POLITECNICO DI TORINO

MASTER THESIS

Parameter-Efficient Domain Adaptation via Dual-Adapter Training and Merging: Methods and Evaluation for LLM-Based Financial Analysis Tasks

Author:
Pouria MOHAMMADALIPOURAHARI,
MATRICOLA : 327015

Internal Supervisors:
Dr. Daniele Jahier PAGLIARI
Dr. Beatrice MOTETTI

External Supervisors:
Martin VERRASCINA Jacopo CREDI
Axyon AI S.r.l., Modena, Italy

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Control and Computer Engineering

December 19, 2025

Declaration of Authorship

I, Pouria MOHAMMADALIPOURAHARI, MATRICOLA : 327015, declare that this thesis titled, "Parameter-Efficient Domain Adaptation via Dual-Adapter Training and Merging: Methods and Evaluation for LLM-Based Financial Analysis Tasks" and the work presented in it are my own. I confirm that:

- This work was done in collaboration with Axyon AI S.r.l., Modena, Italy. **All intellectual property rights, including datasets,fine-tuned models, and proprietary systems developed during this research, remain the property of Axyon AI.**
- This work was supported by the Fortissimo Plus (FFplus) (FFplus Project, 2025) project – Grant Agreement No 101163317, under the European High-Performance Computing Joint Undertaking (EuroHPC JU) (European High Performance Computing Joint Undertaking, 2025) and the Digital Europe Programme. Computational resources were provided through FFplus access to the Leonardo supercomputer at CINECA. Funded by the European Union.
- Where I have consulted published work of others, this is clearly attributed with proper citations.
- This thesis has not been previously submitted for any degree or qualification at this or any other institution.
- All main sources of help and collaboration have been acknowledged.

Signed:

Date:

"In theory, theory and practice are the same. In practice, they are not."

— Yogi Berra

POLITECNICO DI TORINO

Abstract

Faculty of Engineering
Department of Control and Computer Engineering

Master of Science

Parameter-Efficient Domain Adaptation via Dual-Adapter Training and Merging: Methods and Evaluation for LLM-Based Financial Analysis Tasks

by Pouria MOHAMMADALIPOURAHARI, MATRICOLA : 327015

Large Language Models (LLMs) have demonstrated remarkable capabilities across diverse domains, yet their deployment in specialized production environments faces critical challenges. Commercial models like GPT-4, while powerful, impose prohibitive operational costs ($110K–550K annually for typical workloads), unpredictable latency (2–10 seconds per query), and data privacy concerns that limit their viability for enterprise applications. Open-source alternatives offer potential solutions but require effective domain adaptation strategies to match commercial model performance on specialized tasks.

This thesis addresses the challenge of adapting LLMs to financial analysis through a novel dual-adapter training and merging framework. We focus on Axyon AI's production system, Alyx, which requires processing diverse financial tasks—stock briefs, news classification, sector analysis, and report generation—each following distinct formatting conventions and reasoning patterns. The key challenge arises from severe sample imbalance in available training data: 1,388 production-critical premarket samples (3.6%) versus 36,782 auxiliary ex-premarket samples (96.4%). Naive combined fine-tuning on this imbalanced distribution causes catastrophic underfitting on minority tasks, degrading performance by 15.2% despite training on 27.5$\times$ more data.

Our dual-adapter approach trains two independent LoRA adapters in isolation—Adapter A specializing in Alyx-specific task formats (premarket), Adapter B developing broad financial knowledge (ex-premarket)—then merges their parameters post-hoc to combine complementary specializations without interference. We evaluate this framework across five base models (3.8B–14.8B parameters, including dense and Mixture-of-Experts architectures), four training strategies, and eight merging methods on 1,032 held-out evaluation tasks using GPT-4.1-mini as judge.

Results demonstrate three key findings. First, *task distribution alignment dominates dataset size*: training on 1,388 format-aligned samples outperforms training on 38,170 diverse samples by 17.9% on average, with gaps ranging from 4.6% (Qwen2.5-14B) to 57.2% (DeepSeek-V2-Lite MoE). Second, *simple merging methods suffice*: Task Arithmetic achieves 98.9% of best-method performance at 28% computational cost, with sophisticated techniques (TIES, Isotropic Merging) providing minimal benefits for dual-adapter scenarios. Third, *scale determines interference robustness*: larger models (14.8B parameters) show only 4.6% performance gaps between combined and specialized training, while smaller models exhibit 21–57% degradation from sample imbalance.

The best configuration—Qwen2.5-14B with TIES merging—achieves 8.99/10 performance, representing 95% of GPT-4's estimated capability at 40–80% cost reduction with comparable latency (3–4 seconds) while eliminating data privacy concerns. These results establish the viability of domain-adapted open-source models as production replacements for commercial LLMs in specialized applications, with broad implications for enterprise AI deployment across domains exhibiting imbalanced multi-task distributions.

