1$. Therefore, $(R_t^\perp - t)_{t\geq 0}$ is an

$\mathcal{F}_t$-supermartingale, and by the same argument in Lemma 3.9 for any $\mathcal{A}_*$,

$$\begin{aligned}\mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}\left[T^{\text{A-GC}}(h)\right] &\geq \mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}[T_\perp(h)] \\ &= \mathbb{E}^{\mathcal{A}_*}_{\boldsymbol{\tau}(\mathcal{A}_*)}\left[R^\perp_{T_\perp(h)}\right] \\ &\geq e^h,\end{aligned}$$

which establishes (3.36).

Proof of the upper-bound on WADD. Next, we prove (3.37). The argument is structured around five lemmas.

Let $\mathcal{A}$ be any affected subset with $|\mathcal{A}| = K_* + 1$, let $\boldsymbol{\tau}(\mathcal{A})$ be an arbitrary change-point configuration, and write $\rho = (K_* + 1)/N$. The fractional change point is $\nu = \max_{i\in\mathcal{A}} \tau_i$, so that $\rho_t = \rho$ for all $t > \nu$. Write $T \triangleq T^{\text{A-GC}}(h)$ and $\beta(\rho, \rho_*) = D_{\text{KL}}(\mathcal{L}(\cdot;\rho)\|\mathcal{L}(\cdot;\rho_*)) > 0$.

We first record the regularity properties used in the analysis. The only model regularity required is that the Bernoulli parameters are nondegenerate, $0 < q_i < p_i < 1$, and that the clipping interval is nonempty, i.e., $0 < \epsilon < 1 - \rho_*$.

Lemma 3.11 (Regularity Properties of the Adaptive Global-CUSUM)**.** *Let*

$$L_t(r, \rho_*) = \log \frac{\mathcal{L}(Y^\pi_t; r)}{\mathcal{L}(Y^\pi_t; \rho_*)}.$$

Under the system model and uniform sampling, the following properties hold.

(R1) The increment is uniformly bounded: There exists $L_{\max} < \infty$ such that $|L_t(r, \rho_*)| \leq L_{\max}$ a.s. for all $t \geq 1$ and $r \in [\rho_* + \epsilon, 1]$.

(R2) The increment is Lipschitz in r: There exists $\bar{L} < \infty$ such that $|L_t(r, \rho_*) - L_t(s, \rho_*)| \leq \bar{L}\,|r - s|$ a.s. for all $r, s \in [\rho_* + \epsilon, 1]$.

(R3) The transformed variables $Z_s = (Y^\pi_s - p_{I^\pi_s})/(q_{I^\pi_s} - p_{I^\pi_s})$ satisfy $\mathbb{E}[Z_s] = \rho$ and $\text{Var}(Z_s) \leq \sigma^2_Z < \infty$ for $s > \nu$.

Proof. For $y \in \{0,1\}$, the mixture likelihood $\mathcal{L}(y;r)$ is affine in r and strictly positive on the compact set $r \in [\rho_* + \epsilon, 1]$ under the nondegenerate Bernoulli parameters. Hence $L_t(r, \rho_*)$ is bounded on $\{0,1\} \times [\rho_* + \epsilon, 1]$, proving (R1). The same explicit finite-state expression is continuously differentiable in r on this compact set, so its derivative is bounded; the mean-value theorem gives (R2).

For (R3), when $s > \nu$ the affected fraction is ρ and, by Lemma 3.6, uniform sampling makes the sampled client an unbiased draw from the population. Conditional on $I_s^\pi = i$, the transformed variable has mean one if client i is affected and zero otherwise, since

$$\mathbb{E}\left[\frac{Y_s^\pi - p_i}{q_i - p_i} \,\middle|\, I_s^\pi = i\right] = \begin{cases} 1, & i \text{ is affected,} \\ 0, & i \text{ is unaffected.} \end{cases}$$

Averaging over the uniformly sampled client gives $\mathbb{E}[Z_s] = \rho$. Finally, Z_s is bounded because $Y_s^\pi \in \{0,1\}$ and $q_i - p_i = -\Delta \neq 0$, so $\mathrm{Var}(Z_s) \leq \sigma_Z^2 < \infty$ for some finite constant σ_Z^2. □

For $m \in \mathbb{N}$, define the bounded stopping time $T_m = T \wedge (\nu + m)$; we let $m \to \infty$ at the end.

Step 1: Martingale decomposition. Under $\mathbb{P}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}$, for $t > \nu$ the observations Y_t^π are i.i.d. with PMF $\mathcal{L}(\cdot;\rho)$ (Lemma 3.6), so the oracle increments $L_t(\rho, \rho_*)$ are i.i.d. with mean $\beta(\rho, \rho_*)$. The centered partial sums

$$M_n = \sum_{t=\nu+1}^{n} \left(L_t(\rho, \rho_*) - \beta(\rho, \rho_*)\right)$$

form a zero-mean martingale. The optional stopping theorem applied to T_m gives

$$\mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}\left[\sum_{t=\nu+1}^{T_m} L_t(\rho, \rho_*) \,\middle|\, T > \nu\right] = \beta(\rho, \rho_*) \cdot \mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}[T_m - \nu \mid T > \nu]. \tag{3.85}$$

Splitting the oracle sum into the adaptive part based on the estimated $\hat{\rho}_t$ and a residual,

$$\sum_{t=\nu+1}^{T_m} L_t(\rho, \rho_*) = \sum_{t=\nu+1}^{T_m} L_t(\hat{\rho}_t, \rho_*) + \sum_{t=\nu+1}^{T_m} \big(L_t(\rho, \rho_*) - L_t(\hat{\rho}_t, \rho_*)\big), \tag{3.86}$$

and substituting into (3.85) yields

$$\mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}[T_m - \nu \mid T > \nu] = \frac{1}{\beta(\rho, \rho_*)}\big(h + x_1(h, \nu, m) + x_2(h, \nu, m)\big), \tag{3.87}$$

where

$$x_1(h, \nu, m) = \mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}\left[\sum_{t=\nu+1}^{T_m} L_t(\hat{\rho}_t, \rho_*) - h \;\middle|\; T > \nu\right],$$

$$x_2(h, \nu, m) = \mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}\left[\sum_{t=\nu+1}^{T_m} \big(L_t(\rho, \rho_*) - L_t(\hat{\rho}_t, \rho_*)\big) \;\middle|\; T > \nu\right].$$

The proof is completed by establishing the following four results:

- Lemma 3.12: $x_1(h, \nu, m) = O(1)$,
- Lemma 3.13: $\sup_\nu \mathbb{E}[T - \nu \mid T > \nu] \le C_T h$ for large h (the crude detection delay bound),
- Lemma 3.14: that the cumulative squared estimation error is $O(\log h)$,
- Lemma 3.15: $x_2(h, \nu, m) = o(h)$.

Step 2: Overshoot bound.

Lemma 3.12 (Overshoot bound)**.** *Under (R1),*

$$\sup_{\nu \ge 0} \limsup_{m \to \infty} x_1(h, \nu, m) \le L_{\max}.$$

Proof. Let $Z = U(T) + 1$ be the start of the final excursion. Within this excursion the statistic accumulates from zero: $W_T^{\text{A-GC}} = \sum_{s=Z}^{T} L_s(\hat{\rho}_s, \rho_*)$, and $W_{T-1}^{\text{A-GC}} < h \leq W_T^{\text{A-GC}}$, so the overshoot satisfies $0 \leq W_T^{\text{A-GC}} - h \leq L_T(\hat{\rho}_T, \rho_*) \leq L_{\max}$.

Since Z is the start of the excursion that crosses h, the running partial sums $\sum_{s=Z}^{t} L_s(\hat{\rho}_s, \rho_*) > 0$ for all $Z \leq t \leq T$, including $Z \leq \nu$. Therefore,

$$\begin{aligned}\sum_{t=\nu+1}^{T_m} L_t(\hat{\rho}_t, \rho_*) &\leq \sum_{t=Z}^{T} L_t(\hat{\rho}_t, \rho_*) \\ &= W_T^{\text{A-GC}} \\ &\leq h + L_{\max}.\end{aligned}$$

When $Z > \nu$, we have $\sum_{t=\nu+1}^{Z-1} L_t(\hat{\rho}_t, \rho_*) \leq 0$ because the CUSUM reset to zero at time $Z-1$ (i.e., $W_{Z-1}^{\text{A-GC}} = 0$), and any preceding excursions in $[\nu+1, Z-1]$ ended with the statistic at zero or below. Hence the same bound holds. Taking conditional expectations and letting $m \to \infty$ gives $x_1(h, \nu, m) \leq L_{\max}$. □

Step 3: Crude linear delay bound. We show that the expected detection delay grows at most linearly in h. The proof decomposes the post-ν trajectory into CUSUM excursions and analyzes each type separately.

Lemma 3.13 (Crude delay bound)**.** *There exist constants $C_T < \infty$ and $h_0 > 0$ such that for all $h \geq h_0$,*

$$\sup_{\boldsymbol{\tau}(\mathcal{A})} \mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}[T - \nu \mid T > \nu] \;\leq\; C_T\, h. \tag{3.88}$$

Proof. After ν, the observations Y_t^{π} are i.i.d. with PMF $\mathcal{L}(\cdot; \rho)$ (Lemma 3.6). We decompose the post-ν trajectory into excursions defined by the successive reset times of the CUSUM.

(a) Excursion structure.

Define the post-ν successive reset times $U_0 < U_1 < U_2 < \cdots$, with $U_0 = \inf\{t \geq \nu : W_t^{\text{A-GC}} = 0\}$, and $U_j = \inf\{t > U_{j-1} : W_t^{\text{A-GC}} = 0\}$ for $j \geq 1$. We

call $(\nu, U_0]$ the *ongoing excursion*; for $j \geq 1$, each excursion $(U_{j-1}, U_j]$ is *failed* if it resets before hitting h, and the first excursion that hits h is *successful*. If M indexes the successful excursion, then the delay is the sum of the ongoing part, the failed excursions, and the successful excursion:

$$T - \nu = \underbrace{(U_0 - \nu)}_{\text{ongoing excursion}} + \underbrace{\sum_{j=1}^{M-1} (U_j - U_{j-1})}_{\text{failed excursions}} + \underbrace{(T - U_{M-1})}_{\text{successful excursion}} . \tag{3.89}$$

We bound the expected value of each term.

(b) The ongoing excursion contributes $O(h)$.

On $\{T > \nu\}$, if $W_\nu^{\text{A-GC}} > 0$ then the detector is already in an ongoing excursion. This excursion ends when the CUSUM either crosses h or resets to zero. Since increments are bounded by $L_{\max}$ ((R1)), the conditional expected increment $\mathbb{E}_\rho[L_t(r, \rho_*)]$ for any fixed $r \in [\rho_* + \epsilon, 1]$ satisfies

$$\begin{aligned} \mathbb{E}_\rho[L_t(r, \rho_*)] &= \beta(\rho, \rho_*) - D_{\text{KL}}(\mathcal{L}(\cdot; \rho) \| \mathcal{L}(\cdot; r)) \\ &\geq I_{\min}, \end{aligned}$$

where $I_{\min} \triangleq \beta(\rho, \rho_*) - \max_{r \in [\rho_* + \epsilon, 1]} D_{\text{KL}}(\mathcal{L}(\cdot; \rho) \| \mathcal{L}(\cdot; r))$ is a finite constant, $\mathbb{E}_\rho$ denotes the expectation for the case when the affected fraction is ρ. Since $\hat{\rho}_t \in [\rho_* + \epsilon, 1]$, the drift at each step is at least $I_{\min}$.

If $I_{\min} \geq 0$, the excursion reaches h in $O(h)$ expected time by Wald's identity. If $I_{\min} < 0$, the negative drift drives the statistic back to zero in $O(h)$ expected time because the CUSUM starts below h and each step decreases the value by at least $|I_{\min}|$ on average. Hence $\mathbb{E}[U_0 - \nu \mid T > \nu] = O(h)$.

(c) Fresh excursions cross h with positive probability. Consider a fresh excursion starting at reset time $U_j \geq \nu$ (so $W_{U_j}^{\text{A-GC}} = 0$). Within this excursion, the estimator $\hat{\rho}_t$ is computed entirely from post-change observations drawn i.i.d. from $\mathcal{L}(\cdot; \rho)$. We show there exists $\zeta > 0$, independent of h, such that the excursion crosses h before resetting with probability at least ζ.

Fix $\delta_0 \in (0, \beta(\rho, \rho_*)/(2\bar{L}))$ and define the event

$$\xi_j \triangleq \left\{ \left| \frac{1}{n} \sum_{k=1}^{n} Z_{U_j+k} - \rho \right| < \delta_0 \text{ for all } n \geq n_0 \right\}, \tag{3.90}$$

where Z_s are i.i.d. with mean ρ and variance σ_Z^2. Since Z_s is bounded in an interval of length $\frac{1}{\Delta}$, by Hoeffding's inequality applied to each n followed by a union bound over $n \geq n_0$,

$$\begin{aligned} \mathbb{P}(\xi_j^c) &\leq \sum_{k=n_0}^{\infty} 2 \exp\left(-\frac{2k\delta_0^2}{(1/\Delta)^2} \right) \\ &\leq C_1 e^{-C_2 n_0}. \end{aligned} \tag{3.91}$$

for constants $C_1, C_2 > 0$. Choosing n_0 large enough ensures $\mathbb{P}(\xi_j) \geq 1/2$.

On the event ξ_j, for all $n \geq n_0$ the estimator satisfies $|\hat{\rho}_{U_j+n} - \rho| \leq \delta_0$, so by (R2) the adaptive increment satisfies

$$L_t(\hat{\rho}_t, \rho_*) \geq L_t(\rho, \rho_*) - \bar{L}\delta_0, \qquad t \geq U_j + n_0.$$

In particular, within this excursion the CUSUM after time $U_j + n_0$ grows with a drift

$$\begin{aligned} \mathbb{E}_\rho[L_t(\hat{\rho}_t, \rho_*)] &\geq \beta(\rho, \rho_*) - \bar{L}\delta_0 \\ &> \beta(\rho, \rho_*)/2. \end{aligned} \tag{3.92}$$

It remains to lower-bound the probability of surviving the first n_0 steps without a reset. Define the probability that the CUSUM does not reset during the first n_0 steps of the excursion as

$$\zeta_0 \triangleq \mathbb{P}_\rho\Big(\min_{1 \leq m \leq n_0} \sum_{k=1}^{m} L_{U_j+k}(\hat{\rho}_{U_j+k}, \rho_*) \geq 0 \Big) > 0.$$

Since n_0 is a fixed constant, ζ_0 depends only on the joint distribution of the first n_0 post-change increments and is independent of h.

Combining, each fresh excursion crosses h with probability at least $\zeta \triangleq \zeta_0 \cdot \mathbb{P}(\xi_j) > 0$.

(d) Failed excursions have bounded expected length. By part (c), the number of excursions until crossing, $\widetilde{M}$, is stochastically dominated by a geometric random variable with success probability ζ, so $\mathbb{E}[\widetilde{M}] \leq 1/\zeta$.

For a failed excursion (one that resets to zero without crossing h), we need to bound the return time to zero, given that the excursion eventually returns. By the drift bound established in (3.92), from time $U_j + n_0$ onward the excursion behaves like a bounded random walk with positive drift. A standard result for such walks says that, conditional on eventual return to zero, the return time has an exponentially decaying tail; see, e.g., [38, 30]. Therefore, if τ_0 denotes the return time to zero in a failed excursion, then there exist constants $C_3, C_4 > 0$ such that

$$\mathbb{P}(\tau_0 > n \mid \tau_0 < \infty) \leq C_3 e^{-C_4 n}, \qquad n \geq 1,$$

and therefore $\mathbb{E}[\tau_0 \mid \tau_0 < \infty] \leq C_5 < \infty$ for some constant C_5.

(e) The successful excursion takes $O(h)$ steps. In the excursion that crosses h, the CUSUM accumulates from zero to at least h. By the argument in part (c), once $n \geq n_0$ the adaptive CUSUM has drift at least $\beta(\rho, \rho_*)/2 > 0$ with bounded increments. By Wald's identity applied to the random walk $\sum_{k=n_0+1}^{n} L_{U_{\widetilde{M}-1}+k}(\hat{\rho}_{U_{\widetilde{M}-1}+k}, \rho_*)$ stopped at level h, the expected crossing time satisfies

$$\begin{aligned}\mathbb{E}[T - U_{\widetilde{M}-1}] &\leq n_0 + \frac{h}{\beta(\rho, \rho_*)/2} + O(1) \\ &\leq C_6 h.\end{aligned}$$

for a constant $C_6 < \infty$ and all h large enough.

(f) Combining. Taking expectations in (3.89),

$$\begin{aligned}\mathbb{E}[T - \nu \mid T > \nu] &\leq \underbrace{O(h)}_{\text{ongoing}} + \underbrace{(1/\zeta) \cdot C_5}_{\text{failed}} + \underbrace{C_6\, h}_{\text{successful}} \\ &\leq C_T\, h\end{aligned}$$

for $h \geq h_0$ with $C_T = C_6 + 1$ (absorbing the constant C_5/ζ for large h). □

Step 4: Cumulative estimation error.

Lemma 3.14 (Cumulative squared estimation error)**.** *For any integer $d \geq 2$,*

$$\sup_{\boldsymbol{\tau}(\mathcal{A})} \mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}\left[\sum_{t=\nu+1}^{\nu+d}(\hat{\rho}_t - \rho)^2\right] \leq \sigma_Z^2(1+\log d) + C_{\mathrm{pre}},$$

where $C_{\mathrm{pre}} < \infty$ is a constant that accounts for pre-change observations entering the estimator.

