UNIVERSITY OF CALIFORNIA,
IRVINE

Middleware Systems for Efficient and Robust Distributed Machine Learning

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Networked Systems

by

Rahul Atul Bhope

Dissertation Committee:
Professor Nalini Venkatasubramanian, UCI, Chair
Professor Marco Levorato, UCI
Professor Sangeetha Abdu Jyothi, UCI
Professor Sharad Mehrotra, UCI
Dr. Jayaram K. Radhakrishnan, IBM Research

2026

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

My journey through this PhD has been shaped by the kindness, trust, and support of many people who took a chance on me at different points in time. This thesis is as much a reflection of their belief in me as it is of my own efforts.

First and foremost, I am deeply grateful to my advisor, Prof. Nalini Venkatasubramanian, for her unwavering support and the many research opportunities she provided. Her mentorship has not only shaped this thesis but also helped me grow into an independent researcher.

I am thankful to my committee members, Prof. Marco Levorato, Prof. Sharad Mehrotra, Prof. Sangeetha Abdu Jyothi, and Dr. Jayaram K. Radhakrishnan, for their valuable feedback and suggestions that significantly improved this work.

I would especially like to thank Dr. Jayaram K. Radhakrishnan, who has been pivotal in my PhD journey. His guidance, insights, and constant encouragement played a crucial role in shaping my research direction and helping me navigate challenges along the way.

I would like to thank my internship mentors, Dr. Praveen Venkateswaran, Dr. Vatche Isahagian, and Dr. Vinod Muthusamy, for the opportunity to explore research in diverse environments and for their invaluable mentorship. Working with them broadened my perspective on both research and its real-world impact.

To my lab mates and colleagues at UCI, thank you for the stimulating discussions, collaboration, and camaraderie. The environment you all created made even the most challenging times enjoyable: Fangqi Liu, Tung Chun Chan, Andrew Chio, Modeste Mefenya Kenne, Ryan Hildebrant, Jared Macshane, Vishal Chakraborty, Yuqiao Li, and Fernanda Ventorim.

I owe everything to my family, Atul Bhope, Latika Bhope, and Rajas Bhope, whose countless sacrifices, unwavering belief in me, and constant motivation made this journey possible. The values they instilled in me and their support have been the foundation upon which all my achievements stand. This accomplishment is as much theirs as it is mine.

I want to thank my partner, Vedita Babuta, for being my constant source of strength and support. Your patience, encouragement, and belief in me helped me push through the most difficult moments and made this journey all the more meaningful.

I am especially grateful for the many friends I made at UCI, far too many to name individually, who made these years richer, brighter, and more enjoyable. Special thanks go to Sukriti Kapoor, Shreya Mukhopadhyay, Rohith Reddy Gangam, Prakhar Srivastava, Aishwarya Borate, Praveen Ranganath Prabhakar, Sridipta Ghatak, and Nitish Nagesh, whose friendship, encouragement, and shared memories made this journey enjoyable.

To everyone who has supported me along the way, directly or indirectly, thank you!

VITA

Rahul Atul Bhope

EDUCATION

Doctor of Philosophy in Networked Systems — **2021–2026**
University of California, Irvine — *Irvine, California*

Bachelor of Technology in Instrumentation & Control Engineering — **2016–2020**
National Institute of Technology, Tiruchirappalli — *India*

RESEARCH EXPERIENCE

Graduate Research Assistant — **2021–2026**
University of California, Irvine — *Irvine, California*

Research Scientist Intern — **Summer 2025**
IBM Research — *Yorktown Heights, New York*

Research Scientist Intern — **Summer 2024**
IBM Research — *Yorktown Heights, New York*

Research Scientist Intern — **Summer 2022**
IBM Research — *Yorktown Heights, New York*

Project Associate — **2020–2021**
Indian Institute of Science — *Bangalore, India*

Research Intern — **Summer 2019**
Nanyang Technological University — *Singapore*

Research Intern — **Winter 2018**
Indian Institute of Technology — *Kharagpur, India*

TEACHING EXPERIENCE

Teaching Assistant — **2021–2023**
University of California, Irvine — *Irvine, California*

REFEREED CONFERENCE PUBLICATIONS

VOILA: Value-of-Information Guided Fidelity Selection for Cost-Aware Multimodal Question Answering **Jan 2026**
Under Review

Shift Happens: Mixture of Experts based Continual Adaptation in Federated Learning **Dec 2025**
In Proceedings of the 26th International Middleware Conference

Middleware Systems for Efficient and Robust Distributed Machine Learning **Dec 2025**
In Proceedings of the 26th International Middleware Conference (Doctoral Symposium)

In-Context Example Ordering for LLM-Based API Sequence Generation **Dec 2025**
In Proceedings of the 26th International Middleware Conference (Demos)

OptiSeq: Ordering Examples On-The-Fly for In-Context Learning **Nov 2025**
Findings of the Association for Computational Linguistics: EMNLP 2025

DIM-SUM: Dynamic Imputation Methods for Smart Utility Management **July 2025**
Proceedings of the VLDB Endowment (PVLDB)

FLIPS: Federated Learning using Intelligent Participant Selection **Dec 2023**
Proceedings of the 24th International Middleware Conference

REFEREED JOURNAL PUBLICATIONS

Testing CareDEX: SAGAT Methodology Deployed for a Disaster Drill at a Senior Living Community **2023**
Issues in Information Systems (IACIS)

ABSTRACT OF THE DISSERTATION

Middleware Systems for Efficient and Robust Distributed Machine Learning

By

Rahul Atul Bhope

Doctor of Philosophy in Networked Systems

University of California, Irvine, 2026

Professor Nalini Venkatasubramanian, UCI, Chair

Modern machine learning systems increasingly operate across multiple stages of the lifecycle: large-scale training over distributed data sources, continual adaptation as data distributions evolve after deployment, and inference-time decision making for foundation model applications under latency and cost constraints. Across these stages, systems must operate under practical constraints, including data-sharing regulations, data heterogeneity, non-stationary environments, latency budgets, and limited computational resources.

This dissertation offers a unified view of the machine learning lifecycle through the lens of these operational decisions, and argues for a dedicated *middleware layer*: a set of principled decision policies that governs behavior across training, continual adaptation, and inference. In this view, middleware is not merely a communication conduit or infrastructure plumbing layer, but the decision brain between machine learning applications, models, and their operating environments. It interprets available system and model signals, reasons over constraints, and coordinates adaptive behavior.

First, to achieve robust training across data and platform heterogeneity, we present techniques for federated learning that leverage participant-level data characteristics to improve coordination across distributed clients. By analyzing label distribution structure, training can better preserve diversity, improve convergence, and reduce communication overhead.

Second, for continual learning under distribution shifts, we develop mechanisms that monitor evolving data conditions and selectively specialize models as environments change. By using learned representations and label statistics to detect shifts, deployed systems can adapt without the expense of full retraining. Third, for inference-time optimization, we study dynamic context management for large language and vision-language models. We show how outputs such as log probabilities, confidence estimates, and lightweight metadata can guide decisions, including example ordering, retrieval fidelity selection, and cost-aware context construction.

Although these settings appear distinct, they share a common systems pattern: machine learning systems already generate rich internal signals that reflect model state, data state, and execution conditions, yet these signals are rarely exposed as first-class abstractions for system control. This dissertation argues that treating such signals as actionable primitives enables a new decision layer for machine learning systems—one that can observe, analyze, and act on model-generated evidence to build adaptive policies across training, deployment, and inference, thereby enabling efficient, robust, and self-adaptive machine learning systems.

Chapter 1

Introduction

Modern machine learning systems are no longer static artifacts trained once and deployed in isolation. They now span the full lifecycle of model development and deployment, including large-scale training, continual adaptation, and inference-time decision making. Training and inference themselves have received enormous attention, with a wide range of systems solutions for distributed optimization, scalable serving, and efficient execution [114, 195, 180, 252, 99, 157, 158, 236]. However, when these workloads are subject to real-world operational constraints, classical centralized assumptions often break down, and new techniques become necessary [28, 151, 115].

When training data is distributed across users, institutions, or edge devices and cannot be centrally pooled due to data-sharing regulations (GDPR, CCPA, HIPAA) [216, 104, 211], ownership boundaries, or limited bandwidth, techniques such as federated learning [151, 28] offer an effective approach and have demonstrated strong practical value. As deployed environments evolve, data distributions shift, and previous models can become stale [36, 225]. Repeated retraining is often costly, while a single static model may be insufficient to capture multiple evolving distributions. Systems, therefore, need mechanisms that adapt to new data

while preserving prior knowledge and mitigating catastrophic forgetting [150, 61, 145, 225]. Techniques such as continual learning and federated continual learning offer a promising approach for sustained adaptation [31, 244, 225]. At inference time, when model parameters cannot be retrained or modified per query—as is common with foundation model APIs—and applications face latency, token-budget, and cost–quality tradeoffs, techniques such as retrieval-augmented generation, prompt compression, and query-aware routing provide effective means of query-time adaptation [105, 119, 85, 51, 124]. In such settings, system-level decisions—such as which data to use, how to structure inputs, how much computation to allocate, or when to adapt—become as critical as the underlying models themselves [23, 24, 25, 22].

This dissertation argues that these challenges demand a dedicated *middleware layer*—not merely infrastructure plumbing, but a principled decision brain that sits between data sources, models, and applications to govern behavior across the machine learning lifecycle. The central contribution of this work is to develop middleware systems that govern decisions across training under data heterogeneity, continual learning under distributional shifts, and inference-time resource management. Across federated learning, continual learning, and large language and vision-language model inference, this dissertation shows that principled middleware policies—not necessarily architectural changes—are often a key lever for improving efficiency, adaptability, and reliability in real-world deployments [23, 24, 25, 22].

1.1 The Machine Learning Lifecycle and the Role of Middleware

Machine learning systems are increasingly embedded in mission-critical applications whose operational constraints determine how learning and inference must be performed. A motivating example in this dissertation is *CareDEX* [10], an NSF-funded smart-space platform

designed to improve disaster resilience for older adults living in senior housing facilities and age-friendly communities. Through CareDEX, we observed a set of practical problems that recur well beyond senior care: sensitive data is often distributed across organizations and devices, operating environments change after deployment, and time-critical decisions depend on selecting the right information under privacy, latency, and reliability constraints [10, 216, 211]. These observations motivate a broader systems question: how should machine learning applications make principled decisions across the full lifecycle when data, users, environments, and resources are heterogeneous?

Different phases of the machine learning lifecycle require distinct system capabilities. When multiple organizations, users, or edge devices must collaboratively learn without exposing raw data, *federated learning* becomes necessary [151, 28]. *Continual learning* shows up in applications like satellite imagery analysis, where data evolves over time and models must adapt to changing environments and distributions [36, 225]. At inference time, we see applications like agentic multimodal workflows, where models consume context from multiple sources and their behavior depends on how that context is selected and structured [105, 119, 85, 22]. Similarly, in settings like drone-based flood assessment, data is collected in resource-constrained environments, and decisions about what information to use directly impact both efficiency and outcomes. In each case, performance depends not only on the model itself, but on how application-specific decisions are made under operational constraints [23, 24, 25, 22].

This dissertation argues that these recurring decisions demand a dedicated *middleware layer*. Middleware here is not merely communication infrastructure, but rather the decision-making layer between applications, data sources, models, and deployment environments. It coordinates participant selection and aggregation during training, governs adaptation as conditions change after deployment, and manages context retrieval, tool use, and budget-aware inference during application execution. By translating real-world constraints into principled runtime decisions, middleware provides the systems abstraction needed to make machine

learning robust, adaptive, privacy-aware, and operationally effective across the full lifecycle [23, 24, 25, 22].

1.2 Challenges in Real-World Deployments

The deployment of machine learning in real-world environments reveals a set of recurring challenges spanning the full lifecycle—from data collection and model training to inference and adaptation. These challenges are not artifacts of any single model or dataset; they arise from the structural properties of the environments in which modern ML systems must operate, including application-specific privacy requirements, sensing conditions, latency budgets, failure modes, and outcome objectives [28, 36, 51, 10].

Robust Training under Data and Platform Heterogeneity. Federated learning systems must operate under two intertwined sources of heterogeneity that fundamentally challenge efficient and robust training. First, participants hold non-IID data with highly skewed label distributions, violating the assumptions of standard aggregation methods and leading to slow convergence as well as systematic underrepresentation of minority classes [151, 115, 91]. Second, participants exhibit platform heterogeneity, differing in compute capacity, communication bandwidth, and availability, which introduces stragglers that delay or fail to contribute updates in each round [28, 42, 100]. These issues are shaped by the application itself: clinically rare events, geographically localized patterns, disaster-specific observations, or low-connectivity field devices can be exactly the data sources a system most needs, even when they are hardest to access reliably. Crucially, these challenges are not independent: participants that contain rare or statistically important data are often resource-constrained, making them both less likely to be selected and more likely to drop out when selected. This compounding effect exacerbates bias and inefficiency in the learned model, motivating the need for principled participant selection strategies that jointly account for data diversity, application value, and system constraints in a privacy-preserving manner [100, 23].

Continual Adaptation under Distribution Shifts. Deployed federated models operate in environments where data distributions evolve continuously across both time and participants, introducing fundamental challenges for maintaining model performance [36, 46, 225]. One source of drift arises from covariate shift, where the input distribution $P(X)$ changes due to factors such as sensor recalibration, environmental variation, seasonal change, or behavioral drift, rendering previously learned representations less reliable [232, 205, 24]. In parallel, label shift occurs when the label distribution $P(Y)$ evolves over time, altering class prevalences and effectively changing the learning objective, often degrading accuracy on newly dominant classes [241, 24]. The importance of a shift is also application-specific: a rare but safety-critical class, a newly flooded region, or a changing land-cover pattern may require adaptation even when aggregate accuracy appears stable. These challenges are further amplified in federated settings, where different parties experience distinct shifts at different times. In such cases, naively aggregating updates into a single global model leads to negative knowledge transfer, where adapting to one party's distribution harms performance on others [225, 24]. This creates a fundamental systems question: *when and how should deployed models adapt to distributional changes in a way that remains responsive to local shifts while avoiding catastrophic forgetting?* [150, 61, 244, 122].

Dynamic Context Management in Agentic AI Systems. LLM- and VLM-based agentic systems introduce a new class of inference-time decision problems, where system performance depends not only on model capability but also on how inputs are structured and resources are allocated [51, 160, 86]. At the application level, model outputs are highly sensitive to the ordering of in-context examples in few-shot prompts, yet there is no principled way to determine the optimal ordering at inference time [25]. Naïve reliance on log-probability scores is insufficient, as these scores entangle the influence of example order with output confidence, obscuring which orderings are truly effective [25]. At the system level, inference pipelines must also decide the fidelity of retrieved inputs—such as whether

to fetch full-resolution images for vision-language models—under strict latency and bandwidth constraints [41, 81, 248, 22]. These decisions are application-specific: a multimodal agent may need to choose among logs, images, tool outputs, retrieved documents, or sensor streams, while a drone-based flood assessment workflow may need to trade off image resolution, transmission cost, and decision urgency. Existing systems often default to maximum fidelity, incurring unnecessary cost, because they lack mechanisms to predict the required level of detail *before* retrieval [41, 81, 22]. Together, these challenges expose a unified inference-time question: *how can systems leverage available signals to make principled decisions about input structuring and resource allocation without relying on additional training or offline tuning?* [25, 22].

1.3 Problem Statement

Modern machine learning systems increasingly operate across multiple stages of the lifecycle: distributed training over decentralized data sources, deployment in environments where data distributions evolve over time, and inference-time use of foundation models under latency and cost constraints [151, 36, 51]. Across these stages, system performance depends not only on model quality, but also on operational decisions made around the model [23, 24, 25, 22].

This dissertation addresses the following central problem:

> *How can we design middleware systems that enable adaptive decision-making across training, continual learning, and inference under heterogeneity, distribution shift, resource constraints, and dynamic workloads?*

Although these settings differ, they repeatedly expose three practical challenges:

1. **Heterogeneous operating conditions.** Machine learning systems must function across diverse data sources, participants, workloads, and hardware environments, where

uniform strategies are often ineffective [115, 28, 51].

2. **Changing environments over time.** Data distributions, user behavior, and task requirements evolve after deployment, requiring systems to respond without expensive retraining or manual redesign [36, 145, 225].

3. **Resource-constrained decision making.** Communication bandwidth, compute budgets, latency limits, and context windows restrict how models can be trained, updated, and queried in practice [28, 119, 85, 51].

These challenges arise throughout the lifecycle: in federated training when selecting participants under non-IID data, in continual learning when deciding when and how to adapt under shift, and in foundation model inference when choosing how to structure or compress context efficiently. This dissertation studies these recurring decision problems and develops middleware mechanisms that sit around existing models to coordinate more adaptive, efficient, and robust system behavior [23, 24, 25, 22].

1.4 Thesis Contributions

This dissertation develops a unified framework for signal-driven middleware design and instantiates it across four systems.

FLIPS: Intelligent Participant Selection in Federated Learning [23] FLIPS introduces a clustering-based selection framework that embeds label distributions into participant profiles and enforces statistical diversity across communication rounds. This provably improves convergence under heterogeneous data and reduces the number of rounds needed to reach a target accuracy, without centralizing any participant data.

ShiftEx: Shift-Aware Federated Learning [24] ShiftEx proposes a shift-detection and routing mechanism that monitors latent representations and label statistics across streaming

data, triggering dynamic specialization of an underlying mixture-of-experts model. This enables targeted adaptation to new distributions while preserving performance on previously observed ones.

OptiSeq: Example Ordering in In-context Learning [25] OptiSeq frames the ordering of in-context examples as an inference-time middleware problem in distributed agentic systems. Rather than assuming a fixed or heuristic ordering, it treats example sequencing as a critical decision variable that directly influences model outputs, especially in tasks such as API sequence generation. To address this, OptiSeq introduces debiased log-probability scoring to evaluate candidate permutations at inference time without requiring additional training or labeled feedback. By selecting the ordering most likely to yield correct outputs, it enables instance-specific prompt optimization, improving both accuracy and reliability in downstream generation tasks.

VOILA: Cost-Aware Retrieval across Distributed Storage Tiers [22] VOILA is motivated by a fundamental systems constraint: in real-world deployments, multimodal context is distributed across storage tiers with heterogeneous latency, bandwidth, and cost profiles, where high-fidelity inputs are substantially more expensive to access than compressed or cached representations. Rather than uniformly retrieving the highest-fidelity data, VOILA formulates retrieval as a value-of-information decision problem, using calibrated, question-conditioned predictions to estimate the expected benefit of accessing higher-fidelity inputs before any retrieval occurs. It then selects, on a per-query basis, the minimum-cost fidelity whose expected accuracy justifies its acquisition cost, enabling consistent accuracy–cost trade-offs without incurring the overhead of always retrieving high-fidelity context.

1.5 Dissertation Organization

The remainder of this dissertation is organized as follows.

- Chapter 2 surveys related work in distributed and adaptive machine learning systems, continual learning under distribution shift, and inference-time decision making in large language and vision-language models, along with prior approaches to signal- and data-driven system design.

- Chapter 3 introduces the middleware framework for ML lifecycle decision-making, presenting the core abstraction of middleware as a principled decision layer and the observe–analyze–act methodology that underpins system-level decision making across training, continual adaptation, and inference.

- Chapter 4 (*Robust Training under Data Heterogeneity*) presents FLIPS, a middleware system for diversity-aware participant selection in federated learning.

- Chapter 5 (*Continual Adaptation under Distribution Shift*) presents ShiftEx, a shift-aware federated learning framework that adapts models via mixture-of-experts specialization.

- Chapter 6 (*Inference-Time Decision Making in Agentic AI Systems*) presents OptiSeq and VOILA, two middleware systems that address inference-time orchestration as a distributed systems problem: OptiSeq governs the ordering of examples in in-context learning for tasks such as sequencing API and tool invocations across distributed service catalogs, while VOILA manages context fidelity selection across tiered storage to balance retrieval cost against answer quality.

- Chapter 7 synthesizes the key insights from the individual works in this dissertation and outlines directions for future research.

Taken together, this work demonstrates that principled middleware design can significantly improve the efficiency, adaptability, and practicality of modern machine learning systems.

Chapter 2

Related Work

Machine learning systems now operate under increasing operational constraints arising from distributed data, limited resources, and dynamic deployment environments. A large body of prior work has addressed individual aspects of these challenges, including system design for distributed learning, continual adaptation under shift, and inference-time optimization for large models. However, these research directions have largely evolved independently, each community developing tools and abstractions tuned to its own objectives and assumptions, with limited cross-pollination across problem domains.

Research in machine learning has traditionally concentrated on two phases of the pipeline: *training*, where a model is fit to collected data, and *inference*, where it is queried to produce predictions. These phases have attracted enormous research investment, driving progress in optimization, generalization, and benchmark performance. A third phase that sits between and around them—*continual learning*, the ongoing process of keeping deployed models aligned with evolving data distributions and operating conditions—has received comparatively less systematic attention as a full-stack systems problem. Together, these three phases expose a set of challenges that cannot be resolved by model design alone; they require ded-

icated decision-making infrastructure: middleware layers that govern system behavior at each stage independently of the underlying model architecture. Accordingly, this chapter emphasizes representative work that exposes the operational decisions most relevant to the thesis—coordination, adaptation, and compute allocation—rather than attempting an exhaustive survey of every adjacent subfield.

2.1 Systems for Distributed and Adaptive ML

Research on distributed and adaptive machine learning spans system architectures for large-scale training, cluster-level scheduling and resource management, and decentralized paradigms motivated by privacy, locality, and heterogeneous edge environments. Prior surveys have shown that these lines of work repeatedly confront the same core tensions: synchronization versus staleness, throughput versus statistical efficiency, and central coordination versus decentralization [215, 149, 20, 102, 49].

2.1.1 Distributed Training Systems and the Parameter-Server Line

The parameter server (PS) is one of the foundational abstractions for distributed machine learning. In this architecture, worker nodes compute updates while server nodes maintain globally shared parameters. Early PS systems emphasized asynchronous communication, bounded consistency, elasticity, and fault tolerance, and demonstrated scalability to models with billions of parameters and examples [111, 109]. Closely related large-scale systems such as DistBelief further showed that asynchronous replicas and model partitioning could scale deep network training across very large clusters, helping establish distributed deep learning as a systems problem in its own right [48]. Together, these systems established a practical template for large-scale distributed optimization and strongly influenced later distributed learning platforms.

As distributed deep learning matured, general-purpose frameworks such as TensorFlow

broadened the systems view from specialized parameter synchronization to heterogeneous graph execution across CPUs, GPUs, mobile devices, and clusters [1]. In parallel, the literature increasingly formalized the distinction between data parallelism, model parallelism, and pipeline parallelism, with surveys showing that no single strategy is uniformly optimal across models, batch sizes, memory budgets, and network conditions [20, 102, 207]. Subsequent system designs therefore moved toward hybrid forms of parallelism that combine data, model, and pipelined execution to improve scalability for large models and heterogeneous accelerators [163, 167, 57].

Another central thread is communication efficiency. As multiple surveys note, the scalability of distributed deep learning is often limited less by raw compute than by synchronization cost, network contention, and communication frequency [206, 193, 161]. This has led to a large body of work on gradient compression, delayed synchronization, overlap of computation and communication, and alternative communication backends. For example, optimized all-reduce implementations and communication fusion have been shown to substantially reduce end-to-end training time in large GPU clusters [201]. More recent systems further revisit long-standing architectural choices: BAGUA explicitly exposes relaxed communication semantics as a systems interface [65], while newer work argues that parameter-server-like point-to-point communication may again become attractive under imbalanced post-training workloads for large language models [217]. Overall, the systems literature has evolved from fixed synchronization designs toward more adaptive training substrates that jointly tune optimizer behavior, communication structure, and execution policy.

2.1.2 Scheduling, Placement, and Resource Allocation

As distributed learning became a dominant workload in shared GPU clusters, the problem expanded from *within-job* parallelization to *across-job* scheduling and resource allocation. Recent surveys treat workload scheduling for distributed deep learning as a distinct systems

area, emphasizing the effects of GPU heterogeneity, network topology, migration overheads, and multi-objective trade-offs among utilization, fairness, throughput, and job completion time [149, 126]. A key insight in this line of work is that distributed training performance depends not only on how many resources a job receives, but also on where those resources are placed and how the training configuration adapts to them.

One major direction is elastic and co-adaptive scheduling. Pollux models job *goodput* by combining system throughput with statistical efficiency and then reallocates resources dynamically to optimize cluster-wide performance [172]. EDL similarly targets low-overhead elasticity by enabling online changes in training parallelism for migration, transient resource use, and straggler mitigation [227]. Lyra extends this idea to settings where idle inference capacity can be loaned to training jobs, combining capacity borrowing with elastic scaling to reduce queueing and completion times [108]. These systems depart from static user-specified allocations and instead treat the interaction between scheduler decisions and learning dynamics as a first-class concern.

A second direction focuses on network-aware placement and communication-sensitive scheduling. CASSINI shows that accounting for communication patterns during job placement can substantially improve both average and tail completion times in ML clusters [178]. Related work further argues that strict network fair sharing is not always desirable for distributed learning, because certain combinations of jobs can benefit from intentionally non-uniform link allocation when their communication phases interact favorably [179]. At a lower level, TopoOpt co-optimizes training parallelization with topology and routing choices, illustrating that the network itself can be part of the distributed learning design space rather than a fixed substrate [223].

Resource allocation has also been studied inside individual training jobs. In parameter-server settings with heterogeneous workers, H-PS dynamically schedules tasks to mitigate stragglers and combines pipelining with quantization to improve end-to-end efficiency [228]. In-network

aggregation work has similarly shown that switch-memory scheduling can become a bottleneck during large-scale gradient aggregation [173]. More generally, runtime-level techniques such as memory-aware pipeline scheduling and out-of-order backpropagation demonstrate that substantial gains are still available from reordering and overlapping training operations even when the underlying learning algorithm remains unchanged [57, 159]. Taken together, this literature suggests that adaptive distributed ML systems should jointly reason about model execution, communication structure, and cluster-level resource management.

2.1.3 Edge Learning, Federated Learning, and Decentralized Paradigms

A parallel research direction shifts distributed learning away from centralized clusters toward the network edge. In these settings, data locality, privacy constraints, energy limits, and intermittent connectivity become primary design considerations. Surveys on edge learning and distributed learning across cloud, mobile, and edge settings highlight how these environments differ from traditional cluster training by introducing stronger forms of resource and data heterogeneity [93, 207, 215]. In wireless edge scenarios, the objective typically extends beyond throughput to include delay, energy, and service quality [165, 154].

Federated learning (FL) has become the dominant abstraction in this setting. FedAvg demonstrated that local training with periodic model averaging can reduce communication rounds significantly relative to synchronized SGD while still supporting non-IID and unbalanced client data [151]. Subsequent surveys positioned FL as the core systems paradigm for privacy-preserving edge intelligence, while also identifying persistent challenges such as client heterogeneity, limited device capability, stragglers, unstable participation, and expensive communication [19, 129, 209]. At the algorithm–systems boundary, methods such as FedProx, SCAFFOLD, and adaptive federated optimization made explicit that client heterogeneity and optimizer choice can substantially alter stability and convergence in practical deployments [115, 91, 183]. Recent work has therefore emphasized online client scheduling

and joint optimization of communication and computation resources, for example through stochastic control for client participation [67] and energy-aware CPU/GPU and bandwidth allocation in federated edge learning [242]. This literature increasingly treats federated learning as a resource orchestration problem as much as a distributed optimization problem.

More recent decentralized paradigms further relax the assumptions of classical FL. Decentralized federated learning removes the central server and studies peer-to-peer topologies, synchronization mechanisms, and robust overlay construction [239, 79]. Proxy-based decentralized learning additionally allows greater model heterogeneity while reducing communication cost and strengthening privacy guarantees [90]. Split learning takes a different route by partitioning the model itself across client and server so that resource-constrained devices can offload much of the training computation while keeping raw data local [214, 125, 130]. In vehicular and wireless edge settings, recent systems combine federated and split learning, adapt the cut layer dynamically, and jointly optimize model partitioning with resource allocation [171, 237]. Finally, broader decentralized training directions, including volunteer-style collaborative learning and cloud-scale communication minimization for very large models, suggest that the design space is widening beyond traditional parameter-server and centralized-FL assumptions [14, 52, 251, 65]. These developments motivate adaptive distributed ML systems that can operate across a spectrum of topologies, coordination models, and resource environments.

2.2 Continual Learning and Adaptation under Distribution Shift

Research on continual learning under distribution shift sits at the intersection of three mature but historically separate lines of work: catastrophic forgetting in sequential learning, adaptation to domain and test-time shifts, and distributed learning under non-stationarity. Classical

continual learning studies asked how a model can acquire new tasks without overwriting previously learned representations, while more recent work has emphasized that the data stream itself may drift over time, rendering adaptation inseparable from retention. In practical deployments, especially on edge and federated systems, these issues compound because shift is often client-specific, temporally evolving, privacy-constrained, and only partially observable. We therefore situate our work relative to prior research in continual learning, adaptation under distribution shift, and distributed learning under shift and drift.

2.2.1 Catastrophic Forgetting and Continual Learning

Catastrophic forgetting is the central obstacle in continual learning, referring to the sharp degradation of performance on previously learned tasks after training on new data [150, 61]. Early deep continual learning methods can be broadly grouped into rehearsal-based, regularization-based, and parameter-isolation approaches. Rehearsal-based methods retain or regenerate a small set of past examples and interleave them with current data, thereby approximating joint training under bounded memory. Representative works include Learning without Forgetting (LwF), which uses distillation from the previous model without storing old data [123]; iCaRL, which combines exemplar memory, nearest-class-mean classification, and distillation for class-incremental learning [182]; Gradient Episodic Memory (GEM) and its more efficient variant A-GEM, which constrain gradient updates to avoid increasing loss on past tasks [139, 38]; and replay-based online methods such as Experience Replay, MIR, and Dark Experience Replay, which showed that strong baselines often emerge from careful buffer management and logit-level reuse of past knowledge [187, 9, 31]. More recent empirical studies have reinforced that replay remains one of the strongest and most robust families of methods across class-incremental and domain-incremental settings, especially when task boundaries are weak or absent [145].

Regularization-based methods instead preserve past knowledge by penalizing updates to pa-

rameters deemed important for previous tasks. Elastic Weight Consolidation (EWC) introduces a Fisher-information-weighted quadratic penalty, encouraging stability on parameters that are salient for prior tasks [95]. Synaptic Intelligence (SI) and Memory Aware Synapses (MAS) refine this idea by estimating parameter importance online from trajectory-based or sensitivity-based signals, thereby reducing the need for task-specific second-order approximations [243, 8]. These methods are attractive in memory-constrained settings because they do not require storing raw data, but they typically struggle when the incoming stream induces large representational shifts or when task identities are ambiguous. A third line of work allocates disjoint sub-networks or masks to different tasks, as in PackNet, thereby reducing interference through parameter isolation rather than explicit retention of old data [146]. While such methods can be highly effective when task boundaries are known and model capacity is sufficient, they are often less natural in open-ended or domain-drifting streams where future task structure is unknown.

An important trend in recent continual learning research is the move from task-incremental benchmarks toward more realistic online, class-incremental, and domain-incremental settings. In these regimes, forgetting is not caused only by sequential optimization but also by representation drift induced by changing domains, label frequencies, and feature support. Accordingly, several recent works frame continual learning itself as adaptation under non-stationarity rather than as sequential task memorization. This shift is particularly relevant for our setting, where the model must remain plastic enough to track drift while retaining sufficient stability to avoid collapse on previously observed distributions.

2.2.2 Adaptation under Distribution Shift: Domain Adaptation and Test-Time Adaptation

A second relevant line of literature studies how models adapt when train and test distributions differ. Unsupervised domain adaptation (UDA) traditionally assumes access to labeled

source data and unlabeled target data during adaptation. Canonical approaches align feature distributions across domains, often adversarially, as in Domain-Adversarial Neural Networks (DANN) [66], or through normalization/statistics alignment, as in Adaptive Batch Normalization (AdaBN) [121]. Source-free domain adaptation relaxed this assumption by adapting a source-trained model to a target domain without revisiting source samples; SHOT is a prominent example that performs hypothesis transfer using information maximization and pseudo-labeling [127]. Although these methods were not originally developed for continual learning, they directly inform settings where the model must adapt under privacy or storage constraints.

Test-time adaptation (TTA) pushes this idea further by updating the model only at inference time, using the unlabeled test stream itself. Test-Time Training (TTT) adapts using self-supervised auxiliary objectives at test time [203], while Tent showed that even entropy minimization on batch-normalization parameters can substantially improve robustness under corruption and moderate shift [220]. However, standard TTA methods often assume a static target domain. In non-stationary deployment, target distributions evolve over time, so naive self-training can accumulate errors and itself induce catastrophic forgetting. Continual Test-Time Adaptation (CoTTA) explicitly addresses this setting by combining teacher averaging, augmentation-averaged predictions, and stochastic restoration to preserve source knowledge while adapting to a changing target stream [222]. This line of work is especially relevant because it makes explicit that continual adaptation and forgetting are coupled phenomena: adaptation is necessary under drift, but adaptation itself can erase previously useful behavior.

Continual domain adaptation and domain-incremental learning sit between classical CL and TTA. Here the goal is not merely to generalize across a fixed source-target gap, but to remain effective across a sequence of evolving domains with limited memory. Recent work explores combinations of replay, distillation, and representation anchoring for such settings, emphasizing that domain drift can move decision boundaries and destabilize learned features

even when label semantics remain fixed [192, 142]. Compared with task-incremental continual learning, these settings better capture real deployments in which environments change gradually and repeatedly rather than being partitioned into neatly separated tasks.

2.2.3 Distributed, Edge, and Federated Learning under Shift and Drift

The third line of work concerns distributed learning under heterogeneous and drifting data, especially in federated and edge settings. Federated learning was popularized by FedAvg, which aggregates client-side updates without centralizing raw data [151]. While effective under moderate heterogeneity, standard federated optimization is known to degrade under strong client non-IIDness, temporal drift, or locally evolving class support [115, 91, 183]. These problems become more severe when local devices must continually adapt their models from streaming data while preserving previously learned functionality.

Recent work has therefore begun to merge continual learning with federated learning. A central observation is that non-IID data exacerbates forgetting because local adaptation on each client can push the shared model toward client-specific optima that overwrite previously consolidated knowledge. In federated class-continual learning, TARGET alleviates forgetting without storing past real data by combining model-level distillation with generator-based synthetic replay [244]. Other work studies rehearsal-free alternatives for privacy- or memory-sensitive deployments; for example, FedSSI extends synaptic-intelligence-style regularization to heterogeneous continual federated learning, aiming to preserve prior knowledge without local sample buffers [122]. These approaches reflect a broader design tension between privacy, memory, communication, and retention.