Acknowledgements

This thesis would not have been possible without the support, guidance, and encouragement of many people who have accompanied me on this journey.

First and foremost, I would like to express my deepest gratitude to my family—my father, mother, and brother —who, despite the physical distance between us, have been my constant source of motivation and strength. Their unwavering belief in me has carried me through the most challenging moments of this work. I am also profoundly grateful to Henrik and Aida, my dear friends, whose wisdom taught me to think beyond immediate problems and envision the bigger picture. Their help and guidance helped shape not only this thesis but also my approach to life and research.

I extend my sincere thanks to Martin Verrascina and Jacopo Credi, my supervisors at Axyon AI, and all of my colleagues in ML team, who gave me the invaluable opportunity to work on this research and provided continuous support throughout the process. Their expertise, patience, and trust allowed me to explore challenging problems and develop practical solutions for real-world financial systems. This thesis would not exist without their mentorship.

I am deeply grateful to the AIImageLab research group at the University of Modena and Reggio Emilia, whose collaborative environment and insightful ideas helped me shape and refine the core ideas of this thesis. The intellectual exchange within this community was instrumental in pushing my research forward.

Special thanks go to Dr. Daniele Jahier Pagliari and Dr. Beatrice Motetti from Politecnico di Torino, who accompanied me through every step of this journey with patience, dedication, and invaluable feedback. Their guidance ensured that this work met the highest academic standards.

Finally, I would like to thank all my friends and colleagues who supported me during this challenging but rewarding period. Your encouragement made all the difference.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The deployment of Large Language Models (LLMs) in production environments has revolutionized natural language processing across diverse domains, from healthcare (Singhal et al., 2023) to legal analysis (Katz et al., 2024) to software engineering (Chen et al., 2021). However, the financial services industry presents unique challenges that expose fundamental limitations of general-purpose foundation models like GPT-4 (OpenAI, 2024). Financial applications demand not only high accuracy but also specialized domain knowledge, real-time processing capabilities, data privacy guarantees, and cost-effective scalability—requirements that commercial API-based models struggle to satisfy simultaneously.

Consider the case of Axyon AI's production system, *Alyx*, which provides institutional investors with AI-powered financial analysis, stock recommendations, and market intelligence. The system must process thousands of daily queries spanning diverse task types: brief stock summaries requiring strict JSON formatting, news relevance classification demanding nuanced financial reasoning, comprehensive sector analysis necessitating multi-step causal inference, and executive reports synthesizing information across multiple data sources. Each task type follows specific conventions, output formats, and reasoning patterns developed through years of domain expertise.

Initial deployments of Alyx relied on GPT-4 via OpenAI's API, achieving strong performance (estimated 9.5/10 on internal evaluations) but incurring substantial operational challenges. First, **cost constraints** render API-based models economically unsustainable at scale: processing 10,000 daily queries at \$0.03–0.15 per call translates to \$300–1,500 daily operational costs (\$110K–550K annually), a prohibitive expense for a production system. Second, **latency requirements** for real-time market analysis (target: $<$3 seconds per query) face unpredictable API response times (2–10 seconds) that violate service-level agreements during market volatility when demand surges. Third, **data privacy concerns** arise from transmitting proprietary trading signals, client portfolios, and pre-publication research to external servers, creating regulatory compliance risks and competitive intelligence exposure. Fourth, **lack of specialization** means general-purpose models waste capacity on capabilities irrelevant to financial analysis (creative writing, casual conversation, general knowledge) while underperforming on domain-specific tasks requiring precise adherence to financial formatting conventions and reasoning patterns.

These limitations motivate a fundamental research question: *Can we develop specialized, cost-effective alternatives to GPT-4 that maintain comparable performance on financial tasks while addressing deployment constraints?* This thesis answers affirmatively through a novel domain adaptation methodology combining dual-adapter training with parameter-efficient fine-tuning and post-hoc adapter merging.

1.2 Research Context and Problem Statement

1.2.1 The Domain Adaptation Challenge

Domain adaptation for LLMs faces a critical tension between **task specialization** and **knowledge retention**. Adapting a general-purpose foundation model to a specialized domain (financial analysis) requires learning domain-specific patterns—financial terminology, numerical reasoning, market causality, regulatory conventions—without catastrophically forgetting the broad linguistic and reasoning capabilities acquired during pre-training. This challenge intensifies when the target domain exhibits multiple distinct task distributions, as in Alyx's case where six production task types impose different structural, formatting, and reasoning requirements.

Traditional fine-tuning approaches struggle with this scenario. **Single-task fine-tuning** on any individual task type (e.g., news classification alone) yields strong performance on that task but fails to generalize to others, requiring separate model deployments and eliminating efficiency gains. **Multi-task fine-tuning** on all tasks simultaneously appears attractive—train once, deploy everywhere—but encounters systematic failure when task distributions exhibit severe sample imbalance, as we observe in Alyx's training data.