Proof. Fix $t > \nu$. If $U(t) \geq \nu$, all observations entering $\hat{\rho}_t$ are post-change, hence i.i.d. with mean ρ and variance at most σ_Z^2. Conditionally on $U(t)$, $\bar{\rho}_t$ is an average of $t - U(t)$ such terms, so $\mathbb{E}[(\bar{\rho}_t - \rho)^2 \mid U(t)] \leq \sigma_Z^2/(t - U(t))$. Since the clipping operator is a projection onto a convex set containing ρ, it is nonexpansive, giving $\mathbb{E}[(\hat{\rho}_t - \rho)^2 \mid U(t)] \leq \sigma_Z^2/(t - U(t))$.

If $U(t) < \nu$, some pre-change observations enter $\hat{\rho}_t$. Since $\hat{\rho}_t, \rho \in [\rho_* + \epsilon, 1]$, we have $(\hat{\rho}_t - \rho)^2 \leq 1$ deterministically. This can only occur during the ongoing excursion (the one that spans the change point), which contributes at most $U_0 - \nu$ such terms, where U_0 is the first reset after ν.

Splitting the sum at U_0,

$$\sum_{t=\nu+1}^{\nu+d}(\hat{\rho}_t - \rho)^2 \leq (U_0 - \nu) + \sigma_Z^2 \sum_{t=U_0+1}^{\nu+d} \frac{1}{t - U(t)}.$$

Along any stretch of length d, the excursion counter $t - U(t)$ takes each value $1, 2, \ldots$ at most once per excursion, so $\sum_{t=U_0+1}^{\nu+d} \frac{1}{t-U(t)} \leq \sum_{n=1}^{d} \frac{1}{n} \leq 1 + \log d$. Taking expectations and noting $\mathbb{E}[U_0 - \nu \mid T > \nu] \leq C_{\mathrm{pre}} < \infty$ completes the proof. □

Step 5: Discrepancy due to estimation error is $o(h)$.

Lemma 3.15 (Mismatch bound)**.** *Under (R2) and the conclusions of Lemmas 3.13 and 3.14,*

$$\sup_{\nu \geq 0} \limsup_{m \to \infty} \frac{|x_2(h, \nu, m)|}{h} = 0 \qquad \text{as } h \to \infty.$$

Proof. By (R2), $|x_2|$ is bounded by $\bar{L}$ times the expected sum of $|\hat{\rho}_t - \rho|$. Applying the Cauchy–Schwarz inequality twice (first inside the expectation to separate the sum, then to the resulting product of expectations) gives

$$\begin{aligned}
|x_2(h,\nu,m)| &\le \bar{L}\,\mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}\left[\sum_{t=\nu+1}^{T_m} |\hat{\rho}_t - \rho| \;\middle|\; T > \nu\right] \\
&= \bar{L}\,\mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}\left[\sum_{t=\nu+1}^{T_m} 1 \cdot |\hat{\rho}_t - \rho| \;\middle|\; T > \nu\right] \\
&\le \bar{L}\,\mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}\left[\sqrt{T_m - \nu}\,\sqrt{\sum_{t=\nu+1}^{T_m} (\hat{\rho}_t - \rho)^2} \;\middle|\; T > \nu\right] \\
&\le \bar{L}\,\mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}[T_m - \nu \mid T > \nu]^{1/2} \cdot \mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}\left[\sum_{t=\nu+1}^{T_m} (\hat{\rho}_t - \rho)^2 \;\middle|\; T > \nu\right]^{1/2}.
\end{aligned}$$

The first factor is $O(\sqrt{h})$ uniformly in ν by (3.88) and $T_m \le T$. Since $\{T > \nu\} \in \mathcal{F}_\nu$, conditioning on $T > \nu$ only fixes the pre-ν history; the post-change observations after ν remain governed by the same law. Therefore, Lemma 3.14 applies under this conditional measure, and for the second factor it yields

$$\begin{aligned}
&\mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}\left[\sum_{t=\nu+1}^{T_m} (\hat{\rho}_t - \rho)^2 \;\middle|\; T > \nu\right] \\
&\quad\le \sigma_Z^2\big(1 + \mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}[\log(T_m - \nu) \mid T > \nu]\big) + C_{\mathrm{pre}}.
\end{aligned}$$

By Jensen's inequality and (3.88),

$$\begin{aligned}
\mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}[\log(T_m - \nu) \mid T > \nu] &\le \log \mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}[T_m - \nu \mid T > \nu] \\
&\le \log(C_T h),
\end{aligned}$$

so the second factor is $O(\sqrt{\log h})$. Combining,

$$\frac{|x_2(h,\nu,m)|}{h} \;\le\; C\sqrt{\frac{\log h}{h}} \;\xrightarrow[h\to\infty]{}\; 0,$$

uniformly in ν, and the bound is preserved as $m \to \infty$. □

Substituting the bounds from Lemmas 3.12 and 3.15 into the decomposition (3.87) and letting $m \to \infty$ gives, for every $\nu \geq 0$,

$$\begin{aligned}\mathbb{E}^{\mathcal{A}}_{\boldsymbol{\tau}(\mathcal{A})}[T - \nu \mid T > \nu] &= \frac{h + O(1) + o(h)}{\beta(\rho, \rho_*)} \\ &= \frac{h}{\beta(\rho, \rho_*)}\big(1 + o(1)\big).\end{aligned}$$

Taking the supremum over all change-point configurations $\boldsymbol{\tau}(\mathcal{A})$ establishes (3.37).

Chapter 4: Optimal Resource Allocation for ML Model Training and Deployment under Concept Drift

4.1 Introduction

The rapid advancement of machine learning (ML), particularly in generative models such as image generation and large language models (LLMs), is driving widespread adoption across sectors and transforming how we perform both professional and personal tasks [64, 52, 48]. As these models grow more sophisticated, their training and inference have scaled significantly, requiring vast data and substantial computational power, making their development increasingly expensive and resource-intensive. While the growing demand for advanced models and the necessary compute power has 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 [83, 115, 23]. However, despite the benefits of on-device inference, training remains predominantly cloud-dependent due to its intensive compute, storage, and coordination requirements that are beyond the reach of end-user devices.

In many real-world applications, the performance of ML models degrades over time due to shifts in the underlying data distributions. These shifts, collectively referred to as *concept drift*, can arise from changes in the relationship between input features and target variables, evolving class distributions, or the emergence of new, out-of-distribution data. For instance, a rental price prediction model might lose accuracy as market conditions evolve; a medical diagnosis model could be impacted by demographic shifts such as population aging; and a large language model (LLM) may provide incorrect responses to questions about recent events. In all cases, performance degradation stems from a growing mismatch between the training data and the data

encountered during deployment [34, 41, 68].

To sustain performance in the presence of concept drift, ML systems require continuous monitoring, periodic retraining, and redeployment – an operational challenge addressed by the field of MLOps [55, 96, 80]. These processes introduce substantial overhead, compounding the already high costs of model development and training. As ML applications proliferate across diverse domains, the demand for large, powerful ML models is expected to grow. Consequently, operational costs will rise alongside increasing model complexity and user demand, making efficient resource allocation within MLOps workflows increasingly critical for scalable and sustainable deployment.

We consider an ML system in which a service provider delivers trained models to multiple clients, who run inference on their devices. While client devices can support 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 the setting of sudden, discrete concept drift, and introduce 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 the quality of service experienced by clients depends directly on model performance, we further study drift-aware deployment strategies that aim to optimize long-term average performance across clients. The results in this chapter offer practical guidance for designing resilient, cost-efficient ML systems that adapt to evolving data.

4.1.1 Related Work

Scaling Laws and Resource-Aware Training

Extensive empirical studies have investigated the trade-offs among compute, model size, and data volume in ML training. A foundational study in this area [51]

demonstrated that the cross-entropy loss of autoregressive language models follows power-law relationships with model size, dataset size, and total compute. These scaling laws, observed across several orders of magnitude, provide simple prescriptions for allocating fixed compute budgets to achieve target performance levels. Building on this, [44] showed that language models are most compute-efficient when model parameters and training tokens are scaled proportionally. This insight led to the development of Chinchilla (70B parameters, 1.4T tokens), which outperforms much larger models under the same compute constraints. Complementing this work, another line of research focuses on characterizing learning curves to enable cost-effective training strategies [27, 103, 62]. For instance, [27] proposed a probabilistic model for extrapolating learning curves, allowing early termination of unpromising hyperparameter configurations. Similarly, [62] provided empirical evidence that budget-aware learning rate schedules can improve training efficiency under resource constraints. Inspired by such insights into learning dynamics, this chapter extends this body of research to account for deployment-time considerations in nonstationary settings. Specifically, we analyze the trade-offs between training resource allocation and model performance when deployment is subject to sudden concept drift, a setting that has received limited attention in prior scaling and resource-aware studies.

Concept Drift and Non-Stationary Learning

Concept drift poses a fundamental challenge to the assumption of stationary data distributions in model training. Seminal surveys classify drift into gradual (incremental) and sudden (abrupt) shifts in data distribution [34, 68]. Gradual drift, which leads to a slow and predictable decline in model performance, is often addressed through continuous learning techniques such as sliding-window retraining or instance reweighting [54]. Beyond heuristics, time-varying optimization offers a formal framework for understanding and mitigating gradual drift [75, 94]. For example, [94] introduces a two-stage online algorithm that tracks an evolving optimal solution over time, providing theoretical insights into the number of optimization steps – and,

by extension, training resources – required to maintain performance under drift.

In contrast, sudden concept drift, characterized by its unpredictable and often disruptive nature, often necessitates immediate model replacement or ensemble resets triggered by change-point detection methods [11, 15, 79, 70, 116, 87]. While several recent studies [70, 116, 87] have explored the trade-offs between model staleness and retraining costs, they primarily focus on drift detection and updating strategies tailored to specific models or domains. These approaches rarely account for how the statistical properties of drift events affect elastic compute allocation or deployment scheduling, and offer limited guidance for designing resource-aware policies that generalize across deployment scenarios or model classes.

MLOps

Finally, the problem considered in this chapter lies within the emerging realm of MLOps – the practice of reliably developing, deploying, and maintaining machine learning systems at scale [55, 80, 96]. While MLOps primarily aims to establish best practices for ML system operation, the increasing complexity and rising operational costs of modern ML pipelines [26] have spurred interest in cost-aware and compute-efficient retraining and deployment strategies [113, 69, 92, 104, 85].

By introducing a model-agnostic framework that jointly captures the effects of elastic resource allocation and sudden concept drift, this chapter provides foundational insights into how to sustain long-term model performance in resource-constrained settings. This perspective is essential for guiding the development of cost-effective and adaptive training principles that generalize across deployment scenarios.

4.1.2 Contributions

Motivated by the need to sustain performance in distributed ML systems operating under resource constraints and non-stationary data conditions, we develop a model-agnostic framework that captures the interplay between elastic resource allo-

cation 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 deployment policy. The proposed framework enables formal analysis of how these factors jointly influence model performance.

2. We show that optimal training resource allocation policies—those that maximize the long-term performance of server-side models—are of the bang-bang control type, i.e., they switch between extreme values. We prove that a single-switch front-loading policy is optimal if and only if concept durations follow a Decreasing Mean Residual Life (DMRL) distribution. We further demonstrate that intuitive allocation heuristics are sub-optimal under Increasing Mean Residual Life (IMRL) concept duration distributions.

3. We analyze the deployment scheduling problem under communication constraints and establish quasi-convexity of the performance 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 establish theoretical foundations and algorithmic principles for designing resource-aware training and deployment policies in large-scale, real-world ML systems.

4.1.3 Organization

The remainder of this chapter is organized as follows. Section 4.2 formalizes the system model, including concept drift dynamics and performance metrics. Section 4.3 analyzes optimal training resource allocation under budget constraints and

characterizes how different aging properties of concept durations affect the resulting policies. Section 4.4 studies deployment scheduling under communication constraints and develops near-optimal, drift-aware strategies. Section 4.5 concludes the chapter.

4.2 A Framework for Resource Allocation under Sudden Concept Drift

This section introduces a framework to model and analyze the impact of resource allocation strategies for model training and deployment on the performance of a distributed machine learning (ML) system under sudden concept drift. The system consists of a central server, referred to as the ML service provider, which possesses substantial computational resources and access to large-scale datasets, and multiple client devices with limited computational capabilities and data availability. Due to their limited capabilities, clients can only perform inference tasks using a locally deployed version of the model and cannot retrain or update the model on their own. In contrast, the server is both capable of and responsible for training and maintaining the ML model in the face of performance degradations caused by abrupt distributional changes, which affect both the server-side model and its deployed versions at the clients.

4.2.1 Model for Sudden Concept Drift

Gradual and sudden concept drifts are the two main categories of shifts in the underlying data distribution happening over time. Gradual concept drift unfolds over extended periods, causing a slow decline in model performance and is generally more predictable. In contrast, sudden concept drift results in abrupt changes to the data distribution, leading to immediate and significant performance degradation. Given their distinct characteristics, different strategies have been developed to address these two types of drift [41, 68]. Therefore, in this chapter, we specifically focus on modeling and managing the effects of sudden concept drift.

A model's performance is influenced by factors such as its complexity, the quality and quantity of training data, and the computational resources allocated to training. As training progresses and model parameters are iteratively updated, the model performance typically improves. Common metrics for evaluating model performance (e.g., accuracy, F1-score, loss value) computed on an evaluation dataset exhibit fluctuations during training. These fluctuations arise partly from the use of stochastic optimization algorithms (like SGD variants) that process data in mini-batches, and partly because any finite evaluation dataset is only an empirical sample of the true underlying data distribution. Consequently, a performance score calculated on an evaluation set can be viewed as a noisy realization conditioned on the model state and the current data distribution. Evaluating performance consistently under concept drift, where the underlying distribution itself changes, presents additional challenges, motivating a more robust performance measure. To address this, our framework utilizes an idealized metric termed *expected concept loss*, which represents the model's performance in expectation over the true data distribution associated with the current concept. We assume the expected concept loss decreases monotonically as the model improves during training. The rate and pattern of this decrease depend on factors like the model architecture, optimization algorithm, data characteristics, and allocated training resources.

Under sudden concept drift, the evolution of expected concept loss varies across different concepts due to abrupt distributional changes. Let T_i denote the time at which the i^{th} concept begins, and let Y_i be the random variable representing the duration of concept i, such that $T_{i+1} - T_i = Y_i$ and $\mathbb{E}[Y] < \infty$. We assume the evolution of expected concept loss in concept i is captured by a random function $G_i(\cdot) : \mathbb{R}^+ \to \mathbb{R}^+$, drawn from a countable set $\mathcal{G}$ of decreasing convex functions. Here, $G_i(0)$ represents the expected loss level immediately after concept i emerges at time T_i, and $G_i(t)$ models the decay of the expected concept loss over time under a 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).

4.2.2 Elastic Resource Allocation

As previously noted, the resources allocated to training influence the training speed and, consequently, the model's performance at any given time. In supervised learning, training involves iterative parameter updates over a fixed dataset, with more computational resources enabling faster iterations. In online learning, parameters are updated as new data becomes available, with training speed potentially constrained by the data arrival rate. In some applications, this rate can be increased—for example, by increasing a sensor's sampling frequency [47, 24] or acquiring more labeled data through crowdsourcing platforms like Amazon Mechanical Turk [71, 6]. Our framework accommodates both scenarios, referring to them collectively as training resource allocation.

We consider systems where training resources can be dynamically adjusted. Let $e_i(t) : [0, y_i) \to [0, M]$ represent the resource allocation during concept i, where M is the maximum permissible resource level, and y_i is a realization of Y_i. We assume that the effective training speed-up is linearly proportional to the allocated resources $e_i(t)$. We incorporate this dynamic in our framework by applying a horizontal scaling transformation to $G_i(\cdot)$ controlled by the the resource allocation function $e_i(\cdot)$.

The evolution of the expected concept loss at the server, $\mathcal{L}(t)$, considering both the sudden concept drift characterized by the stochastic process $(Y_i, G_i(\cdot))$ and the resource allocation function $e_i(\cdot)$, is then formalized as:

$$\mathcal{L}(t) = G_{I(t)}\left(\int_0^{t-T_{I(t)}} e_{I(t)}(\tau)d\tau\right), \tag{4.1}$$

where $I(t) = \max\{i \in \mathbb{N} : T_i \leq t\}$ identifies the index of the concept active at time t. Note that $\mathcal{L}(t)$ is random, since at time t, both concept $I(t)$ and expected concept loss curve $G_{I(t)}$ are random.

While $\mathcal{L}(t)$ captures the expected concept loss of the model currently being

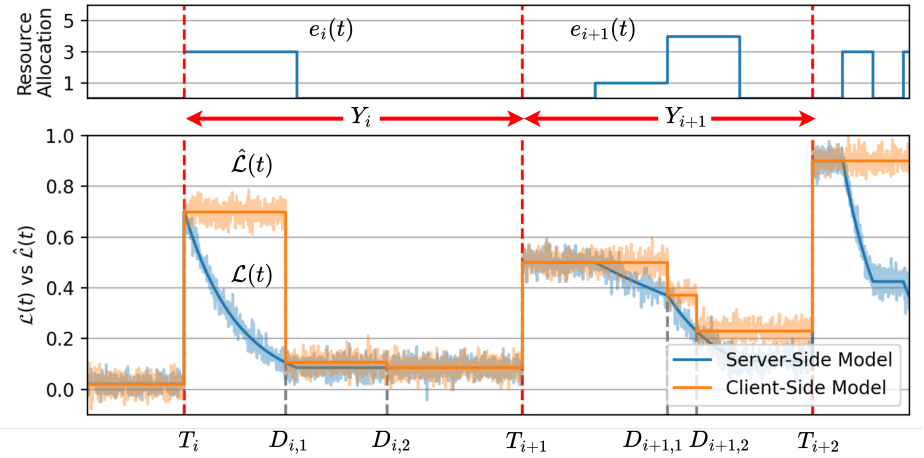


Figure 4.1: The upper figure shows the allocated training resources, denoted by $e_i(t)$. The lower figure depicts the change in a sample path of the expected concept loss of the server-side model, represented as $\mathcal{L}(t)$ (see (4.1)) and in a sample path of the client-side model, represented as $\hat{\mathcal{L}}(t)$ (see (4.2)). The fluctuations around these curves are sample paths for loss based on the evaluation dataset. Here, T_i indicates the time concept i arrives, and $D_{i,j}$ denotes the deployment instances.

trained or held at the server, client devices rely on previously deployed versions. Since deployment may not occur continuously, the model version on client devices often lags behind the server's version. Therefore, the expected concept loss experienced by clients, denoted $\hat{\mathcal{L}}(t)$, may differ from the expected concept loss at the server-side $\mathcal{L}(t)$.