Concept drift in federated systems has also been studied more directly. CDA-FedAvg augments FedAvg with drift detection and adaptation mechanisms, showing that static federated training is often inadequate when local data distributions evolve over time [36]. At the edge,

continual adaptation may also exploit spatial or temporal structure across devices. FedSTIL, for example, studies lifelong person re-identification on distributed edges and distills informative spatial-temporal knowledge across clients, illustrating that drift-aware collaboration can improve both adaptation and communication efficiency [247]. More broadly, recent surveys frame federated continual learning for Edge-AI as a distinct research area encompassing class-, domain-, and task-continuous regimes, with open problems in resource-awareness, privacy-preserving replay, client personalization, and hardware-algorithm co-design [225].

Taken together, the literature suggests that continual learning under distribution shift cannot be adequately addressed by retention mechanisms alone or adaptation mechanisms alone. Rehearsal and regularization help preserve prior knowledge, domain and test-time adaptation help track new environments, and federated/edge methods address privacy and decentralization; yet each family only partially addresses the full problem of learning continuously from heterogeneous, evolving streams. Our work is positioned in this gap, where stability, plasticity, shift robustness, and distributed constraints must be handled jointly rather than in isolation.

2.3 Inference-Time Optimization in Large Language and Vision-Language Models

Inference-time optimization for large language models (LLMs) and vision-language models (VLMs) has rapidly evolved from classical decoding heuristics into a broader family of conditional-computation, retrieval-control, and multi-model orchestration methods. In LLMs, early work largely focused on reducing the sequential bottleneck of autoregressive generation through decoding-time strategies, while more recent work has shifted toward adaptive allocation of compute across layers, tokens, experts, retrieved evidence, and even distinct models. In VLMs, the same agenda appears in a multimodal form, where visual

tokens often dominate the inference cost and therefore motivate aggressive token pruning, sparsification, and routing. We organize the literature into three lines most relevant to inference-time optimization: (i) decoding and single-model adaptive computation, (ii) routing and cascaded multi-model inference, and (iii) retrieval, context compression, and memory selection.

2.3.1 Decoding, reranking, and single-model adaptive computation

A first line of work targets the decoding bottleneck itself. Classical search-based methods such as greedy and beam decoding improve quality but do not fundamentally address the high per-token latency of autoregressive generation. More recent acceleration methods instead reduce the number of expensive target-model passes. Speculative decoding accelerates generation by letting a cheaper draft model propose multiple tokens and letting the target model verify them in parallel, yielding lossless speedups without changing the target distribution [39]. Medusa removes the dependence on a separate draft model by augmenting the backbone with multiple decoding heads that predict several future tokens at once and verify them through tree-based attention [32]. Self-speculative variants further collapse drafting and verification into different portions of the same network. In particular, LayerSkip shows that training with early-exit supervision and progressive layer dropout makes intermediate layers sufficiently predictive to support both early exiting and self-speculative decoding [54].

A second group of methods introduces *adaptive computation* within a single model. Early-exit methods dynamically terminate computation once an intermediate representation is judged sufficient, while dynamic-depth and recursive approaches allocate more layers only to hard inputs or hard tokens. Beyond early exit, recent work explores token-level adaptive depth. For example, Mixture-of-Recursions (MoR) combines parameter sharing with token-wise dynamic recursive depth, allowing different tokens to receive different amounts of computation at inference time [15]. This direction is conceptually aligned with broader

conditional-inference literature in dynamic networks, but LLM-era methods are distinguished by their interaction with autoregressive decoding, KV-cache reuse, and long-context behavior.

Another increasingly important axis is *token pruning* and KV-cache reduction. Rather than spending equal attention on the entire prompt at every step, these methods identify the subset of context most useful for the next-token prediction. Selective Context compresses the input by pruning low-utility text before generation, thereby reducing memory and latency for long-context tasks [119]. LongLLMLingua extends this idea into a prompt-compression framework for long-context settings, explicitly optimizing the density and placement of key information to improve both efficiency and, in some cases, quality [85]. At a finer granularity, LazyLLM dynamically prunes prompt tokens during both prefilling and decoding, allowing different subsets of tokens to remain active at different steps [63]. SnapKV instead compresses the KV cache by exploiting persistent attention patterns across heads, significantly reducing memory growth for long inputs [120]. These methods are particularly relevant for long-context inference, where prompt processing often dominates latency and memory use.

For VLMs, analogous work focuses on the heavy cost of visual tokens. FastV shows that many visual tokens become redundant after shallow layers and can be pruned in a plug-and-play manner with a favorable accuracy–FLOPs trade-off [41]. IVTP further makes pruning instruction-aware by selecting image tokens conditioned on the current prompt, improving the quality-efficiency trade-off for large vision-language systems [81]. More recent work argues that text–visual attention alone is an imperfect importance signal: VisPruner instead uses visual cues and token diversity to preserve salient information under aggressive pruning [248]. Taken together, these results suggest that single-model inference efficiency increasingly depends on *where* compute is spent: over time steps, over layers, and over input tokens.

Decoding-time reranking forms a complementary branch of this literature. In information retrieval and retrieval-augmented generation (RAG), LLMs can serve as powerful rerankers

for candidate passages, but this gain must be balanced against cost. RankGPT shows that properly instructed generative LLMs can act as strong zero-shot rerankers, and that their ranking behavior can be distilled into smaller specialized models for cheaper deployment [202]. This reranking perspective is directly relevant to inference-time optimization because it enables more selective downstream context construction and smaller effective prompts.

2.3.2 Routing, Cascades, MoE-Style Sparsity, and Test-Time Scaling Trade-offs

A second line of research moves beyond a single model and asks how to *route* each query to an appropriate amount of computation. This line is motivated by the observation that easy queries often do not require the strongest available model, while hard queries do. FrugalGPT is one of the earliest and most influential formulations of this idea, proposing budget-aware LLM cascades that route requests across models and stop when the answer is sufficiently strong [40]. Hybrid LLM makes the same principle explicit through quality-aware query routing between a small and a large model, with the operating point adjustable at test time [51]. RouteLLM strengthens this paradigm by training routers from preference data so that inference can dynamically trade off cost and response quality between weak and strong LLMs [160]. More recent work extends fixed-pool routing to dynamic model sets: UniRoute considers settings where unseen models may appear at test time and routes by embedding candidate models into a shared feature space [86].

This literature is closely tied to *query difficulty estimation*. Hybrid LLM explicitly predicts difficulty to decide when escalation is needed, while RouteLLM learns an implicit notion of difficulty through preference supervision. The same principle also underlies cascaded reasoning systems that send only difficult examples to expensive solvers. Recent work pushes this idea further by adding stronger guarantees or finer granularity. C3PO formulates cascade design under probabilistic cost constraints and shows that label-free optimization can

produce cost-controlled cascades for reasoning tasks [213]. CITER moves from query-level to token-level collaboration, routing easy tokens to a small model and hard tokens to a large model during generation [255]. This token-level view blurs the boundary between multi-model cascades and conditional computation inside a model.

Inference-time optimization is also related to sparse *mixture-of-experts* (MoE) computation. Foundational sparse-expert work such as Switch Transformers made token-wise expert routing explicit as a mechanism for scaling model capacity without proportional per-token compute [59]. More recent overviews make clear that the realized efficiency of MoE systems depends not only on sparse activation in principle, but also on routing balance, communication overhead, and hardware scheduling in practice [33]. From the perspective of test-time optimization, MoE routing and multi-model routing are part of the same broader theme: allocate more computation only where it is expected to matter.

A related thread concerns *test-time scaling versus efficiency*. Recent reasoning-oriented models often improve performance by generating longer chains of thought or sampling multiple candidate solutions, but these gains come with sharply rising inference costs. In practice, routing and cascading methods can be interpreted as an attempt to recover some of the benefits of test-time scaling without paying its full cost on every query. Self-speculative decoding, early exiting, cascades, and cheap-to-expensive routing all instantiate the same operational principle: reserve the most expensive computation for the subset of cases where it is necessary. This viewpoint is increasingly important as the field moves from average-case accuracy toward Pareto-efficient deployment.

2.3.3 Retrieval pruning, long-context selection, and memory-aware inference

A third line of work optimizes *what information is presented* to the model at inference time. In RAG pipelines, efficiency depends not only on model choice but also on retrieval depth,

reranking, top-k selection, chunking, and compression. The original RAG formulation established retrieval-augmented generation as a general architecture for coupling parametric and non-parametric memory [105]. RankGPT-style reranking improves the relevance ordering of retrieved candidates before context construction [202], while selective top-k policies reduce both token cost and distraction from marginal evidence. Recent comparisons between RAG and long-context (LC) inference show that this trade-off is nontrivial: when sufficiently resourced, LC models can outperform RAG on average, but RAG often remains much cheaper. Based on this observation, Self-Route dynamically chooses between retrieval-augmented and long-context processing using model self-reflection, achieving a better cost–performance balance than always using either mode alone [124]. This work is especially relevant because it reframes retrieval itself as a routing problem.

Prompt and context compression methods further reduce the amount of retrieved or provided text that must be processed. Selective Context prunes redundant spans before inference [119], while LongLLMLingua provides a more general framework for long-context prompt compression with explicit token budgets [85]. In practice, these methods interact strongly with chunking and filtering choices in RAG: better chunk granularity improves recall, while stronger filtering and compression reduce token cost and mitigate context dilution. Thus, chunking, reranking, top-k selection, and compression should be understood as a joint inference-time optimization problem rather than isolated preprocessing decisions.

Long-context agents extend this problem from one-shot retrieval to *memory selection over time*. MemGPT introduces an OS-inspired memory hierarchy in which relevant information is paged into the active context from external memory, effectively treating context management itself as an inference-time control problem [162]. MemoryBank studies long-term memory with selective retention and forgetting, emphasizing that not all past interactions deserve equal persistence [256]. More recent agentic memory systems such as A-MEM make memory organization itself adaptive by dynamically creating, linking, and updating memory

records for future retrieval [231]. These systems are highly relevant to long-context inference because they replace brute-force context growth with selective memory retrieval and consolidation. In effect, they move inference-time optimization from token pruning within a single prompt to *state pruning across an interaction history.*

2.3.4 Benchmarks vs. Real-World System Constraints

While a large body of work evaluates LLMs and VLMs on standardized benchmarks, there is a growing recognition that such benchmarks often underrepresent the complexity of real-world inference pipelines. Traditional NLP and multimodal benchmarks—such as GLUE [219], SuperGLUE [218], MMLU [76], BIG-bench [200], and multimodal suites like VQAv2 [70] and MM-Vet [238]—primarily measure task accuracy under fixed inputs. These settings assume that all relevant context is already provided and that inference cost is dominated by forward passes through a single model.

However, real-world LLM and VLM systems operate under fundamentally different constraints. In production settings, systems must *retrieve*, *select*, and *compose* context from distributed and heterogeneous sources prior to generation. This includes document stores, vector databases, APIs, and multimodal inputs. Consequently, system performance depends not only on model capability but also on upstream decisions such as retrieval quality, chunking granularity, filtering, reranking, and context budgeting.

Recent work highlights this gap explicitly. For instance, **HELM** [128] and **Holistic Evaluation** frameworks argue that real-world evaluation must incorporate efficiency, robustness, and scenario diversity beyond static benchmarks. Similarly, **Dynabench** [94] introduces dynamic, adversarial data collection to better reflect evolving deployment conditions. In the retrieval-augmented setting, **KILT** [169] evaluates knowledge-intensive tasks with explicit retrieval components, emphasizing that correctness depends on both retrieval and generation.

More recent studies further demonstrate that benchmark conclusions may not transfer to deployed systems. Li et al. [124] show that long-context LLMs can outperform RAG pipelines on certain benchmarks, yet RAG remains significantly more cost-efficient in practice, motivating hybrid routing strategies. Similarly, RAG system design frameworks such as **RAGAS** [56] and **ARES** [188] propose evaluation metrics that jointly assess retrieval quality, answer faithfulness, and generation coherence. These works underscore that evaluating components in isolation (e.g., retrieval recall or generation accuracy) fails to capture end-to-end system behavior.

In multimodal settings, this discrepancy is even more pronounced. Vision-language benchmarks typically assume fully available visual inputs, whereas real systems must decide *which* images, regions, or video segments to process under compute constraints. Emerging benchmarks such as **MME** [62] and **SEED-Bench** [106] attempt to probe multimodal reasoning, but they still largely abstract away the cost of visual token selection, routing, and compression.

Overall, these findings suggest that inference-time optimization cannot be decoupled from system-level context construction. Benchmarks that ignore retrieval, memory management, and routing may overestimate the benefits of scaling model size or context length. As a result, there is increasing interest in *end-to-end evaluation frameworks* that account for both quality and efficiency in realistic pipelines, where the central challenge is not only generating answers but also deciding *what information to process* and *how much computation to allocate* at test time.

Chapter 3

The Approach: Middleware for the ML Lifecycle

3.1 Chapter Overview

Machine learning research has long concentrated on two well-defined phases of the pipeline: *training*, where models are optimized on collected datasets, and *inference*, where trained models are queried to produce predictions. Both phases have received substantial attention, driving advances in optimization algorithms, generalization theory, model architectures, and scalability [114, 195, 180, 252, 99, 236]. Yet the operational decisions that govern how these phases actually execute in real-world deployments—and the third critical phase that bridges them, *continual adaptation*—remain comparatively underserved [145, 36, 225].

This chapter argues that a dedicated *middleware layer* is essential for governing system behavior across all three phases of the ML lifecycle, particularly in distributed and production settings. This layer sits between data sources, models, and applications, and is responsible for making principled decisions about participant selection, adaptation triggering, input

structuring, and resource allocation. Rather than requiring changes to underlying model architectures, middleware operates by observing system conditions, analyzing them to derive actionable insights, and acting to adapt system behavior accordingly [23, 24, 25, 22].

The remainder of this chapter develops this middleware perspective and positions the contributions of this dissertation—FLIPS (training-time participant selection), ShiftEx (deployment-time continual adaptation), and OptiSeq/VOILA (inference-time decision making)—as concrete instantiations of this common framework [23, 24, 25, 22].

3.2 Motivation

Modern ML systems are no longer trained in isolation and queried from a fixed, stable environment. They are increasingly deployed in distributed settings where data is spread across many participants with heterogeneous distributions, computational resources, and network conditions; where the data encountered at deployment shifts continuously from what was seen at training time; and where inference must be performed under strict latency and cost budgets on queries of widely varying complexity [28, 115, 36, 51, 124].

Standard ML pipelines were not designed to address these operational realities. Training procedures treat participant selection and data aggregation as fixed protocols, ignoring the heterogeneity among participants that drives both the efficiency and the quality of what is learned [100, 23]. Inference pipelines apply a uniform computation strategy regardless of query complexity or resource constraints, sacrificing either efficiency or accuracy unnecessarily [51, 119, 85]. And the phase between training and sustained operation—*continual adaptation*—is handled, if at all, through reactive and ad hoc retraining rather than principled, proactive system policy [36, 225, 24].

This gap is not primarily a modeling problem. Better model architectures do not, by themselves, determine how participants should be selected in a federated round, when a deployed

model should be updated in response to distribution shift, or how much retrieval to perform for a given query. These are *system-level decisions* that require a dedicated layer of logic—a middleware layer—that can observe operating conditions, reason about them, and act accordingly [23, 24, 25, 22].

3.3 The Middleware Layer

We define the middleware layer as a model-agnostic decision component that sits between data sources (or upstream systems) and the ML model (or downstream applications). Its role is to make the operational decisions that arise at each phase of the ML lifecycle—decisions that the model itself cannot or should not make, and that standard pipeline infrastructure currently leaves unaddressed [23, 24, 25, 22].

At a high level, the middleware layer follows a consistent three-stage process:

1. **Observe:** The middleware collects information about the current operating context—such as data characteristics across participants, statistics reflecting model behavior on recent data, or properties of an incoming query—without requiring access to raw user data or modifications to model internals. This information may be derived from labels, representations, model outputs, or system-level metadata depending on the lifecycle phase [23, 24, 25, 22].

2. **Analyze:** The middleware processes the collected information to derive structured, reliable representations suitable for decision-making. This may involve clustering data characteristics across participants, computing divergence metrics to characterize distributional change, or normalizing model outputs to enable meaningful comparisons across candidate inputs. The goal is to transform raw observations into quantities that can drive principled decisions [23, 24, 25, 22].

3. **Act:** Based on the analyzed information, the middleware enforces system-level decisions: selecting which participants to include in a training round, deciding whether and how to trigger model adaptation, reordering input examples, or determining retrieval depth for a given query. These actions adapt system behavior without modifying the underlying model [23, 24, 25, 22].

This observe–analyze–act framework is intentionally general. The specific information observed, the analysis performed, and the action space available differ across lifecycle phases and problem settings, but the structure remains consistent. This consistency is what allows FLIPS, ShiftEx, and OptiSeq/VOILA to be understood as instances of a common design pattern despite operating in very different settings [23, 24, 25, 22].

The middleware layer also operates under constraints that are inherent to real-world distributed ML deployments:

- **Model-agnosticism:** Middleware decisions are grounded in observable system conditions rather than model internals, enabling applicability across diverse ML paradigms and architectures [23, 24, 25, 22].

- **Privacy compatibility:** In federated and distributed settings, the middleware must make decisions using aggregate statistics or derived representations rather than raw participant data, preserving the privacy guarantees of the overall system [151, 28, 23, 24].

- **Efficiency:** The observe–analyze–act cycle must impose minimal overhead relative to model training and inference, remaining lightweight enough to execute continuously in production [23, 24, 119, 85, 22].

- **Compatibility:** Middleware decisions must integrate cleanly with existing training and inference pipelines, requiring no changes to model architectures, training proce-

dures, or serving infrastructure [23, 24, 25, 22].

3.4 Middleware Across the ML Lifecycle

We now describe how the middleware perspective manifests at each of the three lifecycle phases addressed in this dissertation.

3.4.1 Training: Participant Selection under Data Heterogeneity

In federated learning, training proceeds through repeated communication rounds in which a subset of participants contributes local model updates. The choice of which participants to include in each round is a critical system-level decision: naive or uniform selection leads to biased aggregation when participant data distributions are heterogeneous, slowing convergence and degrading the generalization of the resulting model [151, 115, 100].

FLIPS instantiates a training-time middleware layer that governs this selection decision. The middleware observes data distribution characteristics across participating clients, analyzes them through clustering to enforce statistical diversity across rounds, and acts by selecting participant subsets that represent a balanced cross-section of the overall data landscape. This decision logic operates entirely outside the model and integrates with the federated training protocol without modifying the underlying aggregation algorithm or model architecture. The result is improved convergence speed and model quality without centralizing any participant data [23].

3.4.2 Continual Adaptation: Responding to Distribution Shift

Once a model is deployed, the environment in which it operates continues to evolve. Real-world data streams are non-stationary, and a model trained on historical data will degrade as input or label distributions shift over time [36, 46, 225]. Maintaining performance in such

settings requires continuous monitoring of deployment conditions and targeted adaptation when warranted—a capability that standard deployment infrastructure does not provide [36, 24].

This phase of the ML lifecycle—between initial training and sustained operation—is precisely where a middleware layer is most needed and yet most absent in conventional systems. ShiftEx introduces a deployment-time middleware layer that governs adaptation decisions. The middleware observes evolving data characteristics through statistics derived from model behavior on incoming data, analyzes them to detect meaningful distributional change, and acts by triggering targeted model specialization when the current model is no longer suited to the observed distribution. Rather than retraining from scratch or applying uniform updates, this middleware enables efficient, targeted adaptation that preserves performance on previously observed distributions while accommodating new ones [24].

3.4.3 Inference: Decision Making under Resource Constraints

At inference time, particularly in systems built on large language and vision-language models, a range of decisions arise that determine both output quality and resource consumption. How should input examples be ordered to maximize model performance? How much retrieval should be performed for a given query? These decisions must be made per-query, under latency and budget constraints, without labeled feedback [51, 119, 85, 25, 22].

OptiSeq and VOILA instantiate inference-time middleware layers that govern these decisions. OptiSeq observes model behavior under different input configurations, analyzes candidate orderings using a debiased scoring mechanism, and acts by selecting the configuration predicted to yield the best output [25]. VOILA observes an initial model response, analyzes model confidence to estimate whether higher-fidelity retrieval is likely to improve the answer, and acts by deciding retrieval depth on a per-query basis [22]. In both cases, the decision logic is embedded in the middleware layer, leaving the underlying model unchanged and

enabling the system to adapt dynamically to the properties of each query [25, 22].

3.5 Summary

This chapter presented middleware as the essential system layer for governing decisions across the ML lifecycle in distributed and production settings. Machine learning research has developed powerful methods for training and inference in controlled settings, but the operational decisions that arise at each lifecycle phase—how to select participants, when and how to adapt to distribution shift, and how to structure inputs and allocate resources at inference—have lacked principled, systematic treatment. A middleware layer that observes operating conditions, analyzes them, and acts accordingly fills this gap in a model-agnostic, privacy-compatible, and deployment-efficient way [23, 24, 25, 22].

The remaining chapters instantiate this perspective concretely. Chapter 4 presents FLIPS, a middleware system for training-time participant selection in federated learning. Chapter 5 presents ShiftEx, a middleware system for continual adaptation under distribution shift. Chapter 6 presents OptiSeq and VOILA, two middleware systems for inference-time decision making in large language and vision-language model pipelines.

Chapter 4

Robust Training under Data and Platform Heterogeneity

4.1 Chapter Overview

Modern machine learning deployments increasingly operate in decentralized environments where data and compute are distributed across heterogeneous participants. Federated Learning (FL) has emerged as a key paradigm to enable such settings while preserving data privacy. However, real-world FL deployments are fundamentally challenged by two forms of heterogeneity: (i) *data heterogeneity*, arising from non-IID distributions across participants, and (ii) *platform heterogeneity*, arising from variability in compute, communication, and availability across participant devices.

In this chapter, we present FLIPS, a middleware system that addresses both forms of heterogeneity through intelligent participant selection. Consistent with the central thesis of this dissertation, FLIPS leverages *signals exposed by ML systems*—in this case, label distributions—to guide system-level decisions. By transforming these signals into actionable

insights for participant selection, FLIPS improves convergence, accuracy, and efficiency in federated training. We describe the design of FLIPS, situate it within prior work on federated optimization and system-aware scheduling, and provide a detailed empirical evaluation demonstrating its benefits. We conclude by discussing how this signal-driven middleware perspective generalizes to subsequent chapters.

4.2 Background

Federated Learning enables multiple participants to collaboratively train a global model without sharing raw data [151]. Each participant performs local training and shares model updates with a central aggregator, which combines them iteratively. While this paradigm preserves privacy and enables learning from decentralized data, it introduces challenges that are absent in centralized settings. Two of these challenges—data heterogeneity and platform heterogeneity—are the central focus of this chapter and are discussed next.

4.2.1 Data Heterogeneity in Federated Learning

In centralized machine learning, training data is typically shuffled and distributed uniformly across compute nodes so that each mini-batch is approximately representative of the overall distribution. This assumption of independent and identically distributed (IID) data underpins the convergence guarantees of standard optimization algorithms such as stochastic gradient descent (SGD). Federated learning fundamentally violates this assumption: each participant's local dataset reflects the data generated or collected in its own environment, and these environments differ substantially across participants.

The term *non-IID data* in federated learning refers to the condition where the data residing on each participant does not share the same underlying distribution. This heterogeneity manifests in several forms. *Label distribution skew* occurs when participants hold data with different proportions of class labels—for instance, a hospital specializing in cardiology will

have a disproportionate number of cardiac diagnoses relative to a general practice. *Feature distribution skew* arises when the input features themselves differ across participants, even for the same class—for example, due to differences in imaging equipment or environmental conditions. *Quantity skew* refers to imbalances in dataset size, where some participants hold far more training examples than others.

The practical consequences of data heterogeneity on federated training are well-documented and severe. When local datasets differ substantially in their distributions, each participant's locally trained model drifts toward its own data characteristics, a phenomenon known as *client drift* [91]. During aggregation, these divergent local updates pull the global model in conflicting directions, producing a "zigzagging" convergence trajectory that slows optimization and reduces final model quality. Empirical studies have consistently demonstrated that increasing the degree of non-IID heterogeneity—for example, by reducing the Dirichlet concentration parameter α used to partition data—degrades both convergence speed and terminal accuracy, with performance dropping from approximately 69% to 55% in controlled experiments as α decreases from IID to highly skewed regimes [143]. In the most severe cases, FL models trained under non-IID conditions fail to converge to an accurate solution altogether [143].

This degradation is particularly problematic in domains where the data is inherently non-IID. In healthcare, for instance, ECG signals collected from wearable devices exhibit strong label imbalance: the vast majority of recorded heartbeats are normal, while abnormal rhythms are rare and concentrated on devices worn by patients with specific cardiac conditions. Under random participant selection, FL algorithms are statistically biased toward selecting participants with predominantly normal heartbeat data, causing the global model to underperform on the clinically important task of detecting arrhythmias.

4.2.2 Platform Heterogeneity and Stragglers

Beyond statistical differences in data, real-world federated deployments face significant variability in the computational capabilities, network conditions, and availability of participating devices—a challenge collectively referred to as *platform heterogeneity*. The devices participating in an FL job range from powerful datacenter servers to resource-constrained edge devices such as smartphones, wearables, and IoT sensors. These devices differ in their CPU and GPU capabilities, available memory, battery life, and network bandwidth [245].

Platform heterogeneity gives rise to the *straggler problem*. In synchronous FL—the dominant training paradigm—the aggregator must wait for all selected participants to complete local training and submit their model updates before proceeding to the next round. A **straggler** is a participant that takes significantly longer than expected to complete its local training, either due to limited compute resources, network congestion, intermittent connectivity, device failures (such as running out of battery), or competing workloads that consume local resources [37]. Stragglers stall the entire training process: faster participants must wait idle until the slowest participant finishes, dramatically reducing system throughput.

The effects of stragglers extend beyond simple latency. When the aggregator imposes a deadline and discards updates from participants that fail to respond in time, the data held by those participants is effectively excluded from the current round. If straggler behavior is correlated with data characteristics—for example, if participants in remote or resource-constrained settings hold underrepresented data distributions—then systematically dropping their updates introduces a bias in the global model that compounds over rounds. The aggregated model becomes skewed toward the data distributions of faster, more reliable participants, undermining the very diversity that federated learning is designed to harness.

4.2.3 The Compounding Effect of Joint Heterogeneity

Data heterogeneity and platform heterogeneity are not independent problems; they interact and amplify each other's effects. Consider a deployment where some participants hold rare but critical data (e.g., minority class labels in a medical setting) and simultaneously operate on resource-constrained devices. Under random selection, these participants are already unlikely to be chosen in any given round. When they are selected, they are disproportionately likely to become stragglers due to their limited compute resources. The result is that the data distributions most important for model generalization are the most frequently excluded from training—a compound failure that neither data-centric nor system-centric solutions can address in isolation.

This interaction motivates the need for a unified approach that simultaneously accounts for what data participants hold and whether they can reliably contribute to training—the precise gap that FLIPS is designed to fill.

4.3 Related Work

4.3.1 Data Heterogeneity: Clustering and Personalization Approaches

A significant body of work has explored techniques to address data heterogeneity in federated learning by leveraging signals derived from local datasets or model updates. A common direction is to group participants with similar data distributions and train specialized models within each group.

Several approaches cluster participants based on model similarity or update statistics. For instance, prior work clusters local model updates using cosine similarity to identify participants with similar data distributions and periodically retrains cluster-specific models [170, 204]. IFCA [69] assigns cluster identities to participants and jointly optimizes cluster-specific mod-

els using alternating minimization, improving convergence under non-IID settings. Other methods rely on dataset-level signals such as label distributions. FedLabCluster [246] clusters participants based on label presence and performs aggregation within each cluster to improve convergence for cluster-specific models. Related work explores sharing encoded representations across participants and applying K-Means clustering to group datasets while preserving privacy [113]. FedCBS [245] introduces a quantitative measure of class imbalance to guide grouping and selection of participants with more balanced data.

Beyond clustering, approaches such as CMFL [221] dynamically filter local updates by comparing them with the global model and discarding those deemed uninformative. Hierarchical federated learning methods further select participants based on statistical divergence measures such as KL divergence between local and aggregated data distributions [50].

While these approaches effectively leverage signals such as gradients, model updates, and label statistics to address data heterogeneity, they primarily focus on *model-level adaptations*—clustering, personalization, or selective aggregation. These techniques often require additional coordination, periodic re-grouping, or assumptions about data similarity across participants. Critically, none of them incorporate awareness of platform heterogeneity into their selection decisions, treating participant reliability as orthogonal to data representativeness.

4.3.2 Platform Heterogeneity: System-Aware and Straggler Mitigation Approaches

A range of system-level techniques have been proposed to mitigate the impact of stragglers and improve training efficiency. Aergia [45] offloads training from slower devices to other participants with similar datasets, reducing latency while maintaining comparable accuracy. Other methods use clustering or hashing techniques to identify redundant or slow participants and selectively discard their updates [96], although such strategies may lead to wasted

computation at the participant level.

Several works explicitly group participants based on system characteristics. FedLesScan [55] categorizes participants into groups such as rookies, active participants, and stragglers, and schedules participation accordingly to reduce training time and cost. Other approaches prioritize faster participants in early training stages to quickly reach a target accuracy, followed by incorporating slower participants to refine the model [185]. Delay-aware aggregation methods assign weights to updates based on arrival times or system performance to mitigate the impact of stragglers [166, 37].

Complementary techniques improve robustness through redundancy and communication efficiency. Gradient coding introduces redundancy in training to tolerate slow or failed participants [189], while communication-efficient frameworks such as FedCS reduce overhead through compression, sparsification, and reconstruction of gradients [135].

These approaches primarily rely on *system-level signals* such as latency, compute capacity, or communication delay to guide participant selection and aggregation. While effective for improving system efficiency, they do not account for the distribution of data across participants. This can result in training dynamics that favor faster or more reliable participants, systematically biasing the global model toward their data distributions at the expense of underrepresented ones.

4.3.3 Limitations and the Case for FLIPS

The preceding discussion reveals a fundamental gap: existing work largely treats data heterogeneity and platform heterogeneity as separate problems. Optimization-based methods address statistical heterogeneity through modified loss functions or aggregation rules, while system-level approaches address resource variability through scheduling and straggler mitigation. However, there is limited work that jointly considers both aspects through a unified

system design.

More specifically, the literature exhibits three key limitations that motivate our approach. First, data-aware methods such as gradient clustering and label-based grouping do not account for participant reliability, meaning that their diversity-aware selections may be undermined by stragglers at deployment time. Second, system-aware methods such as Oort and TiFL do not incorporate distributional signals, meaning they may systematically exclude participants holding rare or underrepresented data. Third, no existing approach leverages the *label distribution*—a compact, privacy-compatible signal that is inherently available at each participant—as the primary basis for participant selection while simultaneously incorporating straggler resilience.

FLIPS is designed to address all three limitations. By clustering participants based on their label distributions and performing diversity-aware selection with straggler compensation, FLIPS ensures that the global model benefits from representative data in every round, even in the presence of unreliable participants.

4.4 FLIPS: Signal-Driven Middleware for Federated Learning

FLIPS is designed as a middleware layer that sits between the FL training process and the underlying distributed infrastructure. Its core objective is to improve training efficiency and robustness by transforming signals from participant datasets into actionable decisions for participant selection.

4.4.1 Leveraging Label Distribution as a Signal: Finding Similar Parties

The key insight behind FLIPS is that the label distribution of a participant's dataset provides a compact representation of its data characteristics. Each participant p_i with dataset d_i is represented by a label distribution vector $ld_i = \{l_1, l_2, \ldots, l_g\}$, where l_j is the number of data points for the j-th label present at the participant and g is the total number of labels. The set of label distributions for all N participants is denoted as $LD = \{ld_1, ld_2, \ldots, ld_N\}$.

FLIPS uses this signal to identify groups of participants with similar data characteristics by solving a clustering problem over label distribution vectors. Formally, the objective is to find k disjoint subsets $L_1, L_2, \ldots, L_k$ of LD that minimize the ratio of intra-cluster distance to inter-cluster distance:

$$\text{minimize}_{S_m} \frac{1}{k} \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{\Delta(L_i) + \Delta(L_j)}{\delta(L_i, L_j)}, \quad i \neq j \tag{4.1}$$

where $\Delta(L_i)$ is the average Euclidean distance between all objects in set L_i and $\delta(L_i, L_j)$ is the average Euclidean distance between objects in sets L_i and L_j. This problem is a subset enumeration problem known to be NP-complete [212]. FLIPS employs K-Means clustering with *kmeans ++* initialization [13] as an efficient heuristic, yielding a k-partition $C = (C_1, C_2, \ldots, C_k)$ that minimizes:

$$\text{minimize} \sum_{x=1}^{N} \sum_{y=1}^{k} \omega_{xy} \|ld_x - c_y\|^2 \tag{4.2}$$

where ω_{xy} is a binary assignment variable and c_y is the centroid of cluster C_y. The optimal

number of clusters k is determined using the Davies-Bouldin index (DBI) [47], with the elbow point in the DBI curve identifying the value of k that best balances cluster cohesion and separation:

$$k_{\text{optimal}} = \arg\min_{k} \left| \frac{dbi(k) - dbi(k-1)}{dbi(k-1)} \right| \tag{4.3}$$

We opted for K-Means due to its simplicity and lower time complexity of $O(NkI \cdot d)$, where N is the number of data points, k is the number of clusters, I is the number of iterations, and d is the number of dimensions. K-Means++ initialization has been demonstrated to scale to millions of data points [17, 156].

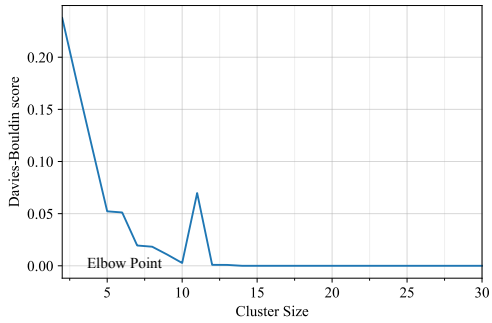


Figure 4.1: Elbow point determination for optimal k

4.4.2 Intelligent Participant Selection

Once clusters, $\mathcal{C}$, are formed, FLIPS performs participant selection in a manner that ensures equitable representation across clusters. In each training round, participants are selected in a round-robin fashion across clusters, cycling through clusters and selecting one participant at a time from each until the required number of participants N_r is reached. FLIPS maintains min-heaps to track the number of times each participant and each cluster has been selected, ensuring both inter-cluster diversity and intra-cluster fairness. This is detailed in Algorithm 1.

This approach directly addresses the core limitation of random selection: under random selection with non-IID data, some rounds will consist entirely of participants with similar data distributions, causing class imbalance in the aggregated update and biasing the global model toward overrepresented classes. By guaranteeing that each round draws from diverse clusters, FLIPS ensures that the gradient updates reflect the full breadth of the data distribution, promoting faster and more stable convergence.