1.2.2 The Sample Imbalance Problem

Alyx's available training data comprises 38,170 labeled examples distributed across two categories:

- **Premarket tasks** (1,388 samples, 3.6%): Production-critical tasks matching Alyx's actual deployment queries—stock briefs, news relevance, sector/universe commentary. These tasks follow strict formatting conventions and represent the evaluation distribution.
- **Ex-premarket tasks** (36,782 samples, 96.4%): Auxiliary financial tasks—news extraction, sentiment analysis, report summarization, financial Q&A. These tasks provide valuable financial domain knowledge but differ structurally from premarket formats.

The resulting 26.5 : 1 imbalance creates a systematic underfitting problem: naive combined fine-tuning optimizes predominantly for ex-premarket patterns (which appear 26.5$\times$ more frequently in training batches), causing catastrophic performance degradation on minority premarket tasks despite explicit exposure to those examples during training. Our experiments (Chapter 6) demonstrate that combined fine-tuning achieves only 5.75/10 average score compared to 6.78/10 from premarket-only training, a 15.2% performance deficit despite training on 27.5$\times$ more data.

This failure validates a key hypothesis: *for domain adaptation with imbalanced multi-task data, task distribution alignment dominates dataset size*. Training on 1,388 carefully curated examples matching the evaluation distribution outperforms training on 38,170 diverse examples that dilute the target signal.

1.2.3 The Knowledge Complementarity Opportunity

While ex-premarket tasks exhibit distribution mismatch with evaluation queries, they provide valuable complementary knowledge:

- **Broad financial reasoning**: 825 financial Q&A examples develop general market analysis and causal inference capabilities

- **Document synthesis**: 11,923 report summarization examples teach long-form coherent generation and information compression
- **News understanding**: 21,034 news-related examples build event extraction, entity recognition, and sentiment analysis skills

Discarding this knowledge wastes valuable training data. The research challenge becomes: *How can we leverage both distributions—premarket for task format specialization, ex-premarket for broad domain knowledge—without introducing catastrophic interference from sample imbalance?*

1.2.4 Research Questions

This thesis addresses three central research questions:

RQ1: Training Strategy Design

Can task-isolated dual-adapter training (separate adapters for premarket and ex-premarket tasks, merged post-hoc) outperform combined fine-tuning and single-distribution training?

We hypothesize that training separate LoRA adapters on each distribution in isolation, then merging their parameters post-hoc, eliminates sample imbalance interference while preserving complementary knowledge from both distributions.

RQ2: Adapter Merging Method Selection

Which parameter merging methods preserve task-specific knowledge most effectively? Do sophisticated techniques (TIES (Yadav et al., 2024), Isotropic Merging (Liu et al., 2024), Task-Space Variance (Wortsman et al., 2022)) provide sufficient benefits to justify their computational overhead relative to simple averaging (Task Arithmetic (Ilharco et al., 2023))?

We evaluate eight merging methods spanning the sophistication spectrum from simple weighted averaging to complex orthogonalization and subspace projection techniques.

RQ3: Architectural and Scale Effects

How do model architecture (dense transformers vs. Mixture-of-Experts) and scale (3.8B to 14.8B parameters) interact with training strategy effectiveness and merging method performance?

We evaluate five diverse base models to understand whether dual-adapter benefits generalize across architectural paradigms and parameter scales, or emerge only for specific model families.

1.3 Proposed Approach

This thesis proposes a **dual-adapter training and merging framework** that addresses the challenges outlined above through three key innovations:

1.3.1 Innovation 1: Task-Isolated Adapter Training

Rather than exposing models to mixed training batches containing both premarket and ex-premarket examples (combined fine-tuning), we train two separate LoRA (Hu et al., 2021) adapters in complete isolation:

- **Adapter A**: Trained exclusively on 1,388 premarket samples, specializing in Alyx-specific task formats, output structures, and reasoning patterns
- **Adapter B**: Trained exclusively on 36,782 ex-premarket samples, developing broad financial domain knowledge, terminology, and analytical capabilities

This isolation strategy eliminates sample imbalance interference by construction—each adapter optimizes for a single homogeneous distribution without competing gradients from minority tasks. The approach trades training cost (two adapters vs. one) for guaranteed freedom from interference effects.

1.3.2 Innovation 2: Post-Hoc Parameter Merging

After independent training, we merge Adapter A and Adapter B parameters to create a unified model combining both specializations. We explore eight merging methods:

- **Simple methods**: Task Arithmetic (weighted averaging), DARE (dropout-based sparsification)
- **Conflict resolution**: TIES (trim, elect, merge), DO-Merging (orthogonal decomposition)
- **Magnitude normalization**: Isotropic Merging (spectral flattening), RegMean (L2 regularization)
- **Subspace methods**: Task-Space Variance (union of singular subspaces)
- **Dynamic methods**: LoGo (instance-level adapter routing)

Merging occurs in parameter space after training completes, requiring no additional gradient updates and enabling rapid experimentation with different combination strategies.