We shall let $D_{i,j}$ denote the time of the j^{th} model deployment within concept i, with the first deployment done at the start time of the concept, $D_{i,0} = T_i$. The expected concept loss of the model deployed to clients at time t is determined by the state of the server model at the time of the most recent deployment before or at t. The expected concept loss of the deployed model at time t, $\hat{\mathcal{L}}(t)$, is then:

$$\hat{\mathcal{L}}(t) = G_{I(t)}\left(\int_0^{D_{I(t),J(t)}-T_{I(t)}} e_{I(t)}(\tau)d\tau\right), \tag{4.2}$$

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

Figure 4.1 illustrates the evolution of the expected concept loss for given training resource allocation. The upper plot shows the allocated resources, which influence the speed of training. In the lower plot, the expected concept loss of the server-side model, represented by the blue curve, exhibits a sudden increase when a new concept arrives. In concept i, the expected concept loss decreases at a constant rate specified by the resource allocation function $e_i(t)$. In concept $i+1$, the amount of allocated resources changes over time. Therefore, the rate expected concept loss decreases changes during concept duration. Furthermore, the expected concept loss of the deployed model, shown by the orange curve, aligns with the server-side model at the deployment instances. The fluctuations around the expected concept loss curves in the figure represent a realization of loss calculated on an evaluation dataset sampled from the data distribution at each specific time.

The above framework captures how training and deployment resource allocations influence both server-side and client-side performance under sudden concept drift. The key assumptions are summarized as follows:

- The expected concept loss remains constant during periods when no training resources are allocated.
- Upon a concept change, data from the new distribution becomes immediately available, allowing training to begin without delay if resources are allocated.
- Each concept persists for an i.i.d random duration Y_i with bounded expectation, $\mathbb{E}[Y] < \infty$.
- The expected loss decay for concept i is modeled by a decreasing convex function $g_i(\cdot) : \mathbb{R}^+ \to \mathbb{R}^+$, which is a realization of $G_i(\cdot)$.
- Training resources are continuously adjustable over time.
- Allocated resources $e_i(t) \in [0, M]$ scale the training speed linearly.

- At each concept change, the expected concept loss of both the deployed and server-side models resets to a common initial level.

Next, we investigate two key questions: (1) Training resource allocation: Given a constraint on total training resources, how should they be allocated over time to mitigate the effects of sudden concept drift and optimize server-side model performance? (Section 4.3) (2) Deployment scheduling: Under a constraint on deployment frequency, what constitutes the optimal deployment policy that maximizes client-side performance under concept drift? (Section 4.4)

The next section formalizes the training resource allocation problem and derives optimal allocation strategies under sudden concept drift.

4.3 Optimizing Resource Allocation for Training under Concept Drift

In this section, we investigate resource allocation policies for optimizing the time average performance of the server-side model under a time average budget constraint in the presence of sudden concept drift, as outlined in the framework introduced in Section 4.2. Since our focus here is on the server-side model, we omit the deployment scheduling problem, which pertains primarily to the client-side model and will be addressed separately.

4.3.1 Problem Formulation

Compute budget constraint on training resources. As previously noted, the model service provider can dynamically adjust 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.

Given the resource allocation functions $e_i(t) : [0, \infty) \to [0, M]$ over the dura-

tion of each concept i the time-average budget constraint is given by:

$$\limsup_{t\to\infty} \frac{\sigma_e}{t} \int_{\tau=0}^{t} e_{I(\tau)}(\tau - T_{I(\tau)})d\tau \leq B, \tag{4.3}$$

where $I(\tau)$ denotes the index of the current concept at time τ, and $T_{I(\tau)}$ is the time at which that concept begins.

Server-side time-average expected loss. Subject to the cost budget constraint (4.3), our objective is to determine a resource allocation policy that minimizes the time-average expected loss of server-side model, expressed as:

$$\limsup_{t\to\infty} \frac{1}{t} \int_{\tau=0}^{t} \mathcal{L}(\tau)d\tau = \limsup_{t\to\infty} \frac{1}{t} \int_{\tau=0}^{t} G_{I(\tau)} \left(\int_{u=0}^{\tau - T_{I(\tau)}} e_{I(\tau)}(u)du \right) d\tau \tag{4.4}$$

$$= \limsup_{t\to\infty} \frac{1}{t} \left[\sum_{i=1}^{I(t)-1} \int_{\tau=0}^{Y_i} G_i(\int_{0}^{\tau} e_i(u)du)d\tau \right], \tag{4.5}$$

where $G_i(\cdot)$ denotes the random expected concept loss function associated with concept i, and $Y_i = T_{i+1} - T_i$ is the duration of that concept. The equality follows by decomposing the time-average loss into a sequence of cycles corresponding to concept durations and their associated losses.

As evident from the objective function in (4.5), the optimal training resource allocation $e_i(t)$ may depend on both the expected loss function $g_i(\cdot)$ and the duration y_i of concept i. Determining such an optimal policy would generally require accurate knowledge of both $g_i(\cdot)$ and y_i, which is often infeasible in real-world settings. To develop a more practical and analytically tractable formulation, we consider stationary *static training resource allocation policies*, wherein the same allocation function is applied across all concept periods, independent of concept-specific characteristics $g_i(\cdot)$ and y_i.

Definition 4.1 (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 $i \in \mathbb{N}$*,*

$$e_i(t) = e(t), \quad t \in [0, \infty], \tag{4.6}$$

where $e(t)$ *is independent of both the concept duration* y_i *and the expected concept loss curve* $g_i(\cdot)$*.*

We assume a static training resource allocation policy $e(\cdot)$, and that the pairs $((Y_i, G_i(\cdot)) : i \in \mathbb{N})$, where each Y_i denotes the concept duration and $G_i(\cdot)$ the corresponding expected concept loss function, are independent and identically distributed. Under these assumptions, the limits in (4.5) and (4.3) exist. By the Renewal-reward Theorem [32, Section 3.4], the server-side time-average expected loss is given by

$$\limsup_{t\to\infty} \frac{1}{t} \int_{\tau=0}^{t} \mathcal{L}(\tau) d\tau = \frac{\mathbb{E}[\int_{\tau=0}^{Y_i} G_i(\int_0^\tau e(u)du)d\tau]}{\mathbb{E}[Y_i]} \tag{4.7}$$

$$= \frac{\mathbb{E}[\int_{\tau=0}^{Y_i} \mathbb{E}[G_i(\int_0^\tau e(u)du)|Y_i]d\tau]}{\mathbb{E}[Y_i]} \tag{4.8}$$

$$= \frac{\mathbb{E}[\int_{\tau=0}^{Y} \bar{g}(\int_0^\tau e(u)du)d\tau]}{\mathbb{E}[Y]}, \tag{4.9}$$

where the equality follows from the independence of G_i and Y_i[1]. We define $\bar{g}(\cdot) = \mathbb{E}[G_i(\cdot)|Y_i] = \mathbb{E}[G_i(\cdot)]$ as the expected loss function, representing the average expected concept loss across different concepts. Since each $g_i(\cdot)$ is assumed convex and decreasing, their expectation $\bar{g}(\cdot)$ inherits these properties[2]. The corresponding time-average budget constraint then becomes

$$\limsup_{t\to\infty} \frac{\sigma_e}{t} \int_{\tau=0}^{t} e_{I(\tau)}(\tau - T_{I(\tau)})d\tau = \sigma_e \frac{\mathbb{E}[\int_0^Y e(t)dt]}{\mathbb{E}[Y]} \leq B. \tag{4.10}$$

[1]Although the time-averages in (4.9) are derived under the assumption that G_i and Y_i are independent, the result extends to more general cases in which the limit exists with a continuous, convex, decreasing function $\bar{g} : [0, \infty) \to \mathbb{R}^+$, not necessarily equal to the mean $\mathbb{E}[G_i(\cdot)]$.

[2]The convexity assumption on individual expected concept loss functions $g_i(\cdot)$ can be relaxed, since our analysis only requires the convexity of the aggregated expected loss curve $\bar{g}(\cdot)$ under static training resource allocation.

To determine the optimal static resource allocation policy minimizing the time-average expected loss in (4.5) while satisfying the budget constraint in (4.3), we formulate an infinite-horizon continuous-time optimal control problem.

We assume the resource allocation function $e(t)$ belongs to the space of measurable and bounded functions on $[0, \infty)$ and lies within 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.

We formulate an infinite-horizon optimal control problem based on the objective function (4.5) and the budget constraint (4.3), both of which are rewritten using Fubini's Theorem. The problem is expressed as follows:

$$\min_{e(\cdot)} \quad J(e) = \int_0^\infty \bar{g}(x(t))\bar{F}_Y(t)dt \tag{4.11}$$

$$\text{s.t.} \quad x'(t) = e(t), \quad x(0) = 0, \quad 0 \leq e(t) \leq M, \quad \forall t \in [0, \infty) \tag{4.12}$$

$$C(e) = \int_0^\infty e(t)\bar{F}_Y(t)dt \leq B_1, \quad \text{where } B_1 = \frac{B \cdot E[Y]}{\sigma_e}. \tag{4.13}$$

Here, $\bar{F}_Y(t) = \mathbb{P}(Y > t)$ denotes the survival function (i.e., the complementary cumulative distribution function) of the concept duration Y. The optimization is taken over static resource allocation functions $e(t)$ that are measurable and bounded on $[0, \infty)$, lying 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 4.2 (Existence of an optimal solution for Problem (4.11))**.** *Let $\bar{g} : [0, \infty) \to \mathbb{R}^+$ be continuously differentiable, convex, and non-increasing. Then the functional optimization problem* (4.11) *is convex and admits at least one optimal solution. In particular, the functional $J(e)$ defined in* (4.11) *and the feasible set of resource allocations*

$$\mathcal{E}_C = \left\{ e \in \mathcal{E} \;\middle|\; \int_0^\infty e(t)\, \bar{F}_Y(t)\, dt \leq B_1 \right\}$$

are both convex over the domain $\mathcal{E} = \{e : e(\tau) \in [0, M],\ \forall \tau \geq 0\}$.

The proof of Lemma 4.2 follows from the linearity of the state function $x(t) = \int_0^t e(\tau)\, d\tau$, the convexity of $\bar{g}(x)$, and the non-negativity of the survival function $\bar{F}_Y(t)$. Complete proof details are provided in Appendix 4.A.1.

Lemma 4.2 establishes the existence of an optimal solution to Problem (4.11). Furthermore, by Arrow's Sufficiency Theorem [88, Theorem 3.1], it follows that the necessary conditions provided by Pontryagin's Maximum Principle (PMP) are also sufficient for global optimality in this setting.

4.3.2 Application of Pontryagin's Maximum Principle to Problem 4.11

To analyze the structure of the optimal solution to the problem in (4.11), we apply Pontryagin's Maximum Principle (PMP). We define the Hamiltonian function $H(t, x, e, p, \nu)$, which combines the integrand from the objective (4.11), a constant Lagrange multiplier $\nu \geq 0$ associated with the isoperimetric constraint (4.13), and the costate variable $p(t)$ corresponding to the state dynamics as

$$H(t, x, e, p, \nu) = \bar{g}(x(t))\bar{F}_Y(t) + (p(t) + \nu\bar{F}_Y(t))e(t). \tag{4.14}$$

The necessary and sufficient conditions (by Lemma 4.2) given by Pontryagin's Maximum Principle (PMP) for the infinite-horizon optimal control problem with free terminal state are as follows [88, Sec. 3.6]:

$$x'(t) = \frac{\partial H}{\partial p} = e(t), \quad x(0) = 0 \tag{4.15}$$

$$p'(t) = -\frac{\partial H}{\partial x} = -\bar{g}'(x(t))\bar{F}_Y(t) \tag{4.16}$$

$$e^*(t) = \underset{e\in\mathcal{E}}{\operatorname{argmin}}\, H(t, x, e, p, \nu) = \underset{e\in\mathcal{E}}{\operatorname{argmin}}(p(t) + \nu\bar{F}_Y(t))e(t) \tag{4.17}$$

$$\lim_{t\to\infty} p(t) = 0 \tag{4.18}$$

$$\nu \geq 0, \quad \nu\left(\int_0^\infty e^*(t)\bar{F}_Y(t)dt - B_1\right) = 0. \tag{4.19}$$

The transversality condition (4.18) follows from the boundedness of the resource allocation function, $e(t) \in [0, M]$, and the finiteness of the expected concept duration,

$\mathbb{E}[Y] < \infty$. Since the Hamiltonian is linear in the control variable $e(t)$, the optimal resource allocation policy must take the form of a *bang-bang* or singular control. We define the corresponding switching function as $\phi(t) \coloneqq p(t) + \nu\, \bar{F}_Y(t)$. From (4.17), the optimal resource allocation policy is given by

$$e^*(t) = \begin{cases} 0 & \text{if } \phi(t) > 0 \\ M & \text{if } \phi(t) < 0 \\ e_s(t) \in [0, M] & \text{if } \phi(t) = 0 \text{ over an interval (singular control).} \end{cases} \tag{4.20}$$

By the transversality condition (4.18) and integration of the costate equation (4.16), we obtain $p(t) = \int_t^\infty \bar{g}'(x(s))\bar{F}_Y(s)ds$. Substituting this expression into the switching function yields

$$\phi(t) = \int_t^\infty \bar{g}'(x(s))\bar{F}_Y(s)ds + \nu\bar{F}_Y(t). \tag{4.21}$$

Furthermore, using (4.16) and the identity $\bar{F}'_Y(t) = -h_Y(t)\bar{F}_Y(t)$, the derivative of the switching function becomes

$$\phi'(t) = p'(t) + \nu\bar{F}'_Y(t) = \bar{F}_Y(t)[-\bar{g}'(x(t)) - \nu h_Y(t)]. \tag{4.22}$$

These expressions form the basis for analyzing the monotonicity of the switching function. In many control problems with monotonic dynamics, the optimal policy often exhibits a single switching time, transitioning from one extreme control value to another. In our setting, this behavior arises under specific structural assumptions about the concept duration distribution Y, particularly when it exhibits monotonic aging characteristics such as Decreasing Mean Residual Life (DMRL) or Increasing Mean Residual Life (IMRL).

As we shall see, the characteristics of the concept duration distribution determine whether the optimal policy exhibits a single switching time. We discuss these characteristics below.

Definition 4.3 (Mean Residual Life (MRL))**.** *The* mean residual life *(MRL) function $m_Y(t)$ of a nonnegative random variable Y is defined as the expected remaining*

lifetime given survival up to time t:

$$m_Y(t) \coloneqq E[Y - t | Y > t] = \frac{\int_t^\infty \bar{F}_Y(u)du}{\bar{F}_Y(t)}, \quad \textit{for } \bar{F}_Y(t) > 0. \tag{4.23}$$

A well-known relationship (see [7]) links the derivative of the MRL function and the hazard rate $m'_Y(t) = h_Y(t)m_Y(t) - 1$, where $h_Y(t) \coloneqq \frac{f_Y(t)}{\bar{F}_Y(t)}$, for $\bar{F}_Y(t) > 0$. Accordingly, the random variable Y is said to have

Decreasing Mean Residual Life (DMRL) if $m_Y(t)$ is non-increasing, i.e.,
$h_Y(t)m_Y(t) \leq 1$.

Increasing Mean Residual Life (IMRL) if $m_Y(t)$ is non-decreasing, i.e.,
$h_Y(t)m_Y(t) \geq 1$.

Many real-world processes naturally exhibit these aging characteristics. DMRL behavior is commonly observed in systems that degrade over time, such as mechanical components, electronic devices, or biological organisms, where aging applies intuitively. In contrast, IMRL behavior characterizes processes in which survival implies increased robustness or reliability, such as the lifecycles of businesses or startups, social and political trends, and the adoption of new technologies or products. Accordingly, depending on the application, the aging properties of concept durations can often be assumed or empirically verified using sample-efficient statistical methods [8, 40, 53].

Next, we analyze the optimality of single-switch policies given the MRL properties of the concept duration distribution. A single-switch resource allocation policy can take one of two forms: (i) *Front-Loading* policy, where $e(t) = M$ for $t \in [0, t^*]$ and $e(t) = 0$ for $t > t^*$; or (ii) *Back-Loading* policy, where $e(t) = 0$ for $t \in [0, t^*]$ and $e(t) = M$ for $t > t^*$. Here, t^* denotes the switching time. For such policies to be optimal, the switch at time t^* must satisfy $\phi(t^*) = 0$, and the sign of the switching function should be opposite on the intervals before and after the switch, while remaining constant within each interval. The following theorems characterize the conditions

under which the front-loading or back-loading policy is optimal, depending on the aging characteristics (DMRL or IMRL) of the concept duration distribution.