4.4.3 Straggler-Aware Selection and Overprovisioning

To address platform heterogeneity, FLIPS incorporates a straggler-aware selection mechanism. When a participant fails to submit its model update within the allotted time, FLIPS records the participant as a straggler and identifies the cluster to which it belongs. In subsequent rounds, FLIPS overprovisions by selecting $\lfloor strg \times N_r \rfloor$ additional participants, where $strg$ is the running average straggler rate. Critically, these additional participants are drawn from the clusters most affected by stragglers—specifically, the clusters that had the highest number of straggling participants in the previous round.

This cluster-aware overprovisioning is what distinguishes FLIPS from naive overprovisioning strategies. Standard approaches (such as Oort's 1.3× overprovisioning) select additional participants without regard to their data distributions, meaning the replacement participants may not compensate for the data diversity lost when a straggler drops. FLIPS ensures that when a participant from a given cluster fails, it is replaced by another participant from the same cluster, preserving the distributional balance that the selection algorithm was designed to achieve.

4.4.4 Privacy-Preserving Middleware Design

FLIPS introduces two additional pieces of sensitive information beyond standard FL: participant label distributions (used for clustering) and cluster membership (used for selection).

Both could be exploited: label distributions reveal the types of data a participant holds, while cluster membership could enable strategic manipulation of the selection process.

To protect these signals, FLIPS executes all clustering and participant selection logic within a Trusted Execution Environment (TEE)—specifically, AMD Secure Encrypted Virtualization (SEV). Each participant establishes a secure channel (e.g., TLS) with the TEE and transmits its label distribution vector. The clustering algorithm runs entirely within the secure enclave, and cluster membership is never revealed to participants. A remote attestation server verifies the integrity of the TEE, and all label distribution data is deleted at the conclusion of the FL job.

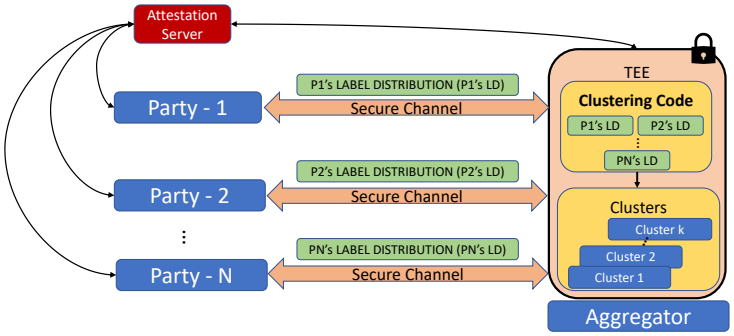


Figure 4.2: End-to-end integrated system design for Private Clustering in FLIPS

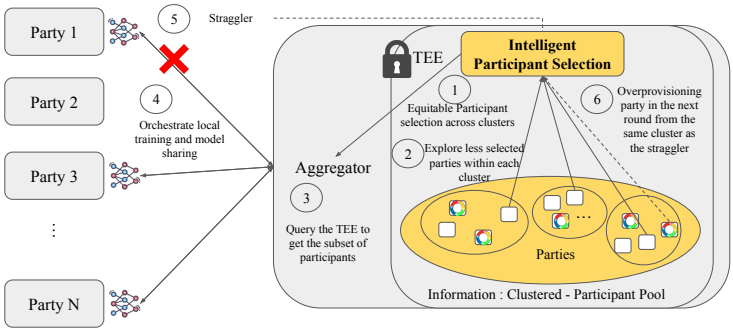


Figure 4.3: Workflow: Intelligent Participant Selection in FLIPS

4.5 Experimental Evaluation

4.5.1 Datasets, Models, and Setup

We evaluate FLIPS on two real-world healthcare datasets that exhibit strong non-IID characteristics, partitioned across 200 participants using Dirichlet allocation at two levels of heterogeneity ($\alpha = 0.3$ and $\alpha = 0.6$).

The **MIT-BIH ECG dataset** [155] comprises digitized electrocardiogram recordings annotated with AAMI labels for arrhythmia identification. The dataset is dominated by normal heartbeats (N class), with abnormal rhythms (S, V, F, Q classes) appearing rarely and concentrated on specific participants. This makes it an ideal testbed for evaluating whether participant selection can improve detection of underrepresented classes. Training uses a 1-D CNN with a learning rate of 0.001 and decay every 20 rounds, for a maximum of 400 rounds.

The **HAM10000 skin lesion dataset** [210] contains 10,015 dermatoscopic images across seven diagnostic categories. The nevus (nv) category dominates, creating a strong class imbalance that mirrors real-world clinical data distributions. Training uses DenseNet-121 with a learning rate of 0.001 and decay every 30 rounds, for a maximum of 400 rounds.

We compare FLIPS against four participant selection strategies: (1) **Random selection** [151], the default in most FL algorithms; (2) **Oort** [100], which prioritizes participants with higher local losses using a utility metric combining statistical and systemic factors; (3) **GradClus** [60], which clusters gradients from participants using hierarchical clustering on gradient similarities; and (4) **TiFL** [42], which groups participants into performance-based tiers and selects from the same tier each round. Experiments are conducted across three FL algorithms—FedYogi, FedProx, and FedAvg—at two participation rates (15% and 20%), with and without stragglers (0%, 10%, 20% straggler rates), yielding a total of 132 experimental configurations.

4.5.2 Results Under Data Heterogeneity

FLIPS consistently outperforms all baseline methods across both datasets, all three FL algorithms, and both levels of non-IID heterogeneity.

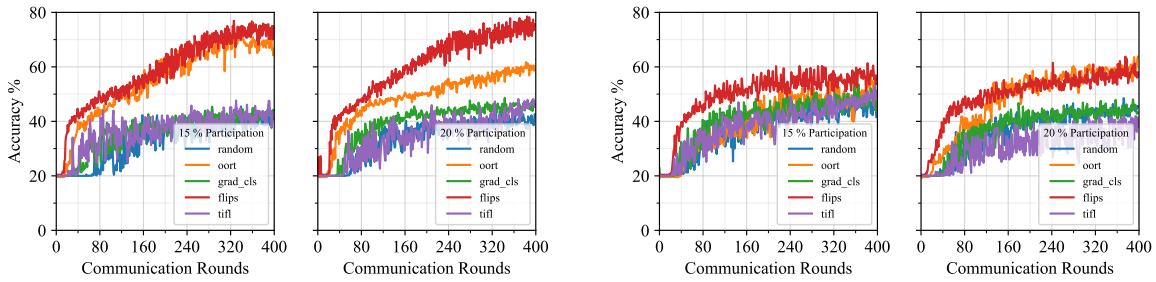
Convergence speed. When evaluating the number of rounds required to reach a target accuracy of 60%, FLIPS demonstrates substantial speedups. Under FedYogi with $\alpha = 0.3$, FLIPS reaches the target in 157–172 rounds on MIT-ECG and 167–190 rounds on HAM10000, while random selection, GradClus, and TiFL all exceed 400 rounds without reaching the target. FLIPS converges 1.08–2.37× faster than Oort when $\alpha = 0.3$ and 1.15–1.86× faster when $\alpha = 0.6$, with similar trends under FedProx and FedAvg.

Peak accuracy. FLIPS achieves the highest terminal accuracy across all configurations. On MIT-ECG with FedYogi and $\alpha = 0.3$, FLIPS reaches 76.92–78.53%, compared to 44.86–45.48% for random selection, 48.21–48.62% for GradClus, 47.67–48.21% for TiFL, and 61.75–71.35% for Oort—an improvement of over 30 percentage points versus random selection and 5–17 percentage points versus Oort. On HAM10000, FLIPS achieves 62.39–66.76%, surpassing random selection by over 20 percentage points and Oort by 2–7 percentage points.

Underrepresented label accuracy. The accuracy gains are most pronounced for underrepresented classes. On MIT-ECG, FLIPS substantially improves detection of arrhythmia categories (S, V, F, Q beats), and on HAM10000, it improves classification of rare lesion types such as basal cell carcinoma (bcc). This confirms that diversity-aware selection directly translates to better representation of minority classes in the global model.

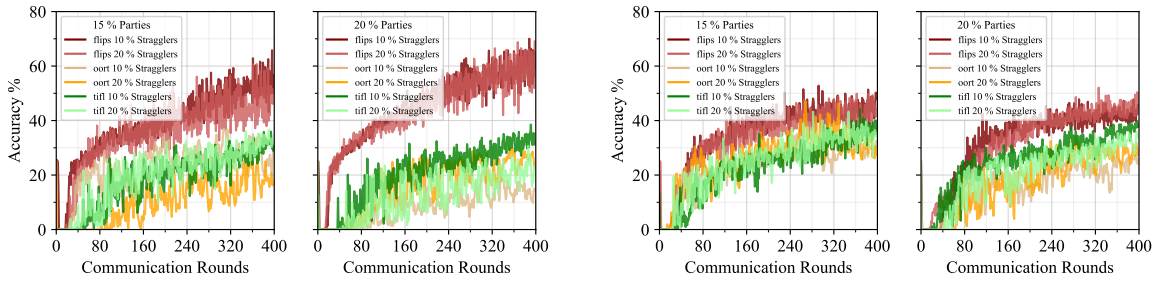
These results connect directly to the limitations identified in Section 3.3. Random selection fails because it cannot guarantee distributional diversity in each round. GradClus, despite also performing clustering, operates on gradients rather than label distributions; since gradient similarity does not directly encode class representation, it fails to ensure balanced label

coverage. TiFL's performance-based tiering groups participants by computational speed rather than data characteristics, so it cannot prevent class imbalance. Oort's utility function partially accounts for statistical value but does not enforce systematic diversity across data distributions, explaining its intermediate performance.



(a) $\alpha = 0.3$ without stragglers (b) $\alpha = 0.6$ without stragglers

Figure 4.4: Convergence plots on the MIT-BIH ECG dataset, FL algorithm: FedYogi



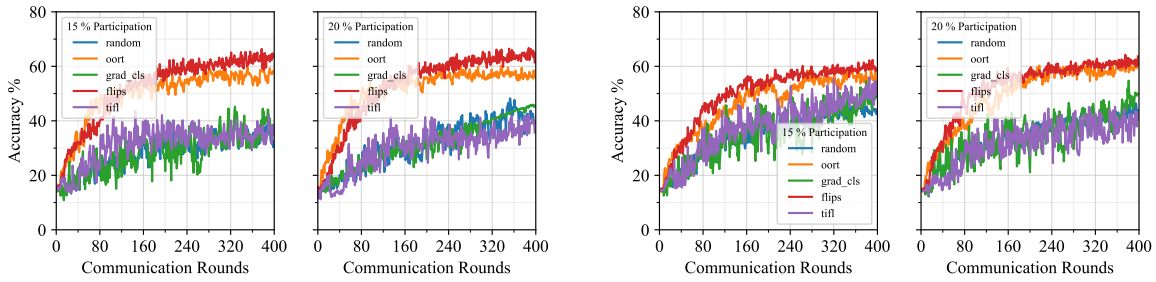
(a) $\alpha = 0.3$ with stragglers (b) $\alpha = 0.6$ with stragglers

Figure 4.5: Convergence plots on the MIT-BIH ECG dataset in the presence of stragglers, FL algorithm: FedYogi

4.5.3 Resilience Under Platform Heterogeneity

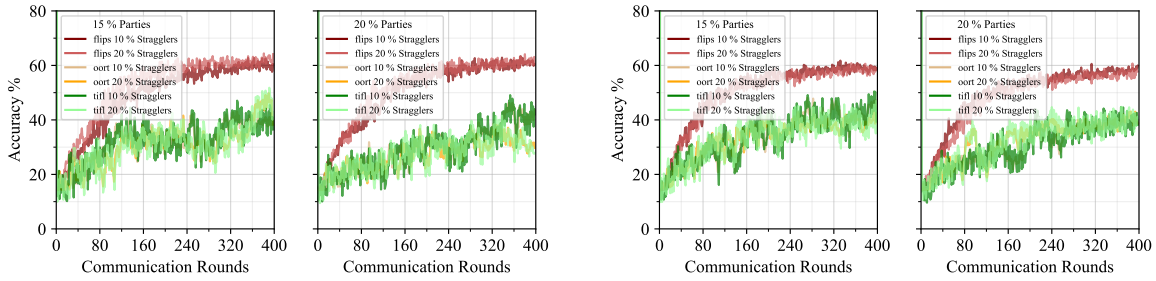
We evaluate FLIPS, Oort, and TiFL under 10% and 20% straggler rates, where stragglers are emulated by dropping the corresponding fraction of selected participants in each round.

Sustained convergence under stragglers. Under a 10% straggler rate on MIT-ECG with FedYogi, Oort and TiFL fail to reach the 60% target accuracy in any of the four settings ($\alpha \in \{0.3, 0.6\}$, participation $\in \{15\%, 20\%\}$), while FLIPS reaches the target in 3 out of 4



(a) $\alpha = 0.3$ without stragglers (b) $\alpha = 0.6$ without stragglers

Figure 4.6: Convergence plots on the HAM10000 (skin lesions) dataset, FL algorithm: FedYogi



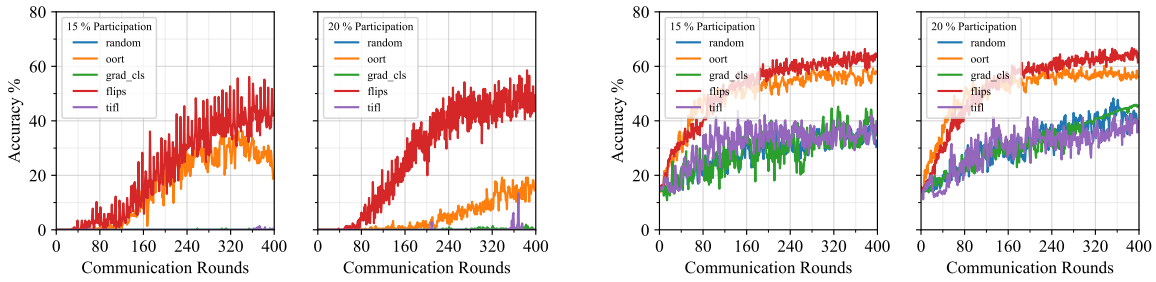
(a) $\alpha = 0.3$ with stragglers (b) $\alpha = 0.6$ with stragglers

Figure 4.7: Convergence plots on the HAM10000 (skin lesions) dataset in the presence of stragglers, FL algorithm: FedYogi

settings, missing only marginally (by 2.8%) in the fourth. Under a 20% straggler rate, the pattern persists: Oort and TiFL fail in all settings, while FLIPS succeeds in 3 out of 4.

Peak accuracy retention. FLIPS attains 74.66% accuracy under 10% stragglers and 74.20% under 20% stragglers on MIT-ECG—only approximately 4 percentage points below the straggler-free scenario. By contrast, Oort drops to 37–56% and TiFL to 44–55%. On HAM10000, FLIPS achieves 60.58–65.76% across straggler conditions, outperforming Oort by 13–26 percentage points and TiFL by 6–17 percentage points.

Faster convergence with stragglers. FLIPS converges 1.2–2.1× faster than both Oort and TiFL across straggler conditions and datasets. This resilience stems directly from FLIPS's cluster-aware overprovisioning: when a straggler drops, the replacement participant



(a) Accuracy of detecting arrhythmia

(b) Accuracy of bcc skin cancer label

Figure 4.8: Convergence curves on underrepresented labels for ECG and HAM10000 datasets

is drawn from the same cluster, preserving the distributional balance that drives convergence. Oort's overprovisioning, by contrast, does not account for which data distributions are lost, so replacement participants may further skew the round's aggregate distribution.

These results validate the hypothesis that addressing data and platform heterogeneity jointly, rather than in isolation, yields substantially more robust training. The straggler-awareness in FLIPS is not a separate mechanism bolted onto a data-aware selection strategy; it is an integral part of the same cluster-based framework, ensuring that the compensatory participants serve the same distributional role as the participants they replace.

Table 4.1: MIT ECG Dataset: Accuracy and Communication savings

Setting		Rounds required to attain 60 % accuracy, FL Algorithm : FedYoGi										
		0 % Stragglers					10 % Stragglers			20 % Stragglers		
α	party %	Random	FLIPS	OORT	GradCls	TiFL	FLIPS	OORT	TiFL	FLIPS	OORT	TiFL
0.3	20	>400	157	373	>400	>400	193	>400	>400	192	>400	>400
0.3	15	>400	172	187	>400	>400	263	>400	>400	214	>400	>400
0.6	20	>400	242	280	>400	>400	>400	>400	>400	326	>400	>400
0.6	15	>400	214	>400	>400	>400	292	>400	>400	>400	>400	>400
Setting		**Highest accuracy attained, FL Algorithm : FedYoGi**										
		0 % Stragglers					10 % Stragglers			20 % Stragglers		
α	party %	Random	FLIPS	OORT	GradCls	TiFL	FLIPS	OORT	TiFL	FLIPS	OORT	TiFL
0.3	20	44.86	78.53	61.75	48.62	48.21	74.66	37.27	50.51	74.20	43.64	44.37
0.3	15	45.48	76.92	71.35	45.40	47.67	71.09	49.19	48.41	67.14	42.34	49.09
0.6	20	48.55	63.79	63.74	48.10	41.97	57.21	42.55	52.18	60.50	47.15	48.20
0.6	15	48.83	61.35	57.47	53.54	53.16	60.55	49.42	54.13	58.18	56.54	54.19

4.5.4 Overhead Analysis

Clustering the label distributions of 200 participants takes approximately 100 ms, with the TEE adding roughly 5% overhead (105.4 ms vs. 100.5 ms). This is negligible relative to the minutes or hours required for each FL round's local training and communication. Clustering

Table 4.2: HAM10000 (Skin lesion) dataset : Accuracy and Communication savings

Setting		Rounds required to attain 60 % accuracy, FL Algorithm : FedYoGi										
		0 % Stragglers					10 % Stragglers			20 % Stragglers		
α	party %	Random	FLIPS	OORT	GradCls	TiFL	FLIPS	OORT	TiFL	FLIPS	OORT	TiFL
0.3	20	>400	167	262	>400	>400	211	>400	>400	202	>400	>400
0.3	15	>400	190	>400	>400	>400	263	>400	>400	190	>400	>400
0.6	20	>400	231	306	>400	>400	313	>400	>400	388	>400	>400
0.6	15	>400	263	>400	>400	>400	265	>400	>400	347	>400	>400
Setting		Highest accuracy attained, FL Algorithm : FedYoGi										
		0 % Stragglers					10 % Stragglers			20 % Stragglers		
α	party %	Random	FLIPS	OORT	GradCls	TiFL	FLIPS	OORT	TiFL	FLIPS	OORT	TiFL
0.3	20	48.26	66.76	61.12	46.48	42.39	63.39	46.28	49.11	64.13	38.25	41.59
0.3	15	41.35	66.41	59.86	45.25	43.70	62.76	43.14	47.09	64.42	49.86	51.81
0.6	20	46.50	62.84	62.36	54.74	45.17	60.58	41.94	44.72	60.71	43.08	44.81
0.6	15	46.55	62.39	59.79	54.66	55.94	61.78	48.13	50.46	60.86	43.46	46.85

is performed once at the start of the FL job (or once per phase if the participant population changes), making its amortized cost effectively zero.

4.5.5 Discussion

The experimental results collectively demonstrate three properties of FLIPS that distinguish it from prior work. First, FLIPS's improvements are *amplified under greater heterogeneity*: the accuracy and convergence gains over baselines are larger when $\alpha = 0.3$ (more non-IID) than when $\alpha = 0.6$ (less non-IID), confirming that signal-driven participant selection is most valuable precisely when it is most needed. Second, FLIPS's benefits are *consistent across FL algorithms*: the improvements hold for FedAvg, FedProx, and FedYogi, indicating that the selection strategy is orthogonal to and compatible with different aggregation and optimization approaches. Third, FLIPS's straggler resilience is *structurally grounded* in its cluster-aware design rather than relying on heuristic overprovisioning, enabling graceful degradation rather than catastrophic failure under platform heterogeneity.

More broadly, the success of FLIPS validates the central thesis hypothesis of this dissertation: *signals generated by machine learning systems can be transformed into actionable insights for system-level optimization.* In FLIPS, label distributions—a byproduct of each participant's data collection process—serve as the basis for a middleware decision that fundamentally alters the dynamics of distributed training. The label distribution is not a training signal in the traditional sense (it is not a gradient or a loss value); rather, it is a structural property

of the data that FLIPS interprets at the system level to orchestrate participant selection.

4.6 Summary and Transition

In this chapter, we introduced FLIPS, a middleware system that addresses data and platform heterogeneity in federated learning through signal-driven participant selection. By clustering participants based on their label distributions and performing diversity-aware selection with cluster-aware straggler compensation, FLIPS achieves 17–20 percentage point accuracy improvements and 20–60% communication cost reductions across real-world healthcare datasets and multiple FL algorithms.

This chapter establishes a key principle that recurs throughout this dissertation: *signals generated by machine learning systems can be transformed into actionable insights for system-level optimization.* In FLIPS, label distributions enable better training dynamics by ensuring each round of federated training draws from a representative cross-section of the participant population.

However, FLIPS operates under an implicit assumption: that the data distributions at each participant remain stationary throughout the training process. In many real-world deployments—particularly those involving streaming data from sensors, cameras, or user interactions—this assumption is violated. Data distributions evolve over time as environmental conditions change, user behaviors shift, and new data sources come online. When such *distribution shifts* occur after training, a model that was well-trained under FLIPS can nonetheless degrade, because the distribution it was trained on no longer reflects the distribution it encounters at deployment time.

This motivates the next chapter, which extends the signal-driven middleware perspective from training to the post-deployment phase.

Algorithm 1 FLIPS Party & Straggler Handling

Participant Side:
RECV global model ($m^{(r)}$) from aggregator
Local model $x^{(r,1)} \leftarrow m^{(r)}$
for $k \in \{1, 2, \ldots, \tau\}$ **do**
    Compute local stochastic gradient $g_i(x^{(r,k)})$
    $x^{(r,k+1)} \leftarrow \text{OPTIMIZER}(x^{r,k}, -g_i(x^{(r,k)}), \eta^{(r)})$
end for
SEND $x^{(r,\tau)}$ to aggregator

Aggregator using FLIPS:
$LD = \{ld_1, ld_2, \ldots, ld_n\}$
$C \leftarrow \text{OPTIMIZED\_CLUSTERS}(LD)$
$H \leftarrow \{\}$ ▷ parties used per cluster
$H_c \leftarrow \text{MIN-HEAP}()$ ▷ clusters used
$H_s^r \leftarrow \{\}$, $H_{sc}^r \leftarrow \text{MAX-HEAP}()$
$count\_strg \leftarrow 0$, $Stragglers \leftarrow False$
Initial model m^1, Parties per round N_r
for $c \in \{1, 2, \ldots, C\}$ **do**
    $c.picks \leftarrow 0$, $\text{INSERT}(H_c, c)$, $h \leftarrow \text{MIN-HEAP}()$
    for $p \in \{1, 2 \ldots c\}$ **do**
        $p.picks \leftarrow 0$, $\text{INSERT}(h, p)$
    end for
    $H[c.id] \leftarrow h$
end for
for $r \in \{1, 2, \ldots, R\}$ **do**
    $S^{(r)} \leftarrow \{\}$
    for $i \in \{1, 2, \ldots, N_r\}$ **do**
        $c \leftarrow \text{EXTRACT\_MIN}(H_c)$, $H \leftarrow H[c.id]$, $p \leftarrow \text{EXTRACT\_MIN}(H)$
        $\text{INCREMENT}(p.picks, 1)$, $\text{INSERT}(H, p)$
        $\text{INCREMENT}(c.picks, 1)$, $\text{INSERT}(H_c, c)$
        Select unique parties: $S^{(r)} \leftarrow S^{(r)} \cup \{p\}$
    end for
    if $Stragglers$ **then**
        for $i \in \{1, 2, \ldots, \lfloor strg \times N^r \rfloor\}$ **do**
            $c \leftarrow \text{EXTRACT\_MAX}(H_{sc}^r)$, $H \leftarrow H[c.id]$
            $p \leftarrow \text{EXTRACT\_MIN}(H)$ where $p \notin H_s^r$
            $S^{(r)} \leftarrow S^{(r)} \cup \{p\}$
        end for
    end if
    SEND $m^{(r)}$ to each $i \in S^{(r)}$
    for each $i \in S^{(r)}$ **do**
        if x_i^r not received **then**
            $H_s^r \leftarrow H_s^r \cup \{i\}$
            $H_{sc}^r \leftarrow H_{sc}^r \cup \{\text{cluster of } i\}$
            $Stragglers \leftarrow True$, $count\_strg \leftarrow count\_strg + 1$
        else
            if $i \in H_s^r$ **then**
                $H_s^r.remove(i)$
                $H_{sc}^r.remove(\text{cluster of } i)$
            end if
        end if
    end for
    if $|H_s^r| = 0$ **then**
        $Stragglers \leftarrow False$
    end if
    Aggregate: $x^{(r)} \leftarrow \frac{1}{N} \sum_{i \in S^{(r)}} n_i x_i^{(r)}$
    $m^{(r+1)} \leftarrow \text{OPTIMIZER}(m^r, x_i^{(r)})$
    $strg \leftarrow \frac{strg \times N^r + count\_strg}{N^r}$
end for

Chapter 5

Continual Adaptation under Distribution Shifts

5.1 Chapter Overview

Federated Learning (FL) is increasingly being deployed in dynamic, real-world environments where data is not static but continuously generated by parties or clients over time. In such settings, models must be updated incrementally as new data arrives, a paradigm referred to as *continual* or *streaming* federated learning [89, 147, 74]. These evolving environments demand efficient mechanisms to handle incoming data, adapt models on the fly, and maintain performance without retraining from scratch.

To support these requirements, FL systems increasingly leverage techniques from traditional stream processing [12, 6]. A central technique is *windowing*, which segments an unbounded stream of party data into finite units, or *windows*, that can be processed independently [6, 137]. This makes it feasible to apply traditional learning algorithms to streaming data. Among windowing strategies, *sliding windows* are the most general form:

windows can overlap, shift, and vary in size. A common special case is the *tumbling window*, which uses non-overlapping, fixed-size windows [240, 137]. This structure allows periodic updates to global models, facilitating continual adaptation to new information while preserving scalability and responsiveness.

A critical challenge in streaming FL environments is that a client's data distributions change over time. As new data arrives, both the input distribution $P(X)$ and the label distribution $P(Y)$ can shift [253]. These distributional shifts, known as *covariate shift* and *label shift* respectively, pose significant challenges for traditional FL approaches that maintain a single global model [235]. When parties experience different distribution shifts at different times, aggregating their model updates can lead to performance degradation and negative knowledge transfer [117]. The global model fails to adapt effectively to the diverse and evolving data patterns across the federation, resulting in poor generalization for parties whose distributions have shifted.

From a middleware systems perspective, streaming FL environments present fundamental challenges in distributed data processing and service orchestration, particularly when dealing with evolving data distributions and shifts that require dynamic system reconfiguration. This chapter introduces ShiftEx, a comprehensive framework for handling both covariate and label shifts in streaming FL using a Mixture of Experts (MoE)-based approach. Each expert model specializes in a distinct covariate regime and is dynamically adapted to evolving label distributions across windows. Our framework detects shifts in real-time, assigns parties to appropriate experts, and mitigates the risk of catastrophic forgetting during adaptation.

5.2 Background and Challenges

Federated Learning (FL) enables decentralized model training across multiple parties while preserving data privacy. Unlike centralized learning, parties train models locally on private

data and share only model updates with a central aggregator. While federated learning has been designed to be privacy-preserving, several attack vectors have been identified, including model inversion attacks that attempt to reconstruct training data from gradients [261], membership inference attacks that determine if specific samples were used in training [152], and poisoning attacks that inject malicious updates to degrade model performance [16]. However, extensive research has developed robust defenses against these vulnerabilities, including differential privacy mechanisms [2], secure aggregation protocols [29], gradient compression techniques [191], and Byzantine-robust aggregation methods [27], effectively mitigating most, if not all, of these attack vectors in practice.

Nevertheless, FL systems deployed in real-world settings face a critical challenge that remains largely unaddressed: party data distributions evolve over time, creating dynamic learning environments that traditional FL approaches struggle to handle effectively.

5.2.1 Distribution Changes in Streaming FL

In windowed streaming FL, distributional changes manifest as differences between consecutive time windows. These changes can occur abruptly, where a party's data distribution shifts dramatically between adjacent windows, or gradually through accumulated changes across multiple windows. Both scenarios pose unique challenges for federated systems, where model aggregation assumes some degree of distributional consistency across participants. **Shift** refers to abrupt, discontinuous changes in the data distribution, often occurring suddenly between consecutive windows and disrupting model performance without warning. Such shifts can be triggered by events like sensor recalibration or sudden changes in class prevalence, and are typically geographically localized and immediate, affecting specific parties or subsets of the FL deployment in a noticeable way [175].

In contrast, **drift** characterizes gradual, continuous changes in data distribution or the underlying data-generating process, unfolding over longer timescales. Examples include seasonal

changes in user behavior or progressive wear in hardware sensors. Rather than being a one-time perturbation, drift consists of a sequence of small shifts that accumulate and degrade model performance over time. Unlike sudden shifts, drift is systemic and harder to detect, often requiring sustained monitoring and adaptation to maintain performance [64]. One effective strategy to mitigate the impact of drift is to adapt the model incrementally as new data arrive, before the accumulation of shifts becomes disruptive. This continual adaptation helps maintain alignment between the model and the evolving data distribution, thereby improving robustness to long-term drift [140].

The federated setting amplifies these challenges in several ways. First, the aggregator cannot directly observe party data to detect shifts or drifts, requiring detection mechanisms on the aggregator side that rely on model updates or limited statistics. Second, parties may experience shifts at different times and rates, making synchronized adaptation difficult. Third, aggregating updates from parties with divergent distributions can lead to negative knowledge transfer, where learning from shifted parties degrades performance on stable ones.

While shift and drift differ in their temporal characteristics and detectability patterns, they fundamentally represent the same underlying challenge: distributional changes that degrade federated model performance. Our framework addresses both types of distributional changes through unified detection and adaptation mechanisms. For clarity and simplicity, throughout the remainder of this chapter, we use the term "shift" to refer to both abrupt shifts and gradual drift, unless the distinction is specifically relevant to the discussion.

5.2.2 Types of Distributional Changes

Covariate Shift occurs when the input distribution $P(x)$ changes while the conditional relationship $P(y|x)$ remains stable. This is common in audio-visual domains where noise and environmental factors affect the appearance of inputs, but not their semantic meaning. For example, an image classifier trained on clear weather images may encounter foggy or rainy

conditions, changing the visual appearance without altering the underlying object categories. In federated settings, different parties may experience covariate shifts due to varying local conditions, device characteristics, or environmental factors. If not addressed, this change in $P(x)$ can lead to reduced model performance despite the label semantics remaining stable.

Label Shift involves changes in the marginal label distribution $P(y)$ while keeping $P(x|y)$ fixed. This commonly occurs in applications like land use classification from satellite imagery, where seasonal or geographic factors change the prevalence of different land types without altering their visual characteristics. In FL systems, label shift can occur when party populations change; for example, in a healthcare federation where disease prevalence varies by region or season.

Concept Shift represents changes in the conditional distribution $P(y|x)$, where the fundamental relationship between inputs and labels evolves. While this is the most challenging form of distributional change, it typically requires labeled data or domain expertise to detect and adapt to, making it less tractable in privacy-preserving FL settings.

Impact on Traditional FL. Traditional FL algorithms (*FedAvg [151], FedProx [115]*), which aggregate model updates into a single global model, struggle in such heterogeneous settings. For covariate shifts, the global model may not generalize to parties with divergent feature distributions. For label shifts, optimization is biased toward dominant labels, degrading performance on less-seen classes. As a result, models trained under such aggregation schemes tend to be brittle and fail to generalize across the diverse distributions encountered during deployment.

5.2.3 The Federated MoE Challenge

A natural way to handle shifts is to allow multiple models to coexist, each specializing in different distributional regimes. This is precisely the insight behind **Mixture of Experts**

(MoE), a machine learning paradigm in which multiple "expert" models are trained to handle different aspects or subsets of the data space. This idea has led to the development of centralized MoE architectures [148, 33]. They typically consist of a gating network that is jointly trained with a fixed set of expert models using full visibility into the global data distribution. This gating function learns to map individual inputs to the appropriate experts by observing how different experts perform on different data or task subsets.

However, this approach presents fundamental challenges in federated learning and in settings with evolving data distributions for several reasons. First, due to privacy constraints, the aggregator cannot access data from parties to train the centralized gating function. Second, a fixed set of experts may be insufficient to handle new or unseen covariate regimes, requiring an architecture that can dynamically train new experts on demand. Third, the participation of individual parties is typically sparse and asynchronous—not all parties are available in every training round—which presents communication challenges and makes it difficult to learn stable routing patterns.

Consequently, we need a method to train a group of expert models in a federated manner such that data never leaves any party. For privacy preservation at inference time, rather than routing data to experts, each party should be associated with an expert model whose parameters are transmitted to that party for local inference.

5.3 Related Work

5.3.1 FL under Distributional Shifts.

Prior work has studied a range of distributional shifts in FL, including inter- and intra-party, external, prior, test-time, label, and concept shifts. Label shift has been tackled via target-aware aggregation [241], dynamic party selection [164], classifier regularization [118], and test-time calibration [230]. Covariate shift has been addressed across multiple axes,

including intra-/inter-party [181], external shift [53], domain-specific shifts [260, 232], and test-time adaptation [205]. However, many approaches rely on fixed groupings or prior knowledge. In contrast, ShiftEx dynamically detects and responds to label, covariate, and test-time shifts through expert specialization, reuse, and reassignment—enabling continual adaptation without static assumptions.

5.3.2 Mixture of Experts in FL.

MoE frameworks have shown promise for personalization and heterogeneity in FL. ShiftEx advances this line by making expert assignment shift-aware and temporal: it dynamically spawns, reuses, and merges experts in response to distributional drift, enabling scalable continual adaptation in real-time federated environments.

5.3.3 Continual and Temporal Adaptation in FL.

To go beyond static training assumptions, recent efforts focus on temporal and streaming FL. Federated continual learning addresses drift while preserving stability and privacy. Distinct from these, ShiftEx offers a modular MoE framework that continuously adapts to both abrupt and gradual changes without relying on explicit drift signals or fixed temporal structures.

5.4 ShiftEx: Shift-Aware Mixture of Experts Framework

This work focuses on covariate and label shifts for several practical reasons. Both can be detected using techniques that preserve FL's privacy constraints—covariate shift through unsupervised methods applied to party model updates or feature statistics, and label shift through party-side label proportion estimation. Additionally, these shifts are well-suited to windowed detection mechanisms, as they often manifest as observable changes between consecutive time windows.

More importantly, covariate and label shifts represent the most common distributional changes in real FL deployments. They can be addressed through specialization strategies where different expert models handle different distributional regimes, providing a natural motivation for mixture-of-experts approaches. This specialization allows the federation to maintain multiple models adapted to different party conditions while avoiding the negative knowledge transfer that occurs when aggregating updates from distributionally diverse parties.