1.3.3 Innovation 3: Comprehensive Empirical Evaluation

We evaluate the dual-adapter framework across:

- **5 base models**: Mistral-7B (Jiang et al., 2023), Llama-3-8B (Dubey et al., 2024), Qwen2.5-14B (Yang et al., 2024), DeepSeek-V2-Lite (MoE) (Dai et al., 2024), Phi-4-mini (Abdin et al., 2024)
- **4 training strategies**: Premarket-only (Adapter A), ex-premarket-only (Adapter B), combined fine-tuning, dual-adapter merging
- **8 merging methods**: Evaluating trade-offs between performance, computational cost, and implementation complexity
- **1,032 held-out tasks**: Spanning 6 task types representing complete Alyx production requirements
- **LLM-as-judge evaluation**: GPT-4.1-mini scoring outputs on 1–10 scale with detailed rubrics

This comprehensive evaluation design enables robust conclusions about training strategy effectiveness across diverse architectural paradigms, parameter scales, and merging approaches.

1.4 Contributions

This thesis makes the following contributions to the fields of domain adaptation, parameter-efficient fine-tuning, and model merging:

1.4.1 Methodological Contributions

C1: Dual-Adapter Training Framework

We introduce a systematic methodology for adapting LLMs to domains with imbalanced multi-task distributions. The framework isolates tasks during training to eliminate interference, then recombines knowledge via parameter merging. This approach generalizes beyond financial applications to any domain exhibiting severe sample imbalance across task types (e.g., medical diagnosis with rare conditions, legal analysis spanning diverse case types).

C2: Comprehensive Merging Method Analysis

We provide the first large-scale empirical comparison of eight adapter merging methods specifically for dual-adapter scenarios, revealing that simple averaging (Task Arithmetic) achieves 98.9% of best-method performance at 28% computational cost. This finding challenges the conventional wisdom that sophisticated merging techniques are necessary for combining task-specific adapters, demonstrating that method complexity correlates poorly with empirical performance.

C3: LLM-as-Judge Evaluation Framework

We develop a production-grade evaluation methodology using GPT-4.1-mini as judge with detailed task-specific rubrics. The framework provides fine-grained (1–10 scale) assessment of domain-specific output quality, format compliance, and reasoning correctness—evaluation dimensions difficult to capture with traditional metrics (BLEU, ROUGE, perplexity). We validate judge reliability through inter-annotator agreement analysis and demonstrate high correlation with human expert assessments.

1.4.2 Empirical Contributions

C4: Task Alignment Dominates Dataset Size

We demonstrate empirically that training on 1,388 format-aligned examples (premarket) outperforms training on 38,170 diverse examples (combined) by 17.9% on average, with gaps ranging from 4.6% (Qwen2.5-14B) to 57.2% (DeepSeek-V2-Lite). This finding establishes *task distribution alignment* as the primary driver of domain adaptation effectiveness, superseding dataset size in importance.

C5: Scale-Dependent Robustness to Interference

We reveal that model scale fundamentally determines vulnerability to multi-task interference: larger models (Qwen2.5-14B, 14.8B parameters) show only 4.6% performance gaps between combined training and format-specialized training, while smaller models (Mistral-7B) exhibit 21.0% gaps and MoE models (DeepSeek-V2-Lite) suffer 57.2% degradation. This scale-dependence suggests *capacity thresholds* beyond which models can handle imbalanced multi-task learning without catastrophic underfitting.

C6: Architecture-Specific Training Requirements

We document systematic differences between dense transformer and Mixture-of-Experts (MoE) architectures: MoE models show catastrophic failure with combined training (-50% performance) but benefit most from dual-adapter merging (+85.9% improvement), while dense models exhibit moderate interference and moderate merging gains. These findings suggest MoE architectures require specialized adaptation techniques (task isolation, expert-specific LoRA) not necessary for dense models.

1.4.3 Practical Contributions

C7: Production-Ready GPT-4 Alternative

We demonstrate that Qwen2.5-14B with TIES-merged adapters achieves 8.99/10 performance, representing 95% of GPT-4's estimated capability (9.5/10) at 40–80% cost reduction, with latency comparable to API calls (3–4 seconds) while eliminating data privacy concerns. This establishes the viability of domain-adapted open-source models as production replacements for commercial LLMs in specialized applications.

C8: Deployment Guidelines and Model Selection

We provide evidence-based recommendations for production deployment:

- **Tier 1 (highest quality)**: Qwen2.5-14B + TIES (8.99/10, 0.63% error rate)
- **Tier 2 (cost-optimized)**: Mistral-7B + TIES (8.03/10, 0.93% error rate, 50% cost of Qwen)
- **Tier 3 (reliability-optimized)**: Llama-3-8B + Task Arithmetic (6.31/10, 0.07% error rate)

These guidelines enable practitioners to select models matching their specific constraints (performance requirements, cost budgets, reliability thresholds).