Theorem 4.4 (Optimality of the Front-Loading Policy)**.** *For any convex, decreasing expected loss function $\bar{g} : [0, \infty) \to \mathbb{R}^+$ and any resource budget $B > 0$, the optimal resource allocation is a single-switch control of the form*

$$e^*(t) = \begin{cases} M & t < t^*_{DMRL} \\ 0 & t \geq t^*_{DMRL}, \end{cases} \tag{4.24}$$

if and only if the concept duration Y has a decreasing mean residual life (DMRL). The switching time t^*_{DMRL} is uniquely determined by the budget constraint $\int_0^{t^*_{DMRL}} \bar{F}_Y(y)dy = \frac{\mathbb{E}[Y]B}{M\sigma_e}$, provided that $B < M\sigma_e$, and $t^*_{DMRL} = \infty$, otherwise.

The proof of Theorem 4.4 relies on analyzing the behavior of the switching function $\phi(t)$ under DMRL concept duration distributions. By leveraging the convexity of the expected loss function $\bar{g}(\cdot)$ and the monotonic aging property of Y, we show that $\phi(t)$ crosses zero at most once, and that its sign pattern aligns with the optimality structure specified in (4.20). The complete proof is provided in Appendix 4.A.2.

Note that Theorem 4.4 states that if Y is DMRL, the optimal policy is front-loading, and conversely, if the optimal policy is front-loading for all budgets, then Y must be DMRL. While such policies may appear intuitive, this result formally rules out the optimality of commonly used heuristics such as fixed or periodic training schedules. Moreover, it implies that if Y is not DMRL, a front-loading policy need not be optimal for all budget levels. The next result, proved in Appendix 4.A.3, shows that when Y is IMRL, the optimal policy exhibits delayed (i.e., idling) resource allocation at the start of each concept duration.

Theorem 4.5 (Optimality of the Back-Loading Policy)**.** *For any convex, decreasing expected loss function $\bar{g} : [0, \infty) \to \mathbb{R}^+$, if the concept duration Y has an increasing*

mean residual life (IMRL) and the budget satisfies $B < M\sigma_e$, the optimal resource allocation exhibits an initial idling period before training begins,

$$e^*(t) = 0, \quad \textit{for } t < t^*_{IMRL}, \tag{4.25}$$

where $t^*_{IMRL} > 0$ denotes the time at which resource allocation starts.

Similar to Theorem 4.4, the proof of Theorem 4.5 leverages the optimality of the bang-bang control for Problem (4.11) and the monotonic aging property of concept duration distribution. A complete proof is provided in Appendix 4.A.3.

Although the IMRL case may represent an edge scenario, Theorem 4.5 characterizes the conditions under which the back-loading policy is optimal. The following corollary provides sufficient conditions for the optimality of such a resource allocation policy.

Corollary 4.6 (Optimality of the Back-Loading Policy)**.** *For any convex, decreasing expected loss function $\bar{g} : [0, \infty] \to \mathbb{R}^+$ with constant derivative $\bar{g}'(t) = -\beta$, and any concept duration Y with an increasing mean residual life (IMRL), the optimal resource allocation is a single-switch control given by*

$$e^*(t) = \begin{cases} 0 & t \le t^*_{IMRL} \\ M & t > t^*_{IMRL} \end{cases} \tag{4.26}$$

where the switching time t^*_{IMRL} is uniquely determined by the budget constraint

$$\int_{t^*_{IMRL}}^{\infty} \bar{F}_Y(y)dy = \frac{\mathbb{E}[Y]B}{M\sigma_e},$$

provided that $B < M\sigma_e$, and $t^*_{IMRL} = 0$ otherwise.

The proof of Corollary 4.6 follows directly from Theorem 4.5 and the fact that a linearly decreasing $\bar{g}(\cdot)$ ensures that the switching function crosses zero at most once. The complete proof is provided in Appendix 4.A.4.

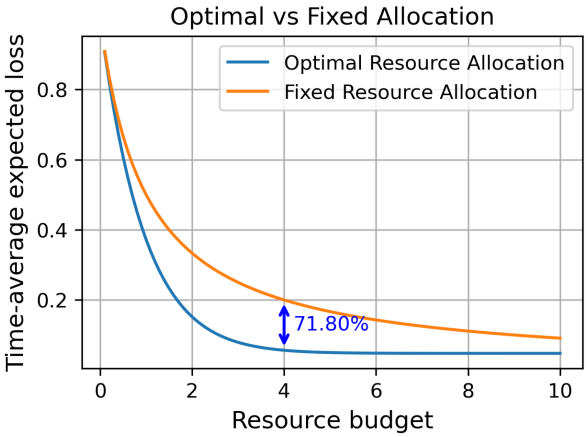
Notably, the strength of Theorems 4.4 and 4.5 lies in their generality: their conclusions do not depend on the specific functional form of $\bar{g}(\cdot)$, but only on its

monotonicity and convexity. These results also demonstrate that training policies that ignore the aging behavior of concept durations are provably suboptimal. The DMRL/IMRL distinction further clarifies how drift dynamics shape the structure of optimal resource allocation policies. Even without precise distributional information, our analysis indicates when front-loading (DMRL) or deferral (IMRL) strategies are most effective. As such, these results provide broadly applicable guidance for designing training resource allocation policies based on the statistical characteristics of training cycles driven by sudden concept changes.

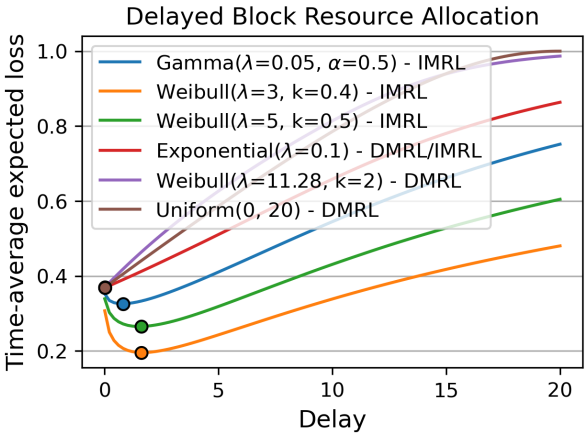
4.3.3 Simulation Results

In Figures 4.2a and 4.2b, we present simulation results illustrating the benefits of optimal training resource allocation and validating the relationship between the aging properties of concept durations and the structure of optimal allocation policies. In Figure 4.2a, we compare the time-average expected loss under two policies: the optimal resource allocation (specifically, the front-loading policy derived from Theorem 4.A.2) and a fixed resource allocation. In this experiment, concept durations follow an exponential distribution, corresponding to the DMRL case, and the expected loss function decays exponentially. The fixed policy assigns a constant resource level such that the budget constraint is exactly met, i.e., $e_{\text{fixed}}(t) = B/\sigma_e \ \forall t$. The results show 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. Additional simulation results using alternative expected loss functions are provided in Appendix 4.A.9.

In Figure 4.2b, we examine the effect of delayed block resource allocation for concept duration distributions with different aging properties. These experiments also use an exponentially decaying expected loss function, and the parameters are chosen such that the budget constraint is binding. Under a delayed block allocation policy, resources are withheld until a delay z, after which the full capacity is applied



(a) Comparison between constraint-binding optimal and fixed resource allocation policies. $Y \sim \text{Exp}(1)$, $\bar{g}(t) = e^{-t}$, $\sigma_e = 1$, $M = 20$. The optimal (front-loading) policy achieves up to 71.8% lower time-average expected loss.



(b) Constraint-binding delayed block allocation with varying delays under different concept duration distributions. $\bar{g}(t) = e^{-t}$, $B = 0.1$, $\sigma_e = 1$, $M = 20$. IMRL distributions (e.g., Gamma, Weibull with $k < 1$) benefit from delay, whereas DMRL distributions degrade.

Figure 4.2: Comparison of optimal, fixed and delayed resource allocation policies under sudden concept drift. All simulations were executed on a MacBook Air M3 with 16 GB of RAM.

until the budget is depleted. Formally, the allocation is defined as

$$e_{\text{block}}(t) = \begin{cases} M & z \le t < T(z) \\ 0 & \text{otherwise} \end{cases}, \quad \text{where } \int_z^{T(z)} \bar{F}_Y(t)dt = \frac{B\mathbb{E}[Y]}{M\sigma_e}. \tag{4.27}$$

Consistent with Theorem 4.4 and 4.5, the results confirm that concept duration distributions with IMRL benefit from delayed resource allocation up to an optimal delay, whereas for DMRL distributions, introducing delay consistently degrades performance. Overall, these findings demonstrate that the aging properties of training cycles driven by sudden concept drift have a significant impact on the structure of optimal training resource allocation strategies. Optimizing training operations based on the characteristics of concept drift yields substantial performance improvements, particularly under budget-constrained scenarios.

4.4 Deployment Optimization under Concept Drift

As discussed in Section 4.2, although improved versions of an ML model may exist on the server side, the performance experienced by clients depends on the version of the model that has been deployed to them. In this section, we study the deployment scheduling problem, aiming to optimize the long-term performance of client-side models subject to a constraint on the deployment rate, which may serve as a proxy for communication costs.

4.4.1 Problem Formulation

To understand the dynamics governing deployment decisions, we analyze deployment policies in a setting where the server-side model is trained under a fixed allocation of training resources. Following the approach in Section 4.3, we restrict our attention to static deployment schedulers, defined as follows:

Definition 4.7 (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 hence stationary and causal – if, for all concepts $i \in \mathbb{N}$,

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

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

Let $N_i = \max\{j : \delta_j < Y_i\}$ denote the random variable representing 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., $e_i(t) = 1, \ \forall t \geq 0$, the time-average expected loss experienced by clients under a static deployment scheduler is given by

$$\begin{aligned}
\limsup_{t\to\infty} \frac{1}{t} \int_{\tau=0}^{t} \hat{\mathcal{L}}(\tau) d\tau &= \limsup_{t\to\infty} \frac{1}{t} \int_{\tau=0}^{t} G_{I(t)} \left(D_{I(t),J(t)} - T_{I(t)} \right) && (4.28) \\
&= \limsup_{t\to\infty} \frac{1}{t} \Bigg[\sum_{i=1}^{I(t)-1} \sum_{j=0}^{N_i-1} G_i(\delta_j)(\delta_{j+1} - \delta_j) \\
&\quad + G_i(\delta_{N_i})(Y_i - \delta_{N_i}) \Bigg] && (4.29) \\
&= \frac{\mathbb{E}[\sum_{j=0}^{N_i-1} G_i(\delta_j)(\delta_{j+1} - \delta_j) + G_i(\delta_{N_i})(Y_i - \delta_{N_i})]}{\mathbb{E}[Y]} && (4.30) \\
&= \frac{\mathbb{E}[\sum_{j=0}^{N-1} \bar{g}(\delta_j)(\delta_{j+1} - \delta_j) + \bar{g}(\delta_N)(Y - \delta_N)]}{\mathbb{E}[Y]} && (4.31) \\
&= \frac{1}{\mathbb{E}[Y]} \sum_{j=0}^{\infty} \bar{g}(\delta_j) \int_{\delta_j}^{\delta_{j+1}} \bar{F}_Y(t) dt, && (4.32)
\end{aligned}$$

where $\bar{g}(\cdot) = \mathbb{E}[G_i(\cdot)]$ denotes the expected loss function, and N_i are i.i.d. random variables representing the number of deployments within each concept durations, with

$N \sim N_i$. Similarly, the time-average deployment rate constraint can be expressed as

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{i=1}^{I(t)} N_i = \frac{\mathbb{E}[N]}{\mathbb{E}[Y]} \tag{4.33}$$

$$= \frac{\mathbb{E}[\sum_{j=1}^{\infty} \mathbb{I}\{\delta_j < Y\}]}{\mathbb{E}[Y]} \tag{4.34}$$

$$= \frac{\sum_{j=1}^{\infty} P(\delta_j < Y)}{\mathbb{E}[Y]} \tag{4.35}$$

$$= \frac{\sum_{j=1}^{\infty} \bar{F}_Y(\delta_j)}{\mathbb{E}[Y]} \tag{4.36}$$

$$\leq r_D, \tag{4.37}$$

where r_D denotes the maximum allowable deployment rate. The optimization problem minimizing the time-average expected loss experienced by clients, subject to the time-average deployment rate constraint, is then given by

$$\min_{\{\Delta_j\}_{j=1}^{\infty}} \quad \sum_{j=0}^{\infty} \bar{g}(\delta_j) \int_{\delta_j}^{\delta_{j+1}} \bar{F}_Y(t)dt \tag{4.38}$$

$$\text{s.t.} \quad \Delta_j = \delta_j - \delta_{j-1} \geq 0, \ \forall j \in \mathbb{N}^+ \quad \delta_0 = 0, \tag{4.39}$$

$$\sum_{j=1}^{\infty} \bar{F}_Y(\delta_j) \leq B_2, \quad \text{where } B_2 = r_D \cdot E[Y]. \tag{4.40}$$

Here $\{\Delta_j\}_{j=1}^{\infty}$ denotes the set of inter-deployment durations. Note that the optimization problem (4.38) is not necessarily convex. However, under suitable conditions, it can be shown to be quasi-convex, as formalized in the following lemma.

Lemma 4.8 (Quasi-convexity of Problem (4.38))**.** *If $\bar{g}$ is a non-negative, convex, and decreasing function, and $\bar{F}_Y$ is strictly decreasing, then the objective function in* (4.38) *is quasi-convex in $\{\Delta_j \geq 0\}_{j\in\mathbb{N}}$. Moreover, if $\bar{F}_Y$ is also convex, then Problem* (4.38) *is quasi-convex.*

A detailed proof of Lemma 4.8 is provided in Appendix 4.A.5. As discussed there, the Karush–Kuhn– Tucker (KKT) conditions are sufficient for global optimality

when the feasible set defined by the deployment rate constraint (4.40) is convex. However, even under this condition, a numerical procedure is typically required to obtain the optimal deployment schedule.

Applying the KKT conditions, for a Lagrange multiplier ν ¿ 0, the optimal deployment scheduler $\{\delta_k^*\}_{k=0}^{\infty}$ satisfies the following necessary condition:

$$\bar{g}'(\delta_k^*)\int_{\delta_k^*}^{\delta_{k+1}^*}\bar{F}_Y(t)dt = \bar{F}_Y(\delta_k^*)[\bar{g}(\delta_k^*) - \bar{g}(\delta_{k-1}^*)] + \nu f_Y(\delta_k^*). \tag{4.41}$$

A numerical approach such as the bisection method can be employed to compute the optimal deployment times, typically by starting from the final deployment and proceeding backward, under an assumed upper bound on the total number of deployments. Motivated by the structure of this solution, we next propose a randomized deployment policy derived from a modified version of *Problem* (4.38), which remains applicable even when the concept duration distribution exhibits a non-convex survival function.

In this modified problem, the deployment rate constraint is replaced by a fixed, deterministic number of deployments, denoted by N_D. The resulting optimization problem is given by

$$\min_{\{\Delta_j\}_{j=1}^{N_D}} \quad \sum_{j=0}^{N_D-1}\bar{g}(\delta_j)\int_{\delta_j}^{\delta_{j+1}}\bar{F}_Y(t)dt + \bar{g}(\delta_{N_D})\int_{\delta_{N_D}}^{\infty}\bar{F}_Y(t)dt \tag{4.42}$$

$$\text{s.t.} \quad \Delta_j = \delta_j - \delta_{j-1} \geq 0,\ \forall j \in [1, N_D] \quad \delta_0 = 0. \tag{4.43}$$

Since the rate constraint has been removed, Lemma 4.8 guarantees that Problem (4.42) remains quasi-convex.

Theorem 4.9 (Optimal Solution for Problem (4.42))**.** *Let $\bar{g}$ be a non-negative, convex, and decreasing function, and let Y be a nonnegative 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*

$$\frac{\bar{g}(\delta_{k-1}^*) - \bar{g}(\delta_k^*)}{-\bar{g}'(\delta_k^*)} = \begin{cases} m_Y(d_{N_D}^*), & k = N_D \\ \frac{\int_{\delta_k^*}^{\delta_{k+1}^*}\bar{F}_Y(t)dt}{\bar{F}_Y(\delta_k^*)}, & k \in [1, N_D - 1], \end{cases} \tag{4.44}$$

where $m_Y(\cdot)$ denotes the mean residual life (MRL) function of the concept duration Y. Furthermore, the effective deployment rate achieved by the optimal scheduler, defined as $r_e(\{\delta_k^*\}_{k=0}^{N_D}) = \sum_{j=1}^{N_D} \bar{F}_Y(\delta_j)$, is monotonically increasing in the number of deployments N_D.

The proof of Theorem 4.9 follows directly from the quasi-convexity of Problem (4.42) and the monotonicity of the survival function $\bar{F}_Y(t)$. Full derivations are provided in Appendix 4.A.6.

In a special case where concept durations follow an exponential distribution and the expected loss decays exponentially, Theorem 4.9 admits a closed-form, chain-structured solution that can be determined iteratively.

Corollary 4.10 (Exponentially distributed concept durations and Exponential Decaying $\bar{g}$)**.** *Assume the expected loss function is given by $\bar{g}(t) = \alpha e^{-\beta t}$, with $\alpha, \beta > 0$, and that the concept durations Y are exponentially distributed with rate parameter λ, i.e., $\bar{F}_Y(t) = e^{-\lambda t}$. Then, the optimal inter-deployment durations $\Delta_k^* = \delta_k^* - \delta_{k-1}^*$ for the problem with a deterministic number of deployment N_D are given by*

$$\Delta_k^* = \begin{cases} \frac{1}{\beta}\ln(\frac{\beta}{\lambda}+1) & k = N_D, \\ \frac{1}{\beta}\ln(\frac{\beta}{\lambda}(1-e^{-\lambda\Delta_{k+1}^*})) & k \in [1, N_D - 1], \end{cases} \tag{4.45}$$

with $\delta_0 = 0$ and $k \in [1, N_D]$.