The combination of detectability, prevalence, and amenability to expert-based solutions makes covariate and label shifts the ideal focus for developing practical shift-aware FL systems that can operate effectively in streaming environments.

5.4.1 Our Approach: Party-Level Expert Assignment

We propose ShiftEx, a shift-aware federated learning framework that sidesteps the gating network challenge by operating at the party level rather than the input level. Instead of routing individual inputs to experts, our system assigns experts to parties based on distributional similarity and detected shifts. This party-to-expert assignment approach offers several advantages. First, it preserves privacy since assignment decisions are made using aggregate statistics or model-level information rather than raw data. Second, it reduces communication overhead by maintaining stable party-expert associations over time windows rather than per-input routing decisions. Third, it naturally aligns with the federated training process, where model updates are already aggregated at the party level.

ShiftEx treats covariate and label shifts as complementary signals for expert management. When covariate shift is detected, the system determines whether existing experts can handle the new input distribution or if new expert specialization is needed. When label shift occurs, the system rebalances participant experts to ensure experts maintain diverse and representative training data across label categories. The framework unifies these responses through

coordinated expert assignment and participant rebalancing: expert models are created and specialized to handle distinct input regimes (addressing covariate shift), while participant assignment strategies ensure balanced learning across label distributions (addressing label shift). This dual approach enables robust adaptation to both types of distributional change while maintaining the scalability and privacy benefits of FL.

5.4.2 ShiftEx Architecture

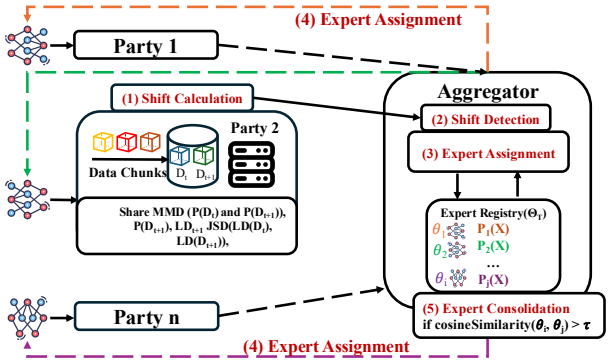


Figure 5.1: System overview of ShiftEx for FL.

ShiftEx differs from traditional continual federated learning (FL) systems by explicitly targeting both covariate and label shifts using an MMD-based shift detector and a distribution-aware Mixture-of-Experts (MoE) model assignment strategy. The system is designed to be modular and can be integrated into existing FL frameworks such as PySyft [262] or Flower [21].

Each client maintains a local dataset and runs a stream processing engine (e.g., Apache Kafka [68] or Flink [35]) to collect, ingest, and preprocess incoming data streams, storing them in a local database. This setup resembles standard streaming data platforms. ShiftEx extends this setup by incorporating a shift detection module that computes latent feature representations and label distributions from local data across consecutive windows. Covariate shift is identified using Maximum Mean Discrepancy (MMD) on latent representations, while label shift is detected via Jensen–Shannon Divergence (JSD) on label histograms. After

computing these statistics, clients transmit only the aggregated embeddings, label distributions, and corresponding MMD and JSD scores to the aggregator. MMD and JSD were selected because they are non-parametric and lightweight, introducing minimal overhead; however, the framework itself is detector-agnostic and can readily accommodate alternative choices if desired.

On the aggregator side, ShiftEx first uses a set of precomputed thresholds to detect covariate and label shift and maintains a registry of expert models, each tagged with its corresponding covariate regime, represented by aggregated latent embeddings from past windows. At window $t = 0$, the registry contains only a single expert. In subsequent windows, the aggregator uses the MMD scores to match each party (or group of parties) to the most appropriate expert. Specifically, parties are first clustered based on similarity in their latent feature distributions. Each cluster is then (1) assigned to an existing expert from the registry if the MMD between the cluster's aggregated representation and the expert's profile is below a threshold, or (2) set up to train a new expert model using a FL process among its members if no suitable match is found. For training, we employ FLIPS [23], which further clusters parties based on their label histograms to promote class-balanced training.

A key distinction from centralized MoE systems is that expert assignment in ShiftEx occurs at the client level rather than at the granularity of individual inputs. Each client is associated with the most suitable expert (or, if necessary, a newly created one) based on latent statistics derived from its local data distribution. At inference time, the process is straightforward: the client simply uses the parameters of the expert model it was previously assigned and locally trained with. This avoids the need for a centralized gating network, ensures that inference is always well defined, preserves privacy by preventing raw inputs from being routed to a server, and reduces overhead by eliminating per-sample routing decisions.

Overall, this architecture allows ShiftEx to be flexibly deployed on top of any existing FL framework, enabling robust, adaptive training in the presence of streaming data shifts while

preserving modularity, scalability, and compatibility with existing systems.

5.4.3 Bootstrap Phase with FLIPS

At system initialization, no global model exists, requiring all parties to collaborate in training an initial baseline model. To ensure representative and balanced training, ShiftEx employs FLIPS (Federated Learning with Intelligent Participant Selection) for participant selection during this bootstrap phase [23]. FLIPS performs label distribution clustering by analyzing the label distribution of each party's data a priori, grouping parties with similar label proportions into clusters. During federated training rounds, FLIPS ensures that participants are selected such that each cluster is equitably represented, preventing bias toward parties with overrepresented label distributions and promoting a more robust initial global model.

This label-aware participant selection is particularly crucial for ShiftEx because the quality of the initial global model affects subsequent shift detection and expert assignment decisions. By ensuring the bootstrap model is trained on representative data across different label distributions, FLIPS provides a stable foundation for detecting when parties begin to experience distributional shifts that warrant expert specialization. The clustering information from FLIPS also provides valuable baseline statistics about party label distributions, which ShiftEx later uses to detect label shift and inform party-to-expert assignment decisions.

5.5 ShiftEx – Party Side Operations

Each participant processes its streaming data using windows of fixed duration. These windows can be of tumbling or sliding in nature, segmenting the continuous data stream into time periods. The system supports both tumbling and sliding window configurations, with the system responding when the shift is greater than a predefined threshold. This windowing strategy ensures synchronized training cycles across all parties while enabling periodic model updates as new data arrives. Within each window, parties accumulate training data that

will be used for local model training and subsequent aggregation.

Algorithm 2 Party-Side: Shift Detection

Require: Current local dataset $\mathcal{D}_t^c = \{(x_i, y_i)\}$, Previous Dataset $\mathcal{D}_{t-1}^c$, Thresholds $\delta_{\text{cov}}, \delta_{\text{label}}$
Ensure: Covariate profile $P_t^c(X)$, label histogram $\mathbf{y}_t^c$, covariate shift Δ_{cov}, label shift Δ_{label}
Compute feature embeddings $\phi(x_i)$ for all $x_i \in \mathcal{D}_t^c$
Estimate $P_t^c(X)$ as the distribution over embeddings
Compute normalized label histogram $\mathbf{y}_t^c$
if $\mathcal{D}_{t-1}^c$ exists **then**
    $\Delta_{\text{cov}} \leftarrow \text{MMD}(P_t^c(X), P_{t-1}^c(X))$
    $\Delta_{\text{label}} \leftarrow \text{JSD}(\mathbf{y}_t^c, \mathbf{y}_{t-1}^c)$
else
    $\Delta_{\text{cov}} \leftarrow 0$, $\Delta_{\text{label}} \leftarrow 0$
end if
Transmit $\{\bar{P}_t^c(X), \mathbf{y}_t^c, \Delta_{\text{cov}}, \Delta_{\text{label}}\}$ to the aggregator

5.5.1 Covariate Shift Detection via Maximum Mean Discrepancy

One of the objectives of ShiftEx is to identify the parties that have undergone covariate and/or label shifts at a given time instant t. After the conclusion of a federated learning round through FLIPS as described above, parties process their local datasets $\mathcal{D}_t^c = \{(x_i, y_i)\}$ through their current model at timestep t to obtain feature embeddings: $P_t^c(x_i) = f_\theta(x_i)$

These embeddings compress raw inputs into a compact, task-relevant feature space. If this latent representation changes substantially from one window to the next, the parties can ascertain that the *underlying distribution* has drifted – even if they never look at the raw pixels or tokens themselves. This makes the covariate shift detection model-, data-, and label-agnostic because raw data can always be translated into a latent representation.

To assess changes in the input distribution, parties compute the Maximum Mean Discrepancy (MMD) between the current and previous window's embeddings: $\text{MMD}(P_t^c(X), P_{t-1}^c(X))$. MMD is a non-parametric statistical measure that compares two probability distributions by embedding them into a reproducing kernel Hilbert space (RKHS) and computing the distance between their means in that space [72].

Formally, given two distributions P and Q with samples $\{x_i\}_{i=1}^{n_P}$ and $\{y_j\}_{j=1}^{n_Q}$, MMD is defined as:

$$\begin{aligned} \text{MMD}^2(P, Q) = &\mathbb{E}_{x,x' \sim P}[k(x, x')] \\ &+ \mathbb{E}_{y,y' \sim Q}[k(y, y')] \\ &- 2\mathbb{E}_{x \sim P, y \sim Q}[k(x, y)] \end{aligned} \tag{5.1}$$

where $k(\cdot, \cdot)$ is a positive definite kernel function (we use the RBF kernel $k(x, y) = \exp(-\gamma \|x - y\|^2)$).

The key advantage of MMD is that it provides a principled way to detect distributional differences without requiring distributional assumptions or labeled data. It operates directly on feature representations, making it ideal for privacy-preserving covariate shift detection in federated settings. A high MMD value ($> \delta_{\text{cov}}$) indicates a significant change in the feature distribution, signaling covariate shift.

5.5.2 Label Shift Detection via Jensen-Shannon Divergence

To detect label distribution changes, we employ Jensen-Shannon Divergence (JSD) [153], a symmetric and bounded measure for comparing discrete probability distributions. Unlike the asymmetric Kullback-Leibler divergence [98], JSD is symmetric ($\text{JSD}(P||Q) = \text{JSD}(Q||P)$) and always finite, making it robust for comparing label distributions that may have non-overlapping support.

For two discrete distributions P and Q, JSD is defined as:

$$\text{JSD}(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M)$$

where $M = \frac{1}{2}(P + Q)$ is the average distribution, and D_{KL} denotes Kullback-Leibler divergence. In our context, we compute JSD between normalized label histograms $\hat{\mathbf{y}}_t^c$ and $\hat{\mathbf{y}}_{t-1}^c$, where $\hat{\mathbf{y}}_t^c[i] = \frac{|\{y \in \mathcal{D}_t^c : y = i\}|}{|\mathcal{D}_t^c|}$ represents the proportion of samples belonging to class i in window t. JSD values range from 0 (identical distributions) to log(2) (completely disjoint distributions), providing an interpretable measure of label distribution shift.

To summarize, at each time window t, parties compute local statistics to detect shift:

- Covariate shift: $\text{MMD}(P_t^c(X), P_{t-1}^c(X)) > \delta_{\text{cov}}$
- Label shift: $\text{JSD}(\hat{\mathbf{y}}_t^c, \hat{\mathbf{y}}_{t-1}^c) > \delta_{\text{label}}$

Parties transmit the computed statistics $\{\text{MMD}_t^c, \text{JSD}_t^c, P_t^c(X), \hat{\mathbf{y}}_t^c\}$ to the aggregator, enabling privacy-preserving shift detection without sharing raw data.

5.6 ShiftEx – Aggregator Side Operations

On the aggregator side, ShiftEx orchestrates the specialized expert creation, assignment, and consolidation using the shift metrics received from parties. The aggregator maintains an expert pool initialized with a single expert at $t = 0$, and dynamically expands or consolidates this pool based on detected distributional shifts across consecutive windows.

5.6.1 Shift Detection and Party Clustering

At each round t, the aggregator receives aggregated latent features $\bar{P}_t^c(X)$ and label histograms $\mathbf{y}_t^c$ along with computed MMD and JSD scores from all participating parties. Using predefined thresholds δ_{cov} and δ_{label}, the aggregator identifies the subset of parties $\mathcal{C}^{\text{shift}}$ experiencing significant distributional change.

If $\mathcal{C}^{\text{shift}} \neq \emptyset$, the aggregator performs K-Means clustering on the shifted parties using their latent feature representations $\bar{P}_t^c(X)$. This clustering groups parties with similar covariate

Algorithm 3 Aggregator-Side: Shift-Aware Federated Learning with Expert Assignment

Require: Initial model θ_0, party set $\mathcal{C}$, thresholds $\delta_{\text{cov}}, \delta_{\text{label}}, \tau, \epsilon$, minimum cluster size γ
Ensure: Final expert pool $\Theta_T = \{\theta_T^{(1)}, \ldots, \theta_T^{(K)}\}$

// Initialize expert pool and assign to all parties
Initialize expert pool $\Theta_0 \leftarrow \{\theta_0\}$; assign $M_0(c) \leftarrow \theta_0$ for all $c \in \mathcal{C}$
for each round $t = 1$ to T **do**
    // Receive latent and label statistics from all parties
    Receive $\{\bar{P}_t^c(X), \mathbf{y}_t^c, \bar{P}_{t-1}^c(X), \mathbf{y}_{t-1}^c\}$ from all $c \in \mathcal{C}$
    // Perform shift detection using thresholds
    $\mathcal{C}^{\text{shift}} \leftarrow \{c \in \mathcal{C} \mid \text{MMD}(\bar{P}_t^c(X), \bar{P}_{t-1}^c(X)) > \delta_{\text{cov}} \text{ or } \text{JSD}(\mathbf{y}_t^c, \mathbf{y}_{t-1}^c) > \delta_{\text{label}}\}$
    if $\mathcal{C}^{\text{shift}} \neq \emptyset$ **then**
        // Cluster shifted parties using latent representations
        $\{G_1, \ldots, G_m\} \leftarrow \text{KMeans}(\mathcal{C}^{\text{shift}}, \bar{P}_t^c(X))$
        for all groups G_j **do**
            if $|G_j| \geq \gamma$ **then**
                // Match to existing expert or create a new one
                Aggregate $P_j(X) \leftarrow \bigcup_{c \in G_j} \bar{P}_t^c(X)$
                if $\text{MatchExpert}(P_j(X), \Theta_{t-1}, \epsilon)$ **then**
                    $\theta_k \leftarrow$ closest matching expert
                    $\mathcal{C}_j^{\text{balanced}} \leftarrow \text{FLIPS}(G_j)$
                    $\theta_k \leftarrow \text{FL}(\mathcal{C}_j^{\text{balanced}}, \theta_k)$ // Label-balanced training
                else
                    $\theta_{\text{new}} \leftarrow \text{Clone}(\theta_0)$
                    $\mathcal{C}_j^{\text{balanced}} \leftarrow \text{FLIPS}(G_j, y_c^t)$
                    $\text{SAVE}(\theta_k, P_j(X))$
                    $\theta_{\text{new}} \leftarrow \text{FL}(\mathcal{C}_j^{\text{balanced}}, \theta_{\text{new}})$
                    $\Theta_t \leftarrow \Theta_{t-1} \cup \{\theta_{\text{new}}\}$
                    Assign $M_t(c) \leftarrow \theta_{\text{new}}$ for $c \in G_j$
                end if
            else
                //Cluster too small – trigger local finetuning
                $\text{LocalFineTune}(G_j, M_{t-1}, M_t)$
            end if
        end for
    end if
    //Merge experts with high similarity to prevent redundancy
    for all pairs $(\theta_i, \theta_j) \in \Theta_t$ **do**
        if $\text{ModelSimilarity}(\theta_i, \theta_j) > \tau$ **then**
            $\theta_{\text{merged}} \leftarrow \text{ConsolidateExperts}(\theta_i, \theta_j)$
            $\Theta_t \leftarrow (\Theta_t \setminus \{\theta_i, \theta_j\}) \cup \{\theta_{\text{merged}}\}$
            Update M_t for affected parties
        end if
    end for
end for
return Θ_T

characteristics into clusters $\{G_1, \ldots, G_m\}$, where each cluster represents a distinct distributional regime.

5.6.2 Expert Assignment and Creation

For each cluster G_j with sufficient size ($|G_j| \geq \gamma$), the aggregator follows a two-stage process:

Stage 1: Expert Matching. The aggregator computes the aggregated latent representation $P_j(X) = \bigcup_{c \in G_j} \bar{P}_t^c(X)$ for the cluster and compares it against all existing experts in the pool Θ_{t-1} using MMD distance. If an existing expert θ_k has a covariate profile within threshold ϵ of $P_j(X)$, the cluster is assigned to that expert for continued training.

Stage 2: Expert Creation. If no suitable match is found, a new expert θ_{new} is instantiated by cloning the initial model θ_0. The cluster's aggregated covariate profile $P_j(X)$ is stored alongside θ_{new} in the expert registry for future matching.

5.6.3 Label-Balanced Training with FLIPS

Once parties are assigned to experts, ShiftEx employs FLIPS to ensure balanced label representation during training. Within each expert's assigned party group, FLIPS performs secondary clustering based on label histograms and selects participants in a round-robin manner from each label cluster. This prevents dominant classes from overwhelming the expert's training signal and ensures that the expert maintains robust performance across all label categories.

5.6.4 Expert Consolidation

To prevent unbounded growth of the expert pool under prolonged heterogeneity, ShiftEx periodically consolidates experts with high similarity. After each round t, the aggregator computes pairwise model similarity between all expert pairs $(\theta_i, \theta_j) \in \Theta_t$ using cosine simi-

larity of flattened parameter vectors. If similarity exceeds threshold τ, the two experts are merged through parameter averaging, and all parties previously assigned to either expert are reassigned to the consolidated version. This consolidation step ensures the expert pool remains manageable while preserving specialization where needed.

For clusters below the minimum size threshold γ, parties perform local fine-tuning by continuing to train their previously assigned expert model without aggregation. This preserves personalization for parties experiencing idiosyncratic shifts while avoiding the overhead of creating experts for statistically insignificant clusters.

5.7 Experimental Strategy

Our goal is to evaluate the adaptability of ShiftEx compared to existing FL techniques under streaming/continual learning conditions, where data distributions shift across windows due to natural or synthetic causes. We focus on the model's ability to maintain high accuracy and fast convergence despite changes in input or label distributions.

5.7.1 Datasets

We evaluate our framework on five datasets spanning diverse covariate and label shifts. **FMoW** (Functional Map of the World) [43] contains over a million satellite images labeled by functional use (e.g., hospital, airport, school) across regions and time; we select 10 labels [88]. **Tiny-ImageNet-C** [77] augments Tiny-ImageNet with 15 corruption types at 5 severities, while **CIFAR-10-C** [77] applies the same corruptions to CIFAR-10 for small-scale evaluation. **FEMNIST** [34], a federated variant of EMNIST, includes handwritten characters from hundreds of users, and **Fashion-MNIST** [229] consists of clothing images widely used as a benchmark.

We simulate 200 parties for CIFAR-10-C, FEMNIST, and Fashion-MNIST to capture fine-

grained heterogeneity in non-IID settings. For FMoW, we instead use 50 parties: its high-resolution imagery and naturally diverse covariates (geography, time, land use) already induce strong variability, so fewer parties suffice while ensuring each has enough samples for stable shift detection. In all cases, 20% of parties are sampled per FL round.

5.7.2 Distributional Shifts

We introduce a range of natural and synthetic covariate and label shifts critical for federated learning evaluation. In FMoW, covariate shifts emerge from visual differences across geographic locations, while label shifts reflect changes in land use distributions [88]. These represent natural shifts, as we partition the data across different time windows based on temporal metadata. For Tiny-ImageNet-C, we group corruption types and randomly sample severity levels across time windows to simulate unpredictable, environment-driven covariate shifts. CIFAR-10-C enables similar shift modeling on a smaller scale by using controlled corruptions to emulate weather-related covariate shifts.

For FEMNIST and Fashion-MNIST, we simulate synthetic shifts using PyTorch image transformations (e.g., rotation, scaling, color jitter) to induce covariate shifts, and Dirichlet sampling to generate skewed label distributions across time windows. In each window, 50% of the participating clients retain their previous data distribution, while the remaining 50% receive a new distribution, simulating partial population shift over time.

5.7.3 Models and Baselines

We select architectures standard for each dataset: LeNet-5 [103] for FEMNIST and Fashion-MNIST, DenseNet-121 [80] for FMoW, ResNet-18 [75] for CIFAR-10-C, and ResNet-50 [75] for Tiny-ImageNet-C. To detect covariate shifts, we extract encoder-based representations from the penultimate (pre-logit) layer of each model, which captures task-relevant features.

To benchmark ShiftEx, we evaluate it against four prominent FL baselines tailored for het-

erogeneous or streaming scenarios. **FedProx** [115] extends FedAvg [151] with a proximal term to stabilize training under non-IID data but relies on a single global model and lacks any mechanism to detect or adapt to evolving distributions over time. **OORT** [101] optimizes party selection using utility-based scores that balance informativeness and system constraints, yet assumes static utility and ignores temporal shifts in data. **Fielding** [110] re-clusters parties based on label distributions to train balanced experts, but overlooks covariate shifts and does not adapt clusters as party distributions change across windows. **FedDrift** [88] introduces lightweight shift detection by clustering parties based on local loss patterns to assign expert models, but it offers only coarse adaptation and lacks explicit modeling of covariate or label shift dynamics. In contrast, ShiftEx explicitly detects both types of shift, dynamically instantiates or reuses experts, and optimizes shift-aware assignment to maintain accuracy and robustness in continual/streaming FL environments.

5.7.4 Windowing Strategy

We use two schemes matched to dataset scale and shift dynamics. For large datasets (FMoW, Tiny-ImageNet-C), we employ *tumbling* windows: each window is disjoint and drawn from distinct covariate distributions—e.g., different timestamps in FMoW and different corruption types in Tiny-ImageNet-C—thereby inducing strong between-window shifts. For smaller datasets (CIFAR-10-C, FEMNIST, Fashion-MNIST), we use *sliding* windows that allow overlap across windows to capture gradual change.

5.7.5 Evaluation Metrics

To evaluate model performance under distributional shifts, we focus on three complementary metrics: **Accuracy Drop** measures the immediate decline in performance after a shift, indicating the model's brittleness to sudden changes—larger drops imply greater sensitivity. **Recovery Time** captures the number of rounds required to regain 95% of pre-shift performance, reflecting the model's adaptability and convergence speed. **Max Accuracy**

denotes the highest accuracy achieved within a window, serving as a measure of the model's generalization ability under the new distribution.

5.8 Results and Discussion

We evaluate ShiftEx on five diverse datasets, covering both large-scale real-world settings (FMoW, TinyImageNet-C) and widely used academic benchmarks (CIFAR-10-C, FEMNIST, and Fashion-MNIST). We report results over 4 windows for FMoW and CIFAR-10-C, and 5 windows for TinyImageNet-C, FEMNIST, and Fashion-MNIST. The first window (W0) is used for initialization and burn-in, and all subsequent windows (W1 onward) evaluate adaptation to new, shifted distributions.

FMoW												
Tech.	**W1**			**W2**			**W3**			**W4**		
	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max
FedProx	20.39 ± 1.36	44	63.48 ± 1.38	39.57 ± 1.43	>51	52.23 ± 1.69	23.45 ± 2.47	>51	45.07 ± 1.79	18.24 ± 2.51	>51	36.95 ± 2.51
Fielding	21.32 ± 1.29	>51	45.66 ± 1.46	17.07 ± 1.83	>51	48.29 ± 1.57	15.76 ± 1.55	>51	50.57 ± 2.08	16.70 ± 2.18	>51	48.34 ± 2.18
OORT	8.60 ± 1.37	>51	52.11 ± 1.37	7.36 ± 1.39	>51	50.80 ± 1.85	4.68 ± 2.10	>51	50.79 ± 2.10	3.94 ± 2.70	>51	50.06 ± 2.70
FLIPS	22.57 ± 0.64	13	66.68 ± 0.59	14.43 ± 0.76	19	66.23 ± 0.75	11.78 ± 0.91	6	65.76 ± 1.07	10.49 ± 1.09	4	66.14 ± 1.09
FedDrift	18.43 ± 1.37	15	64.25 ± 1.43	26.17 ± 1.63	>51	62.33 ± 1.70	20.84 ± 2.11	>51	59.50 ± 2.11	18.33 ± 1.89	>51	54.13 ± 1.89
CIFAR-10-Corruptions												
Tech.	**W1**			**W2**			**W3**			**W4**		
	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max
FedProx	17.80 ± 1.33	>51	72.56 ± 1.18	15.55 ± 1.48	>51	77.55 ± 1.81	17.01 ± 2.02	>51	75.64 ± 2.02	18.64 ± 2.14	>51	76.35 ± 1.96
Fielding	19.76 ± 1.48	>51	72.25 ± 1.72	16.81 ± 2.16	>51	77.33 ± 1.58	17.24 ± 2.03	>51	75.48 ± 2.03	15.39 ± 1.93	>51	75.54 ± 2.53
OORT	20.03 ± 1.36	>51	72.24 ± 1.28	16.66 ± 1.44	>51	76.92 ± 1.54	18.33 ± 2.06	>51	75.37 ± 2.06	16.39 ± 2.69	>51	75.39 ± 2.69
FLIPS	37.51 ± 0.53	>51	81.14 ± 0.65	34.07 ± 0.83	9	90.86 ± 0.64	39.46 ± 1.02	14	87.77 ± 1.02	32.84 ± 1.09	19	87.74 ± 1.09
FedDrift	32.16 ± 1.60	>51	78.47 ± 1.20	27.01 ± 1.55	18	86.95 ± 1.63	33.15 ± 1.83	18	84.67 ± 1.83	29.73 ± 1.91	23	83.68 ± 1.91

Table 5.1: Comparison of Accuracy Drop (Drop %), Recovery Time (Time in rounds), and Max Accuracy (Max %) across Windows 1–4. Top: FMoW dataset. Bottom: CIFAR-10-Corruptions dataset.

5.8.1 Adaptation Time Improvements

Across all datasets, ShiftEx achieves significantly faster adaptation than all baselines. On the large-scale FMoW dataset, where most baselines fail to recover within 51 rounds, ShiftEx consistently recovers within 1–13 rounds across all windows. In TinyImageNet-C, recovery time drops to 27–29 rounds, whereas other methods plateau with no recovery even after 40 rounds. In CIFAR-10-C, ShiftEx completes adaptation in 9–19 rounds, while baselines

TinyImagenet-C

Tech.	W1			W2			W3			W4			W5		
	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max
FedProx	27.19 ± 1.01	>40	38.76 ± 1.40	15.99 ± 1.53	>40	37.37 ± 1.44	11.49 ± 1.63	>40	34.21 ± 2.12	6.91 ± 2.76	>40	40.01 ± 2.16	11.43 ± 2.39	>40	36.34 ± 2.57
Fielding	29.93 ± 1.28	>40	39.86 ± 1.26	17.11 ± 1.40	>40	38.42 ± 1.52	12.40 ± 1.64	>40	38.75 ± 1.64	11.33 ± 1.91	>40	43.61 ± 2.42	15.05 ± 2.05	>40	40.70 ± 2.05
OORT	29.72 ± 1.13	>40	54.76 ± 1.18	18.87 ± 1.47	>40	52.66 ± 1.47	14.23 ± 2.12	>40	52.30 ± 2.12	10.23 ± 2.21	>40	49.91 ± 2.56	6.01 ± 2.56	>40	50.83 ± 2.56
FLIPS	34.87 ± 0.52	29	62.15 ± 0.61	20.73 ± 0.69	30	60.78 ± 0.74	14.07 ± 0.91	34	59.91 ± 0.91	10.87 ± 0.99	33	59.49 ± 0.99	8.53 ± 1.09	30	59.89 ± 1.09
FedDrift	33.43 ± 1.46	>40	52.94 ± 1.36	20.79 ± 1.49	>40	51.12 ± 1.49	9.76 ± 1.61	>40	53.27 ± 1.61	10.99 ± 1.87	>40	52.75 ± 1.87	9.92 ± 2.21	>40	51.36 ± 2.21

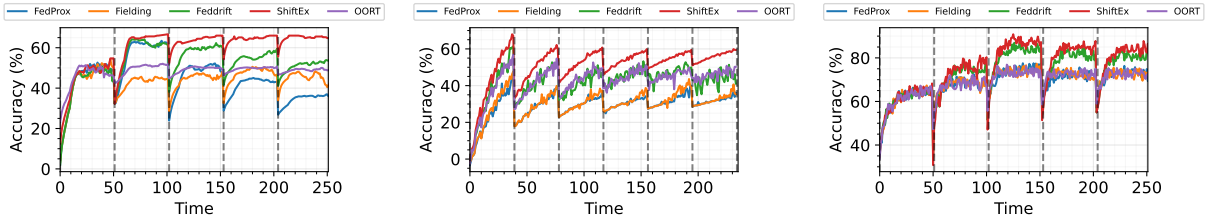
FEMNIST

Tech.	W1			W2			W3			W4			W5		
	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max
FedProx	9.36 ± 1.47	>51	59.20 ± 1.13	11.76 ± 1.19	>51	55.26 ± 1.53	2.24 ± 1.69	>51	57.94 ± 1.81	4.02 ± 1.96	>51	56.38 ± 1.96	1.38 ± 2.68	>51	58.18 ± 2.68
Fielding	9.14 ± 1.42	48	60.18 ± 1.06	11.12 ± 1.57	>51	56.80 ± 1.64	4.68 ± 1.92	>51	57.86 ± 1.92	4.72 ± 2.33	>51	57.68 ± 2.33	3.12 ± 2.59	>51	58.24 ± 2.59
OORT	6.76 ± 1.29	>51	59.98 ± 1.20	9.96 ± 1.45	>51	54.44 ± 1.40	4.40 ± 1.74	>51	53.96 ± 1.95	3.74 ± 2.19	>51	54.78 ± 2.60	0.26 ± 1.89	>51	57.56 ± 2.31
FLIPS	8.90 ± 0.60	42	62.72 ± 0.71	10.62 ± 0.75	35	61.58 ± 0.73	4.32 ± 0.74	14	63.56 ± 0.75	3.94 ± 0.96	1	64.98 ± 0.84	3.12 ± 1.02	0	66.36 ± 1.02
FedDrift	8.14 ± 1.49	40	62.68 ± 1.58	15.04 ± 1.63	50	60.40 ± 1.41	10.56 ± 1.72	44	60.68 ± 1.72	9.82 ± 2.00	>51	59.12 ± 2.00	8.02 ± 2.30	49	60.46 ± 2.30

Fashion MNIST

Tech.	W1			W2			W3			W4			W5		
	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max	Drop	Time	Max
FedProx	11.60 ± 1.15	44	60.14 ± 1.19	9.28 ± 1.80	>51	59.68 ± 1.55	5.96 ± 1.93	>51	59.06 ± 1.83	9.09 ± 2.32	>51	59.42 ± 2.32	9.33 ± 2.31	>51	59.21 ± 2.31
Fielding	8.68 ± 1.34	30	62.16 ± 1.29	11.08 ± 1.45	47	60.64 ± 1.45	5.14 ± 1.78	>51	59.96 ± 1.78	2.80 ± 2.41	19	61.78 ± 2.41	0.34 ± 2.70	0	63.90 ± 2.70
OORT	10.10 ± 1.12	33	61.64 ± 1.08	6.54 ± 1.75	41	61.18 ± 1.87	4.12 ± 1.78	32	61.36 ± 1.78	3.26 ± 1.83	19	60.68 ± 2.49	-0.86 ± 1.74	0	63.04 ± 2.70
FLIPS	7.56 ± 0.56	5	67.10 ± 0.54	8.90 ± 0.75	4	66.32 ± 0.75	6.58 ± 0.74	1	66.35 ± 0.89	3.24 ± 0.76	0	67.68 ± 0.80	1.14 ± 1.03	0	69.00 ± 1.03
FedDrift	8.68 ± 1.29	20	63.74 ± 1.51	8.09 ± 1.73	13	65.86 ± 1.14	15.48 ± 1.50	22	63.46 ± 2.23	13.82 ± 2.50	16	64.62 ± 1.88	15.64 ± 2.51	10	64.06 ± 2.51

Table 5.2: Accuracy Drop (Drop %), Recovery Time (Time in rounds), and Max Accuracy (Max %) across Windows 1–5. Top: TinyImagenet-C. Bottom: Fashion MNIST.

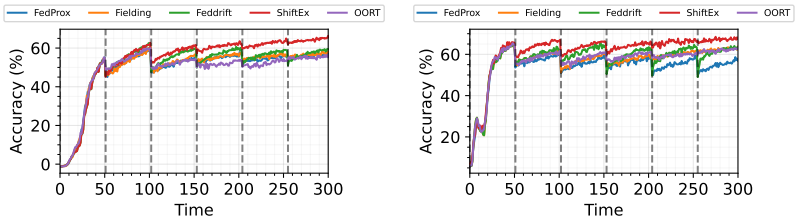


(a) FMoW (b) TinyImageNet-C (c) CIFAR-10-C

Figure 5.2: Convergence plots showing test accuracy over rounds for FMoW, TinyImageNet-C, and CIFAR-10-C.

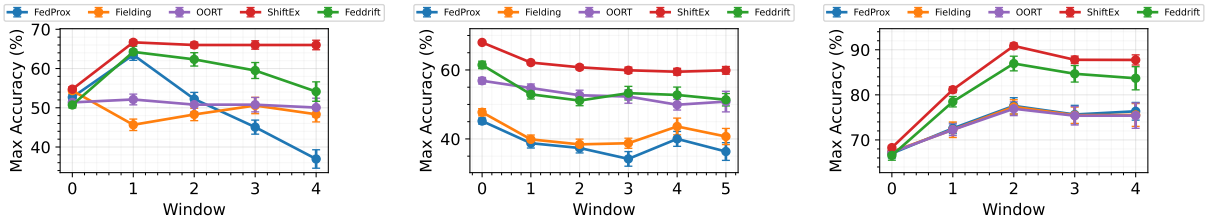
remain stagnant beyond 51 rounds. On FEMNIST and Fashion-MNIST, ShiftEx achieves recovery within 0–42 rounds, including several windows where adaptation completes in under 5 rounds—markedly outperforming methods that require 50+ rounds without convergence.

These results demonstrate that ShiftEx's dynamic expert creation and shift-aware assignment enable rapid adaptation to new distributional regimes. Even in scenarios where all clients simultaneously experience a shift, clustering still groups similar clients together and triggers creation of new experts, ensuring robustness without collapse.



(a) FEMNIST (b) FashionMNIST

Figure 5.3: Convergence plots showing test accuracy over rounds for FEMNIST and FashionMNIST.