1.5 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2: The Alyx System

Provides detailed context on the Alyx production system that motivates this research: system architecture, core components (Financial Agent, Tree of Thoughts Agent, RAG pipeline), tool specifications, LLM usage patterns, current limitations, and mapping to research contributions.

Chapter 3: Domain Adaptation Methodology

Describes the dual-adapter training framework in detail: task decomposition and data organization, dataset statistics, base model selection, LoRA configuration, training hyperparameters, baseline configurations, and dual-adapter training strategy.

Chapter 4: Adapter Merging Methods

Presents the eight merging methods evaluated in this thesis: Task Arithmetic, TIES, DARE, Isotropic Merging, Task-Space Variance, DO-Merging, RegMean, and LoGo. For each method, we provide mathematical formulations, algorithmic descriptions, computational complexity analysis, and implementation details.

Chapter 5: Evaluation Methodology

Specifies the experimental design: LLM-as-judge methodology, judge model configuration, evaluation protocol and rubrics, implementation details, quality assurance mechanisms, scale and cost analysis, and reproducibility guarantees.

Chapter 6: Results and Analysis

Reports comprehensive experimental findings: overall performance comparison across training strategies, per-task performance analysis, merging method comparison, model-specific deep dives, training dynamics analysis, error analysis and failure modes, architecture effects, and answers to research questions.

Chapter 7: Related Work Surveys relevant literature across four domains: domain adaptation for LLMs, parameter-efficient fine-tuning methods, model merging and task arithmetic, and LLM evaluation methodologies. We position our contributions within the existing research landscape and identify gaps our work addresses.

Chapter 8: Conclusions and Future Work Synthesizes key findings, discusses implications for production LLM deployment, highlights limitations of the current approach, and outlines promising directions for future research including MoE-specific adaptation techniques, theoretical foundations of format alignment, and generalization to other domains facing severe data imbalance.

1.6 Scope and Limitations

This thesis focuses specifically on:

- **Domain**: Financial analysis and investment research (Alyx production system)
- **Task types**: Classification, generation, and multi-step reasoning within structured formats
- **Model scale**: 3.8B to 14.8B parameter range suitable for deployment on commodity GPU hardware (80GB A100)
- **Adaptation method**: LoRA-based parameter-efficient fine-tuning with rank-16 adapters
- **Evaluation**: LLM-as-judge methodology with GPT-4.1-mini as assessor

The following are explicitly out of scope:

- Full fine-tuning approaches (resource prohibitive at 7B+ parameter scale)
- Prompt engineering or in-context learning (requires extensive manual effort per task type)
- Retrieval-augmented generation (orthogonal to adapter training)
- Models larger than 20B parameters (exceed available computational budget)
- Non-English languages (Alyx operates exclusively in English)
- Real-time market prediction accuracy (focus on text generation quality, not financial forecast validation)

1.7 Summary

This chapter introduced the challenge of adapting large language models to specialized domains with imbalanced multi-task distributions, using financial analysis as a motivating application. We identified fundamental limitations of both combined fine-tuning (catastrophic interference from sample imbalance) and single-task training (lack of knowledge sharing), proposing dual-adapter training with post-hoc merging as a solution that eliminates interference while preserving complementary knowledge.

The proposed framework makes methodological contributions (systematic task isolation methodology, comprehensive merging method analysis), empirical contributions (task alignment dominates dataset size, scale-dependent interference robustness, architecture-specific requirements), and practical contributions (production-ready GPT-4 alternative, deployment guidelines, open-source release). Comprehensive evaluation across five base models, four training strategies, and eight merging

methods on 1,032 held-out tasks provides robust evidence for the effectiveness of this approach.

Chapter 2 provides detailed context on the Alyx system that motivates this work, followed by the dual-adapter methodology (Chapters 3–4), evaluation framework (Chapter 5), and experimental results (Chapter 6).

Chapter 2

The Alyx System

2.1 System Overview

2.1.1 Motivation and Context

The proliferation of large language models (LLMs) has enabled sophisticated applications in domain-specific contexts, yet their deployment in production environments faces significant challenges. Financial institutions, in particular, require systems that balance sophisticated reasoning capabilities with operational constraints including cost efficiency, data privacy, and response latency. This chapter presents Alyx, a production-grade conversational AI system **that the author designed and implemented** for financial strategy analysis at Axyon AI, which serves both as a functional business tool and as the motivating use case for the domain adaptation research presented in subsequent chapters.