The derivation of Corollary 4.10 is provided in Appendix 4.A.7. As shown in Equation (4.45), the final inter-deployment duration is constant, while the preceding durations are determined recursively in a backward fashion, resulting in a chain-structured solution. This structure implies that as the number of deployments N_D increases, the optimal deployment times adjust to accommodate the additional deployments, while previously determined inter-deployment intervals remain unchanged.

4.4.2 Randomized Deployment Scheduler

We now introduce a randomized deployment scheduling policy to address the deployment optimization problem under the rate constraint (4.38). The proposed approach leverages the optimal deterministic schedulers obtained from Problem (4.42) for different deployment counts. The randomized policy operates over the convex hull of these optimal schedulers with consecutive numbers of deployments, assigning probabilities such that the resulting expected effective deployment rate exactly matches the allotted deployment rate limit.

Theorem 4.11 (Randomized Deployment Scheduler)**.** *Let $\{\delta_k^*\}_{k=0}^{\infty}$ denote the optimal deployment scheduler for Problem* (4.38)*, and let $\{\delta_k^*\}_{k=0}^{N_D}$ denote the optimal deployment scheduler for Problem* (4.42) *with a deterministic number of deployments N_D. Assume that the expected loss function $\bar{g}(\cdot)$ is non-negative, convex, and decreasing, and that the concept duration Y has survival function $\bar{F}_Y(t) > 0$ for all $t \in [0, \infty)$. Then, for any optimal scheduler $\{\delta_k^*\}_{k=0}^{\infty}$, there exists a deployment count $N_D^* \in \mathbb{N}^+$ and a scalar parameter $\gamma \in [0, 1]$ such that a convex combination of the effective deployment rates corresponding to the optimal schedulers with N_D^* and $N_D^* + 1$ deployments exactly matches the allotted deployment rate limit, i.e.,*

$$\gamma \cdot r_e(\{\delta_k^*\}_{k=0}^{N_D^*}) + (1-\gamma) \cdot r_e(\{\delta_k^*\}_{k=0}^{N_D^*+1}) = r_e(\{\delta_k^*\}_{k=0}^{\infty}) = r_D \mathbb{E}[Y], \tag{4.46}$$

where r_D denotes the allotted deployment rate limit.

A randomized deployment policy that, at the beginning of each concept, uses the scheduler $\{\delta_k^*\}_{k=0}^{N_D^*}$ with probability γ and $\{\delta_k^*\}_{k=0}^{N_D^*+1}$ with probability $1-\gamma$ achieves an expected effective deployment rate exactly equal to the deployment rate constraint.

A detailed proof of Theorem 4.11 is presented in Appendix 4.A.8.

The randomized deployment scheduler provides an approximate solution to the constrained deployment optimization problem (4.38), which is inherently non-convex. It guarantees full utilization of the allotted deployment budget and does not

rely on the convexity of the survival function of the concept duration distribution. Although the method requires knowledge of the expected loss curve and the probability distribution of concept duration, it offers a computationally efficient alternative to high-dimensional grid search or unstable non-convex optimization techniques. While the randomized policy assumes that both the expected loss curve and concept duration distribution can be estimated – an assumption that is feasible in many practical scenarios – an alternative approach is to use a parametric scheduler that emulates the key structural property of the optimal solution, namely, decreasing inter-deployment durations. In this context, the randomized policy serves as a computationally efficient baseline for testing and tuning such parametric algorithms.

4.4.3 Simulation Results

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 and computationally efficient choice for deployment under concept drift. Figure 4.3 presents simulation results highlighting the performance gains achieved by optimal deployment policies relative to a periodic deployment baseline and demonstrates the near-optimality of the randomized deployment policy across different concept duration distributions. In all experiments, the expected loss follows an exponential decay, and distribution parameters are chosen such that $\mathbb{E}[Y] = 1$. Except for the Exponential case, where a closed-form solution is available, the optimal deployment policies are computed numerically. For Exponential and Weibull($k = 2$) distributions, 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 under the same effective deployment rate. In these settings, the randomized policy, constructed as a convex combination of optimal solutions for adjacent deployment counts N_D, achieves performance nearly indistinguishable from the optimal policy. In contrast, for the Erlang-2 distribution, the performance gap between optimal and periodic policies narrows, owing both to the distribution's reduced variability (which limits the

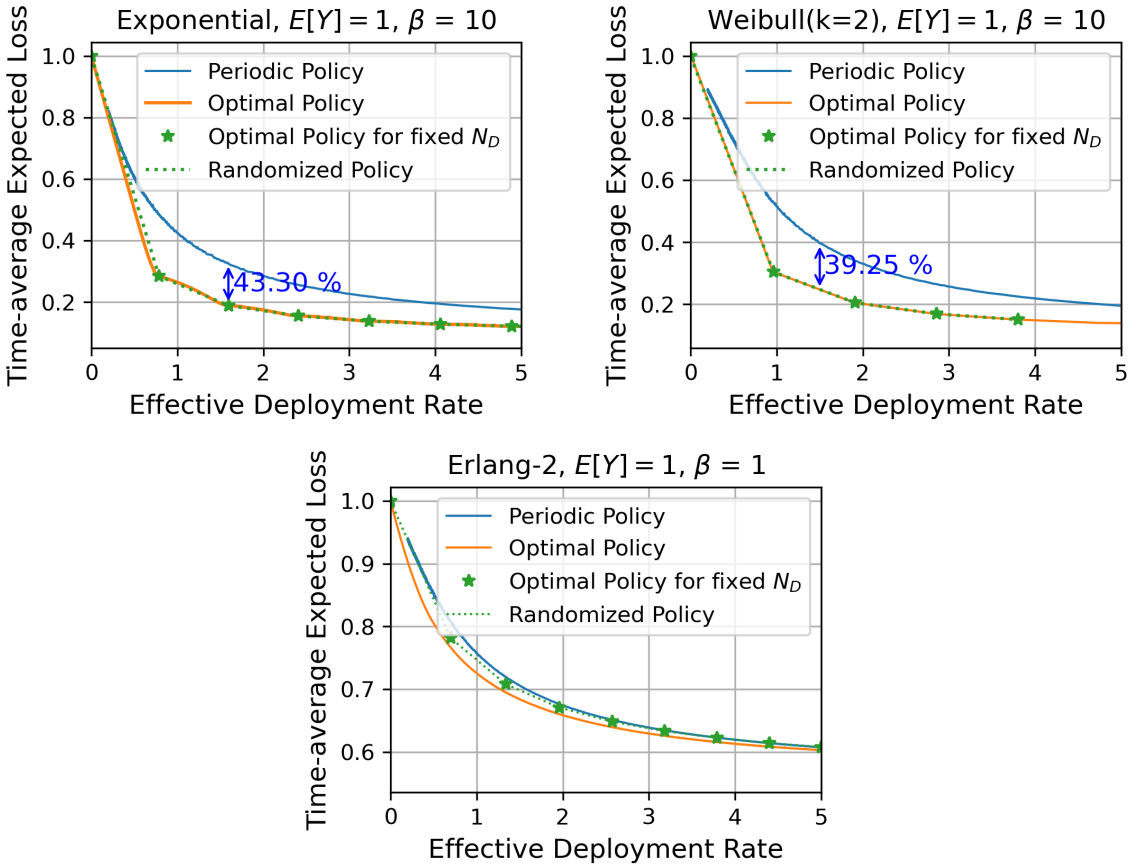


Figure 4.3: Comparison of periodic, optimal, and randomized deployment policies under different concept duration distributions (Exponential, Weibull($k = 2$), and Erlang-2), with $\mathbb{E}[Y] = 1$ and exponentially decaying expected loss $\bar{g}(t) = e^{-\beta t}$. Simulations were run on a MacBook Air M3 (16 GB RAM).

policy's leverage) and to the smaller decay rate β of the loss function, which slows the decline of expected penalties. Under these conditions, the randomized policy no longer matches the optimal performance, providing a concrete example where the optimal policy strictly outperforms its randomized counterpart.

4.5 Conclusion

We have presented a formal framework for studying training resource allocation and model deployment in distributed machine learning 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 must optimize when to allocate resources and when to deploy updated models. Our analysis shows that the structure of optimal policies depends fundamentally on the statistical properties of concept durations and loss functions. As machine learning systems become increasingly ubiquitous, the monitoring-retraining-redeployment cycle will represent a growing component of their operational costs. The results of this chapter lay the foundation for principled, cost-efficient training and deployment strategies that adapt to non-stationary environments.

4.A Technical Appendices

4.A.1 Proof of Lemma 4.2

Proof. Consider any $e_1, e_2 \in \mathcal{E}$ and any $\lambda \in [0,1]$. It holds that

$$J(\lambda e_1 + (1-\lambda)e_2) = \int_0^\infty \bar{g}\left(\int_0^t (\lambda e_1(\tau) + (1-\lambda)e_2(\tau))d\tau\right)\bar{F}_Y(t)dt \tag{4.47}$$

$$\leq \int_0^\infty \left(\lambda \bar{g}\left(\int_0^t e_1(\tau)d\tau\right)\right.$$

$$\left. + (1-\lambda)\bar{g}\left(\int_0^t e_2(\tau)d\tau\right)\right)\bar{F}_Y(t)dt \tag{4.48}$$

$$= \lambda J(e_1) + (1-\lambda)J(e_2), \tag{4.49}$$

where the inequality (4.48) follows from the fact that $\bar{g}(\cdot)$ is a convex function. Therefore, $J(e)$ is a convex functional on the domain $\mathcal{E}$. Similarly, the functional $C(e)$ is affine since

$$C(\lambda e_1 + (1-\lambda)e_2) = \int_0^\infty (\lambda e_1(t) + (1-\lambda)e_2(t))\,\bar{F}_Y(t)dt \tag{4.50}$$

$$= \lambda C(e_1) + (1-\lambda)C(e_2), \tag{4.51}$$

which completes the proof. □

4.A.2 Proof of Theorem 4.4

Proof. For the front-loading policy to be optimal, we need to show that the switching function crosses zero at most once from negative to positive. Suppose $0 < B < M\sigma_e$. Here, t^* is uniquely determined by the budget constraint (4.13), with $0 < t^* < \infty$. Since the budget constraint is binding, by the complementary slackness (4.19), we expect $\nu > 0$. Given the policy (4.24), the state and costate trajectories are

$$x^*(t) = \begin{cases} Mt & t < t^* \\ Mt^* & t \geq t^* \end{cases} \tag{4.52}$$

$$p^*(t) = \begin{cases} \int_t^{t^*} \bar{g}'(Ms)\bar{F}_Y(s)ds + \bar{g}'(Mt^*)m_Y(t^*)\bar{F}_Y(t^*) & t < t^* \\ \bar{g}'(Mt^*)m_Y(t)\bar{F}_Y(t) & t \geq t^* \end{cases} \tag{4.53}$$

At the switch point t^*, we require $\phi(t^*) = 0$, i.e.,

$$\phi(t^*) = p^*(t^*) + \nu \bar{F}_Y(t^*) = \bar{F}_Y(t^*)(\bar{g}'(Mt^*)m_Y(t^*) + \nu) = 0. \tag{4.54}$$

Since $\bar{F}_Y(t) > 0$ and $\bar{g}'(t) < 0$, for $0 < t < \infty$, we have $\nu > 0$ and

$$\nu = -\bar{g}'(Mt^*)m_Y(t^*), \tag{4.55}$$

which is consistent with (4.19).

Now, we verify (4.17) by showing $\phi(t) > 0$ for $t \in (t^*, \infty)$, and $\phi(t) < 0$ for $t \in [0, t^*)$. For $t \in (t^*, \infty)$, where $e^*(t) = 0$, the switching function becomes

$$\phi(t) = p^*(t) + \nu \bar{F}_Y(t) = \bar{g}'(Mt^*)m_Y(t)\bar{F}_Y(t) + (-\bar{g}'(Mt^*)m_Y(t^*))\bar{F}_Y(t) \tag{4.56}$$

$$= \bar{F}_Y(t)\bar{g}'(Mt^*)[m_Y(t) - m_Y(t^*)]. \tag{4.57}$$

Since $\bar{F}_Y(t) \geq 0$ and $\bar{g}'(Mt^*) < 0$, for $\phi(t) > 0$ we need $m_Y(t) < m_Y(t^*)$. Since $t > t^*$, the condition $m_Y(t) < m_Y(t^*)$ holds if $m_Y(t)$ is decreasing for $t > t^*$.

For $t \in [0, t^*)$, where $e^*(t) = M$, the first derivative of the switching function is

$$\begin{aligned} \phi'(t) &= \bar{F}_Y(t)[-\bar{g}'(x^*(t)) - \nu h_Y(t)] \\ &= \bar{F}_Y(t)[-\bar{g}'(Mt) + \bar{g}'(Mt^*)m_Y(t^*)h_Y(t)]. \end{aligned} \tag{4.58}$$

Since $\bar{g}'(Mt) < \bar{g}'(Mt^*)$ for $t < t^*$,

$$-\bar{g}'(Mt) + \bar{g}'(Mt^*)m_Y(t^*)h_Y(t) > -\bar{g}'(Mt^*)(1 - m_Y(t^*)h_Y(t)). \tag{4.59}$$

If $m_Y(t) > m_Y(t^*)$ for $t < t^*$, then

$$1 - m_Y(t^*)h_Y(t) > 1 - m_Y(t)h_Y(t) > 0,$$

which since $\bar{F}_Y(t) > 0$ means $\phi'(t) > 0$ for $t \in [0, t^*)$. Given that $\phi(t^*) = 0$, this implies $\phi(t) < 0$ for $t \in [0, t^*)$.

For the case $B \geq M\sigma_e$, since the budget constraint is not binding, by (4.19), $\nu = 0$. Since $e^*(t) = M$ for $t \in [0, \infty)$, the state function satisfies $x^*(t) = Mt$, the switching function $\phi(t) = p(t)$, and the derivative of the costate $p'(t) = -\bar{g}'(Mt)\bar{F}_Y(t)$. Since $\bar{g}'(t) < 0$ and $\bar{F}_Y(t) > 0$, $p'(t) > 0$ for $t \in [0, \infty)$. Since by the transversality condition (4.18) $\lim_{t\to\infty} p(t) = 0$ and $p(t)$ is increasing, we conclude $\phi(t) = p(t) < 0$. □

4.A.3 Proof of Theorem 4.5

Proof. Suppose there is an optimal resource allocation policy that idles until time t^*, i.e., $e^*(t) = 0,\ \forall t \in [0, t^*]$, and thus $x(t) = 0,\ \forall t \in [0, t^*]$.

At the switch point t^* we require $\phi(t^*) = 0$,

$$\phi(t^*) = p^*(t^*) + \nu\bar{F}_Y(t^*) = 0. \tag{4.60}$$

Since the resource budget is binding, i.e., $B < M\sigma_e$, by (4.60) and Mean Value Theorem there exists x_0 such that $0 \leq x_0 < \infty$, which yields

$$\nu = -\frac{p^*(t^*)}{\bar{F}_Y(t^*)} \tag{4.61}$$

$$= -\frac{\int_{t^*}^{\infty} \bar{g}'(x^*(s))\bar{F}_Y(s)ds}{\bar{F}_Y(t^*)} \tag{4.62}$$

$$= -\frac{\bar{g}'(x_0)\int_{t^*}^{\infty} \bar{F}_Y(s)ds}{\bar{F}_Y(t^*)} \tag{4.63}$$

$$= -\bar{g}'(x_0)m_Y(t^*) \tag{4.64}$$

$$> 0, \tag{4.65}$$

where (4.19) is satisfied since $\bar{g}'(t) < 0$.

By (4.17) and (4.20), for $t \in [0, t^*)$ we have $\phi(t) > 0$. Since $\phi(t^*) = 0$, the first derivative of the switching function for $t \in [0, t^*)$ should be positive, i.e.,

$$\begin{aligned}\phi'(t) &= \bar{F}_Y(t)[-\bar{g}'(x^*(t)) - \nu h_Y(t)] \\ &= \bar{F}_Y(t)[-\bar{g}'(0) + \bar{g}'(x_0)m_Y(t^*)h_Y(t)] > 0.\end{aligned} \tag{4.66}$$

Since $\bar{g}'(0) \leq \bar{g}'(x_0)$ for $0 \leq x_1 < \infty$,

$$-\bar{g}'(0) + \bar{g}'(x_0) m_Y(t^*) h_Y(t) \geq -\bar{g}'(x_0)(1 - m_Y(t^*) h_Y(t)). \tag{4.67}$$

If $m_Y(t) > m_Y(t^*)$ for $t < t^*$, then

$$1 - m_Y(t^*) h_Y(t) > 1 - m_Y(t) h_Y(t) > 0,$$

where the last inequality follows from the IMRL property. Thus, since $\bar{F}_Y(t) > 0$, if Y has IMRL, then $\phi'(t) > 0$ for $t \in [0, t^*)$. □

4.A.4 Proof of Corollary 4.6

Proof. For the back-loading policy to be optimal, we need to show that the switching function crosses zero at most once from positive to negative. Suppose $0 < B < M\sigma_e$. Here, t^* is uniquely determined by the budget constraint (4.13), with $0 < t^* < \infty$. Since the budget constraint is binding, by the complementary slackness (4.19), we expect $\nu > 0$. Given the policy (4.26), the state and costate trajectories are

$$x^*(t) = \begin{cases} 0 & t < t^* \\ M(t - t^*) & t \geq t^* \end{cases} \tag{4.68}$$

$$p^*(t) = \begin{cases} \int_{t^*}^{\infty} \bar{g}'(M(s - t^*)) \bar{F}_Y(s) ds & t < t^* \\ \int_{t}^{\infty} \bar{g}'(M(s - t^*)) \bar{F}_Y(s) ds & t \geq t^*. \end{cases} \tag{4.69}$$

At the switch point t^* we require $\phi(t^*) = 0$,

$$\phi(t^*) = p^*(t^*) + \nu \bar{F}_Y(t^*) = 0. \tag{4.70}$$

By (4.70) and Mean Value Theorem, there exists x_1 such that $0 \leq x_1 < \infty$ which yields

$$\nu = -\frac{p^*(t^*)}{\bar{F}_Y(t^*)} \tag{4.71}$$

$$= -\frac{\int_{t^*}^{\infty} \bar{g}'(M(s-t^*))\bar{F}_Y(s)ds}{\bar{F}_Y(t^*)} \tag{4.72}$$

$$= -\frac{\bar{g}'(x_1)\int_{t^*}^{\infty} \bar{F}_Y(s)ds}{\bar{F}_Y(t^*)} \tag{4.73}$$

$$= -\bar{g}'(x_1)m_Y(t^*) \tag{4.74}$$

$$> 0, \tag{4.75}$$

where (4.19) is satisfied since $\bar{g}'(t) < 0$.