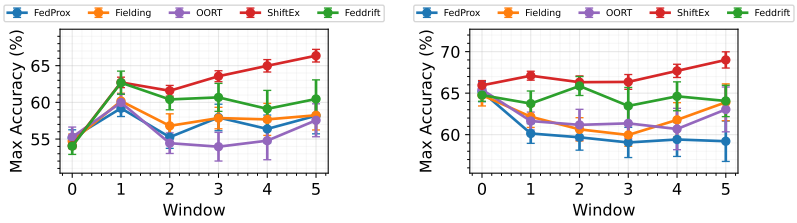


(a) FMoW (b) TinyImageNet-C (c) CIFAR-10-C

Figure 5.4: Maximum accuracy per window for FMoW, TinyImageNet-C, and CIFAR-10-C.

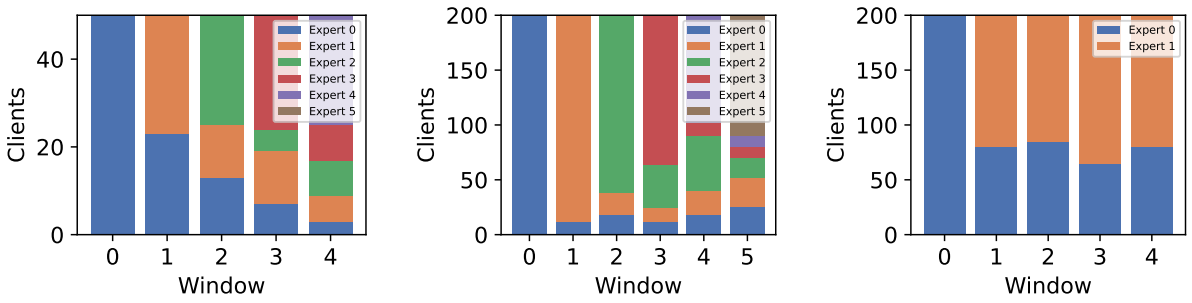
5.8.2 Post-Adaptation Accuracy

On top of faster recovery, ShiftEx maintains the highest peak accuracy across all settings. In FMoW, it reaches maximum accuracies between 66.14% and 66.68% across windows, surpassing all other baselines by 15–30 percentage points. In TinyImageNet-C, ShiftEx sustains accuracy above 59% in every window and peaks at 62.15%, while other methods remain stuck in the 38–54% range. In CIFAR-10-C, ShiftEx achieves 90.86% in W2 and consistently maintains accuracy above 87%, which is significantly higher than the 72–78% ceiling of other approaches. Even on benchmark datasets like FEMNIST and Fashion-MNIST, ShiftEx ends with 66.36% and 69.00%, respectively, outperforming all baselines that either plateau or decline over time.



(a) FEMNIST (b) FashionMNIST

Figure 5.5: Maximum accuracy per window for FEMNIST and FashionMNIST.

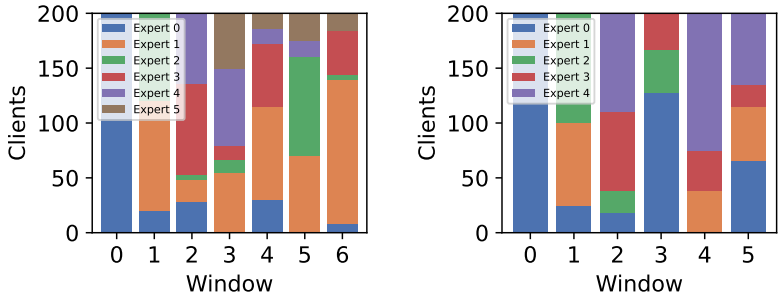


(a) FMoW (b) TinyImageNet-C (c) CIFAR-10-C

Figure 5.6: Expert distribution across clients for FMoW, TinyImageNet-C, and CIFAR-10-C.

5.8.3 Baseline Behavior Analysis

The behavior of baselines provides insight into why ShiftEx succeeds where others struggle. FedProx and Fielding rely on a single global model or static label-aware clustering and therefore struggle under covariate shifts, leading to sharp drops and limited recovery. OORT shows a smaller initial accuracy drop, but this reflects underreaction rather than robustness: its utility-based client selection masks distributional changes, preventing effective adaptation and resulting in limited recovery and lower final accuracy. FedDrift performs better under moderate covariate drift, as seen with controlled corruptions on CIFAR-10-C, because it clusters clients by loss patterns. However, it lacks explicit handling of label shifts and cannot adapt when class distributions skew significantly, as they do in larger datasets such as FMoW and Tiny-ImageNet-C. In contrast, ShiftEx explicitly detects both covariate and label shifts and instantiates new experts when needed, which explains its accuracy dips



(a) FEMNIST (b) FashionMNIST

Figure 5.7: Expert distribution across clients for FEMNIST and FashionMNIST.

followed by rapid recovery and consistently higher post-shift accuracy.

5.8.4 System Overheads

ShiftEx is designed to impose minimal additional memory and computational cost beyond standard FL training, making it practical for real-world deployments. In our ResNet-50 experiments (where a full model is ≈100 MB), each party stores a single d-dimensional latent feature vector (with $d = 2048$ for ResNet-50). On the aggregator side, ShiftEx maintains: (1) expert centroids (5 experts × 2048 floats = ≈40 KB); (2) party-to-expert mappings (200 integers = ≈0.8 KB); (3) a fixed reference set of 200 RGB images (224×224×3, float32), which requires approximately 114 MB; and (4) up to 6 experts, totaling ≈600 MB. Altogether, this amounts to approximately ≈714 MB.

In terms of runtime, ShiftEx adds minimal latency during shift detection and adaptation. Kernel-based MMD drift detection takes an average of 154 ± 17 ms, clustering 200 parties' latent representations takes 1389 ms, and expert assignment and creation add only 0.15 ms, yielding a total adaptation overhead of approximately 1.55 seconds per shift window. These operations are infrequently triggered, only when a shift is detected, and are amortized over multiple training rounds.

5.8.5 Expert Dynamics and Long-Term Adaptation

To analyze how ShiftEx adapts over time, we track the distribution of parties across experts in each window. At initialization (W0), all parties are assigned to a single expert. This ensures a shared starting point across all methods and datasets, allowing us to isolate and measure adaptation purely as a result of evolving data distributions. From W1 onward, we observe how parties progressively migrate to newly created experts as shifts occur, indicating clear, data-driven reassignments.

In FMoW, parties quickly diversify across multiple experts as new satellite scenes with regional or temporal variations are introduced. By W4, the party population is distributed across five distinct experts, showing that ShiftEx recognizes and preserves major distributional changes as separate regimes. In TinyImageNet-C, which contains progressive corruptions (e.g., fog, blur, noise), parties spread smoothly across six experts by W5. The fact that each window results in new dominant experts indicates that ShiftEx not only detects shifts but actively expands its capacity to encode new visual distortions without overriding prior knowledge.

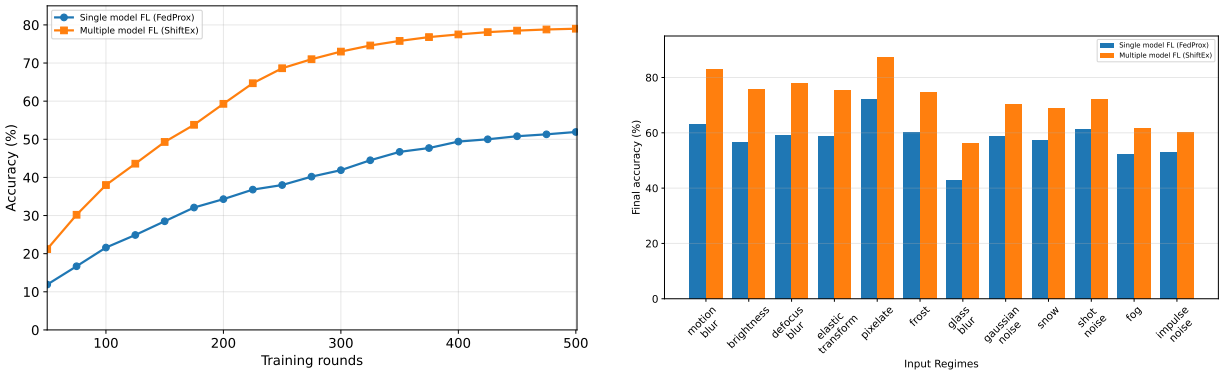
For CIFAR-10-C, we observe a more contained dynamic: parties gradually shift toward a second expert, stabilizing into a two-expert configuration. This reflects fewer major shifts, and showcases ShiftEx's ability to remain compact and resist unnecessary expert proliferation when the environment is relatively stable. In FEMNIST, where character distributions and writing styles evolve gradually, parties adapt by migrating across five experts with some reuse over time. This highlights ShiftEx's ability to handle subtle drift without full resets, preserving older experts when still relevant.

FashionMNIST shows mixed behavior, with parties jumping to new experts in W1–W2, reconsolidating around one in W3–W4, and redistributing again in W5. This cyclical pattern reflects changing but partially repeating shifts, demonstrating how ShiftEx supports both

fresh specialization and expert reuse over time. These evolving expert assignment patterns confirm that ShiftEx supports both short-term responsiveness and long-term memory. By allowing parties to switch experts only when necessary and retaining expert states across windows, ShiftEx balances adaptation with stability, crucial for robust learning in dynamic federated environments.

5.8.6 Long-Horizon Training: Single Model vs. Multiple Model FL

A natural question when evaluating ShiftEx against single-model baselines is whether the performance gap reflects an insufficient training budget or a deeper structural limitation of maintaining a single global model under heterogeneous data distributions. To investigate this, we train FedProx and ShiftEx for 500 rounds on TinyImageNet-C, evaluating every 25 rounds across 200 clients with 20% participation per round using ResNet-18 [75]. Each client is assigned data from a distinct corruption regime at severity levels 1–5, with both methods initialized from the same randomized weights.



(a) Test accuracy over 500 training rounds for FedProx and ShiftEx on TinyImageNet-C with 200 clients and 20% participation per round.

(b) Final accuracy at round 500 broken down by input regime. ShiftEx outperforms FedProx across all input regimes.

Figure 5.8: Comparison of FedProx and ShiftEx on TinyImageNet-C.

As shown in Figure 5.8a, FedProx saturates at approximately 52% by round 500, while ShiftEx converges to almost 80%, with no sign of the two trajectories converging. This confirms that the gap is structural rather than a training budget effect. When clients simultane-

ously present data from divergent covariate regimes, the global model receives conflicting gradient signals each round, converging to a parameter compromise that is suboptimal for every individual distribution — a manifestation of negative knowledge transfer [117]. This limitation is further exacerbated by the finite representational capacity of a single model: while it can approximate a limited set of distributions, it cannot simultaneously capture a large number of highly divergent regimes without degradation. In contrast, maintaining multiple specialized models allows each to focus on a narrower subset of distributions, effectively expanding the system's overall capacity. Figure 5.8b shows this holds across all 12 different input regimes without exception, with gaps ranging from 10 to over 20 percentage points. These results demonstrate that the advantage of expert specialization in ShiftEx is not recoverable through extended single-model training.

5.9 Conclusion

We present a shift-aware mixture of experts framework for federated learning under non-stationary data, addressing covariate and label shifts in streaming environments. Through shift detection, latent memory for expert reuse, and optimization, our method enables dynamic expert instantiation and scalable adaptation. Extensive experiments show significant gains in accuracy and adaptation speed over FL baselines, offering a practical, privacy-preserving middleware for real-world deployments.

Traditional FL middleware approaches assume relatively stable workload characteristics, but our framework addresses the unique challenge of building middleware that can detect, respond to, and manage distributional shifts in real-time across federated participants. The system implements key middleware patterns, including event-driven architectures (responding to detected shifts), load balancing (distributing clients across experts), and service discovery (expert reuse and consolidation). The windowing-based stream processing and latent memory mechanisms function as middleware services that enable scalable, real-time adapta-

tion without requiring application-level awareness of underlying distributional changes.

Limitations include potential expert pool growth under extreme heterogeneity, reliance on frozen encoders, and honest client reporting. Under extreme heterogeneity, the number of experts may proliferate; however, our consolidation step periodically merges similar experts to mitigate this risk. Future work will explore expert compression via online distillation, secure reporting under adversarial settings, and tighter integration with hardware enclaves for end-to-end privacy.

Chapter 6

Dynamic Context Management

6.1 Chapter Overview

Modern AI deployment is increasingly dominated by foundational-model inference workloads. Large language models (LLMs) and the multimodal vision-language models (VLMs) derived from them now support search, question answering, code generation, scientific assistants, and autonomous decision making at large scale [30, 176, 131]. In these settings, the model is typically a fixed pretrained system serving a stream of heterogeneous queries rather than a task-specific model that can be updated during deployment.

This shift makes *inference-time variability* a first-class systems problem. Queries differ in intent, difficulty, modality, and information requirements, yet the model weights cannot be fine-tuned during inference to accommodate each case. The main control lever is therefore *context*: the instructions, demonstrations, retrieved evidence, and input fidelity provided to the model. This chapter frames that problem as *dynamic context management*.

In this thesis, we have examined how middleware and system-generated signals can improve decision making at each stage of the machine learning lifecycle. Earlier chapters addressed

training-time challenges under data heterogeneity and deployment-time adaptation under distribution shift. This chapter focuses on the final stage: *inference-time decision making*.

A central premise of this chapter is that context is the main lever available to steer a fixed foundation model toward the task at hand. We focus on two complementary forms of context that every LLM- or VLM-based system must manage:

- **Application-level context** refers to *what is placed in the prompt*—the task instructions, demonstrations, retrieved examples, and their ordering. It determines how the task is presented to the model during in-context learning.
- **System-level context** refers to *what information is retrieved and at what cost*—the fidelity of retrieved materials, storage and bandwidth overhead, and latency constraints. It determines the quality and acquisition cost of inputs before they ever reach the model.

Dynamic context management requires reasoning over both dimensions jointly: one shapes the information the model sees, while the other shapes the information the system can afford to provide.

This chapter studies two instances of this problem:

- **OptiSeq** addresses application-level context by optimizing the *ordering* of in-context examples in few-shot prompting. It leverages token-level log probabilities as a signal, removes the confounding influence of example order, and re-scores candidate outputs in an *example-free* setting to identify the ordering that best separates correct from incorrect outputs.
- **VOILA** (Value-Of-Information guided Latency-Aware fidelity selection) addresses system-level context by predicting, *before retrieval*, which input fidelity—caption, thumbnail, compressed image, or full resolution—is sufficient for a given query. It transforms

historical VLM correctness patterns into calibrated probability estimates over fidelity levels and uses a value-of-information decision rule to select the lowest-cost fidelity that preserves accuracy.

Together, OptiSeq and VOILA show how inference-time signals—whether log probabilities during generation or calibrated correctness predictions derived from prior model outputs—can be converted into routing decisions that improve accuracy and efficiency without changing model weights. Section 6.2 develops this framing, and Sections 6.4 and 6.5 instantiate it at the application and system levels, respectively.

6.2 Background and Motivation

6.2.1 The Rise of LLM-Based Agentic AI

Large language models and their multimodal extensions have become reusable foundation models for a wide range of downstream tasks. Models such as GPT-4 [30], the LLaMA family [71], Mixtral [84], and multimodal systems such as BLIP [107], LLaVA [257], Flamingo [7], and Qwen-VL [18] are invoked across text, code, vision, and reasoning workloads. Their central strength is that they can be steered to new tasks from instructions and demonstrations provided at inference time, without updating model parameters—the core mechanism behind *in-context learning* (ICL) [30].

Agentic AI systems build on LLMs by equipping them with the ability to plan, reason over multiple steps, access external tools, retrieve information from memory, and produce structured outputs [168, 144]. Retrieval-Augmented Generation [105] extends the effective context of LLMs with dynamically retrieved documents; tool-using agents [174, 199] invoke external APIs and services; multimodal agents [7, 131] combine language with visual perception.

These systems are deployed under highly variable workloads, where both response quality

and aggregate inference cost matter. Because model weights remain fixed at serving time, performance depends heavily on inference-time decisions above the model layer: which examples to include in a prompt, how to order them, which visual fidelity to retrieve, or which storage tier to query.

6.2.2 Application-Level Context: Structuring the Prompt

For LLM-based systems, the most direct lever available at inference time is the construction of the prompt itself. In-context learning allows models to generalize from a small number of task-specific examples embedded in the prompt [30, 176]. However, ICL is highly sensitive to how those examples are presented: their selection [132, 233], their quantity [250, 4], and crucially their *ordering* [141, 254] each significantly influence model output.

The sensitivity to ordering arises from the autoregressive nature of LLMs. The model processes the prompt as a sequential token stream, and the order in which examples appear shapes the probability distributions over subsequent tokens. An ordering that places the most contextually relevant example immediately before the query can prime the model toward the correct output; a poorly ordered prompt can mislead it. This effect is not uniform: the optimal order varies across test instances, tasks, and models [141, 73], ruling out any static precomputed strategy.

The problem is compounded in *online* inference settings where there is no validation data to precompute orderings, no fixed label space, and no guarantee that the examples are drawn from the same distribution as the query. Existing approaches either require labeled development sets with balanced classes [141, 73] or resort to embedding-based ranking that conflates *example relevance* with *prompt ordering quality*—a distinction we formalize in Section 6.4. An effective solution must be inference-time, training-free, and generalizable across tasks and model families.

6.2.3 System-Level Context: Retrieval Cost and Information Fidelity

For multimodal and agentic AI systems, the challenge extends beyond prompt construction to the upstream problem of *what information to retrieve* before the model is ever invoked. Real-world multimodal systems store inputs across heterogeneous storage tiers: mobile devices cache low-resolution thumbnails and text captions locally, while full-resolution images reside in cloud storage with higher retrieval latency and bandwidth cost [138, 208]. Enterprise systems use hot/cold storage hierarchies with retrieval penalties ranging from milliseconds to minutes [11]. Crucially, these costs are incurred *before* any model is invoked, making post-hoc model routing [40, 160] insufficient for end-to-end efficiency.

Recent work on adaptive inference focuses on *model selection*—routing queries to lightweight or heavyweight models based on difficulty [40, 51, 194]. These approaches assume inputs are already retrieved at full fidelity and optimize *how* to compute given fixed context. They do not address the prior question of *what fidelity of information to retrieve in the first place*. Different queries have fundamentally different minimum information requirements: a question about the dominant color of an image can be answered from a low-resolution thumbnail, while a question about fine-grained text requires full resolution. If this variability can be predicted from the query alone—before retrieval—significant cost savings become achievable without sacrificing accuracy.

6.2.4 Inference-Time Signals: Log Probabilities and Calibrated Predictions

A key insight unifying both application-level and system-level optimization is that LLMs and VLMs either generate or enable the derivation of rich signals at inference time, and those signals can be interpreted and transformed to guide routing decisions.

For application-level context, the token-level log probabilities computed during autoregressive generation serve as a direct signal of model confidence. When the model generates an output conditioned on a given prompt, the sum of log probabilities reflects how "expected" that output is under the provided context. However, this signal is confounded by the in-context examples: the same output may receive different log probabilities under different orderings because the examples themselves shift the model's probability distribution. OptiSeq resolves this confound by re-scoring each candidate output *without* the in-context examples, conditioning only on the task instruction. The deconfounded score reflects inherent output quality rather than example-order-induced probability shifts, enabling reliable selection of the best ordering.

For system-level context, a different signal regime applies. Since retrieval must occur *before* inference, the VLM's log probabilities are unavailable at decision time. Instead, VOILA leverages a derived signal: historical patterns of VLM correctness at different fidelity levels, collected during an offline training phase and transformed through gradient-boosted regression and isotonic calibration into reliable probability estimates. These calibrated estimates reflect the expected accuracy of each fidelity level for a given query type, serving as a lightweight, inference-time proxy for the full retrieval-and-inference pipeline.

Both approaches share a common design principle: identify the relevant signal, remove or correct for confounding factors (example order for OptiSeq; retrieval cost and model variance for VOILA), and use the deconfounded signal to make a principled routing decision.

6.3 Related Work and Research Gaps

6.3.1 In-context learning and prompt sensitivity.

A substantial body of work studies how to select and structure ICL examples. Techniques range from BM25 and embedding-based retrieval [132] to graph-based diversification [233],

curriculum ordering [136], and sequential selection with beam search [234]. However, these methods target offline or batch settings with static example pools and labeled data. Online inference-time ordering—where examples are dynamic and no validation data is available—remains underexplored. Existing inference-time approaches are restricted to classification tasks with balanced labels [141, 73] and do not generalize to generation tasks such as API sequence generation. Furthermore, there is a gap between *ranking* examples by relevance and *ordering* them optimally in the prompt; prior work conflates these two problems.

6.3.2 Adaptive inference and model routing.

Methods such as FrugalGPT [40], RouteLLM [160], and hybrid routing [51] reduce inference cost by directing queries to models of different capacities. Early-exit mechanisms [258, 190] adapt computation depth per input. These approaches assume that inputs are already available at full fidelity and optimize the compute applied to fixed context—not the context itself.

6.3.3 Visual context compression and pre-retrieval selection.

Compression-based approaches reduce the visual token count fed to VLMs after retrieval [226, 58]. Learned context compression [112] summarizes visual context into latent representations. None of these approaches operate *before retrieval*; they assume full-resolution inputs have already been obtained. The problem of deciding which fidelity to retrieve as a function of the query—the pre-retrieval context selection problem formalized in VOILA—has not been previously studied.

6.3.4 Value-of-information and cost-sensitive acquisition.

Value-of-information (VOI) frameworks [26, 92] study when the expected benefit of acquiring additional information exceeds its cost. Active feature acquisition [97] extends this to

sequential settings. VOILA applies VOI to the pre-retrieval multimodal perception problem, connecting classical decision theory to modern vision-language systems.

The gaps identified above motivate the two contributions of this chapter: OptiSeq for inference-time application-level context structuring, and VOILA for system-level context selection.

6.4 OptiSeq: Structuring Application-Level Context

In-context learning offers a powerful inference-time capability, but its performance is sensitive to a dimension that prior work has largely overlooked: the *ordering* of examples within the prompt. This section presents OptiSeq, a system that addresses the application-level context structuring problem by leveraging log probability signals to identify, at inference time and without any training or validation data, the example ordering that maximizes the LLM's ability to distinguish correct outputs from incorrect ones.

6.4.1 Sensitivity of ICL to Example Ordering

We begin with an empirical characterization of ordering sensitivity under online inference conditions.

Naive ICL fails to distinguish correct from incorrect outputs. Figure 6.1 illustrates the performance variation for `llama-3-8b-instruct` across all six orderings of three in-context examples on an API sequence generation task from ToolBench [174]. Only one specific ordering yields the correct answer; the remaining five produce incorrect API sequences with varying precision and recall. This variability shows that reordering examples is not a minor perturbation—it fundamentally alters the input context and shifts the token probability distributions that drive generation.

To understand why naive ICL cannot self-correct for this, Figure 6.2 shows the log prob-

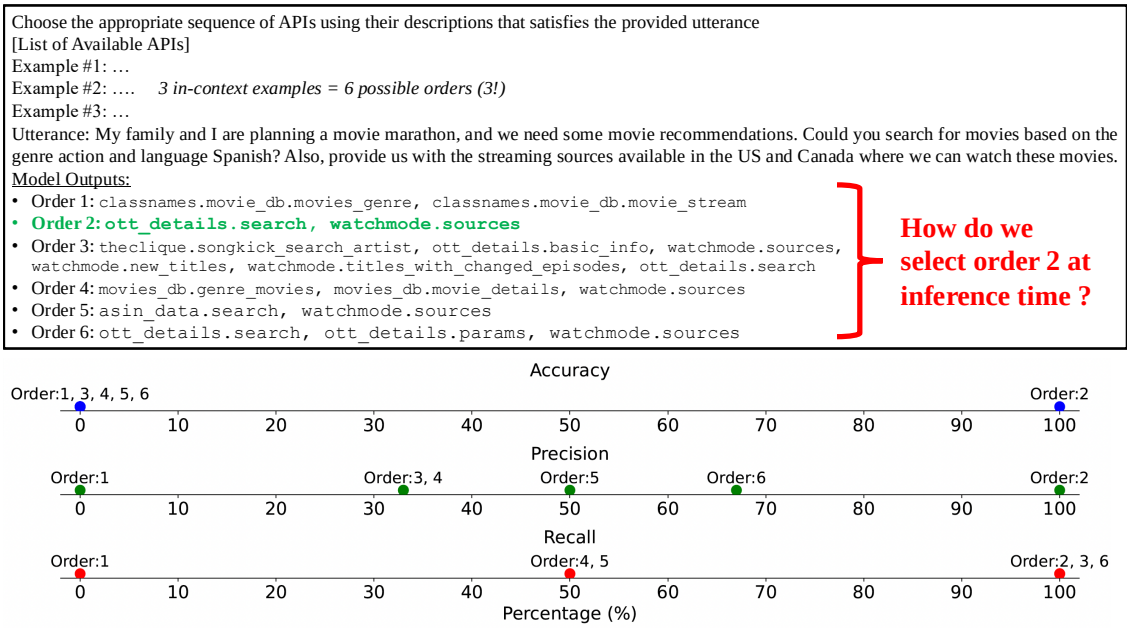


Figure 6.1: `llama-3-8b-instruct` performance variation across all six in-context example orderings for a ToolBench API generation task.

abilities assigned by the LLM to both correct and incorrect output sequences across all orderings. Naive ICL assigns *similar* log probabilities to both correct and incorrect outputs for every ordering—the model cannot distinguish them by confidence alone. Ideally, a model should assign higher probabilities to correct outputs; the inability to do so under standard prompting is the core limitation that OptiSeq addresses.

Optimal order varies across instances and model families. Figure 6.3 evaluates six orderings of three AG News examples [249] across three model families. The optimal ordering is not universal: it varies across test instances and models. An ordering that achieves the highest accuracy for `llama-3-8b-instruct` may underperform for `granite-20b-code-instruct` and vice versa. This rules out any static, precomputed strategy and motivates an adaptive, instance-specific approach that is also model-aware.

The cost of a wrong ordering can be large. Figure 6.4 measures accuracy variation across all orderings in a 3-shot ICL setting for five datasets. Performance fluctuates by

Choose the appropriate sequence of APIs using their descriptions that satisfy the provided utterance
[List of Available APIs and their descriptions]
Example #1: ….
Example #2: …. *3 in-context examples = 6 possible orders (3!)*
Example #3: ….
Utterance: My company is hosting a charity event and needs items for a gift auction. Can you find popular, highly-rated products in the ID region with details on price, image, and seller location?

- **Order 1:** `shopeeapi.search.products, shopeeapi.product.details`
- **Order 2:** `realtor_data_api.realtorpropertylist, asin_data.product`
- **Order 3:** `realtor_data_api.realtorpropertylist, demo.get_product`
- **Order 4:** `realtor_data_api.realtorpropertylist, asin_data.product`
- **Order 5:** `shopeeapi.search.products, shopeeapi.product.details`
- **Order 6:** `shopeeapi.search.products, shopeeapi.product.details`

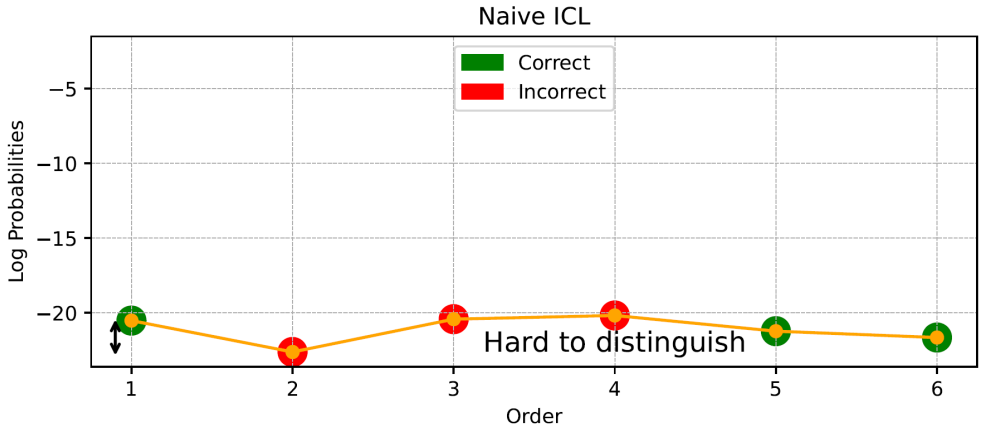


Figure 6.2: Naive ICL log probabilities show high overlap between correct and incorrect output sequences across all orderings, making it impossible to select the best ordering by confidence alone.

approximately 12 percentage points for API generation and 17 percentage points for text classification. This magnitude implies that choosing the wrong ordering can be as costly as using a substantially weaker model.

6.4.2 Methodology

The core question is: *How does the ordering of in-context examples affect an LLM's ability to distinguish between possible outputs?* The ordering of examples provides a context signal that shifts the LLM's probability distribution. Outputs generated under different orderings may have comparable log probabilities because the in-context examples themselves absorb probability mass proportional to their order within the autoregressive context. Removing that confound—evaluating output likelihood *without* the examples—enables a fairer com-

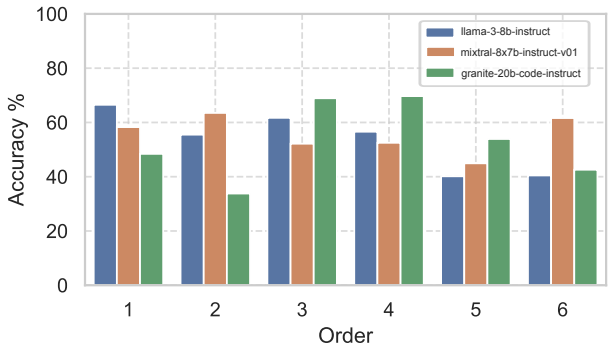


Figure 6.3: AG News classification accuracy across different orderings and model families, showing that no single ordering generalizes across instances or models.

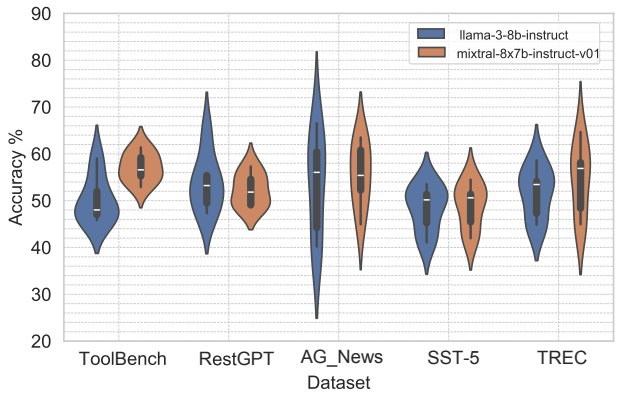


Figure 6.4: Variation in average accuracy across all permutations of three in-context examples for five datasets, demonstrating the high sensitivity of LLM performance to example ordering.

parison of candidate outputs across orderings and yields a cleaner signal for selecting the best one.

OptiSeq

OptiSeq is an *example-free* method that optimizes in-context example ordering by leveraging the log probabilities of LLM-generated outputs (Figure 6.5). The process consists of four steps:

- **Generate outputs for all permutations.** Given task instruction $\mathcal{I}$ and in-context examples $\mathcal{E}$, construct $k = |\mathcal{E}|!$ prompts, one per ordering. Each prompt is fed to the

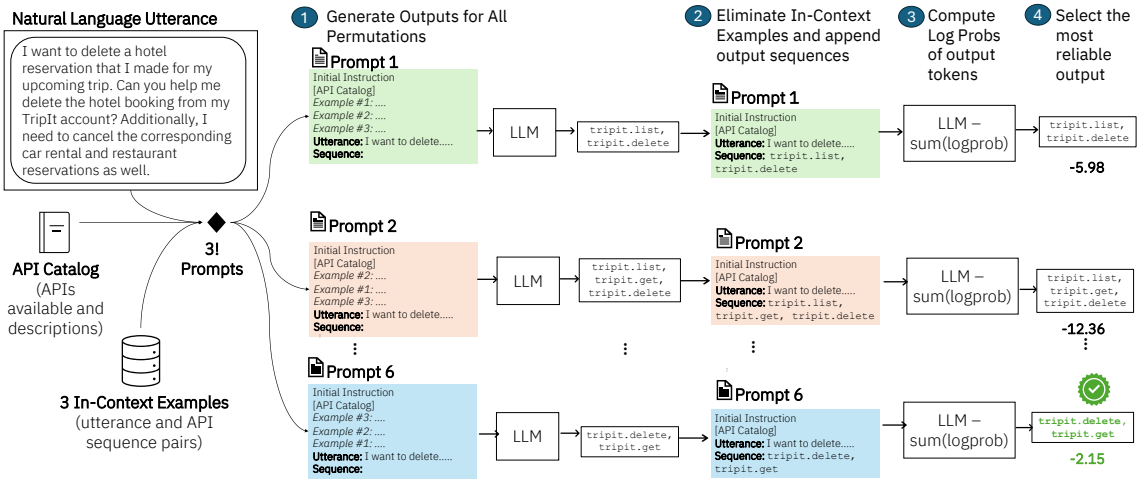


Figure 6.5: OptiSeq overview: outputs are generated for all example orderings using standard ICL, then re-scored with the in-context examples removed. The ordering whose output receives the highest example-free log probability is selected as optimal.

LLM to produce a candidate output $o_k \in \mathcal{O}$:

$$\log P(o_k \mid \mathcal{I}, \mathcal{E}_k) = \sum_{i=1}^{n} \log P(x_{ik} \mid \mathcal{I} \oplus \mathcal{E}_k \oplus x_{jk<ik}) \tag{6.1}$$

where x_{ik} is the i^{th} token of output o_k, $\mathcal{E}_k$ is the k^{th} permutation, and $x_{jk<ik}$ denotes preceding tokens.

- **Eliminate in-context examples.** For each candidate output o_k, construct a modified prompt containing only the task instruction $\mathcal{I}$ (no examples).

- **Compute example-free log probabilities.** Using the same LLM, compute the sum of log probabilities of o_k conditioned solely on $\mathcal{I}$:

$$\Phi_k = \sum_{i=1}^{n} \log P(x_{ik} \mid \mathcal{I} \oplus x_{jk<ik}) \quad \forall\, o_k \in \mathcal{O} \tag{6.2}$$

- **Select the optimal ordering.** Choose the ordering k^* that maximizes the example-

Algorithm 4 OptiSeq

Inputs: Task instruction $\mathcal{I}$, In-context examples $\mathcal{E}$, Large Language Model M
Outputs: Optimal example ordering $\mathcal{E}_{k^*}$

Construct $k = |\mathcal{E}|!$ permutations $\mathcal{E}_k$
for each permutation $\mathcal{E}_k$ **do**
    $o_k \leftarrow M(\mathcal{I}, \mathcal{E}_k)$ // Generate outputs for all orderings
end for
for each generated output o_k **do**
    $\Phi_k \leftarrow \sum_{i=1}^{n} \log P(x_{ik} \mid \mathcal{I} \oplus x_{jk<ik})$ // Log probs without examples
end for
$k^* = \underset{k}{\text{argmax}}\ \Phi_k$ // Select optimal ordering
return $\mathcal{E}_{k^*}$

free log probability:

$$k^* = \underset{k}{\text{argmax}}\ \Phi_k \tag{6.3}$$

Figure 6.6 shows the effect of this transformation for the same utterance from Figure 6.2: the example-free scores clearly separate correct from incorrect outputs, whereas the standard ICL log probabilities could not. The full algorithm is given in Algorithm 4.