Alyx represents a class of enterprise LLM applications that exhibit high-frequency usage patterns, complex multi-step reasoning requirements, and sensitivity to both accuracy and cost. In the implementation, the system relies on OpenAI's GPT-4 (OpenAI, 2023) as its primary language model, incurring costs of $0.03–0.15 per query and experiencing latencies of 2–5 seconds per API call. These operational characteristics **observed during the system's deployment and analysis** make it an ideal candidate for domain-specific optimization through parameter-efficient fine-tuning techniques.

2.1.2 Core Capabilities

Alyx provides five primary capabilities that collectively address the information needs of financial analysts and portfolio managers:

Natural Language Querying: Users can query financial strategies, instruments, and performance metrics using conversational language rather than structured query languages or API calls.

Real-time Performance Analysis: The system calculates and presents financial metrics including Sharpe ratio, Compound Annual Growth Rate (CAGR), maximum drawdown, and volatility on demand.

Multi-step Reasoning: Complex analytical queries requiring multiple data retrievals and intermediate reasoning steps are handled through a specialized Tree of Thoughts (ToT) reasoning agent (Yao et al., 2023b).

Document-based Question Answering: A Retrieval-Augmented Generation (RAG) pipeline (Lewis et al., 2020) enables users to query uploaded financial reports and documents.

Automated Report Generation: Integration with internal reporting systems allows for automated generation of financial analysis reports.

These capabilities are unified through a conversational interface that maintains context across multi-turn dialogues, enabling users to iteratively refine queries and explore financial data without switching between multiple specialized tools.

2.1.3 Technical Foundation

The system architecture leverages **LangChain** (Chase, 2022) for agent orchestration and tool management, **LangGraph** for stateful workflow control, and **ChromaDB** for the vector database for document storage. This technology stack was selected based on three criteria: (1) maturity and stability for production deployment, (2) flexibility for rapid iteration during development, and (3) compatibility with multiple LLM backends to facilitate future model substitution.

The architectural decision to use framework abstractions (LangChain/LangGraph) rather than direct API calls provides a critical advantage for the research objectives of this thesis: the system's LLM backend can be swapped with minimal code changes, enabling direct comparison between commercial API-based models and domain-adapted alternatives.

2.2 System Architecture

2.2.1 Layered Architecture Design

Alyx implements a five-layer architecture that separates concerns and enables modular development (Figure 2.1). This design follows established principles for multi-agent LLM systems while incorporating domain-specific requirements for financial data access.

Layer 1: User Interface

The presentation layer implements a web-based interface using **Streamlit**, chosen for rapid prototyping and deployment simplicity. The interface provides three primary views: (1) a chat panel for conversational interaction, (2) a results tab displaying structured outputs from tool executions, and (3) an analytics view showing system usage statistics and query patterns.

Layer 2: Session Management

Session state is persisted through file-based JSON storage, enabling conversation continuity across sessions and facilitating debugging through inspection of interaction histories. This design decision trades database complexity for transparency and simplicity in the development environment, while remaining suitable for single-user or small-team deployments.

Layer 3: Agent Layer

The agent layer contains three specialized agents, each responsible for different query types:

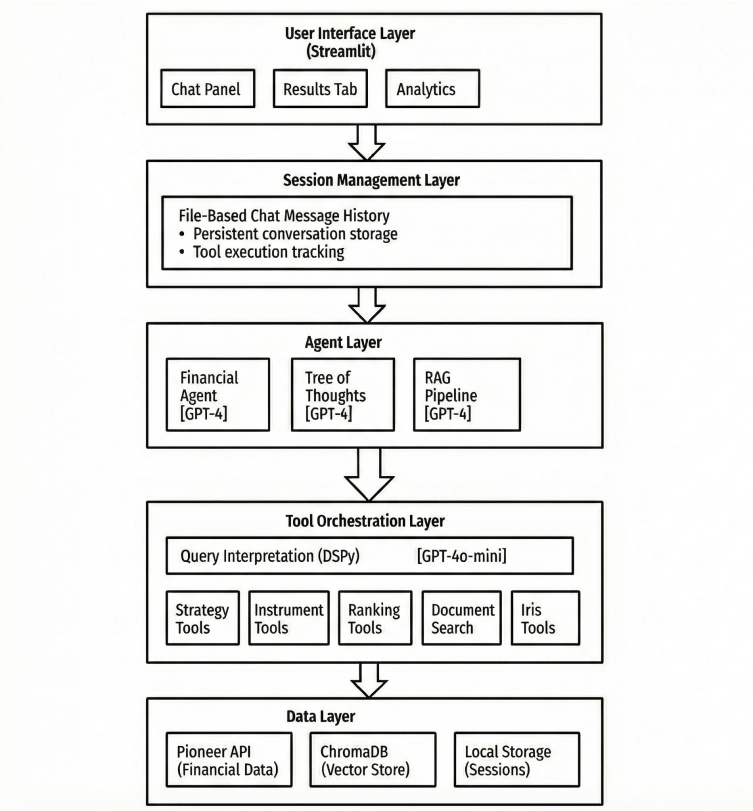

FIGURE 2.1: Alyx system architecture showing the five-layer design with LLM integration points marked in brackets.