Now, we verify (4.17) by showing $\phi(t) > 0$ for $t \in [0, t^*)$, and $\phi(t) < 0$ for $t \in (t^*, \infty)$. For $t \in [0, t^*)$, where $e^*(t) = 0$, the first derivative of the switching function is

$$\begin{aligned}\phi'(t) &= \bar{F}_Y(t)[-\bar{g}'(x^*(t)) - \nu h_Y(t)] \\ &= \bar{F}_Y(t)[-\bar{g}'(0) + \bar{g}'(x_1)m_Y(t^*)h_Y(t)].\end{aligned} \tag{4.76}$$

Since $\bar{g}'(0) \leq \bar{g}'(x_1)$ for $0 \leq x_1 < \infty$,

$$-\bar{g}'(0) + \bar{g}'(x_1)m_Y(t^*)h_Y(t) \geq -\bar{g}'(x_1)(1 - m_Y(t^*)h_Y(t)). \tag{4.77}$$

If $m_Y(t) > m_Y(t^*)$ for $t < t^*$, then

$$1 - m_Y(t^*)h_Y(t) > 1 - m_Y(t)h_Y(t) > 0,$$

which since $\bar{F}_Y(t) > 0$ means $\phi'(t) > 0$ for $t \in [0, t^*)$, and the last inequality follows from the IMRL property. Given that $\phi(t^*) = 0$, this implies $\phi(t) < 0$ for $t \in [0, t^*)$.

For $t \in (t^*, \infty)$, where $e^*(t) = M$, the switching function becomes

$$\phi(t) = \int_t^{\infty} \bar{g}'(M(s-t^*))\bar{F}_Y(s)ds - \frac{\int_{t^*}^{\infty} \bar{g}'(M(s-t^*))\bar{F}_Y(s)ds}{\bar{F}_Y(t^*)}\bar{F}_Y(t). \tag{4.78}$$

The condition $\phi(t) < 0$ for $t \in (t^*, \infty)$ is satisfied if the following inequality holds for $t \in (t^*, \infty)$:

$$\frac{\int_t^\infty \bar{g}'(M(s-t^*))\bar{F}_Y(s)ds}{\bar{F}_Y(t)} < \frac{\int_{t^*}^\infty \bar{g}'(M(s-t^*))\bar{F}_Y(s)ds}{\bar{F}_Y(t^*)}. \tag{4.79}$$

For the expected loss function with constant negative first derivative, i.e., $\bar{g}'(t) = -\beta,\ \beta > 0$, Equation (4.79) becomes

$$m_Y(t) > m_Y(t^*), \quad \forall t \in (t^*, \infty). \tag{4.80}$$

For the case $B \geq M\sigma_e$, since the budget constraint is not binding, by (4.19), $\nu = 0$. Since $e^*(t) = M$ for $t \in [0, \infty)$, the state function is $x^*(t) = Mt$, the switching function is $\phi(t) = p(t)$, and the derivative of the costate is $p'(t) = -\bar{g}'(Mt)\bar{F}_Y(t)$. Since $\bar{g}'(t) < 0$ and $\bar{F}_Y(t) > 0$, $p'(t) > 0$ for $t \in [0, \infty)$. Since due to the transversality condition (4.18) $\lim_{t\to\infty} p(t) = 0$ and $p(t)$ is increasing, $\phi(t) = p(t) < 0$. □

4.A.5 Proof of Lemma 4.8

Proof. Let us define a step function $s : [0, \infty) \to \mathbb{R}$ associated with the deployment offsets $\{\delta_j\}_{j=0}^K$ as

$$s(t) = \begin{cases} \delta_j & \text{if } t \in [\delta_j, \delta_{j+1}) \text{ for } j = 0, \ldots, K-1, \\ \delta_K & \text{if } t \in [\delta_K, \infty), \end{cases} \tag{4.81}$$

where $K = \sup\{k : k \in \mathbb{N}\}$, $\delta_{K+1} = \infty$ and $\delta_0 = 0$. Due to (4.81), the objective function (4.38) can be written as a single integral,

$$h(\{\delta_j\}_{j=1}^K) = \int_0^\infty \bar{g}(s(t))\bar{F}_Y(t)dt. \tag{4.82}$$

Let $\{\delta_j^{(1)}\}_{j=1}^K$ and $\{\delta_j^{(2)}\}_{j=1}^K$ be two different set of offsets, and $z_j = \lambda\delta_j^{(1)} + (1-\lambda)\delta_j^{(2)},\ \lambda \in [0,1]$ be the component-wise convex combination. Then $\{z_j\}_{j=1}^K$ also satisfies the ordering constraint and is feasible. Let $s^{(1)}(t),\ s^{(2)}(t),\ s^{(z)}(t)$ be the corresponding step functions.

Let $i^{(1)}$, $i^{(2)}$ be such that $t \in [\delta^{(1)}_{i^{(1)}}, \delta^{(1)}_{i^{(1)}+1})$ and $t \in [\delta^{(2)}_{i^{(2)}}, \delta^{(2)}_{i^{(2)}+1})$, respectively. Let $m = \min\{i^{(1)}, i^{(2)}\}$. Since $\delta^{(1)}_m, \delta^{(2)}_m \leq t$, their convex combination $z_m \leq t$. Hence, the left endpoint of the interval in which t falls under $\{z_j\}_{j=1}^K$ is at most t, implying that $s^{(z)}(t) \geq z_m \geq \min\{\delta^{(1)}_m, \delta^{(2)}_m\}$. Thus,

$$s^{(z)}(t) \geq \min\{s^{(1)}(t), s^{(2)}(t)\}. \tag{4.83}$$

Since $\bar{g}$ is non-increasing, $\bar{F}_Y(t) \geq 0, \ \forall t$, and by (4.83) it follows that for any fixed t

$$\bar{g}(s^{(z)}(t))F_Y(t) \leq \bar{g}(\min\{s^{(1)}(t), s^{(2)}(t)\})F_Y(t) \tag{4.84}$$

$$\leq \max\{\bar{g}(s^{(1)}(t))F_Y(t), \bar{g}(s^{(2)}(t))F_Y(t)\}. \tag{4.85}$$

Thus, the function $\{\delta_j\}_{j=0}^K \mapsto \bar{g}(s(t))F_Y(t)$ is quasi-convex in $\{\delta_j\}_{j=0}^K$ for any fixed $t \in [0, \infty)$.

Under the assumption that the offset sets $\{\delta^{(1)}_j\}_{j=1}^K$ and $\{\delta^{(2)}_j\}_{j=1}^K$ are component-wise greater or equal, i.e., $\delta^{(1)}_j \geq \delta^{(2)}_j, \ \forall j$, since the pointwise dominance of either of the offset preserved for all t, it follows that

$$\int_0^\infty \bar{g}(s^{(z)}(t))F_Y(t)dt \leq \max\Big\{\int_0^\infty \bar{g}(s^{(1)}(t))F_Y(t)dt, \int_0^\infty \bar{g}(s^{(2)}(t))F_Y(t)dt\Big\}. \tag{4.86}$$

□

4.A.6 Proof of Theorem 4.9

Proof. Given the objective function (4.42),

$$h(\{\delta_j\}_{j=1}^{N_D}) := \sum_{j=0}^{N_D-1} \bar{g}(\delta_j) \int_{\delta_j}^{\delta_{j+1}} \bar{F}_Y(t)dt + \bar{g}(\delta_{N_D}) \int_{\delta_{N_D}}^\infty \bar{F}_Y(t)dt. \tag{4.87}$$

By the first order optimality conditions, we have

$$\frac{\partial h}{\partial \delta_{N_D}}\bigg|_{\{\delta^*_j\}_{j=1}^{N_D}} = \bar{g}(\delta^*_{N_D-1})\bar{F}_Y(\delta^*_{N_D}) + \bar{g}'(\delta^*_{N_D})$$

$$\int_{\delta^*_{N_D}}^\infty \bar{F}_Y(t)dt - \bar{g}(\delta^*_{N_D})\bar{F}_Y(\delta^*_{N_D}) = 0, \tag{4.88}$$

and for $j = 1, \dots N_D - 1$ it holds that

$$\left.\frac{\partial h}{\partial \delta_j}\right|_{\{\delta_j^*\}_{j=1}^{N_D}} = \bar{g}(\delta_{j-1}^*)\bar{F}_Y(\delta_j^*) + \bar{g}'(\delta_j^*) \int_{\delta_{j-1}^*}^{\delta_j^*} \bar{F}_Y(t)dt - \bar{g}(\delta_j^*)\bar{F}_Y(\delta_j^*) = 0. \tag{4.89}$$

Rearranging these terms yields (4.44).

To prove that the effective rate is increasing in N_D, let $\delta_j^*(N_D)$ denote the j-th optimal deployment offset where N_D is the total number of deployments. By the convexity of $\bar{g}$, the monotonicity of $\bar{F}_Y$, and the KKT conditions, one can show that

$$\delta_j^*(N_D + 1) < \delta_j^*(N_D), \quad \text{for } j = 1, \cdots, N_D. \tag{4.90}$$

Since $\bar{F}_Y$ is decreasing, (4.90) implies that

$$\bar{F}_Y(\delta_j^*(N_D + 1)) > \bar{F}_Y(\delta_j^*(N_D)), \quad \text{for } j = 1, \cdots, N_D. \tag{4.91}$$

Therefore, the difference between $r_e(\{\delta_j^*\}_{j=0}^{N_D+1})$ and $r_e(\{\delta_j^*\}_{j=0}^{N_D})$ is positive, i.e.,

$$r_e(\{\delta_j^*\}_{j=0}^{N_D+1}) - r_e(\{\delta_j^*\}_{j=0}^{N_D}) = \sum_{j=1}^{N_D+1} \bar{F}_Y(\delta_j^*(N_D + 1)) - \sum_{j=1}^{N_D} \bar{F}_Y(\delta_j^*(N_D)) \tag{4.92}$$

$$= \bar{F}_Y(\delta_{N_D+1}^*(N_D + 1)) + \sum_{j=1}^{N_D} \bar{F}_Y(\delta_j^*(N_D + 1)) \tag{4.93}$$

$$- \bar{F}_Y(\delta_j^*(N_D)) > 0. \tag{4.94}$$

□

4.A.7 Proof of Corollary 4.10

Proof. For $\bar{g}(t) = \alpha e^{-\beta t}$ and $\bar{F}_Y(t) = e^{-\lambda t}$, by Theorem 4.9 it holds that

$$\frac{1}{\beta} e^{\beta \Delta_k^*}(1 - e^{-\beta \Delta_k^*}) = \begin{cases} 1 & k = N_D \\ \frac{1}{\lambda}(1 - e^{-\lambda \Delta_{k+1}^*}) & k = 1, \cdots, N_D - 1. \end{cases} \tag{4.95}$$

Rearranging the terms yields (4.45). □

4.A.8 Proof of Theorem 4.11

Proof. Under the randomized policy described in Theorem 4.11, the number of deployments within concept durations depends on whether the deployment schedule in concept i is $\pi_i = \{\delta_k^*\}_{k=0}^{N_D^*}$ or $\pi_i = \{\delta_k^*\}_{k=0}^{N_D+1^*}$. Therefore, the time-average deployment rate under randomized deployment policy can be written as

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{i=1}^{I(t)} N_i = \frac{\mathbb{E}[N\cdot\mathbb{I}\{\pi=\{\delta_k^*\}_{k=0}^{N_D^*}\}]+\mathbb{E}[N\cdot\mathbb{I}\{\pi=\{\delta_k^*\}_{k=0}^{N_D+1^*}\}]}{\mathbb{E}[Y]} \tag{4.96}$$

$$= \frac{1}{\mathbb{E}[Y]}\Bigg(\mathrm{P}(\pi=\{\delta_k^*\}_{k=0}^{N_D^*})\sum_{j=1}^{N_D^*}\bar{F}_Y(\delta_j) + \mathrm{P}(\pi=\{\delta_k^*\}_{k=0}^{N_D+1^*})\sum_{j=1}^{N_D+1^*}\bar{F}_Y(\delta_j)\Bigg) \tag{4.97}$$

$$= \frac{\gamma r_e(\{\delta_k^*\}_{k=0}^{N_D^*})+(1-\gamma)r_e(\{\delta_k^*\}_{k=0}^{N_D+1^*})}{\mathbb{E}[Y]}, \tag{4.98}$$

where (4.97) follows from the independence of the number of deployments and the fixed deployment policy selections in each concept.

As shown in Theorem 4.9, since 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 the number of deployments N_D within the range $[0,\infty)$, there exists $N_D^*\in\mathbb{N}$ and $\gamma\in[0,1]$ which ensure that the time-average deployment rate in (4.98) matches the deployment rate limit r_D. □

4.A.9 Additional Simulation Results with Alternative Expected Loss Curves

Note that our analysis of the optimal policies applies to any convex decreasing expected loss curve. However, the improvements over intuitive deployment heuristics depend on the probability distribution of concept duration, functional form of the expected loss function, and additional system parameters. In the figures below, we provide additional simulation results for the training resource allocation problem with different expected loss curves.

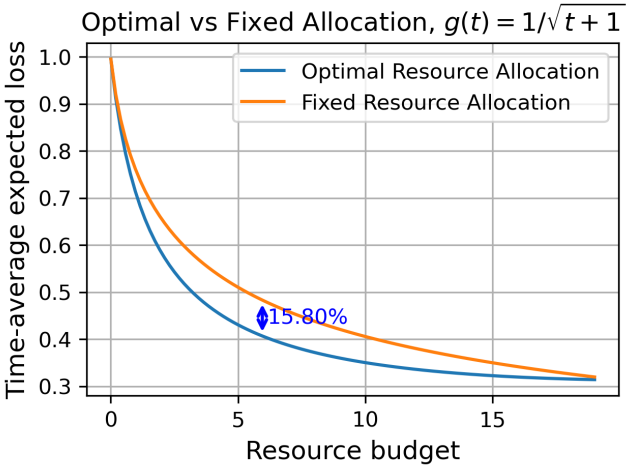


Figure 4.4: Comparison between constraint-binding optimal and fixed resource allocation policies. $Y \sim \text{Exp}(1)$, $\bar{g}(t) = 1/\sqrt{1+t}$, $\sigma_e = 1, M = 20$.

The simulation results demonstrate that the improvement over fixed resource allocation is greater when the expected loss curve decreases more rapidly in the initial phase, as front-loading policy effectively utilizes resources at the beginning of a concept.

We emphasize that the observed performance gains depend critically on the available resource budget. When the budget for model updates is very limited, improvements are naturally constrained, whereas wit h an abundant budget the different deployment policies converge to similar performance. The results shown in the figures correspond to intermediate budgets, where gains are most pronounced. These findings – both qualitative and quantitative – offer valuable guidance for selecting training and deployment strategies according to the provider's cost sensitivity and available computational resources.

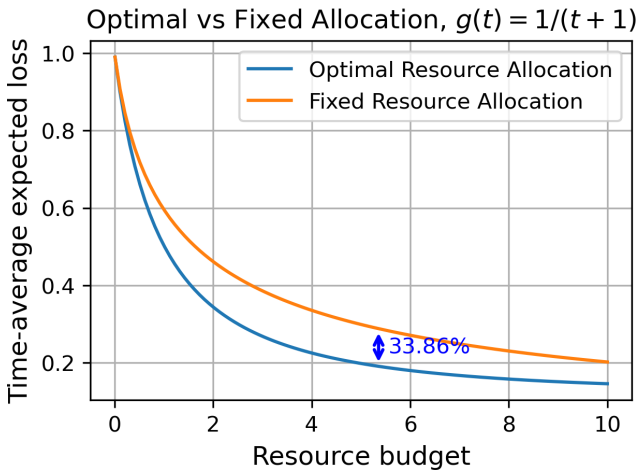


Figure 4.5: Comparison between constraint-binding optimal and fixed resource allocation policies. $Y \sim \text{Exp}(1)$, $\bar{g}(t) = 1/(1+t)$, $\sigma_e = 1, M = 20$.