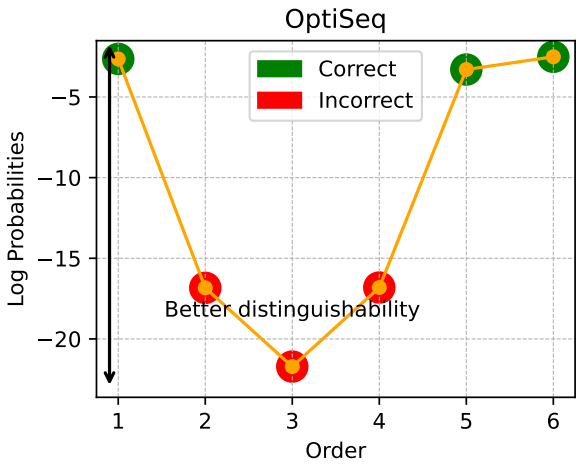


Figure 6.6: OptiSeq log probabilities after removing in-context examples, showing clear separation between correct and incorrect outputs.

EOptiSeq

While OptiSeq provides reliable ordering selection, evaluating all $|\mathcal{E}|!$ permutations grows factorially with the number of examples. EOptiSeq prunes the search space by anchoring the first example position using embedding similarity, motivated by the finding that placing the most relevant example first yields the highest accuracy [133].

Given test instance x_{test} and examples $\{e_1, \ldots, e_E\}$, EOptiSeq computes Sentence-BERT embeddings [184]:

$$\mathbf{v}_i = \text{SBERT}(e_i), \quad \mathbf{v}_{\text{test}} = \text{SBERT}(x_{\text{test}}) \tag{6.4}$$

and cosine similarities $s_i = \frac{\mathbf{v}_i \cdot \mathbf{v}_{\text{test}}}{\|\mathbf{v}_i\| \, \|\mathbf{v}_{\text{test}}\|}$. The highest-ranked example is anchored in the first position, reducing the search from $\mathcal{E}!$ to $(\mathcal{E}-1)!$ permutations. The final ordering is selected using the same example-free log probability scoring as OptiSeq.

6.4.3 Experimental Strategy

We evaluate OptiSeq and EOptiSeq across two tasks, five datasets, and five LLMs under three-shot ICL with greedy decoding.

Datasets

We consider two tasks: **API sequence generation** and **text classification**. API generation requires producing a sequence from a set of API candidates (multi-label prediction over a large combinatorial output space), a setting that prior ordering methods do not address [73, 141]. For text classification we use (i) AG News [249], (ii) SST-5 [197], and (iii) TREC [78]. For API generation we use (iv) RestGPT [199] and (v) ToolBench [174].

Models

We evaluate five models from three families over a parameter range of 8B to 70B:

- `llama-3-8b-instruct` [71]
- `llama-3-70b-instruct` [71]
- `granite-13b-instruct-v2` [83]
- `granite-20b-code-instruct` [83]
- `mixtral-8x7b-instruct-v01` [84]

Baselines

We compare against **Random** ordering, **Top-K** (cosine similarity ranking), **LocalE** [141] (median-entropy ordering computed over a development set assuming balanced labels), and **Influence score** [73] (per-ordering deviation from average predicted probability). LocalE and Influence require corpus-level statistics and labeled development data; comparisons are provided as reference but are not strictly apples-to-apples. All methods operate on identical example sets to isolate ordering effects.

Metrics

For classification tasks we report accuracy (%). For API sequence generation tasks we report accuracy (exact sequence match), recall (fraction of correct APIs in ground truth), and precision (fraction of correct APIs in predicted sequence), all ignoring ordering within the API sequence for recall and precision.

Dataset	Model	Random	Top-K	OptiSeq	EOptiSeq	LocalE	Influence
ToolBench	llama-3-8b-instruct	43.99	44.80	**54.18**	51.12	48.77	50.30
	llama-3-70b-instruct	58.24	59.06	**68.43**	65.37	63.03	64.56
	granite-20b-code-instruct	53.76	56.21	**62.93**	60.28	60.32	61.54
	granite-13b-instruct-v2	43.42	44.41	**47.76**	46.37	43.31	44.64
	mixtral-8x7b-instruct-v01	43.38	42.56	**48.26**	46.23	46.90	42.26
RestGPT	llama-3-8b-instruct	40.76	42.04	**61.15**	52.32	48.46	51.09
	llama-3-70b-instruct	54.14	57.96	**69.42**	64.34	61.48	64.97
	granite-20b-code-instruct	35.03	39.49	**46.49**	42.43	38.15	38.15
	granite-13b-instruct-v2	24.84	24.20	**30.57**	28.02	25.87	26.71
	mixtral-8x7b-instruct-v01	41.41	43.31	**55.41**	51.59	43.17	42.15
AGNews	llama-3-8b-instruct	72.13	73.89	**78.54**	75.64	74.44	76.53
	llama-3-70b-instruct	75.00	76.50	**81.00**	78.00	77.50	79.03
	granite-20b-code-instruct	62.90	66.31	**73.94**	69.27	61.99	65.92
	granite-13b-instruct-v2	59.81	58.69	**61.36**	59.98	59.42	60.52
	mixtral-8x7b-instruct-v01	85.45	85.97	**89.60**	86.37	86.78	87.62
SST-5	llama-3-8b-instruct	53.10	53.10	**57.10**	54.10	54.87	54.00
	llama-3-70b-instruct	55.00	55.50	**60.00**	56.00	57.20	56.80
	granite-20b-code-instruct	27.80	30.50	**32.70**	30.70	28.66	29.16
	granite-13b-instruct-v2	46.90	47.90	**50.10**	48.90	47.15	47.45
	mixtral-8x7b-instruct-v01	52.90	53.90	**55.70**	54.90	54.38	54.85
TREC	llama-3-8b-instruct	60.60	63.20	**67.80**	66.40	63.80	64.20
	llama-3-70b-instruct	62.00	65.00	**70.00**	68.00	65.60	66.00
	granite-20b-code-instruct	50.60	53.40	**58.00**	55.60	50.98	51.34
	granite-13b-instruct-v2	40.00	43.20	**46.40**	45.20	40.36	40.69
	mixtral-8x7b-instruct-v01	62.20	66.80	**73.20**	70.60	68.94	69.40

Table 6.1: Accuracy (%) comparison across datasets and models. Bold indicates the highest accuracy; underlined indicates the second-highest.

6.4.4 Results and Discussion

OptiSeq consistently outperforms all baselines across tasks, datasets, and model families. Table 6.1 shows that OptiSeq achieves an average improvement of 10.5 percentage points over Random and 9.05 points over Top-K for API generation, and approximately 6 points over both for classification. Gains over LocalE and Influence score average 5.5–6.5 points for API generation and 4 points for classification. The key differentiator is the example-free re-scoring: LocalE and Influence evaluate log probabilities while examples are still present, which preserves the ordering-induced probability shifts and reduces distinguishability between permutations. By removing the examples before re-scoring, OptiSeq obtains a cleaner, ordering-agnostic confidence estimate that reliably identifies the best permutation.

Strategic ordering of fewer examples outperforms unordered many-shot ICL. Figure 6.7 shows that 3-shot OptiSeq outperforms 4-shot and 5-shot ICL with Random and Top-K ordering. This is significant because context-length constraints make adding more

examples impractical in production settings, and adding examples without optimizing their order can introduce noise or redundancy [134]. Optimizing the ordering of a smaller set is more effective than naively expanding it.

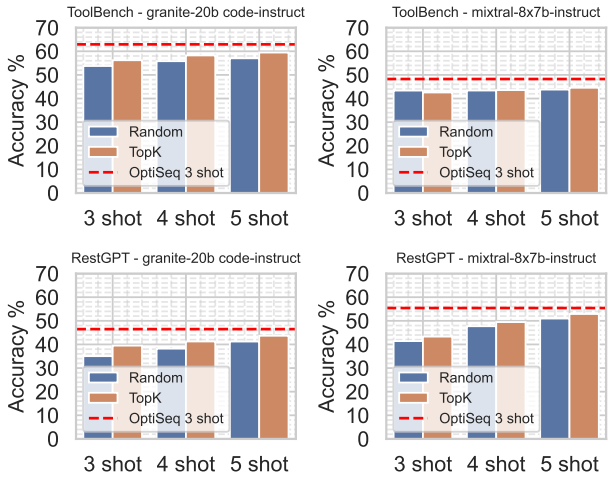


Figure 6.7: 3-shot OptiSeq (red) outperforms 4- and 5-shot ICL with Random and Top-K ordering, showing that optimized ordering of fewer examples beats unordered many-shot prompting.

OptiSeq generalizes under out-of-distribution examples. Figure 6.8 presents OOD experiments where ToolBench queries use RestGPT in-context examples and vice versa. OptiSeq with OOD examples outperforms Random and Top-K with in-distribution examples by 6.15 and 5.35 percentage points respectively. This is a particularly valuable property in online settings where in-domain examples may be unavailable: optimized ordering compensates for domain mismatch.

EOptiSeq trades modest accuracy for improved efficiency. By anchoring the first example position via embedding similarity, EOptiSeq reduces the permutation space from $\mathcal{E}!$ to $(\mathcal{E}-1)!$. It consistently outperforms Top-K and Random but lags OptiSeq by a small margin—an acceptable trade-off in latency-sensitive deployments.

Practical impact in deployed pipelines. In an edge-cloud middleware deployment for API sequence generation, OptiSeq-guided prompting reduces retry attempts by 35–45% com-

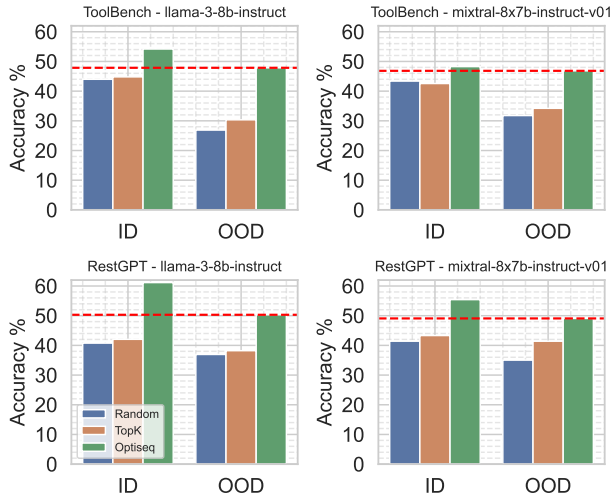


Figure 6.8: In-distribution (ID) vs. out-of-distribution (OOD) performance on API generation. OptiSeq with OOD examples outperforms Random/Top-K with ID examples.

pared to Random and Top-K ordering, with llama-3-8b-instruct on RestGPT achieving 95.5% Success@3 with roughly half the retries of Top-K and a 19% higher first-try success rate.

Limitations. OptiSeq requires evaluating $|\mathcal{E}|!$ permutations, which grows factorially. This is practical for 3–5 shot settings but prohibitive for many-shot ICL (> 10 examples). Search-based strategies such as beam search over the permutation space are a natural extension for such settings.

6.5 VOILA: Selecting System-Level Context

While OptiSeq optimizes *what* the prompt contains, there is a complementary question that arises before the prompt is even constructed: *what information should be retrieved in the first place?* For multimodal and agentic systems, retrieval itself has a cost—in bandwidth, latency, and storage—and different queries require fundamentally different amounts of visual information. VOILA addresses this system-level context selection problem by learning to predict, from the query alone and before any retrieval, the minimum-cost input fidelity required to answer correctly.

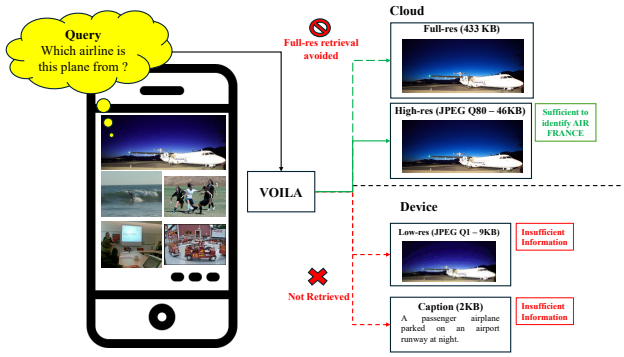


Figure 6.9: **VOILA overview.** Example of pre-retrieval fidelity selection: low-cost representations may be insufficient, while an intermediate fidelity can answer the query correctly without incurring full-resolution retrieval cost.

6.5.1 Motivation and Problem Formulation

Modern multimodal systems store visual content across heterogeneous storage tiers. Mobile services like Google Photos and iCloud Photos cache compressed thumbnails and captions locally while full-resolution originals reside in cloud storage [138, 208]. Enterprise systems use hot/cold storage hierarchies with retrieval latencies ranging from milliseconds to minutes [11]. Crucially, these costs are incurred *before* any model is invoked.

A naive system always retrieves at maximum fidelity—incurring the maximum cost regardless of query requirements. But queries have fundamentally different minimum information needs: "What color is the car?" can be answered from a low-resolution thumbnail, while "What text appears on this sign?" requires a full-resolution image. Given the query *"Which airline is this plane from?"*, a VLM cannot answer from a 2 KB text caption alone but correctly identifies "Air France" from a moderately compressed 46 KB JPEG, avoiding retrieval of the 433 KB full-resolution image—a 9× bandwidth reduction.

We formalize this as a *cost-sensitive information acquisition problem*. Let (q, x) denote query q and image x. Given a discrete, ordered set of input fidelities $\mathcal{F} = \{f_1, \ldots, f_K\}$ sorted by increasing acquisition cost, the objective is to select the fidelity $f^\star$ maximizing expected

utility:

$$f^\star = \arg\max_{f \in \mathcal{F}} \left(\Pr(\hat{a}_f \text{ is correct} \mid q) - \lambda\, c(f) \right)$$

where $c(f)$ captures retrieval bandwidth, latency, and storage overhead, λ is a cost-sensitivity parameter, and the probability of correctness must be estimated *before retrieving fidelity f or running the VLM.*

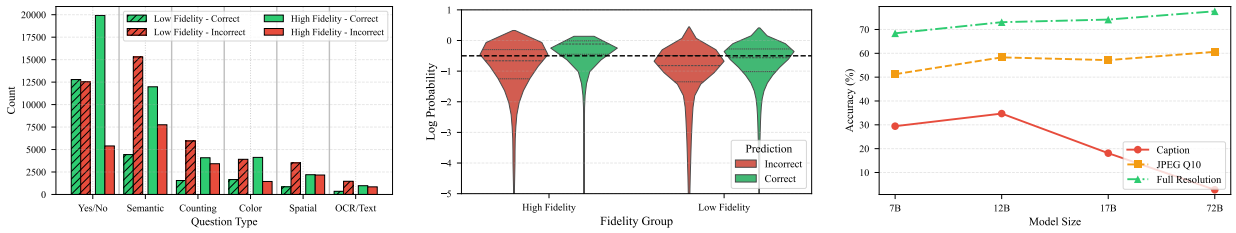
Pre-retrieval constraint. Because fidelity selection must occur before retrieval, VOILA cannot rely on post-hoc model signals such as token log probabilities or hidden activations—the very signals that OptiSeq exploits. Instead, it must reason using only signals derivable from the question text alone, which distinguishes VOILA from all model routing approaches that assume inputs are already available.

Pre-retrieval fidelity selection is relevant in three deployment scenarios: (i) **Edge-Cloud VQA Systems**, where network latency and bandwidth costs dominate query latency and captions/thumbnails are cached on-device while full-resolution images reside in the cloud; (ii) **Agentic Memory Systems** [144, 168], where long-context agents store episodic memories across storage tiers with varying retrieval costs; and (iii) **Mission-Critical Cyber-Physical Systems** [177], where autonomous drones transmit imagery over bandwidth-constrained networks.

Empirical Observations

To motivate VOILA's design, we present three empirical observations based on Pixtral-12B on VQA-v2, examining how model behavior varies across input fidelities from text-only captions to full-resolution images.

Observation 1: Information Requirements Vary Systematically Across Queries. Figure 6.10a quantifies how different question types over the same images require different



(a) Query-dependent accuracy across fidelities.

(b) Model confidence vs. context sufficiency.

(c) Effect of model scale under fixed fidelity.

Figure 6.10: **Empirical evidence motivating VOILA's design** (Pixtral-12B, VQA-v2). (a) Low-fidelity inputs succeed on semantically coarse queries but fail on visually demanding ones; no single fidelity dominates. (b) Incorrect low-fidelity answers often receive log-probabilities comparable to correct answers, making post-retrieval confidence an unreliable escalation signal. (c) Scaling model size yields limited gains when input fidelity is insufficient; model capacity cannot substitute for missing visual evidence.

minimum input fidelities. Low-fidelity representations (captions, 32×32 thumbnails) succeed on semantically coarse queries such as yes/no and broad object identification, but fail disproportionately on counting, color, spatial, and OCR tasks that require fine-grained visual evidence. High-fidelity inputs substantially improve accuracy on visually demanding categories while offering limited gains on queries already solvable at low fidelity. No single fidelity dominates across query types, motivating query-adaptive fidelity selection.

Observation 2: Model Confidence Is an Unreliable Signal for Context Sufficiency. A common adaptive heuristic is to retrieve low-cost context first and escalate to higher-fidelity inputs only when the model exhibits low confidence (token-level log-probabilities). However, as shown in Figure 6.10b, incorrect answers produced from low-fidelity inputs often receive log-probability scores comparable to correct answers—frequently exceeding confidence thresholds that would prevent escalation. This leads to confident but wrong predictions being returned without ever accessing more informative representations. Post-retrieval confidence is thus an unreliable indicator of context sufficiency, motivating pre-retrieval, query-conditioned decision mechanisms.

Observation 3: Model Scaling Does Not Compensate for Insufficient Information.

Figure 6.10c shows that under fixed input fidelity, scaling yields limited and non-monotonic gains: caption-only inputs exhibit early saturation and even degradation at larger model sizes, while moderately compressed inputs (JPEG Q10) show only marginal improvement. Full-resolution inputs consistently benefit from increased capacity. These trends indicate that model failures under low-fidelity inputs stem from missing visual evidence rather than inadequate reasoning, and that information sufficiency is a prerequisite for effective scaling.

Together, these three observations establish that (i) information requirements are predictable from query features, (ii) post-retrieval confidence signals are insufficient for escalation decisions, and (iii) model scaling cannot substitute for pre-retrieval information selection. This motivates VOILA's formalization of pre-retrieval context selection as a cost-sensitive learning problem.

6.5.2 Methodology

VOILA employs a two-stage pipeline to estimate fidelity-specific success probabilities and use them for cost-aware routing.

Question-Conditioned Success Scoring

For each fidelity $f \in \mathcal{F}$, a training set is constructed by executing the VLM at fidelity f on labeled examples and recording whether the output is correct ($y_f \in \{0, 1\}$). A question feature representation $\phi(q)$ is extracted consisting of TF-IDF features of the question text, lexical statistics (length, numeric indicators), and coarse question-type features (counting, color, spatial). A Gradient Boosting Regressor (GBR) is trained per fidelity:

$$r_f(q) = g_f(\phi(q))$$

to produce a raw score correlating with the VLM's correctness likelihood at fidelity f. GBR is chosen for its robustness in sparse, mixed feature spaces; the framework is agnostic to the

specific regressor.

Fidelity-Specific Probability Calibration

Raw scores $r_f(q)$ are not guaranteed to be well-calibrated as probabilities. To obtain reliable estimates suitable for decision-making, a fidelity-specific isotonic regression calibrator is applied:

$$\psi_f : r_f(q) \mapsto \hat{p}_f(q)$$

fitted on a held-out calibration split. Isotonic regression enforces monotonicity while remaining non-parametric, correcting systematic over- and under-confidence without parametric assumptions. At inference time, only the lightweight calibrated predictor $\hat{p}_f(q)$ is evaluated—no VLM invocation or retrieval is needed.

The key effect of calibration is probability separation. As illustrated in Figure 6.11, the VLM's own token probabilities cluster in a narrow, overlapping band for both correct and incorrect predictions—providing little discriminative signal under low-fidelity inputs. In contrast, the GBR + isotonic pipeline produces well-separated, monotonic probabilities: queries solvable at low fidelity concentrate near high $\hat{p}_f$ values, while those requiring additional visual detail receive lower estimates. This separation is what enables reliable VOI decisions.

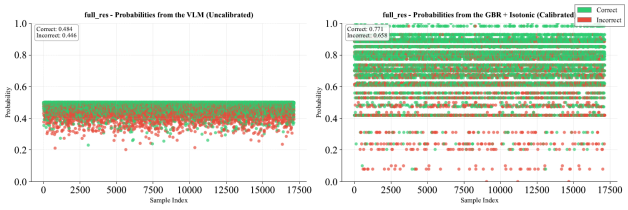


Figure 6.11: **Calibrated vs. uncalibrated probabilities** (full-resolution fidelity, VQA-v2). *Left:* VLM token probabilities are compressed into a narrow band (correct mean: 0.484; incorrect mean: 0.446), providing negligible discriminative signal. *Right:* GBR + isotonic calibration achieves clear separation (correct mean: 0.771; incorrect mean: 0.658), enabling reliable value-of-information decisions across fidelities.

Algorithm 5 VOILA Inference Procedure

Input: Query q, fidelity set $\mathcal{F} = \{f_1, \ldots, f_K\}$, trade-off λ
Output: Selected fidelity f^* and answer $\hat{a}_{f^*}$
// Phase 1: Fidelity selection (question-only, no VLM)
Initialize $f \leftarrow f_1$ ▷ Start with lowest-cost fidelity
Compute $\hat{p}_f(q)$ using calibrated predictor $\psi_f(r_f(q))$
for $f' \in \{f_2, \ldots, f_K\}$ in ascending cost order **do**
    Compute $\hat{p}_{f'}(q)$ using calibrated predictor $\psi_{f'}(r_{f'}(q))$
    VOI $\leftarrow [\hat{p}_{f'}(q) - \hat{p}_f(q)] - \lambda \cdot c(f')$
    if VOI > 0 **then**
        $f \leftarrow f'$ ▷ Escalate to higher fidelity
        $\hat{p}_f(q) \leftarrow \hat{p}_{f'}(q)$
    else
        break ▷ No further escalation justified
    end if
end for
// Phase 2: Execute VLM once at selected fidelity
Retrieve visual representation at fidelity f
Execute VLM to produce answer $\hat{a}_f$
return $f, \hat{a}_f$

Value-of-Information-Based Fidelity Selection

Given calibrated success probabilities $\hat{p}_f(q)$, VOILA selects fidelities by comparing their expected utility via the Value of Information (VOI):

$$\text{VOI}(f_j \mid f_i) = \hat{p}_{f_j}(q) - \hat{p}_{f_i}(q) - \lambda \cdot (c(f_j) - c(f_i))$$

Starting from the lowest-cost fidelity, VOILA escalates to the next fidelity only when VOI $>$ 0, i.e., when the expected accuracy gain justifies the additional retrieval cost. The selected fidelity is the lowest-cost option that cannot be profitably escalated further. This implements a near Bayes-optimal policy: utility regret is bounded by calibration error, so improvements in calibration directly translate to better routing. The complete inference procedure is given in Algorithm 5.

6.5.3 Experimental Strategy

Models and Datasets

We evaluate six state-of-the-art VLMs spanning 7B to 235B parameters: *Pixtral-12B* [5], *LLaMA-4-Maverick (17B×12E)* [3], *LLaVA-1.5-7B* [257], *Qwen2-VL-72B*, *Qwen2.5-VL-7B*, and *Qwen3-VL-235B-A22B* [18]. This range allows assessment of whether fidelity selection remains beneficial as model capacity scales.

We evaluate on five datasets spanning complementary deployment regimes: **VQA-v2** [70] and **GQA** [82] for standard compositional VQA; **TextVQA** [196] for OCR-dependent reasoning; **FloodNet** [177] for mission-critical aerial imagery; and **LoCoMo** [144] for long-horizon agentic memory retrieval (VQA-style pairs generated from conversational memory items via GPT-4o).

Input Fidelities and Cost Model

For each image, we construct an ordered fidelity set

$\mathcal{F} = \{\texttt{caption}, \texttt{resize}_{32\times32}, \texttt{jpeg\_q1}, \texttt{jpeg\_q10}, \texttt{full}\}$

reflecting representations already materialized in real-world systems. Acquisition cost is modeled as:

$$c(f) = \alpha\, B(f) + \beta\, L(f) + \gamma\, S(f)$$

where $B(f)$, $L(f)$, $S(f)$ capture bandwidth ratio, retrieval latency, and storage access penalty. Costs are normalized so $c(\texttt{full}) = 120$. Table 6.2 summarizes the normalized costs for each fidelity.

Table 6.2: **Normalized acquisition costs per fidelity.** Costs reflect bandwidth, retrieval latency, and storage tier penalties, normalized such that $c(\texttt{full}) = 120$.

Fidelity	Caption	Resize 32×32	JPEG Q1	JPEG Q10	Full
Cost	9.1	18.1	45.4	63.9	120.0

Evaluation Metrics

We report task accuracy and average acquisition cost per query. Brier score is reported in ablation studies to assess calibration quality.

6.5.4 Results and Discussion

Main Results: Pareto Frontier Analysis

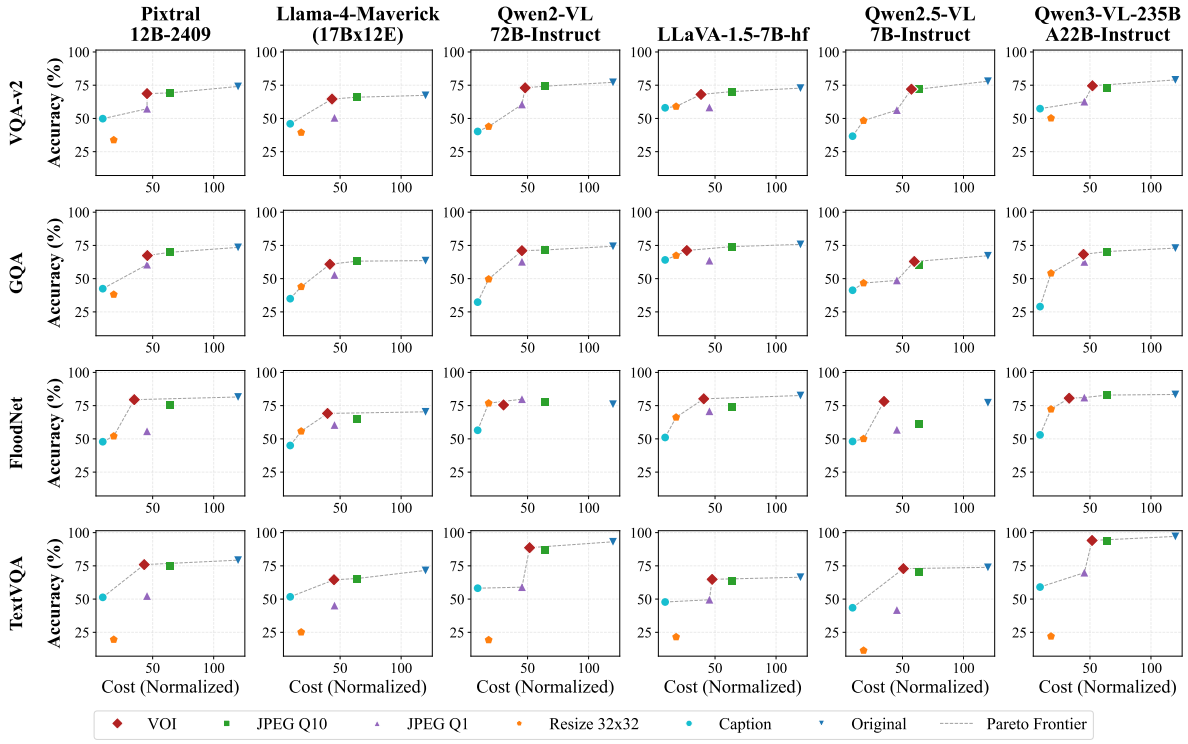


Figure 6.12: **Accuracy–cost Pareto frontiers across datasets and models.** Each subplot compares VOILA to fixed-fidelity baselines in the accuracy vs. normalized retrieval cost plane (upper left is better). Dashed curves denote Pareto frontiers. Across all datasets and model scales, VOILA lies on or near the frontier, achieving near-full-resolution accuracy at 50–60% lower average cost. No single fixed fidelity is Pareto-optimal.

VOILA matches near-full-resolution accuracy at 50–60% lower cost. The Pareto

frontiers in Figure 6.12 show that VOILA lies consistently close to the full-resolution operating point across all datasets and models, while operating at substantially lower cost. In most cases VOILA achieves accuracy within 2–6 percentage points of always using full-resolution images, yet reduces normalized retrieval cost by roughly 50–60%. This pattern is stable across diverse visual regimes, including OCR-heavy TextVQA and bandwidth-constrained FloodNet, indicating that VOILA reliably identifies when high-fidelity visual information is genuinely required and avoids unnecessary retrieval otherwise.

Complete Accuracy and Cost Results

Gains persist under model scaling. The same qualitative Pareto improvements hold across model sizes from 7B to 235B parameters and across datasets with distinct visual and linguistic distributions. Larger models improve absolute accuracy, but the relative gap between fixed-fidelity baselines persists, and VOILA continues to recover most of the full-resolution accuracy at a fraction of the cost. Model scaling alone cannot compensate for insufficient visual information: when critical details are missing, even very large models fail. VOILA's consistent gains reflect a complementary optimization axis—adaptive information acquisition—rather than an artifact of model capacity.

Fidelity Selection Distributions

Figure 6.13 reveals the learned routing behavior of VOILA. Across all models and datasets, VOILA overwhelmingly routes queries to low-cost representations: captions handle queries that require only semantic understanding, while higher-fidelity inputs are reserved for visually demanding tasks. This adaptive distribution is the key mechanism behind VOILA's cost savings—rather than blindly using expensive full-resolution inputs, VOILA learns that the majority of queries can be answered from compact representations already available on-device or in hot storage.

Table 6.3: **Complete comparison across VQA-v2, GQA, FloodNet, and TextVQA.** Accuracy (%) and average normalized cost for each method. **VOI** rows are bolded. VOILA achieves accuracy within 2–6% of full-resolution at 50–60% lower cost across all model–dataset pairs.

Model	Method	VQA-v2		GQA		FloodNet		TextVQA	
		Acc	Cost	Acc	Cost	Acc	Cost	Acc	Cost
Pixtral-12B-2409	Caption	49.83	9.1	42.45	9.1	47.84	9.1	51.31	9.1
	VOI	**68.67**	**45.41**	**67.37**	**45.66**	**79.48**	**34.94**	**75.92**	**43.09**
	JPEG Q10	69.21	63.9	69.81	63.9	75.75	63.9	74.82	63.9
	JPEG Q1	57.27	45.4	60.53	45.4	55.70	45.4	52.28	45.4
	Resize 32×32	33.80	18.1	38.10	18.1	52.05	18.1	19.57	18.1
	Original	74.06	120.0	73.52	120.0	81.51	120.0	79.29	120.0
Llama-4-Maverick (17Bx12E)	Caption	45.98	9.1	34.93	9.1	45.01	9.1	51.70	9.1
	VOI	**64.61**	**43.37**	**60.87**	**41.64**	**69.14**	**39.68**	**64.52**	**44.98**
	JPEG Q10	65.99	63.9	63.17	63.9	65.23	63.9	65.48	63.9
	JPEG Q1	50.45	45.4	52.76	45.4	60.41	45.4	45.11	45.4
	Resize 32×32	39.42	18.1	43.92	18.1	55.73	18.1	25.11	18.1
	Original	67.37	120.0	63.56	120.0	70.35	120.0	71.56	120.0
Qwen2-VL-72B-Instruct	Caption	40.21	9.1	32.34	9.1	56.48	9.1	58.19	9.1
	VOI	**73.01**	**48.00**	**71.03**	**45.37**	**75.51**	**30.30**	**88.65**	**51.64**
	JPEG Q10	74.27	63.9	71.57	63.9	78.07	63.9	86.84	63.9
	JPEG Q1	60.49	45.4	62.68	45.4	79.84	45.4	58.96	45.4
	Resize 32×32	43.90	18.1	49.63	18.1	76.85	18.1	19.27	18.1
	Original	77.16	120.0	74.35	120.0	76.08	120.0	93.11	120.0
LLaVA-1.5-7B-hf	Caption	58.01	9.1	64.09	9.1	51.05	9.1	47.80	9.1
	VOI	**68.07**	**38.52**	**71.18**	**26.86**	**80.12**	**40.77**	**64.85**	**47.62**
	JPEG Q10	70.21	63.9	74.11	63.9	73.64	63.9	63.25	63.9
	JPEG Q1	58.25	45.4	63.51	45.4	70.76	45.4	49.50	45.4
	Resize 32×32	58.99	18.1	67.37	18.1	66.22	18.1	21.50	18.1
	Original	72.80	120.0	75.77	120.0	82.61	120.0	66.45	120.0
Qwen2.5-VL-7B-Instruct	Caption	36.65	9.1	41.28	9.1	48.06	9.1	43.48	9.1
	VOI	**72.01**	**57.31**	**62.92**	**59.66**	**78.17**	**34.85**	**72.90**	**50.72**
	JPEG Q10	72.10	63.9	60.14	63.9	61.24	63.9	70.26	63.9
	JPEG Q1	56.26	45.4	48.66	45.4	56.81	45.4	41.74	45.4
	Resize 32×32	48.38	18.1	46.70	18.1	50.06	18.1	11.35	18.1
	Original	77.98	120.0	67.22	120.0	77.19	120.0	73.87	120.0
Qwen3-VL-235B-A22B	Caption	57.36	9.1	28.97	9.1	52.99	9.1	59.05	9.1
	VOI	**74.57**	**52.04**	**68.28**	**44.70**	**80.62**	**32.94**	**94.10**	**51.75**
	JPEG Q10	72.58	63.9	70.40	63.9	82.95	63.9	93.65	63.9
	JPEG Q1	62.62	45.4	62.58	45.4	81.06	45.4	69.85	45.4
	Resize 32×32	50.21	18.1	54.08	18.1	72.31	18.1	22.00	18.1
	Original	79.01	120.0	72.98	120.0	83.39	120.0	97.15	120.0

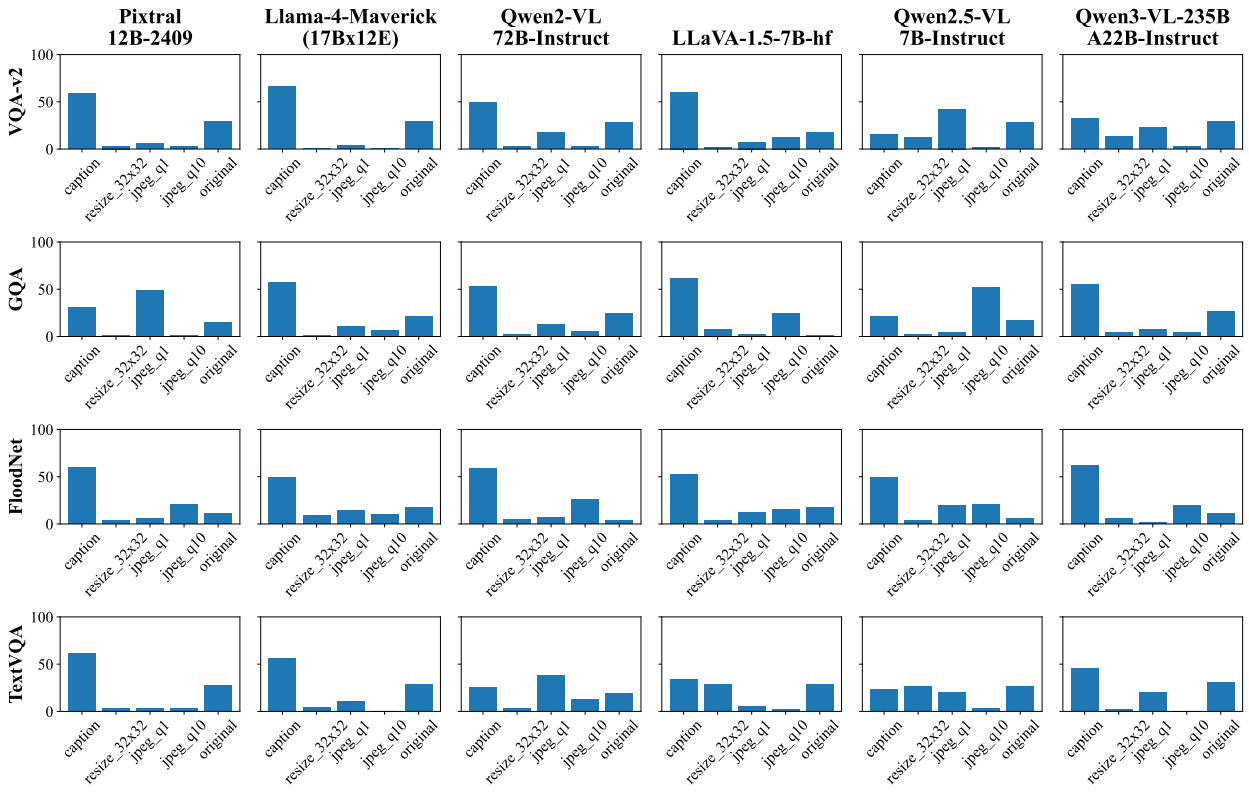


Figure 6.13: **Learned fidelity selection distributions under VOILA routing.** Each subplot reports the percentage of queries routed to each input fidelity across four datasets (rows) and six VLMs (columns). Low-cost representations (captions and thumbnails) dominate across all settings, while higher-cost fidelities (JPEG Q1/Q10 and full-resolution) are selected selectively when expected utility justifies their acquisition cost.