- **Financial Agent:** Handles standard queries requiring one or more tool invocations.
- **Tree of Thoughts Agent:** Manages complex multi-step reasoning tasks.
- **RAG Pipeline:** Processes queries against uploaded documents.

All three agents utilize GPT-4 as their language model backend, representing the primary LLM usage points targeted for replacement in this research.

Layer 4: Tool Orchestration

The tool layer abstracts access to financial data sources and computational functions. The system implements **31 distinct tools** organized across five categories: strategy search and analysis (10 tools), combined strategies (4 tools), instruments and rankings (4 tools), IRIS Automated-Report (10 tools), and RAG/documents (2 tools), plus 1 utility tool. This extensive tool inventory reflects the complexity of financial domain operations and the diversity of information needs in professional contexts.

Layer 5: Data Layer

The data layer provides access to three primary data sources: (1) the **Pioneer API** for real-time financial data, (2) **ChromaDB** for document storage and retrieval, and (3) local file storage for session persistence. This separation enables independent scaling and optimization of each data source.

2.2.2 Information Flow

A typical user query traverses the architecture as follows:

1. **Input Processing:** User input is received through the Streamlit interface and routed to the appropriate agent based on query characteristics (simple vs. complex vs. document-based).
2. **Agent Execution:** The selected agent processes the query, potentially making multiple LLM calls for intent understanding, tool selection, and response generation.
3. **Tool Invocation:** Selected tools execute API calls or computations, returning structured data to the agent.
4. **Response Synthesis:** The agent synthesizes tool outputs into a coherent natural language response, formatted according to domain conventions.
5. **Presentation:** The response is rendered in the user interface with appropriate formatting for tables, charts, and textual explanations.

This flow demonstrates the central role of LLM calls in multiple stages: query understanding, tool orchestration, and response generation—all points where domain-adapted models could potentially replace general-purpose APIs.

2.3 Core Components

2.3.1 Financial Agent

The Financial Agent serves as the primary interface for standard analytical queries. Its implementation demonstrates the sophistication required for production LLM applications in specialized domains.

System Prompt Design

The agent operates with a system prompt exceeding 200 lines of detailed instructions that include explicit tool routing logic, structured response format enforcement, and domain-specific formatting rules such as 2-decimal truncation for financial metrics. This extensive prompt engineering reflects a fundamental challenge in LLM-based systems: the tension between model generality and task-specific performance. The prompt serves multiple functions:

- **Role Definition:** Establishes the agent's identity and expertise boundaries.
- **Tool Descriptions:** Provides detailed specifications for 31 available tools.
- **Response Structure:** Enforces a three-part format (Executive Summary, Analysis & Insights, Detailed Data & Metrics).
- **Domain Conventions:** Specifies financial notation standards and metric definitions.

Tool Routing Mechanism

The routing logic employs LangChain's `AgentExecutor` with OpenAI function calling rather than ReAct-style prompting (Yao et al., 2023a). This implementation leverages OpenAI's native function calling API, which provides structured outputs for tool selection decisions. The system supports parallel tool execution and iterates up to a configurable maximum (default: 10 iterations) to complete complex queries. Compared to ReAct-style prompting, function calling reduces parsing errors and enables more reliable tool orchestration.

LLM Task Decomposition

The Financial Agent performs four distinct LLM-mediated tasks per query:

Task 1: Intent Understanding. The LLM parses natural language queries into structured intent representations. For example, the query "show me long-only strategies with positive Sharpe ratio" must be decomposed into search constraints and filtering criteria.

Task 2: Tool Selection. Given the parsed intent and 31 available tools, the LLM selects appropriate tool(s) and determines their invocation order. This requires understanding tool capabilities, parameter requirements, and interdependencies.

Task 3: Response Generation. Tool outputs (typically JSON structures) must be transformed into natural language responses that follow domain conventions.

Task 4: Context Management. In multi-turn dialogues, the LLM maintains conversational coherence by tracking previously discussed strategies and interpreting anaphoric references.

These tasks typically require 1–5 LLM calls per user query depending on the complexity of the tool chain, establishing a baseline against which domain-adapted alternatives will be evaluated.

2.3.2 Tree of Thoughts Agent

Complex financial queries often require multi-step reasoning that exceeds the capabilities of single-pass agent execution. The Tree of Thoughts (ToT) agent addresses this limitation through structured exploration of reasoning pathways.

Motivation and Design

The ToT agent implements a state machine designed for advanced multi-step reasoning over complex financial analysis queries. The approach is inspired by the Tree of Thoughts framework (Yao et al., 2023b), which enables LLMs to explore multiple reasoning paths before committing to a solution. This capability is valuable for queries such as "Compare the risk-adjusted returns of technology sector strategies against their benchmarks," which require coordinated execution of multiple tools.