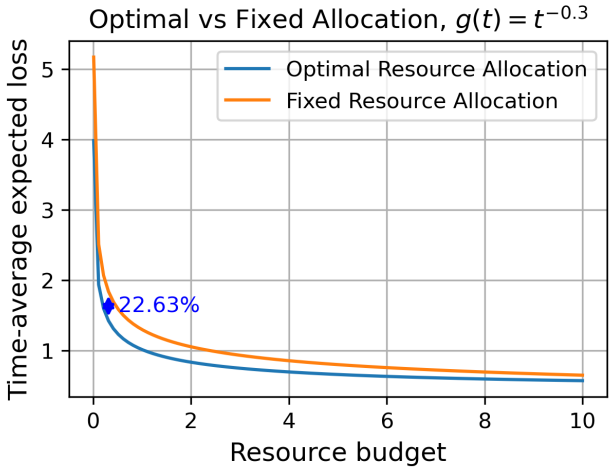


Figure 4.6: Comparison between constraint-binding optimal and fixed resource allocation policies. $Y \sim \text{Exp}(1)$, $\bar{g}(t) = t^{-0.3}$, $\sigma_e = 1, M = 20$.

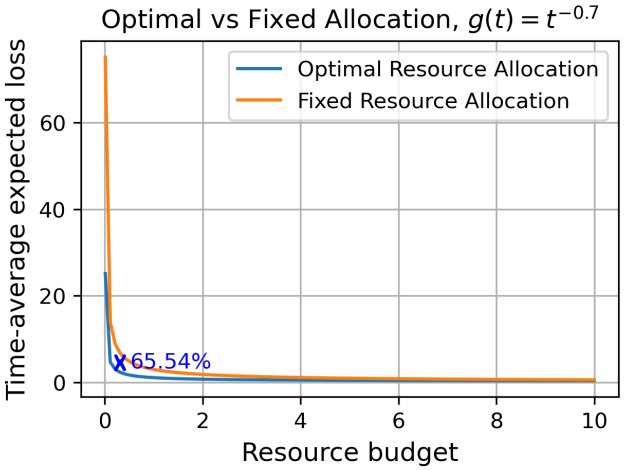


Figure 4.7: Comparison between constraint-binding optimal and fixed resource allocation policies. $Y \sim \text{Exp}(1)$, $\bar{g}(t) = t^{-0.7}$, $\sigma_e = 1, M = 20$.

Chapter 5: Conclusion

In this dissertation, we investigated resource allocation and decision-making problems that arise in the operation and maintenance of large-scale deployments of machine learning systems. A central focus was on settings in which client-side and server-side resources are coordinated to improve inference efficiency, monitor the performance of deployed models, and support model updating under resource constraints.

To provide a comprehensive treatment of the operational challenges that arise in such systems, this dissertation examined three closely related problem domains: congestion-aware inference task offloading, performance monitoring and concept drift detection, and resource allocation for model training and deployment. Together, these problems capture key stages in the lifecycle of distributed machine learning systems, including serving inference requests, detecting degradation in deployed models, and taking remedial actions through retraining and model updates.

Addressing this broad set of challenges required a diverse set of theoretical tools from online learning, queueing theory, sequential hypothesis testing, optimal control and renewal theory. These tools allowed us to develop formal problem formulations and algorithms with provable performance guarantees, complementing the heuristic-driven methods that are often used in practical system design.

We conclude this dissertation by highlighting the salient contributions of the preceding chapters and outlining several promising directions for future research.

5.1 Congestion-Aware Inference Task Offloading

In Chapter 2, we studied the problem of congestion-aware inference task offloading in hierarchical inference systems. To address this problem, we proposed a novel multi-armed bandit algorithm, which we call Lyapunov-EXP4 (Ly-EXP4). This

algorithm combines the EXP4 framework for adversarial contextual bandits with Lyapunov optimization techniques, allowing it to minimize expected regret while ensuring that a time-average constraint is satisfied. We then used Ly-EXP4 to guide offloading decisions in a hierarchical inference system where clients have heterogeneous task distributions and offloading costs. In this setting, the goal was to maximize system-level accuracy subject to a resource utilization constraint. Through this approach, which admits provable near-optimal performance guarantees and can be implemented in a distributed manner, we characterized the delay-accuracy-fairness trade-offs achievable in hierarchical inference systems.

While the hierarchical inference system considered in Chapter 2 consists of a single server, several natural extensions remain open for future work. One important direction is to consider systems with multiple servers that have heterogeneous communication and computation capacities and may be organized either in parallel or in a cascading manner. Such extensions would allow the framework to capture a broader class of practical deployment scenarios. Another promising direction is the design of client clustering schemes that can better balance system performance and fairness by grouping clients according to their task distributions, resource requirements, or offloading costs. More broadly, the Lyapunov optimization perspective developed in this chapter provides a principled way to incorporate long-term constraints into online learning algorithms. This suggests that similar ideas may be useful beyond contextual bandits, including in reinforcement learning and other constrained online decision-making problems.

5.2 Performance Monitoring in Large-Scale Deployments

Performance monitoring for concept drift detection is one of the core mechanisms needed for maintaining machine learning systems in production environments. Designing such mechanisms becomes particularly challenging in large-scale, distributed deployments where clients rely on on-device inference and the central server has lim-

ited access to the clients' data and feedback. In such settings, the data collection and detection mechanisms must be designed jointly to meet the detection objective in a sample-efficient and timely manner. Traditional change point detection problems often focus on detecting the earliest change in a single data stream or across multiple streams. Chapter 3 instead studied an alternative formulation with greater operational relevance for large-scale deployments, that is detecting the times when a significant fraction of the clients experience significant performance degradations. To address this problem, we proposed Greedy Partial Sum CUSUM and Adaptive Global CUSUM, which belong to two complementary classes of algorithms with different strengths and limitations, namely individual-statistics based and global-statistics based detectors. Through the analysis of detection delay and false alarm characteristics, we identified the regimes in which each algorithm is most effective, including how their behavior depends on the population size, as well as the spread and magnitude of the performance degradations across clients.

In Chapter 3, we considered the problem of change point detection in a population under limited feedback, with a fairness aspect naturally induced by the detection objective. Even this relatively simple formulation makes clear that the policies governing where to collect information and how to aggregate it play a nontrivial role in detection performance and hints at a connection between resource allocation and change point detection problems. A unified treatment of fairness in change point detection with adaptive sampling-driven policies therefore represents a promising direction for future work.

5.3 Resource Allocation for Model Training and Deployment

As machine learning systems become increasingly widespread, and models grow more complex and data-hungry, periodic maintenance operations like retraining and redeployment in response to performance degradation represent a growing share of operational cost. Consequently, deciding when to train, how much to train, and

when to deploy updated models is central to the cost-efficient operation of ML systems. Despite their practical importance, however, the design and optimization of these processes are often treated through heuristic-based approaches, both in practice and in the existing literature.

In Chapter 4, we proposed a formal framework for studying training resource allocation and deployment scheduling under resource constraints to analyze how the characteristics of concept drift processes and the structure of loss functions affect the form of optimal policies. In particular, we showed that the aging properties of concept drift processes, especially their mean residual life, play a fundamental role in determining the structure of optimal training resource allocation. The main insight from this analysis is that the optimal training resource allocation policy does not depend directly on the evolution of the loss function, and it is not always optimal to train the model immediately after drift is detected. This is especially true when the concept drift process has increasing mean residual life, since in such cases delaying training may save resources that would otherwise be spent on concepts that are likely to be short-lived.

In contrast, optimal model deployment policies do depend on both the evolution of the loss function and the distribution of concept durations. Our analysis showed that optimal deployment schedules typically have increasing inter-deployment times, thereby prioritizing early deployments when they are most beneficial. While Chapter 4 treated training resource allocation and deployment scheduling separately, the joint optimization of these two processes represents a promising direction for future work.

Taken together, this dissertation has studied several core processes involved in the operation and maintenance of large-scale deployments of ML systems. By developing analytical formulations and algorithms with provable guarantees, it provides theoretical foundations for improving the efficiency and scalability of inference serving, performance monitoring, training, and deployment under resource constraints.

Acknowledgement of Support

This material is based upon work supported in part by the National Science Foundation under Grant CNS-2212202 and NSF Award 2148224, DOD Award W911NF-24-2-0185, funds from OUSD R&E, NIST, and industry partners as specified in the Resilient & Intelligent NextG Systems (RINGS) program, a Samsung Fellowship Award, and WNCG/6G@UT.

References

[1] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming Throughput-Latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 117–134, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL `https://www.usenix.org/conference/osdi24/presentation/agrawal`.

[2] Shipra Agrawal and Nikhil R. Devanur. Bandits with concave rewards and convex knapsacks. In *Proceedings of the fifteenth ACM conference on Economics and computation*, EC '14, pages 989–1006, New York, NY, USA, June 2014. Association for Computing Machinery. ISBN 978-1-4503-2565-3.

[3] Ghina Al-Atat, Andrea Fresa, Adarsh Prasad Behera, Vishnu Narayanan Moothedath, James Gross, and Jaya Prakash Champati. The case for hierarchical deep learning inference at the network edge. In *Proceedings of the 1st International Workshop on Networked AI Systems*, NetAISys '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702129. doi: 10.1145/3597062.3597278. URL `https://doi.org/10.1145/3597062.3597278`.

[4] Ahmet Gunhan Aydin and Haris Vikalo. Viewport prediction via adaptive edge offloading. *IEEE Networking Letters*, 7(1):21–25, 2025. doi: 10.1109/LNET.2024.3480149.

[5] Ashwinkumar Badanidiyuru, Robert D. Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 207–216, 2013.

[6] Jihwan Bang, Hyunseo Koh, Seulki Park, Hwanjun Song, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on a contaminated data stream with blurry task boundaries. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9275–9284, 2022.

[7] Richard E. Barlow and Frank Proschan. Statistical theory of reliability and life testing: Probability models. In *Statistical Theory of Reliability and Life Testing: Probability Models*, International Series in Decision Processes. Holt, Rinehart and Winston, New York, 1974/1975. ISBN 0-03-085853-4.

[8] Bo Bergman and Bengt Klefsjö. A family of test statistics for detecting monotone mean residual life. *Journal of Statistical Planning and Inference*, 21(2): 161–178, February 1989. ISSN 0378-3758. doi: 10.1016/0378-3758(89)90002-5.

[9] Hasan Burhan Beytur, Ahmet Gunhan Aydin, Gustavo de Veciana, and Haris Vikalo. Optimization of offloading policies for accuracy-delay tradeoffs in hierarchical inference. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, pages 1989–1998, 2024. doi: 10.1109/INFOCOM52122.2024.10621325.

[10] Hasan Burhan Beytur, Gustavo de Veciana, Haris Vikalo, and Kevin S Chan. Optimal resource allocation for ml model training and deployment under concept drift, 2025. URL `https://arxiv.org/abs/2512.12816`.

[11] Albert Bifet and Ricard Gavaldà. Learning from Time-Changing Data with Adaptive Windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)*, Proceedings, pages 443–448. Society for Industrial and Applied Mathematics, April 2007. ISBN 978-0-89871-630-6. doi: 10.1137/1.9781611972771.42.

[12] Ilai Bistritz, Zhengyuan Zhou, Xi Chen, Nicholas Bambos, and Jose Blanchet. Online exp3 learning in adversarial bandits with delayed feedback. *Advances in Neural Information Processing Systems*, 32, 2019.

[13] George Box and José Ramírez. Cumulative score charts. *Quality and Reliability Engineering International*, 8(1):17–27, 1992. doi: https://doi.org/10.1002/qre.4680080105. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.4680080105.

[14] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 33: 1877–1901, 2020.

[15] M. Budka and B. Gabrys. Change-point detection in evolving data streams using ensembles of classifiers. *IEEE Signal Processing Letters*, 25(9):1353–1357, 2018. doi: 10.1109/LSP.2018.2849385.

[16] Fernando E. Casado, Dylan Lema, Marcos F. Criado, Roberto Iglesias, Carlos V. Regueiro, and Senén Barro. Concept drift detection and adaptation for federated and continual learning. *Multimedia Tools and Applications*, 81(3): 3397–3419, January 2022. ISSN 1573-7721. doi: 10.1007/s11042-021-11219-x.

[17] Semih Cayci, Swati Gupta, and Atilla Eryilmaz. Group-fair online allocation in continuous time. *Advances in Neural Information Processing Systems*, 33: 13750–13761, 2020.

[18] Semih Cayci, Yilin Zheng, and Atilla Eryilmaz. A lyapunov-based methodology for constrained optimization with bandit feedback. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4):3716–3723, Jun. 2022. doi: 10.1609/aaai.v36i4.20285. URL https://ojs.aaai.org/index.php/AAAI/article/view/20285.

[19] Ayan Chakrabarti, Roch Guérin, Chenyang Lu, and Jiangnan Liu. Real-time edge classification: Optimal offloading under token bucket constraints. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 41–54, December 2021. doi: 10.1145/3453142.3492329.

[20] Hock Peng Chan. Optimal sequential detection in multi-stream data. *The Annals of Statistics*, 45(6), December 2017. ISSN 0090-5364. doi: 10.1214/17-AOS1546.

[21] Kit Yan Chan, Bilal Abu-Salih, Raneem Qaddoura, Ala' M. Al-Zoubi, Vasile Palade, Duc-Son Pham, Javier Del Ser, and Khan Muhammad. Deep neural networks in the cloud: Review, applications, challenges and research directions. *Neurocomputing*, 545:126327, 2023. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.126327.

[22] Anamitra Chaudhuri, Georgios Fellouris, and Ali Tajer. Round Robin Active Sequential Change Detection for Dependent Multi-Channel Data, March 2024.

[23] D. Chen, S. Yang, and J. Li. What role do small models play in a world of giants? In *International Conference on Learning Representations*, 2024.

[24] Weihao Cheng, Sarah Erfani, Rui Zhang, and Ramamohanarao Kotagiri. Learning datum-wise sampling frequency for energy-efficient human activity recognition. 32(1), 2018.

[25] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL `https://proceedings.mlr.press/v15/coates11a.html`.

[26] Ben Cottier, Robi Rahman, Loredana Fattorini, Nestor Maslej, Tamay Besiroglu, and David Owen. The rising costs of training frontier ai models, 2025. URL https://arxiv.org/abs/2405.21015.

[27] T. Domhan, J. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3460–3468, 2015.

[28] Georgios Fellouris and Grigory Sokolov. Second-Order Asymptotic Optimality in Multisensor Sequential Change Detection. *IEEE Transactions on Information Theory*, 62(6):3662–3675, June 2016. ISSN 1557-9654. doi: 10.1109/TIT.2016.2549042.

[29] Jean Feng, Alexej Gossmann, Gene A Pennello, Nicholas Petrick, Berkman Sahiner, and Romain Pirracchio. Monitoring machine learning-based risk prediction algorithms in the presence of performativity. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 919–927. PMLR, 02–04 May 2024. URL https://proceedings.mlr.press/v238/feng24b.html.

[30] Sergey Foss and Timofei Prasolov. Moments of the first descending epoch for a random walk with negative drift. *Statistics & Probability Letters*, 189:109547, 2022. doi: 10.1016/j.spl.2022.109547.

[31] Andrea Fresa and Jaya Prakash Varma Champati. An offloading algorithm for maximizing inference accuracy on edge device in an edge intelligence system. In *Proceedings of the 25th International ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 15–23, New York, NY, USA, October 2022. ISBN 978-1-4503-9482-6. doi: 10.1145/3551659.3559044.

[32] Robert G. Gallager. *Discrete Stochastic Processes*. Springer US, Boston, MA, 1996. ISBN 978-1-4613-5986-9 978-1-4615-2329-1. doi: 10.1007/978-1-4615-2329-1.

[33] Robert G. Gallager. Discrete stochastic processes, 2011. Draft of 2nd edition, January 31, 2011.

[34] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44:1–44:37, March 2014. doi: 10.1145/2523813.

[35] Salvatore Greco, Bartolomeo Vacchetti, Daniele Apiletti, and Tania Cerquitelli. Unsupervised Concept Drift Detection from Deep Learning Representations in Real-time. *IEEE Transactions on Knowledge and Data Engineering*, 37(10): 6232–6245, October 2025. ISSN 1041-4347, 1558-2191, 2326-3865. doi: 10.1109/TKDE.2025.3593123.

[36] Kun Guo, Ruifeng Gao, Wenchao Xia, and Tony Q. S. Quek. Online learning based computation offloading in mec systems with communication and computation dynamics. *IEEE Transactions on Communications*, 69(2):1147–1162, February 2021. ISSN 1558-0857. doi: 10.1109/TCOMM.2020.3038875.

[37] Ragini Gupta, Shinan Liu, Ruixiao Zhang, Xinyue Hu, Xiaoyang Wang, Hadjer Benkraouda, Pranav Kommaraju, Phuong Cao, Nick Feamster, and Klara Nahrstedt. Generative active adaptation for drifting and imbalanced network intrusion detection, 2025. URL `https://arxiv.org/abs/2503.03022`.

[38] Allan Gut. *Stopped Random Walks: Limit Theorems and Applications*. Springer, 2009. doi: 10.1007/978-0-387-87835-5.

[39] Bruce Hajek. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability*, 14(3):502–525, 1982.

[40] W. J. Hall and Jon A. Wellner. Estimation of mean residual life. In Anthony Almudevar, David Oakes, and Jack Hall, editors, *Statistical Modeling for Biological Systems: In Memory of Andrei Yakovlev*, pages 169–189. Springer International Publishing, Cham, 2020. ISBN 978-3-030-34675-1. doi: 10.1007/978-3-030-34675-1_10.

[41] Meng Han, Zhiqiang Chen, Muhang Li, Hongxin Wu, and Xilong Zhang. A survey of active and passive concept drift handling methods. *IEEE Transactions on Knowledge and Data Engineering*, 38(4):1492–1535, 2022. ISSN 1467-8640. doi: 10.1111/coin.12520. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/coin.12520`.

[42] Soroush Heydari and Qusay H. Mahmoud. Tiny machine learning and on-device inference: A survey of applications, challenges, and future directions. *Sensors*, 25(10), 2025. ISSN 1424-8220. doi: 10.3390/s25103191. URL `https://www.mdpi.com/1424-8220/25/10/3191`.