Out-of-Distribution Generalization

Table 6.4 reports accuracy and average acquisition cost when VOILA is trained on one dataset and evaluated on the other. Across all six VLMs, VOILA exhibits stable accuracy–cost behavior under both VQA→GQA and GQA→VQA transfer, with only modest degradation. Notably, average acquisition costs remain close to the in-distribution operating point rather than collapsing to full-resolution retrieval under distribution shift. These results confirm that VOILA's adaptive fidelity selection generalizes across datasets, and that the minimum visual information required by a query is primarily determined by question characteristics rather than dataset-specific statistics.

Table 6.4: **Out-of-distribution generalization.** Accuracy (%) / avg. cost when training on one dataset and testing on the other. VOILA's routing policy is stable across dataset transfer, confirming that query-driven fidelity requirements generalize across distributions.

Model	VQA→GQA	GQA→VQA
Llama-4-Maverick (17Bx12E)	58.57 / 48.55	64.77 / 45.87
Pixtral-12B-2409	64.77 / 45.87	63.34 / 42.26
Qwen2.5-VL-7B	62.87 / 40.71	62.44 / 34.93
LLaVA-1.5-7B	69.12 / 64.60	69.76 / 58.64
Qwen3-VL-235B	74.49 / 73.40	75.98 / 67.98
Qwen2-VL-72B	65.61 / 34.23	66.24 / 33.13

Agentic Memory Retrieval (LoCoMo)

Table 6.5: **LoCoMo memory recall (Acc % / Cost).** Despite being trained on VQA-v2 and GQA, VOILA generalizes to this out-of-distribution agentic memory setting, consistently achieving strong accuracy–cost tradeoffs across all models.

Model	Caption	Original	VOILA
Pixtral-12B-2409	32.5 / 10	46.7 / 120	**43.6** / **69.5**
Qwen2.5-VL-7B-Instruct	26.8 / 10	73.0 / 120	**67.1** / **52.3**
LLaVA-1.5-7B-hf	43.7 / 10	64.7 / 120	**61.5** / **53.0**
Qwen3-VL-235B-A22B	27.4 / 10	76.7 / 120	**72.6** / **51.5**
Llama-4-Maverick (17Bx12E)	29.3 / 10	60.8 / 120	**56.2** / **50.9**
Qwen2-VL-72B-Instruct	26.4 / 10	76.0 / 120	**68.1** / **40.8**

Table 6.5 shows VOILA's performance on LoCoMo evaluating long-horizon agentic memory retrieval in an out-of-distribution setting. VOILA is trained exclusively on static VQA benchmarks (VQA-v2 and GQA), yet generalizes effectively to LoCoMo's constrained memory setting. For Qwen2.5-VL-7B, VOILA attains 67.1% accuracy at cost 52.3—substantially better than caption-only retrieval (26.8%) while avoiding the high cost of always retrieving the original image (73.0% at cost 120). For the 235B-parameter Qwen3-VL model, VOILA recovers 94.6% of full-resolution accuracy (72.6% vs. 76.7%) at less than half the retrieval cost. These results demonstrate that VOILA learns query-driven signals about when additional visual information is valuable, rather than overfitting to specific datasets or fidelity structures.

6.5.5 Ablation Studies

Effect of Calibration Method

Table 6.6: **Ablation: calibration method and decision rule across 6 VLMs on VQA-v2 and GQA.** Entries report **Acc (%) / Cost**. *None*: uncalibrated raw GBR scores; *Iso*: isotonic regression (VOILA default, bolded); *Temp*: temperature scaling; *Acc*: accuracy-only routing (ignore cost); *Thr*: fixed threshold. Isotonic calibration consistently yields the best cost–accuracy tradeoff.

	VQA-v2					GQA				
VLM	**None**	**Iso**	**Temp**	**Acc**	**Thr**	**None**	**Iso**	**Temp**	**Acc**	**Thr**
Pixtral-12B	66.24/52.18	**68.67/45.41**	66.89/56.32	74.06/120.0	69.21/63.90	64.12/51.23	**67.37/45.66**	63.84/49.67	73.52/120.0	69.81/63.90
Qwen3-VL-235B	72.13/58.91	**74.57/52.04**	73.24/61.08	79.01/120.0	72.58/63.90	66.45/51.84	**68.28/44.70**	65.72/48.23	72.98/120.0	70.40/63.90
LLaVA-1.5-7b	65.84/44.73	**68.07/38.52**	64.23/46.91	72.80/120.0	70.21/63.90	68.92/32.45	**71.18/26.86**	67.18/35.62	75.77/120.0	74.11/63.90
LLaMA-4-Maverick	62.15/49.84	**64.61/43.37**	61.34/51.92	67.37/120.0	65.99/63.90	58.23/47.91	**60.87/41.64**	57.45/49.38	63.56/120.0	63.17/63.90
Qwen2-VL-72B	70.48/54.72	**73.01/48.00**	71.15/58.34	77.16/120.0	74.27/63.90	68.34/51.28	**71.03/45.37**	67.92/53.71	74.35/120.0	71.57/63.90
Qwen2.5-VL-7B	69.23/63.84	**72.01/57.31**	68.91/67.15	77.98/120.0	72.10/63.90	60.18/65.73	**62.92/59.66**	58.94/68.42	67.22/120.0	60.14/63.90

Isotonic calibration (VOILA default) consistently and substantially outperforms uncalibrated routing across all model–dataset pairs, validating the importance of reliable probability estimation for cost-aware decisions. Temperature scaling underperforms isotonic regression because it assumes a parametric miscalibration form that may not hold across fidelities. Accuracy-only routing achieves the highest accuracy by always picking full-resolution (cost = 120) but incurs maximum retrieval cost—confirming that the VOI cost-penalty term captures useful signal. Fixed-threshold routing reduces costs but is less accurate than VOILA as the threshold is dataset-agnostic and fails to adapt to per-query information requirements.

Predictor Robustness

Table 6.7: **Predictor robustness under fixed isotonic calibration** (Pixtral-12B, VQA-v2 and GQA). Performance is broadly similar across predictors, confirming that VOILA's gains are driven primarily by calibration quality rather than predictor architecture. Entries report **Acc (%) / Cost**.

Dataset	**GBR**	**LogReg**	**Ridge**	**MLP**
VQA-v2	68.67/45.41	67.92/47.23	68.14/46.58	67.35/48.91
GQA	67.37/45.66	67.81/46.84	67.73/45.12	68.05/49.27

Accuracy and cost are broadly similar across all four predictor architectures (GBR, logistic regression, ridge regression, MLP), confirming that VOILA's performance is not brittle to

predictor choice. The primary driver of VOILA's gains is calibration quality—any regressor that produces monotonic scores and can be calibrated yields comparable results—though GBR provides consistently strong and efficient performance as the default choice.

Extended Ablation: All Models and Datasets

Table 6.8: **Extended ablation across all 6 VLMs and 4 datasets.** Each entry reports **Acc (%) / Cost / Brier Score**. Isotonic calibration (VOILA default, bolded) consistently achieves the best cost–accuracy tradeoff. Lower Brier score indicates better-calibrated probabilities.

Model	Method	Calibration Variants (VOI Routing)			Alternative Decision Rules (Isotonic)	
		None	Isotonic	Temp. Scaling	Acc-only	Fixed-Thr.
			VQA-v2			
Pixtral-12B	Acc / Cost / Brier	66.24/52.18/0.256	**68.67/45.41/0.231**	66.89/56.32/0.248	74.06/120.0/0.231	69.21/63.90/0.231
Llama-4-Maverick	Acc / Cost / Brier	62.15/49.84/0.289	**64.61/43.37/0.264**	61.34/51.92/0.281	67.37/120.0/0.264	65.99/63.90/0.264
Qwen2-VL-72B	Acc / Cost / Brier	70.48/54.72/0.223	**73.01/48.00/0.198**	71.15/58.34/0.215	77.16/120.0/0.198	74.27/63.90/0.198
LLaVA-1.5-7B	Acc / Cost / Brier	65.84/44.73/0.267	**68.07/38.52/0.241**	64.23/46.91/0.259	72.80/120.0/0.241	70.21/63.90/0.241
Qwen2.5-VL-7B	Acc / Cost / Brier	69.23/63.84/0.238	**72.01/57.31/0.212**	68.91/67.15/0.231	77.98/120.0/0.212	72.10/63.90/0.212
Qwen3-VL-235B	Acc / Cost / Brier	72.13/58.91/0.219	**74.57/52.04/0.195**	73.24/61.08/0.211	79.01/120.0/0.195	72.58/63.90/0.195
			GQA			
Pixtral-12B	Acc / Cost / Brier	64.12/51.23/0.278	**67.37/45.66/0.253**	63.84/49.67/0.271	73.52/120.0/0.253	69.81/63.90/0.253
Llama-4-Maverick	Acc / Cost / Brier	58.23/47.91/0.312	**60.87/41.64/0.287**	57.45/49.38/0.305	63.56/120.0/0.287	63.17/63.90/0.287
Qwen2-VL-72B	Acc / Cost / Brier	68.34/51.28/0.241	**71.03/45.37/0.218**	67.92/53.71/0.235	74.35/120.0/0.218	71.57/63.90/0.218
LLaVA-1.5-7B	Acc / Cost / Brier	68.92/32.45/0.245	**71.18/26.86/0.221**	67.18/35.62/0.239	75.77/120.0/0.221	74.11/63.90/0.221
Qwen2.5-VL-7B	Acc / Cost / Brier	60.18/65.73/0.289	**62.92/59.66/0.265**	58.94/68.42/0.282	67.22/120.0/0.265	60.14/63.90/0.265
Qwen3-VL-235B	Acc / Cost / Brier	66.45/51.84/0.256	**68.28/44.70/0.232**	65.72/48.23/0.249	72.98/120.0/0.232	70.40/63.90/0.232
			FloodNet			
Pixtral-12B	Acc / Cost / Brier	76.84/41.28/0.198	**79.48/34.94/0.176**	77.12/43.56/0.191	81.51/120.0/0.176	75.75/63.90/0.176
Llama-4-Maverick	Acc / Cost / Brier	66.73/45.12/0.267	**69.14/39.68/0.243**	67.21/47.38/0.259	70.35/120.0/0.243	65.23/63.90/0.243
Qwen2-VL-72B	Acc / Cost / Brier	73.18/36.84/0.212	**75.51/30.30/0.189**	73.92/39.12/0.205	76.08/120.0/0.189	78.07/63.90/0.189
LLaVA-1.5-7B	Acc / Cost / Brier	77.45/46.92/0.187	**80.12/40.77/0.165**	78.23/49.18/0.181	82.61/120.0/0.165	73.64/63.90/0.165
Qwen2.5-VL-7B	Acc / Cost / Brier	75.62/40.91/0.203	**78.17/34.85/0.181**	76.18/43.27/0.196	77.19/120.0/0.181	61.24/63.90/0.181
Qwen3-VL-235B	Acc / Cost / Brier	78.34/38.72/0.182	**80.62/32.94/0.161**	79.12/40.91/0.175	83.39/120.0/0.161	82.95/63.90/0.161
			TextVQA			
Pixtral-12B	Acc / Cost / Brier	73.21/49.84/0.221	**75.92/43.09/0.198**	72.84/52.13/0.214	79.29/120.0/0.198	74.82/63.90/0.198
Llama-4-Maverick	Acc / Cost / Brier	62.18/51.23/0.289	**64.52/44.98/0.265**	61.73/53.47/0.282	71.56/120.0/0.265	65.48/63.90/0.265
Qwen2-VL-72B	Acc / Cost / Brier	86.12/58.34/0.142	**88.65/51.64/0.119**	86.84/60.72/0.136	93.11/120.0/0.119	86.84/63.90/0.119

The extended ablation in Table 6.8 confirms findings across all four datasets and includes Brier scores as an explicit measure of calibration quality. The reduction in Brier score from uncalibrated to isotonic calibration (e.g., from 0.256 to 0.231 for Pixtral-12B on VQA-v2) directly correlates with improved accuracy–cost performance, providing empirical support for the theoretical regret bound: better-calibrated probability estimates lead to better VOI decisions.

6.6 Chapter Summary

This chapter framed inference-time decision making as a problem of *dynamic context management* for foundation models. In deployed LLM- and VLM-based systems, workloads consist of heterogeneous queries and tasks while the model weights remain fixed at serving time. Because the model cannot be fine-tuned for each request, performance depends on how effectively the system selects, structures, and retrieves context. We studied this problem at two levels: application-level context, which determines how the task is presented in the prompt, and system-level context, which determines what information is retrieved and at what cost.

Application-level context. OptiSeq demonstrated that the ordering of in-context examples is a critical and underexplored dimension of prompt design. By leveraging example-free log probability scoring—removing the confounding influence of the examples themselves before re-evaluating output likelihood—OptiSeq identifies the ordering that best distinguishes correct outputs, achieving gains of 5.5–10.5 percentage points over baselines across two tasks, five datasets, and five LLMs ranging from 8B to 70B parameters. Crucially, the approach requires no training data, no validation set, and no modification to the underlying model.

System-level context. VOILA demonstrated that different queries require fundamentally different minimum input fidelities, and that this variability is predictable from the query alone before any retrieval occurs. By combining gradient-boosted regression with isotonic calibration and a value-of-information decision rule, VOILA selects the minimum-cost fidelity that preserves accuracy, achieving 50–60% cost reductions while retaining 90–95% of full-resolution accuracy across five datasets and six VLMs from 7B to 235B parameters.

Viewed jointly, these systems show that context is the practical control surface for fixed foundation models at inference time. OptiSeq manages prompt-side context by ordering demonstrations to improve task resolution, while VOILA manages retrieval-side context

by selecting the cheapest sufficient fidelity before inference. In both cases, a lightweight middleware layer converts available signals into runtime routing decisions without modifying model weights.

Taken together, these results suggest that dynamic context management is a general design principle for agentic AI systems. Optimizing both how context is structured in the prompt and what context is retrieved before prompting opens a broad systems design space for improving accuracy, efficiency, and cost at scale.

Chapter 7

Key Takeaways and Future Directions

7.1 Overview

Chapter 1 established that modern machine learning systems must contend with three structurally distinct phases of their lifecycle—*training under data heterogeneity*, *continual adaptation to distributional shift*, and *inference-time resource management*—and that each phase exposes a class of decisions that model design alone cannot resolve. Which participants should be selected in a federated round? When should a deployed model be updated as its environment changes? How should in-context examples be ordered, and what fidelity of input should be retrieved for a given query? These are not algorithmic questions about model architecture; they are system-level decisions that require a principled governing layer.

The central response developed in this dissertation is that a *middleware layer*—sitting between data sources, models, and applications—should govern these decisions in a model-agnostic, scalable, and deployment-compatible way. But what enables middleware to behave sensibly across such varied settings without access to labeled feedback or model internals? The answer, developed fully in Chapter 3, is a single unifying design principle: *a unified*

signal abstraction layer for machine learning systems.

Machine learning systems continuously produce rich intermediate signals during standard training and inference—label distributions capturing participant heterogeneity, latent representations encoding input structure, and log probabilities and calibrated confidence reflecting prediction uncertainty. Despite being readily available, these signals are typically transient and underutilized within the system stack. This dissertation argues that they should be treated as first-class system abstractions. Doing so requires explicitly observing these signals, analyzing them through principled transformations, and routing them through a middleware layer that translates observations into actionable system decisions.

The *observe–analyze–act* methodology, introduced in Chapter 3, operationalizes this principle:

1. **Observe:** Signals are extracted from the standard model and data computations without requiring architectural modification or additional supervision. FLIPS observes label distributions at federated participants [23]; ShiftEx computes latent embeddings and divergence statistics from streaming client data [24]; OptiSeq and VOILA read log-probability distributions and calibrated confidence estimates from LLM and VLM outputs [25, 22].

2. **Analyze:** Raw signals are transformed—through clustering, debiasing, normalization, divergence estimation, or calibration—into structured representations that are reliable and comparable for downstream decision-making. This step is critical: raw signals are often noisy, biased, and distribution-dependent, and cannot be used directly to drive decisions.

3. **Act:** Analyzed signals are mapped to executable system-level decisions that improve efficiency, robustness, or cost-awareness without modifying the underlying model: participant selection policies in FLIPS, expert creation and routing decisions in ShiftEx,

prompt permutation selection in OptiSeq, and retrieval fidelity choices in VOILA.

This framework provides the unifying thread across all four contributions. FLIPS, ShiftEx, OptiSeq, and VOILA differ in domain, model family, and deployment context, but each instantiates the same observe–analyze–act pattern at a different stage of the machine learning lifecycle. The sections that follow summarize what each contribution demonstrated, distill the broader lessons, and chart directions for future work.

7.2 Summary of Contributions

This dissertation instantiated this unified signal abstraction perspective through four systems, each corresponding to a different stage of the machine learning lifecycle.

FLIPS: training-time adaptation through distribution signals. The first contribution, FLIPS, addressed federated learning under data and platform heterogeneity. FLIPS used label distributions as an explicit signal of participant diversity and incorporated that signal into middleware-driven participant selection. By clustering participants based on label distribution and ensuring equitable representation during training, FLIPS showed that middleware can improve convergence and robustness without modifying the underlying learning algorithm. In the language of this thesis, FLIPS demonstrated that even simple distribution-level signals, when elevated to first-class system abstractions, can materially improve how training is orchestrated.

ShiftEx: deployment-time adaptation through representation signals. The second contribution, ShiftEx, extended this signal abstraction perspective to continual adaptation in streaming federated environments. Here, the relevant signals were no longer static label histograms, but evolving latent representations and label-distribution statistics over time. ShiftEx used these signals to detect covariate and label shifts, assign clients to appropriate experts, and adapt the model pool through a middleware layer that coordinated

expert creation, reuse, and consolidation. This contribution showed that deployment-time adaptation requires more than periodic retraining: it requires a system that can observe how the data-generating environment is changing and react accordingly.

OptiSeq: inference-time adaptation through probabilistic signals. The third contribution, OptiSeq, shifted focus to large language models operating in the in-context learning regime. OptiSeq showed that the order of examples in a prompt can substantially influence model behavior, and that this sensitivity can be exploited through inference-time optimization. It used output log probabilities as a signal for comparing candidate example orderings and selecting better prompting configurations without any retraining or additional supervision. This highlighted an important thesis theme: even when the model remains fixed, middleware can still improve performance by interpreting signals emitted during inference and using them to guide decision making.

VOILA: cost-aware multimodal inference through value-of-information signals. The fourth contribution, VOILA, generalized this inference-time view to multimodal systems operating under retrieval and fidelity costs. VOILA used question-conditioned signals to estimate whether a given visual fidelity would be sufficient before retrieval, and then selected the minimum-cost representation that preserved expected utility. This contribution expanded the notion of inference-time adaptation beyond prompt construction to include information acquisition itself. In doing so, it showed that middleware can operate not only on model outputs, but also on decisions about what information should be retrieved and processed in the first place.

A unified systems view. Taken together, these four contributions support a single claim: *signals serve as a unifying abstraction layer for machine learning systems that need to adapt at runtime.* FLIPS, ShiftEx, OptiSeq, and VOILA differ in domain, task, and model family, but they share a common structure. Each observes signals generated by the learning system,

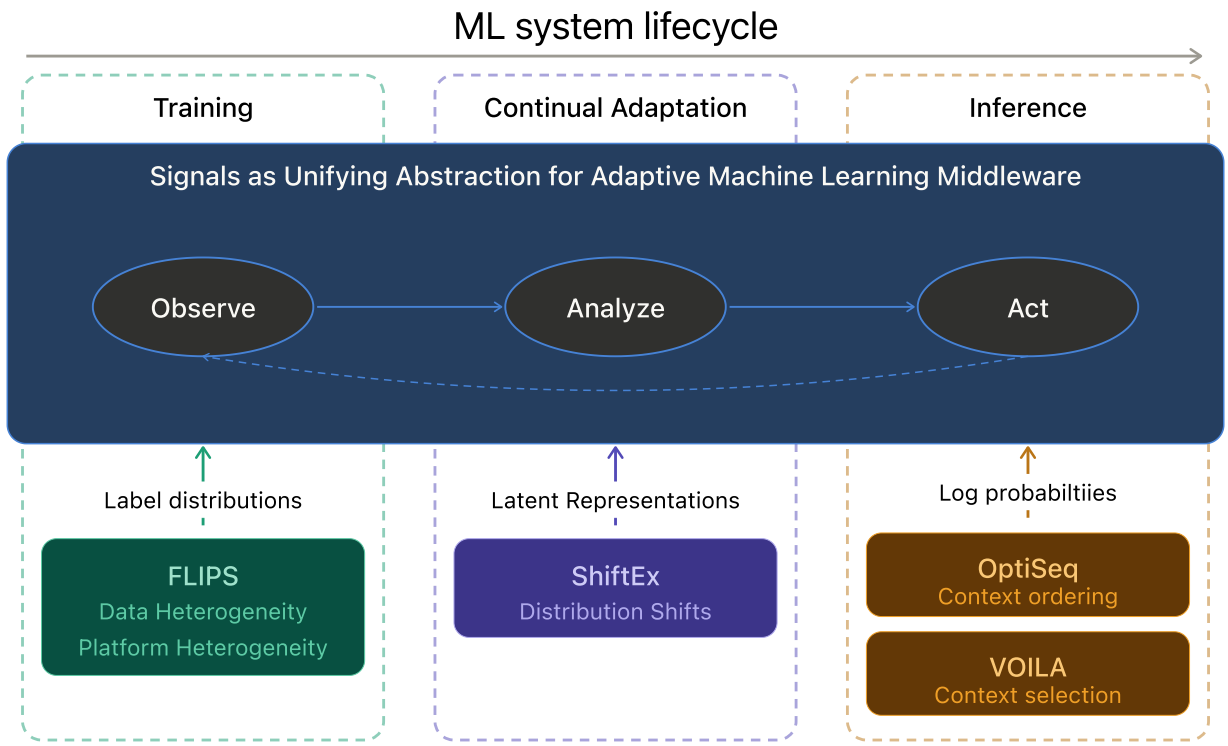


Figure 7.1: Middleware across the ML lifecycle. Dedicated decision layers govern system behavior at each phase—training under data heterogeneity, continual adaptation to distributional shift, and inference-time resource management—enabling model-agnostic policies that improve efficiency, adaptability, and reliability in real-world deployments.

analyzes them into representations suitable for system-level control, and acts on those representations through middleware decisions that improve efficiency, robustness, or adaptability. The result is a thesis that is not simply about federated learning, continual learning, or LLM inference in isolation, but about how machine learning systems can become more self-aware and adaptive by systematically observing, analyzing, and acting on their own internal signals, as shown in Figure 7.1.

7.3 Key Takeaways

This dissertation leads to several broader takeaways about the design of adaptive machine learning systems.

Signal abstractions for system-level decisions. Internal system artifacts—label distributions, embeddings, latent statistics, log probabilities, and calibrated confidence scores—are produced during standard training and inference at little additional cost, yet they are rarely surfaced as reusable decision primitives. This work demonstrates that these signals, when elevated to first-class abstractions and paired with a dedicated analysis layer, can support meaningful system-level interventions across training, deployment, and inference. The insight is not that any single signal type is universally powerful, but that systematically observing and acting on what the model already produces is a principled path to adaptive behavior.

Middleware is the right architectural layer for signal abstraction. Embedding adaptation logic directly into the model reduces modularity, couples system-level concerns to the model's internals, and requires repeated retraining as operational requirements change. Middleware, by contrast, is model-agnostic: it observes signals exposed through standard interfaces, applies transformation and analysis logic independently of the learner, and delivers decisions that govern system behavior without modifying weights or architectures. This separation of concerns makes signal-aware systems easier to compose, deploy across heterogeneous model families, and update as deployment conditions evolve.

Raw signals require transformation to become reliable. A consistent finding across all four systems is that raw signals cannot be used directly. Label distributions require normalization and clustering before they can drive equitable participant selection. Latent embeddings require divergence estimation to serve as stable shift indicators. Log probabilities require debiasing to correct for positional and length effects before they can rank candidate outputs fairly. In each case, the transformation step is not incidental—it is the core technical contribution that makes the signal actionable. Designing principled transformations tailored to each signal type is a fundamental engineering challenge in building reliable signal-driven decision layers.

Adaptation must be understood as a lifecycle property, not a single-stage concern. The challenges addressed in this dissertation emerge at structurally different moments—training under data heterogeneity, deployment under non-stationarity, and inference under prompt sensitivity and resource constraints—yet they share a common structure. All can be addressed by observing internal signals, analyzing them into reliable decision representations, and acting on the resulting system behavior accordingly. A unified signal abstraction perspective reveals these as manifestations of the same underlying design problem, and the observe–analyze–act abstraction as a common solution framework that applies across all three lifecycle phases.

Resource-awareness must be a first-class optimization objective. Accuracy alone is an insufficient design criterion for real-world deployments. Practical systems must contend with communication cost, latency budgets, retrieval bandwidth, memory footprint, and monetary expenditure—constraints that interact in non-trivial ways with model behavior. The contributions in this dissertation show that internal signals provide a natural bridge between model behavior and these operational concerns. OptiSeq improves prompt accuracy without any additional training; VOILA maintains near-full-resolution accuracy while cutting retrieval cost by 50–60%. Treating resource-awareness as a first-class objective, informed by signal-driven middleware, yields systems that are not only accurate but deployable at scale.

Privacy and decentralization are compatible with effective signal-driven adaptation. A distinctive property of a unified signal abstraction layer is that it can operate on aggregated or encoded representations—label distribution histograms, embedding statistics, divergence scores—rather than raw data. FLIPS and ShiftEx demonstrate that powerful adaptation decisions can be made in federated settings without centralizing participant data, relying solely on summary statistics that carry minimal private information. This compatibility is not a side benefit; it is a necessary condition for deploying adaptive middleware in healthcare, financial services, and edge computing domains where raw data cannot be

shared.

The observe–analyze–act framework is a reusable systems design pattern. Although instantiated across four systems in specific settings, the observe–analyze–act framework does not depend on any particular model class, domain, or deployment paradigm. Distribution-level signals are relevant wherever data heterogeneity exists; representation-level shift indicators apply to any model with an accessible embedding space; log-probability-based decision-making applies to any model that produces output distributions. This generality suggests that the unified signal abstraction layer is not a collection of domain-specific heuristics but a principled and reusable design pattern for adaptive machine learning systems broadly.

7.4 Practical Future Directions

The contributions of this dissertation establish a principled framework for lifecycle-aware decision making in machine learning systems. At the same time, they surface several concrete next steps that are technically realistic, immediately useful, and well aligned with how modern ML systems are already deployed.

7.4.1 Operationalizing a Unified Signal Abstraction Layer for ML Systems

Each system in this dissertation—FLIPS, ShiftEx, OptiSeq, and VOILA—defines its own signal types, extraction pipelines, and analysis logic in isolation. A natural next step is to consolidate these capabilities under a shared infrastructure: a *unified signal abstraction layer* that standardizes how system-level observations are registered, calibrated, cached, versioned, and consumed. Recent work already points in this direction from multiple angles: internal representation signals can guide tool invocation, task-aware cache policies can shrink

long-context memory costs, and output-distribution signals can drive inference-time compute allocation without task-specific retraining [116, 259, 87]. What remains missing is the systems interface that exposes these signals once and lets multiple downstream policies reuse them.

In practical terms, such a layer should provide four concrete services: a signal registry with typed schemas, calibration and normalization modules that make signals comparable across models, privacy-preserving summarization for distributed settings, and offline replay tools for testing policies before deployment. This is a tractable systems problem rather than a speculative one. Much like feature stores and observability stacks became standard once production ML matured, a shared signal layer can make adaptive policies reusable across training, deployment, and inference without forcing every new project to rebuild the same plumbing from scratch.

7.4.2 Context Budgeting and Tool Invocation for Agentic Systems

One of the most pressing open problems in deployed language systems is context management under hard token, memory, and latency budgets. Current approaches—truncation, sliding windows, or hand-crafted summarization—are often static and only weakly connected to what the model is actually using. The observe–analyze–act framework offers a more practical alternative: use runtime signals to decide what context to retain, what to compress, and when external tools are actually worth invoking.

Recent results suggest that this direction is already practical. Task-aware KV-cache compression methods now adapt memory retention to the active task, and reusable compressed caches can preserve broad knowledge while lowering repeated retrieval and long-context costs [259, 44]. At the same time, long-context studies show that preserving instruction fidelity and the right contextual cues remains a first-order concern, not an afterthought [186]. A concrete next step for this thesis is therefore not a fully autonomous memory manager, but a lightweight controller that scores context segments, preserves high-value instructions

and evidence, and compacts lower-value history only when budgets become tight.

The same reasoning extends directly to tool use. Rather than invoking retrieval, search, or code execution by default, the system can estimate whether the current uncertainty profile justifies the call. Recent work on adaptive tool-use triggers supports exactly this kind of selective invocation policy [116]. This is a natural and usable extension of VOILA: replace fidelity selection over visual inputs with budgeted decisions over retrieval depth, tool calls, and context expansion, all driven by signals the model already emits during inference.

7.4.3 Distribution-Grounded Model Routing and Escalation

A third concrete direction concerns model selection at serving time: given a query, which model in a heterogeneous pool should process it? This is now a practical deployment problem rather than a hypothetical one, as many systems rely on mixtures of small, fast models and larger, more expensive ones. Recent routing systems already optimize quality, latency, and cost trade-offs over heterogeneous model pools [224, 198].

A natural extension of ShiftEx is to ground routing decisions in distributional similarity rather than in hand-tuned heuristics alone. Instead of learning a router from expensive labeled comparisons for every new model pool, a deployment can maintain embedding summaries for each model's strong operating region and compare incoming traffic to those summaries using the same style of divergence estimates used for expert assignment in ShiftEx. In practice, this can begin as a simple escalation policy: handle easy, in-distribution queries with cheaper models, and escalate when confidence drops or when the query distribution drifts away from the region where a smaller model is reliable.

The near-term opportunity here is not a fully general marketplace of autonomous experts, but an auditable routing layer over a small number of models with different cost-quality profiles. That makes the evaluation story much clearer: one can measure routing regret,

escalation frequency, latency, and failure recovery directly, while still keeping the core thesis contribution intact—signals and distribution summaries provide the right abstraction for making these decisions without entangling them with model internals.

7.5 Reflection

A broader message of this dissertation is that machine learning systems research should move beyond a purely model-centric view. As models become larger, more capable, and more expensive to retrain, many practical improvements will come not from modifying the model itself, but from designing better system layers around it. This is especially true in real-world deployments, where the key challenges are often not architectural limitations, but the need to operate robustly under heterogeneity, shift, uncertainty, and cost constraints.

This dissertation showed that internal signals provide an important foundation for this systems view. They make it possible to reason about what the model is seeing, how the environment is changing, how confident the system is, and what information is worth acquiring. Middleware, in turn, provides the mechanism for translating those observations into action. In that sense, the central contribution of this dissertation is not only a set of four systems, but a design philosophy: *adaptive machine learning systems should be built by coupling expressive models with a unified signal abstraction layer that can observe, analyze, and adapt through their internal signals.*

7.6 Closing Remarks

In summary, this dissertation presented a unified signal abstraction framework for adaptive machine learning systems. Across federated training, continual adaptation, and inference-time optimization, it demonstrated that modern machine learning systems expose rich signals that, when properly analyzed and calibrated, yield reliable decision policies. By elevating

those signals into a dedicated decision layer, this work provides a path toward machine learning systems that are not only more accurate, but also more efficient, robust, cost-aware, and responsive to changing real-world conditions.

More broadly, this thesis argues for a shift in how we think about machine learning systems: from static pipelines built around fixed models to adaptive systems that continually observe themselves and respond. A unified signal abstraction layer for ML systems is the abstraction that makes this shift possible.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *CCS '16*, 2016.