[43] Fabian Hinder, Valerie Vaquet, and Barbara Hammer. One or two things we know about concept drift—a survey on monitoring in evolving environments. Part A: Detecting concept drift. *Frontiers in Artificial Intelligence*, 7, June 2024. ISSN 2624-8212. doi: 10.3389/frai.2024.1330257.

[44] J. Hoffmann, S. Borgeaud, A. Mensch, P. Buchlovsky, T. Cai, E. Rutherford, K. Millican, C. Jones, B. Bos, S. Gray, C. Leahy, E. Conway, Z. Dai, A. Mirhoseini, and E. Grefenstette. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[45] Andrew Howard, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, Yukun Zhu, Ruoming Pang, Hartwig Adam, and Quoc Le. Searching for mobilenetv3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019. doi: 10.1109/ICCV.2019.00140.

[46] Chuang Hu, Wei Bao, Dan Wang, and Fengming Liu. Dynamic adaptive dnn surgery for inference acceleration on the edge. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 1423–1431, 2019. doi: 10.1109/INFOCOM.2019.8737614.

[47] Weiqiang Huang, Juecen Zhan, Yumeng Sun, Xu Han, Tai An, and Nan Jiang. Context-aware adaptive sampling for intelligent data acquisition systems using dqn. *arXiv preprint arXiv:2504.09344*, 2025.

[48] R. Hussein and N. Gupta. Chatgpt's impact across sectors: A survey. *International Journal of Human–Computer Studies*, 152:102761, 2025. doi: 10.1016/j.ijhcs.2024.102761.

[49] Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, and Phillip B. Gibbons. Federated Learning under Distributed Concept Drift. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 5834–5853. PMLR, April 2023.

[50] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, page 615–629, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450344654. doi: 10.1145/3037697.3037698. URL `https://doi.org/10.1145/3037697.3037698`.

[51] J. Kaplan, S. McCandlish, T. Henighan, T. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[52] H. Kim and S. Park. Tools for understanding how large language models work. *arXiv preprint arXiv:2501.01234*, 2025.

[53] Subhash C. Kochar, Hari Mukerjee, and Francisco J. Samaniego. Estimation of a Monotone Mean Residual Life. *The Annals of Statistics*, 28(3):905–921, 2000. ISSN 0090-5364.

[54] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. In *Proceedings of the 2007 IEEE International Conference on Data Mining*, pages 123–132, 2007.

[55] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (mlops): Overview, definition, and architecture, 2023.

[56] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[58] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[59] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[60] Juhyun Lee, Nikolay Chirkov, Ekaterina Ignasheva, Yury Pisarchyk, Mogan Shieh, Fabio Riccardi, Raman Sarokin, Andrei Kulik, and Matthias Grundmann. On-device neural net inference with mobile gpus. In *Efficient Deep Learning for Computer Vision CVPR 2019 (ECV2019)*, 2019.

[61] Afroditi Letsiou, Vishnu Narayanan Moothedath, Adarsh Prasad Behera, Jaya Prakash Champati, and James Gross. Hierarchical inference at the edge: A batch pro-

cessing approach. In *2024 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 476–482, 2024. doi: 10.1109/SEC62691.2024.00055.

[62] Mengtian Li, Ersin Yumer, and Deva Ramanan. Budgeted training: Rethinking deep neural network training under resource constraints. *arXiv preprint arXiv:1905.04753*, 2019.

[63] Haijun Liao, Zhenyu Zhou, Bo Ai, and Mohsen Guizani. Learning-based energy-efficient channel selection for edge computing-empowered cognitive machine-to-machine communications. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–6, May 2020. doi: 10.1109/VTC2020-Spring48590.2020.9128780.

[64] Ahmed Ali Linkon, Mujiba Shaima, Md Shohail Uddin Sarker, Badruddowza, Norun Nabi, Md Nasir Uddin Rana, Sandip Kumar Ghosh, Mohammad Anisur Rahman, Hammed Esa, and Faiaz Rahat Chowdhury. Advancements and Applications of Generative Artificial Intelligence and Large Language Models on Business Management: A Comprehensive Review. *Journal of Computer Science and Technology Studies*, 6(1):225–232, March 2024. ISSN 2709-104X. doi: 10.32996/jcsts.2024.6.1.26.

[65] Xin Liu, Bin Li, Pengyi Shi, and Lei Ying. An efficient pessimistic-optimistic algorithm for stochastic linear bandits with general constraint. In *Advances in Neural Information Processing Systems*, volume 34, pages 24075–24086. Curran Associates, Inc., 2021.

[66] Xin Liu, Bin Li, Pengyi Shi, and Lei Ying. POND: Pessimistic-Optimistic oNline Dispatching, 2021. URL `https://arxiv.org/abs/2010.09995`.

[67] Gary Lorden and Moshe Pollak. Sequential Change-Point Detection Procedures That are Nearly Optimal and Computationally Simple. *Sequential Analysis*, 27(4):476–512, November 2008. ISSN 0747-4946, 1532-4176. doi: 10.1080/07474940802446244.

[68] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2019. ISSN 1558-2191. doi: 10.1109/TKDE.2018.2876857. URL `https://ieeexplore.ieee.org/document/8496795/`. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[69] Sandeep Madireddy, Prasanna Balaprakash, Philip Carns, Robert Latham, Glenn K Lockwood, Robert Ross, Shane Snyder, and Stefan M Wild. Adaptive learning for concept drift in application performance modeling. In *Proceedings of the 48th International Conference on Parallel Processing*, pages 1–11, 2019.

[70] Ananth Mahadevan and Michael Mathioudakis. Cost-effective retraining of machine learning models. *arXiv preprint arXiv:2310.04216*, 2023.

[71] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[72] Dimitrios Michael Manias, Ibrahim Shaer, Li Yang, and Abdallah Shami. Concept Drift Detection in Federated Networked Systems. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Madrid, Spain, December 2021. IEEE. ISBN 978-1-7281-8104-2. doi: 10.1109/GLOBECOM46510.2021.9685083.

[73] Yuyi Mao, Jun Zhang, and Khaled B. Letaief. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12):3590–3605, December 2016. ISSN 1558-0008. doi: 10.1109/JSAC.2016.2611964.

[74] Y. Mei. Efficient scalable schemes for monitoring a large number of data streams. *Biometrika*, 97(2):419–433, 2010. ISSN 0006-3444.

[75] Aryan Mokhtari, Shahin Shahrampour, Ali Jadbabaie, and Alejandro Ribeiro. Online optimization in dynamic environments: Improved regret rates for strongly convex problems. pages 7195–7201, 2016. doi: 10.1109/CDC.2016.7799379. URL https://ieeexplore.ieee.org/abstract/document/7799379?casa_token=Q4c2p0MPTMUAAAAA:kFT1P1YKdX4sEishu_jHF-0tsLA9LbTPzhuOauZdFTWEI-WigtZF62yEWrI8S3CgQ.

[76] Vishnu Narayanan Moothedath, Jaya Prakash Champati, and James Gross. Online algorithms for hierarchical inference in deep learning applications at the edge, 2023. arXiv:2304.00891.

[77] Vishnu Narayanan Moothedath, Jaya Prakash Champati, and James Gross. Getting the Best Out of Both Worlds: Algorithms for Hierarchical Inference at the Edge. *IEEE Transactions on Machine Learning in Communications and Networking*, 2:280–297, 2024. ISSN 2831-316X. doi: 10.1109/TMLCN.2024.3366501.

[78] Vishnu Narayanan Moothedath, Umang Agarwal, Umeshraja N, James Richard Gross, Jaya Prakash Champati, and Sharayu Moharir. Inference offloading for cost-sensitive binary classification at the edge. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(29):24449–24457, Mar. 2026. doi: 10.1609/aaai.v40i29.39627. URL https://ojs.aaai.org/index.php/AAAI/article/view/39627.

[79] F. Moreno-García, J. del Campo-Ávila, N. García-Pedrajas, and S. Ventura. Dynamic ensemble selection based on local accuracy for non-stationary environments. *Data Mining and Knowledge Discovery*, 34(3):736–770, 2020. doi: 10.1007/s10618-019-00658-8.

[80] Maksim Muravev, Brazhenko , Dmitry, Somenkova , Anzhela, Golovkov , Alexander, and Ilia and Sergunin. MLOps Architecture as a Future of Machine

Learning. *Journal of Computer Information Systems*, 0(0):1–13, 2025. ISSN 0887-4417. doi: 10.1080/08874417.2025.2483826.

[81] Michael Neely. *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, 2010.

[82] Ivana Nikoloska and Nikola Zlatanov. Data selection scheme for energy efficient supervised learning at iot nodes. *IEEE Communications Letters*, 25(3):859–863, March 2021. ISSN 1558-2558. doi: 10.1109/LCOMM.2020.3034992.

[83] J. Park, K. Lee, and M. Cho. LLamaDuo: Dual-mode inference for on-device llms. In *Proceedings of EMNLP*, pages 1123–1135. ACL, 2024.

[84] Moshe Pollak. Optimal Detection of a Change in Distribution. *The Annals of Statistics*, 13(1):206–227, March 1985. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1176346587.

[85] Aurick Qiao, Sang Keun Choe, Suhas Jayaram Subramanya, Willie Neiswanger, Qirong Ho, Hao Zhang, Gregory R Ganger, and Eric P Xing. Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, 2021.

[86] Partha Pratim Ray. A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 34(4): 1595–1623, April 2022. ISSN 1319-1578. doi: 10.1016/j.jksuci.2021.11.019.

[87] Florence Regol, Leo Schwinn, Kyle Sprague, Mark Coates, and Thomas Markovich. When to retrain a machine learning model, 2025. URL `https://openreview.net/forum?id=iGX0lwpUYj`.

[88] Suresh P. Sethi. *Optimal Control Theory: Applications to Management Science and Economics*. Springer Texts in Business and Economics. Springer Inter-

national Publishing, Cham, 2021. ISBN 978-3-030-91744-9 978-3-030-91745-6. doi: 10.1007/978-3-030-91745-6.

[89] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, Portugal, 2018.

[90] A. Shehabi, A. Newkirk, S. Smith, A. Hubbard, N. Lei, M. Siddik, et al. 2024 United States Data Center Energy Usage Report. Technical Report LBNL-2001637, Lawrence Berkeley National Laboratory, 2024.

[91] Li Shen, Yan Sun, Zhiyuan Yu, Liang Ding, Xinmei Tian, and Dacheng Tao. On efficient training of large-scale deep learning models: A literature review, April 2023. arXiv:2304.03589.

[92] Ya Shen, Gang Chen, Hui Ma, and Mengjie Zhang. Cost-aware dynamic cloud workflow scheduling using self-attention and evolutionary reinforcement learning. In *International Conference on Service-Oriented Computing*, pages 3–18. Springer, 2024.

[93] Md. Maruf Hossain Shuvo, Syed Kamrul Islam, Jianlin Cheng, and Bashir I. Morshed. Efficient acceleration of deep learning inference on resource-constrained edge devices: A review. *Proceedings of the IEEE*, 111(1):42–91, 2023. doi: 10.1109/JPROC.2022.3226481.

[94] Andrea Simonetto, Aryan Mokhtari, Alec Koppel, Geert Leus, and Alejandro Ribeiro. A class of prediction-correction methods for time-varying convex optimization. *IEEE Transactions on Signal Processing*, 64(17):4576–4591, 2016. ISSN 1053-587X, 1941-0476. doi: 10.1109/TSP.2016.2568161. URL `http://ieeexplore.ieee.org/document/7469393/`.

[95] Thitirat Siriborvornratanakul. From human annotators to ai: The transition and the role of synthetic data in ai development. In Helmut Degen and Stavroula Ntoa, editors, *Artificial Intelligence in HCI*, pages 379–390, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-93429-2.

[96] Jasper Stone, Raj Patel, Farbod Ghiasi, Sudip Mittal, and Shahram Rahimi. Navigating mlops: Insights into maturity, lifecycle, tools, and careers. In *2025 IEEE Conference on Artificial Intelligence (CAI)*, pages 643–650, 2025. doi: 10.1109/CAI64502.2025.00118.

[97] Alexander Tartakovsky, Igor Nikiforov, and Michele Basseville. *Sequential Analysis: Hypothesis Testing and Changepoint Detection*. Chapman and Hall/CRC, New York, August 2014. ISBN 978-0-429-15234-4. doi: 10.1201/b17279.

[98] Alexander G. Tartakovsky. Asymptotically Optimal Quickest Change Detection in Multistream Data—Part 1: General Stochastic Models. *Methodology and Computing in Applied Probability*, 21(4):1303–1336, December 2019. ISSN 1573-7713. doi: 10.1007/s11009-019-09735-3.

[99] Alexander G. Tartakovsky and Venugopal V. Veeravalli. Asymptotically Optimal Quickest Change Detection in Distributed Sensor Systems. *Sequential Analysis*, 27(4):441–475, November 2008. ISSN 0747-4946. doi: 10.1080/07474940802446236.

[100] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. BranchyNet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, December 2016. doi: 10.1109/ICPR.2016.7900006.

[101] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you

need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[102] Venugopal V. Veeravalli, Georgios Fellouris, and George V. Moustakides. Quickest Change Detection With Controlled Sensing. *IEEE Journal on Selected Areas in Information Theory*, 5:1–11, 2024. ISSN 2641-8770. doi: 10.1109/JSAIT.2024.3362324.

[103] Tom Viering and Marco Loog. The shape of learning curves: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(6):7799–7819, June 2023. ISSN 0162-8828. doi: 10.1109/TPAMI.2022.3220744. URL `https://doi.org/10.1109/TPAMI.2022.3220744`.

[104] Shaoqi Wang, Aidi Pi, and Xiaobo Zhou. Elastic parameter server: Accelerating ml training with scalable resource scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 33:1128–1143, 2021.

[105] Xubin Wang, Zhiqing Tang, Jianxiong Guo, Tianhui Meng, Chenhao Wang, Tian Wang, and Weijia Jia. Empowering edge intelligence: A comprehensive survey on on-device ai models. *ACM Comput. Surv.*, 57(9), April 2025. ISSN 0360-0300. doi: 10.1145/3724420. URL `https://doi.org/10.1145/3724420`.

[106] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017. URL `https://arxiv.org/abs/1708.07747`.

[107] Yao Xie and David Siegmund. Sequential multi-sensor change-point detection. *The Annals of Statistics*, 41(2), April 2013. ISSN 0090-5364. doi: 10.1214/13-AOS1094.

[108] Jie Xu, Lixing Chen, and Pan Zhou. Joint service caching and task offloading for mobile edge computing in dense networks. In *IEEE INFOCOM 2018 -*

IEEE Conference on Computer Communications, pages 207–215, April 2018. doi: 10.1109/INFOCOM.2018.8485977.

[109] Qunzhi Xu and Yajun Mei. Asymptotic optimality theory for active quickest detection with unknown postchange parameters. *Sequential Analysis*, 42(2): 150–181, April 2023. ISSN 0747-4946. doi: 10.1080/07474946.2023.2187417.

[110] Qunzhi Xu, Yajun Mei, and George V. Moustakides. Optimum Multi-Stream Sequential Change-Point Detection With Sampling Control. *IEEE Transactions on Information Theory*, 67(11):7627–7636, November 2021. ISSN 1557-9654. doi: 10.1109/TIT.2021.3074961.

[111] En Yu, Jie Lu, Kun Wang, Xiaoyu Yang, and Guangquan Zhang. Drift-aware Collaborative Assistance Mixture of Experts for Heterogeneous Multistream Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40 (19):16199–16207, March 2026. ISSN 2374-3468. doi: 10.1609/aaai.v40i19.38656.

[112] Hang Yu, Weixu Liu, Jie Lu, Yimin Wen, Xiangfeng Luo, and Guangquan Zhang. Detecting group concept drift from multiple data streams. *Pattern Recognition*, 134:109113, February 2023. ISSN 00313203. doi: 10.1016/j.patcog.2022.109113.

[113] Menglu Yu, Ye Tian, Bo Ji, Chuan Wu, Hridesh Rajan, and Jia Liu. Gadget: Online resource optimization for scheduling ring-all-reduce learning jobs. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1569–1578. IEEE, 2022.

[114] Ming Zhou and Jie Lu. Continuous Graph Learning-Based Self-Adaptation for Multi-Stream Concept Drift. *IEEE Transactions on Cybernetics*, 55(8): 3760–3773, August 2025. ISSN 2168-2275. doi: 10.1109/TCYB.2025.3569816.

[115] X. Zhou, L. Chen, and T. Wu. TinyLLaVA: A lightweight vision–language assistant. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9876–9885, 2024.

[116] Indrė Žliobaitė, Marcin Budka, and Frederic Stahl. Towards cost-sensitive adaptation: When is it worth updating your predictive model? *Neurocomputing*, 150:240–249, 2015.

[117] Shaofeng Zou, Venugopal V. Veeravalli, Jian Li, and Don Towsley. Quickest Detection of Dynamic Events in Networks. *IEEE Transactions on Information Theory*, 66(4):2280–2295, April 2020. ISSN 0018-9448, 1557-9654. doi: 10.1109/TIT.2019.2948350.

Vita

Hasan Burhan Beytur received the B.Sc. and M.Sc. degrees in Electrical and Electronics Engineering from Middle East Technical University, Ankara, Turkey, in 2017 and 2020, respectively. From 2018 to 2020, he was an R&D Engineer with Turk Telekom. He is pursuing his Ph.D. degree in Electrical and Computer Engineering at The University of Texas at Austin, USA, under the supervision of Professor Gustavo de Veciana. His research interests include communication networks, machine learning systems, and stochastic modeling and analysis.

Address: hbbeytur@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.