[3] A. A. Abdullah, A. Zubiaga, S. Mirjalili, A. H. Gandomi, F. Daneshfar, M. Amini, A. S. Mohammed, and H. Veisi. Evolution of meta's llama models and parameter-efficient fine-tuning of large language models: a survey, 2025.

[4] R. Agarwal, A. Singh, L. M. Zhang, B. Bohnet, L. Rosias, S. Chan, B. Zhang, A. Anand, Z. Abbas, A. Nova, et al. Many-shot in-context learning. *arXiv preprint arXiv:2404.11018*, 2024.

[5] P. Agrawal, S. Antoniak, E. B. Hanna, B. Bout, D. Chaplot, J. Chudnovsky, D. Costa, B. De Monicault, S. Garg, T. Gervet, et al. Pixtral 12b. *arXiv preprint arXiv:2410.07073*, 2024.

[6] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. Fernandez-Moctezuma, R. Lax, S. McVeety, D. Mills, F. Perry, E. Schmidt, et al. The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of the VLDB Endowment*, 8(12):1792–1803, 2015.

[7] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[8] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 144–161, 2018.

[9] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11849–11860, 2019.

[10] A. Allred and A. Arauz. Testing caredex: Sagat methodology deployed for a disaster drill at a senior living community. *Issues in Information Systems*, 2023.

[11] Amazon Web Services. Amazon s3 storage classes. `https://docs.aws.amazon.com/AmazonS3/latest/userguide/storage-class-intro.html`, 2024.

[12] M. Armbrust, T. Das, J. Torres, B. Yavuz, R. S. Xin, A. Ghodsi, I. Stoica, and M. Zaharia. Structured streaming: A declarative api for real-time applications in apache spark. In *Proceedings of the 2018 International Conference on Management of Data*, pages 601–613, 2018.

[13] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[14] M. Atre, B. Jha, and A. Rao. Distributed deep learning using volunteer computing-like paradigm. In *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 933–942, 2021.

[15] S. Bae, Y. Kim, R. Bayat, S. Kim, J. Ha, T. Schuster, A. Fisch, H. Harutyunyan, Z. Ji, A. Courville, and S.-Y. Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025.

[16] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[17] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. *Proc. VLDB Endow.*, 5(7):622–633, mar 2012.

[18] J. Bai, S. Bai, S. Yang, S. Wang, S. Tan, P. Wang, J. Lin, C. Zhou, and J. Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023.

[19] P. Bellavista, L. Foschini, and A. Mora. Decentralised learning in federated deployment environments. *ACM Computing Surveys*, 54(1):1–38, 2021.

[20] T. Ben-Nun and T. Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys*, 52(4):1–43, 2019.

[21] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[22] R. A. Bhope, K. R. Jayaram, V. Muthusamy, R. Kumar, V. Isahagian, and N. Venkatasubramanian. Voila: Value-of-information guided fidelity selection for cost-aware multimodal question answering, 2026.

[23] R. A. Bhope, K. R. Jayaram, N. Venkatasubramanian, A. Verma, and G. Thomas. Flips: Federated learning using intelligent participant selection. In *Proceedings of the 24th International Middleware Conference*, Middleware '23, page 301–315, New York, NY, USA, 2023. Association for Computing Machinery.

[24] R. A. Bhope, K. R. Jayaram, P. Venkateswaran, and N. Venkatasubramanian. Shift happens: Mixture of experts based continual adaptation in federated learning. In *Proceedings of the 26th International Middleware Conference*, Middleware '25, page 455–468, New York, NY, USA, 2025. Association for Computing Machinery.

[25] R. A. Bhope, P. Venkateswaran, K. R. Jayaram, V. Isahagian, V. Muthusamy, and N. Venkatasubramanian. Optiseq: Ordering examples on-the-fly for in-context learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

[26] M. Bilgic and L. Getoor. Value of information lattice: Exploiting probabilistic independence for effective feature subset acquisition. *Journal of Artificial Intelligence Research*, 41:69–95, 2011.

[27] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 118–128, 2017.

[28] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

[29] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. ACM, 2017.

[30] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[31] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara. Dark experience for general continual learning: A strong, simple baseline. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[32] T. Cai, Y. Li, S. Goyal, H. Cheng, Y. Mei, W. Zeng, D. Chen, et al. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

[33] W. Cai, J. Jiang, F. Wang, J. Tang, S. Kim, and J. Huang. A survey on mixture of experts. *arXiv preprint arXiv:2407.06204*, 2024.

[34] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar. Leaf: A benchmark for federated settings, 2019.

[35] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas. Apache flink: Stream and batch processing in a single engine. *The Bulletin of the Technical Committee on Data Engineering*, 38(4), 2015.

[36] F. E. Casado, D. Lema, M. F. Criado, R. Iglesias, C. V. Regueiro, and S. Barro. Concept drift detection and adaptation for federated and continual learning. *Multimedia Tools and Applications*, pages 1–23, 2022.

[37] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala. Fedat: A high-performance and communication-efficient federated learning system with asynchronous tiers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery.

[38] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations (ICLR)*, 2019. Poster.

[39] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[40] L. Chen, M. Zaharia, and J. Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

[41] L. Chen, H. Zhao, T. Liu, S. Bai, J. Lin, C. Zhou, and B. Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision (ECCV)*, 2024.

[42] Y. J. Cho, J. Wang, and G. Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

[43] G. Christie, N. Fendley, J. Wilson, and R. Mukherjee. Functional map of the world, 2018.

[44] G. Corallo, O. Weller, F. Petroni, and P. Papotti. CacheNotes: Task-aware key-value cache compression for reasoning-intensive knowledge tasks. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6571–6590, Rabat, Morocco, 2026. Association for Computational Linguistics.

[45] B. Cox, L. Y. Chen, and J. Decouchant. Aergia: Leveraging heterogeneity in federated learning systems. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference*, Middleware '22, page 107–120, New York, NY, USA, 2022. Association for Computing Machinery.

[46] M. F. Criado, F. E. Casado, R. Iglesias, C. V. Regueiro, and S. Barro. Non-iid data and continual learning processes in federated learning: A long road ahead. *Information Fusion*, 88:263–280, 2022.

[47] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[48] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

[49] M. Dehghani and Z. Yazdanparast. From distributed machine to distributed deep learning: A comprehensive survey. *Journal of Big Data*, 10(1):1–21, 2023.

[50] Y. Deng, F. Lyu, J. Ren, Y. Zhang, Y. Zhou, Y. Zhang, and Y. Yang. Share: Shaping data distribution at edge for communication-efficient hierarchical federated learning. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 24–34, 2021.

[51] D. Ding, A. Mallick, C. Wang, R. Sim, S. Mukherjee, V. Rühle, L. V. S. Lakshmanan, and A. H. Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[52] M. Diskin, A. Bukhtiyarov, M. Ryabinin, L. Saulnier, Q. Lhoest, A. Sinitsin, D. Popov, D. Pyrkin, M. Kashirin, A. Borzunov, A. Villanova del Moral, D. Mazur, I. Kobelev, Y. Jernite, T. Wolf, and G. Pekhimenko. Distributed deep learning in open collaborations. *arXiv preprint arXiv:2106.10207*, 2021.

[53] Z. Du, J. Sun, A. Li, P.-Y. Chen, J. Zhang, H. H. Li, and Y. Chen. Rethinking normalization methods in federated learning. In *Proceedings of the 3rd International Workshop on Distributed Machine Learning*, pages 16–22, 2022.

[54] M. Elhoushi, A. Shrivastava, D. Liskovich, B. Hosmer, B. Wasti, L. Lai, A. Mahmoud, B. Acun, S. Agarwal, A. Roman, A. A. Aly, B. Chen, and C.-J. Wu. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[55] M. Elzohairy, M. Chadha, A. Jindal, A. Grafberger, J. Gu, M. Gerndt, and O. Abboud. Fedlesscan: Mitigating stragglers in serverless federated learning, 2022.

[56] S. Es, J. James, J. Espinosa, and et al. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2024.

[57] S. Fan, Y. Rong, C. Meng, Z. Cao, S. Wang, Z. Zheng, C. Wu, G. Long, J. Yang, L. Xia, L. Diao, X. Liu, and W. Lin. DAPPLE: A pipelined data parallel approach for training large models. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 431–445, 2021.

[58] M. Farazi, S. Khan, and N. Barnes. Accuracy vs. complexity: a trade-off in visual question answering models. *Pattern Recognition*, 120:108106, 2021.

[59] W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[60] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3407–3416. PMLR, 18–24 Jul 2021.

[61] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[62] C. Fu et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023.

[63] Q. Fu, M. Cho, et al. Lazyllm: Dynamic token pruning for efficient long context llm inference. *arXiv preprint arXiv:2407.14057*, 2024.

[64] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):1–37, 2014.

[65] S. Gan, X. Lian, R. Wang, J. Chang, C. Liu, H.-f. Shi, S. Zhang, X. Li, T. Sun, J. Jiang, B. Yuan, S. Yang, J. Liu, and C. Zhang. BAGUA: Scaling up distributed learning with system relaxations. *arXiv preprint arXiv:2107.01499*, 2021.

[66] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. S. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

[67] Z. Gao, Z. Zhang, Y. Zhang, T. Wang, Y. Gong, and Y. Guo. Online client scheduling and resource allocation for efficient federated edge learning. *arXiv preprint arXiv:2410.10833*, 2024.

[68] N. Garg. *Apache kafka*. Packt Publishing, 2013.

[69] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran. An efficient framework for clustered federated learning. *arXiv preprint arXiv:2006.04088*, 2020.

[70] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017.

[71] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[72] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

[73] Q. Guo, L. Wang, Y. Wang, W. Ye, and S. Zhang. What makes a good order of examples in in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14892–14904, 2024.

[74] P. Hamedi, R. Razavi-Far, and E. Hallaji. Federated continual learning: Concepts, challenges, and solutions, 2025.

[75] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[76] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[77] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[78] E. H. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C.-Y. Lin. Question answering in webclopedia. In *TREC*, volume 52, pages 53–56, 2000.

[79] Y. Hua, K. Miller, A. Bertozzi, C. Qian, and B. Wang. Efficient and reliable overlay networks for decentralized federated learning. *arXiv preprint arXiv:2112.15486*, 2021.

[80] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2018.

[81] K. Huang et al. Ivtp: Instruction-guided visual token pruning for large vision-language models. In *European Conference on Computer Vision (ECCV)*, 2024.

[82] D. A. Hudson and C. D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[83] IBM Granite Team. Granite 3.0 language models. `https://github.com/ibm-granite/granite-3.0-language-models`, 2024. Technical Report, IBM Research.

[84] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[85] H. Jiang, Q. Wu, X. Luo, D. Li, C.-Y. Lin, Y. Yang, and L. Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[86] W. Jitkrittum, H. Narasimhan, A. S. Rawat, J. Juneja, Z. Wang, C.-Y. Lee, P. Shenoy, R. Panigrahy, A. K. Menon, and S. Kumar. Universal model routing for efficient llm inference. *arXiv preprint arXiv:2502.08773*, 2025.

[87] S.-h. Jo, Y. Ko, S.-g. Lee, and J. Seol. Task-aware block pruning with output distribution signals for large language models. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 6089–6107, Rabat, Morocco, 2026. Association for Computational Linguistics.

[88] E. Jothimurugesan, K. Hsieh, J. Wang, G. Joshi, and P. B. Gibbons. Federated learning under distributed concept drift, 2023.

[89] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. A. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, O. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. X. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and open problems in federated learning. *ArXiv*, abs/1912.04977, 2019.

[90] S. Kalra, J. Wen, J. C. Cresswell, M. Volkovs, and H. Tizhoosh. Decentralized federated learning through proxy model sharing. *Nature Communications*, 14, 2023.

[91] S. P. Karimireddy et al. Scaffold: Stochastic controlled averaging for federated learning. In *ICML*, 2020.

[92] K. Kärkkäinen, M. Kachuee, O. Goldstein, and M. Sarrafzadeh. Cost-sensitive feature-value acquisition using feature relevance. *arXiv preprint arXiv:1912.08281*, 2019.

[93] A. R. Khouas, M. R. Bouadjenek, H. Hacid, and S. Aryal. Training machine learning models at the edge: A survey. *arXiv preprint arXiv:2403.02619*, 2024.

[94] D. Kiela, M. Bartolo, Y. Nie, D. Kaushik, A. Geiger, Z. Wu, B. Vidgen, G. Prasad, A. Singh, P. Ringshia, et al. Dynabench: Rethinking benchmarking in nlp. In *Proceedings of the 2021 conference of the North American chapter of the Association for Computational Linguistics: human language technologies*, pages 4110–4124, 2021.

[95] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[96] G. Kollias, T. Salonidis, and S. Wang. Sketch to skip and select: Communication efficient federated learning using locality sensitive hashing. In R. Goebel, H. Yu, B. Faltings, L. Fan, and Z. Xiong, editors, *Trustworthy Federated Learning*, pages 72–83, Cham, 2023. Springer International Publishing.

[97] J. Kossen, C. Cangea, E. Vértes, A. Jaegle, V. Patraucean, I. Ktena, N. Tomasev, and D. Belgrave. Active acquisition for multimodal temporal data: a challenging decision-making task. *arXiv preprint arXiv:2211.05039*, 2022.

[98] S. Kullback. Kullback-leibler divergence, 1951.

[99] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

[100] F. Lai et al. Oort: Efficient federated learning via guided participant selection. In *OSDI*, 2021.

[101] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury. Efficient federated learning via guided participant selection. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2021.

[102] M. Langer, Z. He, W. Rahayu, and Y. Xue. Distributed training of deep learning models: A taxonomic perspective. *IEEE Transactions on Parallel and Distributed Systems*, 31(12):2802–2818, 2020.

[103] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.

[104] C. Legislature. California consumer privacy act of 2018, AB 375. `https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201720180AB375`, 2018. Accessed: 2026-04-22.

[105] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[106] B. Li et al. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023.

[107] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[108] J. Li, H.-Y. Xu, Y. Zhu, Z. Liu, C. Guo, and C. Wang. Lyra: Elastic scheduling for deep learning clusters. In *Proceedings of the Eighteenth European Conference on Computer Systems*, 2023.

[109] M. Li, D. G. Andersen, J. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 583–598, 2014.

[110] M. Li, D. Avdiukhin, R. Shahout, N. Ivkin, V. Braverman, and M. Yu. Federated learning clients clustering with adaptation to data drifts, 2024.

[111] M. Li, L. Zhou, Z. Yang, A. Li, D. Andersen, and A. Smola. Parameter server for distributed machine learning, 2013. Preprint.

[112] S. Li, C. Gong, Y. Zhu, C. Luo, Y. Hong, and X. Lv. Context-aware multi-level question embedding fusion for visual question answering. *Information Fusion*, 102:102000, 2024.

[113] S. Li, S. Hou, B. Buyukates, and S. Avestimehr. Secure federated clustering, 2022.

[114] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*, 2020.

[115] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[116] W. Li, D. Li, K. Dong, C. Zhang, H. Zhang, W. Liu, Y. Wang, R. Tang, and Y. Liu. Adaptive tool use in large language models with meta-cognition trigger. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13346–13370, Vienna, Austria, 2025. Association for Computational Linguistics.

[117] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.

[118] X.-C. Li and D.-C. Zhan. Fedrs: Federated learning with restricted softmax for label distribution non-iid data. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 995–1005, 2021.

[119] Y. Li, B. Dong, C. Lin, and F. Guerin. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*, 2023.

[120] Y. Li, Y. Huang, B. Yang, B. Venkitesh, A. Locatelli, H. Ye, T. Cai, P. Lewis, and D. Chen. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*, 2024.

[121] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018.

[122] Y. Li, Y. Wang, H. Wang, Y. Qi, T. Xiao, and R. Li. Fedssi: Rehearsal-free continual federated learning with synergistic synaptic intelligence. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 36151–36169. PMLR, 2025.

[123] Z. Li and D. Hoiem. Learning without forgetting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 614–629, 2016.

[124] Z. Li, C. Li, M. Zhang, Q. Mei, and M. Bendersky. Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach. *arXiv preprint arXiv:2407.16833*, 2024.

[125] Z. Li, C. Yan, X. Zhang, G. Gharibi, Z. Yin, X. Jiang, and B. Malin. Split learning for distributed collaborative training of deep learning models in health informatics. *AMIA Annual Symposium Proceedings*, pages 1047–1056, 2023.

[126] F. Liang, Z. Zhang, H. Lu, C. Li, V. C. M. Leung, Y. Guo, and X. Hu. Resource allocation and workload scheduling for large-scale distributed deep learning: A survey. *arXiv preprint arXiv:2406.08115*, 2024.

[127] J. Liang, D. Hu, and J. Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 6028–6039, 2020.

[128] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[129] B. Lim. *Resource Allocation for Federated Learning Enabled Edge Intelligence*. PhD thesis, Nanyang Technological University, 2022.

[130] Z. Lin, G. Qu, X. Chen, and K. Huang. Split learning in 6g edge networks. *IEEE Wireless Communications*, 31(2):170–176, 2024.

[131] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[132] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

[133] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[134] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[135] Y. Liu, S. Chang, and Y. Liu. Fedcs: Communication-efficient federated learning with compressive sensing. In *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 17–24, 2023.

[136] Y. Liu, J. Liu, X. Shi, Q. Cheng, Y. Huang, and W. Lu. Let's learn step by step: Enhancing in-context learning ability with curriculum learning. *arXiv preprint arXiv:2402.10738*, 2024.

[137] C. Lohrmann et al. Elastic stream processing with latency guarantees. In *IEEE ICDE*, 2015.

[138] S. Lonn, P. Radeva, and M. Dimiccoli. Smartphone picture organization: A hierarchical approach, 2019.

[139] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6467–6476, 2017.

[140] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

[141] Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021.

[142] F. Lyu, D. Liu, L. Zhao, Z. Zhang, F. Shang, F. Hu, W. Feng, and L. Wang. Overcoming domain drift in online continual learning. *arXiv preprint arXiv:2405.09133*, 2024.

[143] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin. A state-of-the-art survey on solving non-iid data in federated learning. *Future Generation Computer Systems*, 135:244–258, 2022.

[144] A. Maharana, D.-H. Lee, S. Tulyakov, M. Bansal, F. Barbieri, and Y. Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

[145] Z. Mai, R. Li, J. Jeong, D. Quispe, H. J. Kim, and S. Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

[146] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7765–7773, 2018.

[147] O. Marfoq, G. Neglia, L. Kameni, and R. Vidal. Federated learning for data streams, 2023.

[148] S. Masoudnia and R. Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42:275–293, 2014.

[149] R. Mayer and H.-A. Jacobsen. Scalable deep learning on distributed infrastructures: Challenges, techniques, and tools. *ACM Computing Surveys*, 53(1):1–37, 2019.

[150] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press, 1989.

[151] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[152] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy*, pages 691–706. IEEE, 2019.

[153] M. L. Menéndez, J. A. Pardo, L. Pardo, and M. d. C. Pardo. The jensen-shannon divergence. *Journal of the Franklin Institute*, 334(2):307–318, 1997.

[154] M. Merluzzi, P. Di Lorenzo, and S. Barbarossa. Wireless edge machine learning: Resource allocation and trade-offs. *IEEE Access*, 9:45377–45398, 2021.

[155] G. Moody and R. Mark. The impact of the mit-bih arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.

[156] K. O. Mortensen, F. Zardbani, M. A. Haque, S. Y. Agustsson, D. Mottin, P. Hofmann, and P. Karras. Marigold: Efficient k-means clustering in high dimensions. *Proc. VLDB Endow.*, 16(7):1740–1748, may 2023.

[157] NVIDIA. FasterTransformer: Transformer related optimization. `https://github.com/NVIDIA/FasterTransformer`, 2022. GitHub repository.

[158] NVIDIA. TensorRT-LLM: Optimized large language model inference on NVIDIA GPUs. `https://github.com/NVIDIA/TensorRT-LLM`, 2023. GitHub repository.

[159] H. Oh, J. Lee, H. Kim, and J. Seo. Scheduling optimization techniques for neural network training. *arXiv preprint arXiv:2110.00929*, 2021.

[160] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

[161] S. Ouyang, D. Dong, Y. Xu, and L. Xiao. Communication optimization strategies for distributed deep learning: A survey. *arXiv preprint arXiv:2003.03009*, 2020.

[162] C. Packer, S. Wooders, K. Lin, V. Fang, S. G. Patil, I. Stoica, and J. E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2024.

[163] S. Pal, E. Ebrahimi, A. Zulfiqar, Y. Fu, V. Zhang, S. Migacz, D. Nellans, and P. Gupta. Optimizing multi-gpu parallelization strategies for deep learning training. *IEEE Micro*, 39(5):91–101, 2019.

[164] J. Pang, H. Du, Z. Liu, X. Xu, and Y. Feng. Feddcs: Dynamical client selection for federated learning in mobile scenarios with label distribution shift. In *2024 10th International Conference on Big Data Computing and Communications (BigCom)*, pages 103–110. IEEE, 2024.

[165] J. Park, S. Samarakoon, A. Elgabli, J. Kim, M. Bennis, S.-L. Kim, and M. Debbah. Communication-efficient and distributed learning over wireless networks: Principles and applications. *Proceedings of the IEEE*, 109(5):796–819, 2021.

[166] J.-G. Park, D.-J. Han, M. Choi, and J. Moon. Sageflow: Robust federated learning against both stragglers and adversaries. In *Neural Information Processing Systems*, 2021.

[167] J. H. Park, G. Yun, C. Yi, N. T. Nguyen, S. Lee, J. Choi, S. Noh, and Y.-r. Choi. HetPipe: Enabling large DNN training on (whimpy) heterogeneous GPU clusters through integration of pipelined model parallelism and data parallelism. *arXiv preprint arXiv:2005.14038*, 2020.

[168] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA, 2023. Association for Computing Machinery.

[169] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard, et al. Kilt: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, 2021.

[170] R. Presotto, G. Civitarese, and C. Bettini. Fedclar: Federated clustering for personalized sensor-based human activity recognition. In *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 227–236, 2022.

[171] X. Qiang, Z. Chang, Y. Hu, L. Liu, and T. Hämäläinen. Adaptive and parallel split federated learning in vehicular edge computing. *IEEE Internet of Things Journal*, 12(5):4591–4604, 2025.

[172] A. Qiao, W. Neiswanger, Q. Ho, H. Zhang, G. Ganger, and E. Xing. Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning. *arXiv preprint arXiv:2008.12260*, 2020.

[173] Y. Qin, C.-I. Lao, Y. Le, W. Wu, and K. Chen. Efficient data-plane memory scheduling for in-network aggregation. *arXiv preprint arXiv:2201.06398*, 2022.

[174] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

[175] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

[176] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[177] M. Rahnemoonfar, T. Chowdhury, A. Sarkar, D. Varshney, M. Yari, and R. R. Murphy. Floodnet: A high resolution aerial imagery dataset for post flood scene understanding. *IEEE Access*, 9:89644–89654, 2021.

[178] S. Rajasekaran, M. Ghobadi, and A. Akella. CASSINI: Network-aware job scheduling in machine learning clusters. *arXiv preprint arXiv:2308.00852*, 2023.

[179] S. Rajasekaran, M. Ghobadi, G. Kumar, and A. Akella. Congestion control in machine learning clusters. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, 2022.

[180] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pages 1–16. IEEE, 2020.

[181] A. Ramezani-Kebrya, M. Zhang, and C.-J. Hsieh. Federated learning under covariate shifts with generalization guarantees. *arXiv preprint arXiv:2303.14843*, 2023.

[182] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, 2017.

[183] S. Reddi et al. Adaptive federated optimization. In *ICLR*, 2021.

[184] N. Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[185] A. Reisizadeh, I. Tziotis, H. Hassani, A. Mokhtari, and R. Pedarsani. Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity. *IEEE Journal on Selected Areas in Information Theory*, 3(2):197–205, 2022.

[186] P. K. Robinette, A. Hard, S. Ramaswamy, E. Amid, R. Mathews, and T. T. Johnson. We are what we repeatedly do: Improving long context instruction following. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 4855–4884, Rabat, Morocco, 2026. Association for Computational Linguistics.

[187] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 348–358, 2019.

[188] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia. Ares: An automated evaluation framework for retrieval-augmented generation systems. *arXiv preprint arXiv:2311.09476*, 2024.

[189] R. Schlegel, S. Kumar, E. Rosnes, and A. G. i. Amat. Codedpaddedfl and codedsecagg: Straggler mitigation and secure aggregation in federated learning. *IEEE Transactions on Communications*, 71(4):2013–2027, 2023.

[190] T. Schuster, A. Fisch, J. Gupta, M. Dehghani, D. Bahri, V. Tran, Y. Tay, and D. Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

[191] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Interspeech 2014*, pages 1058–1062, 2014.

[192] H. Shi and H. Wang. A unified approach to domain incremental learning with memory: Theory and algorithm. *arXiv preprint arXiv:2310.12244*, 2023.

[193] S. Shi, Z. Tang, X. Chu, C. Liu, W. Wang, and B. Li. Communication-efficient distributed deep learning: Survey, evaluation, and challenges. *arXiv preprint arXiv:2005.13247*, 2020.

[194] T. Shnitzer, A. Ou, M. Silva, K. Soule, Y. Sun, J. Solomon, N. Thompson, and M. Yurochkin. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*, 2023.

[195] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[196] A. Singh, V. Natarajan, M. Shah, Y. Jiang, X. Chen, D. Batra, D. Parikh, and M. Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019.

[197] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[198] W. Song, Z. Huang, C. Cheng, W. Gao, B. Xu, G. Zhao, F. Wang, and R. Wu. IRT-router: Effective and interpretable multi-LLM routing via item response theory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15629–15644, Vienna, Austria, 2025. Association for Computational Linguistics.

[199] Y. Song, W. Xiong, D. Zhu, W. Wu, H. Qian, M. Song, H. Huang, C. Li, K. Wang, R. Yao, et al. Restgpt: Connecting large language models with real-world restful apis. *arXiv preprint arXiv:2306.06624*, 2023.

[200] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*, 2023.

[201] P. Sun, W. Feng, R. Han, S. Yan, and Y. Wen. Optimizing network performance for distributed dnn training on gpu clusters: Imagenet/alexnet training in 1.5 minutes. *arXiv preprint arXiv:1902.06855*, 2019.

[202] W. Sun, L. Yan, X. Ma, S. Wang, P. Ren, Z. Chen, D. Yin, and Z. Ren. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[203] Y. Sun, X. Wang, Z. Liu, J. Miller, A. A. Efros, and M. Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 9229–9248, 2020.

[204] A. Z. Tan, H. Yu, L. Cui, and Q. Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–17, 2022.

[205] Y. Tan, C. Chen, W. Zhuang, X. Dong, L. Lyu, and G. Long. Is heterogeneity notorious? taming heterogeneity to handle test-time shift in federated learning. *Advances in Neural Information Processing Systems*, 36:27167–27180, 2023.

[206] Z. Tang, S. Shi, X. Chu, W. Wang, and B. Li. Communication-efficient distributed deep learning: A comprehensive survey. *arXiv preprint arXiv:2003.06307*, 2020.

[207] M. Threadgill and A. Gerstlauer. A survey of distributed learning in cloud, mobile, and edge settings. *arXiv preprint arXiv:2405.15079*, 2024.

[208] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. Variable rate image compression with recurrent neural networks, 2016.

[209] S. Trindade, L. Bittencourt, and N. Fonseca. Management of resource at the network edge for federated learning. *arXiv preprint arXiv:2107.03428*, 2021.

[210] P. Tschandl, C. Rosendahl, and H. Kittler. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5(1), aug 2018.

[211] U.S. Department of Health & Human Services. The Health Insurance Portability and Accountability Act of 1996 (HIPAA). `https://www.hhs.gov/hipaa/index.html`, 1996. Accessed: 2026-04-22.

[212] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[213] A. Valkanas, S. Pal, P. Rumiantsev, Y. Zhang, and M. Coates. C3po: Optimized large language model cascades with probabilistic cost constraints for reasoning. *arXiv preprint arXiv:2511.07396*, 2025.

[214] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[215] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer. A survey on distributed machine learning. *ACM Computing Surveys*, 53(2):1–33, 2019.

[216] P. Voigt and A. v. d. Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer Publishing Company, Incorporated, 1st edition, 2017.

[217] X. Wan, P. Qi, G. Huang, C. Ruan, M. Lin, and J. Li. Revisiting parameter server in LLM post-training. In *International Conference on Learning Representations (ICLR)*, 2026.

[218] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pages 3261–3275, 2019.

[219] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP*, 2018.

[220] D. Wang, E. Shelhamer, S. Liu, B. A. Olshausen, and T. Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations (ICLR)*, 2021.

[221] L. WANG, W. WANG, and B. LI. Cmfl: Mitigating communication overhead for federated learning. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 954–964, 2019.

[222] Q. Wang, O. Fink, L. V. Gool, and D. Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7201–7211, 2022.

[223] W. Wang, M. Khazraee, Z. Zhong, Z. Jia, D. Mudigere, Y. Zhang, A. Kewitsch, and M. Ghobadi. TopoOpt: Optimizing the network topology for distributed DNN training. *arXiv preprint arXiv:2202.00433*, 2022.

[224] X. Wang, Y. Liu, W. Cheng, X. Zhao, Z. Chen, W. Yu, Y. Fu, and H. Chen. MixLLM: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10912–10922, Albuquerque, New Mexico, 2025. Association for Computational Linguistics.

[225] Z. Wang, F. Wu, F. Yu, Y. Zhou, J. Hu, and G. Min. Federated continual learning for edge-ai: A comprehensive survey. *arXiv preprint arXiv:2411.13740*, 2024.

[226] W. Weng, J. Zhu, X. Meng, H. Zhang, R. Zhang, and C. Yuan. Learning to compress contexts for efficient knowledge-based visual question answering. *arXiv preprint arXiv:2409.07331*, 2024.

[227] Y. Wu, K. Ma, X. Yan, Z. Liu, and J. Cheng. Elastic deep learning in multi-tenant GPU cluster. *arXiv preprint arXiv:1909.11985*, 2019.

[228] L. Xian, B. Li, J. Liu, Z. Guo, and D. K. Du. H-PS: A heterogeneous-aware parameter server with distributed neural network training. *IEEE Access*, 9:44049–44058, 2021.

[229] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[230] J. Xu, H. Lin, Y. Xu, Y. Li, N. Z. Gong, and M. Wang. A joint training-calibration framework for test-time personalization with label shift in federated learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023.

[231] W. Xu, Z. Liang, K. Mei, H. Gao, J. Tan, and Y. Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

[232] Y. Xu, S. R. Basu, A. Bhattacharya, and J. Ghosh. Federated covariate shift adaptation for missing target output values. *arXiv preprint arXiv:2302.08436*, 2023.

[233] Z. Yang, Y. Zhang, D. Sui, C. Liu, J. Zhao, and K. Liu. Representative demonstration selection for in-context learning with two-stage determinantal point process. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5443–5456, 2023.

[234] J. Ye, Z. Wu, J. Feng, T. Yu, and L. Kong. Compositional exemplars for in-context learning. In *International Conference on Machine Learning*, pages 39818–39833. PMLR, 2023.

[235] T. Yoon, S. Shin, S. J. Hwang, and E. Yang. Fedmix: Approximation of mixup under mean augmented federated learning. In *International Conference on Learning Representations*, 2021.

[236] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538, Carlsbad, CA, July 2022. USENIX Association.

[237] L. Yu, Z. Chang, Y. Jia, and G. Min. Model partition and resource allocation for split learning in vehicular edge networks. *IEEE Transactions on Intelligent Transportation Systems*, 26:17851–17865, 2025.

[238] T. Yu et al. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*, 2024.

[239] L. Yuan, L. Sun, P. S. Yu, and Z. Wang. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, 11:34617–34638, 2024.

[240] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 423–438, 2013.

[241] E. L. Zec, A. Breitholtz, and F. D. Johansson. Overcoming label shift in targeted federated learning. *arXiv preprint arXiv:2411.03799*, 2024.

[242] Q. Zeng, Y. Du, K. Huang, and K. Leung. Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing. *IEEE Transactions on Wireless Communications*, 20(12):7947–7962, 2021.

[243] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3987–3995, 2017.

[244] J. Zhang, C. Chen, W. Zhuang, and L. Lv. Addressing catastrophic forgetting in federated class-continual learning. *arXiv preprint arXiv:2303.06937*, 2023.

[245] J. Zhang, A. Li, M. Tang, J. Sun, X. Chen, F. Zhang, C. Chen, Y. Chen, and H. Li. Fed-cbs: A heterogeneity-aware client sampling mechanism for federated learning via class-imbalance reduction, 2022.

[246] J. Zhang and S. Lv. Fedlabcluster: A clustered federated learning algorithm based on data sample label. In *2021 International Conference on Electronic Information Engineering and Computer Science (EIECS)*, pages 423–428, 2021.

[247] L. Zhang, G. Gao, and H. Zhang. Spatial-temporal federated learning for lifelong person re-identification on distributed edges. *IEEE Transactions on Circuits and Systems for Video Technology*, 35(2):1884–1896, 2025.

[248] Q. Zhang, A. Cheng, M. Lu, R. Zhang, Z. Zhuo, J. Cao, S. Guo, Q. She, and S. Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. *arXiv preprint arXiv:2412.01818*, 2024.

[249] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

[250] Y. Zhang, K. Zhou, and Z. Liu. What makes good examples for visual in-context learning? *Advances in Neural Information Processing Systems*, 36:17773–17794, 2023.

[251] Z. Zhang, S. Zheng, Y. Wang, G. Karypis, T. M. Chilimbi, M. Li, and X. Jin. MiCS: Near-linear scaling for training gigantic model on public cloud. *Proceedings of the VLDB Endowment*, 16(1):37–50, 2022.

[252] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

[253] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[254] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR, 2021.

[255] W. Zheng, Y. Chen, W. Zhang, S. Kundu, Y. Li, Z. Liu, E. P. Xing, H. Wang, and H. Yao. Citer: Collaborative inference for efficient large language model decoding with token-level routing. *arXiv preprint arXiv:2502.01976*, 2025.

[256] W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[257] B. Zhou, Y. Hu, X. Weng, J. Jia, J. Luo, X. Liu, J. Wu, and L. Huang. Tinyllava: A framework of small-scale large multimodal models. *arXiv preprint arXiv:2402.14289*, 2024.

[258] W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341, 2020.

[259] X. Zhou, W. Wang, M. Zeng, J. Guo, X. Liu, L. Shen, M. Zhang, and L. Ding. DynamicKV: Task-aware adaptive KV cache compression for long context LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 8042–8057, Suzhou, China, 2025. Association for Computational Linguistics.

[260] H. Zhu, J. Bai, N. Li, X. Li, D. Liu, D. L. Buckeridge, and Y. Li. Fedweight: mitigating covariate shift of federated learning on electronic health records data through patients re-weighting. *npj Digital Medicine*, 8(1):1–19, 2025.

[261] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, pages 14774–14784, 2019.

[262] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose, et al. Pysyft: A library for easy federated learning. *Federated learning systems: Towards next-generation AI*, pages 111–139, 2021.