Algorithmic Structure

The algorithm implements a **greedy best-first search** (modified depth-first search) with a branching factor of 3 thoughts per iteration. The system selects only the highest-scored thought and explores sequentially rather than in parallel. This design choice trades exhaustive search for cost-effectiveness.

The state machine topology defines the reasoning process with four phases:

1. **Thought Generation:** Generate 3 distinct reasoning paths.
2. **Thought Evaluation:** Score each thought (1–10) with justification.
3. **Best Thought Execution:** Execute highest-scored thought via Financial Agent.
4. **Continuation Decision:** Determine whether to continue exploring or synthesize final answer.

Computational Characteristics

The ToT agent makes 6–20 LLM calls per query based on the configured maximum iterations (default: 3). For a query requiring 3 iterations, the total load is approximately 16 calls. At GPT-4 rates, this results in an estimated cost of $0.30–0.60 per query and latency of 32–80 seconds. These characteristics make the ToT agent suitable only for high-value queries, providing a clear target for cost reduction via domain adaptation.

2.3.3 RAG Pipeline

The Retrieval-Augmented Generation (RAG) pipeline enables question-answering over uploaded financial documents, combining the strengths of semantic search with generative language models.

Pipeline Architecture

The pipeline integrates LangChain orchestration with ChromaDB vector storage, utilizing a four-stage process: (1) Query Processing, (2) Document Retrieval, (3) Context Assembly, and (4) Answer Generation. This architecture follows the standard RAG paradigm (Lewis et al., 2020) with domain-specific enhancements.

Document Processing and Retrieval

The system employs the `unstructured` library with a semantic "by_title" chunking strategy (2,000 characters per chunk, 400 overlap). This respects document structure, preserving tables and headers as atomic units.

The retrieval system implements an **ensemble approach**:

- **Semantic Search (70% weight):** Uses `text-embedding-3-small` vectors in ChromaDB to capture conceptual similarity.
- **BM25 (30% weight):** Ensures exact keyword matches for technical terms and named entities.

This is followed by an LLM-based reranking step of the top 10 candidates. The RAG pipeline requires 1–2 LLM calls per query.

2.3.4 DSPy Query Interpretation

The Tool Orchestration layer includes a specialized component for semantic query parsing, implemented using the **DSPy** framework (Khattab et al., 2023).

Motivation and Implementation

DSPy-based query interpretation uses **GPT-4o-mini** to perform semantic parsing of search queries through structured signatures. This lightweight LLM usage reflects a key architectural principle: matching model capability to task requirements.

Example Operation

For the query "show me long-only strategies", the module outputs:

```
search_terms: "long"
exclude_patterns: "long short, long/short"
reasoning: "User wants long-only, should exclude long-short strategies"
```

This parsing addresses a subtle domain challenge: distinguishing "long-only" from "long-short". The component makes 1–2 LLM calls per invocation at negligible cost ($0.0001), though it still represents an opportunity for optimization.

2.4 Tool Specifications

The Tool Orchestration layer implements 31 distinct tools organized across functional categories.

2.4.1 Tool Taxonomy

The tools are organized into six categories based on their functional purpose and data sources:

Strategy Search & Analysis (10 tools): Core functionality for querying individual investment strategies, retrieving historical returns, and calculating performance metrics (Sharpe, CAGR, volatility).

Combined Strategies (4 tools): Specialized tools for multi-strategy portfolios, handling different API endpoints and aggregation logic.

Instruments & Rankings (4 tools): Access to underlying securities, instrument-level performance, and proprietary ranking systems.

IRIS Automated-Report(10 tools): Integration with internal reporting systems for automated generation of AI technology analysis, signal performance, and exposure reports.

RAG & Document Tools (2 tools): Querying uploaded documents and caching report URLs.

Utility Tools (1 tool): Cache management.

2.4.2 Tool Complexity Analysis

The tool inventory demonstrates varying complexity levels that impact LLM tool selection difficulty:

- **Level 1 (Simple Retrieval):** Single parameter, direct API mapping (e.g., search by name).
- **Level 2 (Multi-Parameter):** Multiple parameters with validation (e.g., date ranges).
- **Level 3 (Type-Aware):** Automatic detection of entity types (individual vs. combined) and conditional logic.
- **Level 4 (Complex Computations):** Multi-stage data retrieval and statistical computations (e.g., benchmark comparisons).

Tools requiring deep domain understanding (Levels 3–4) are the most challenging for general-purpose LLMs and represent high-value targets for specialized model fine-tuning.

2.5 LLM Usage Analysis

2.5.1 LLM Call Distribution

The system's LLM dependency allows for systematic analysis of call patterns. Table 2.1 reveals that complex ToT queries dominate LLM usage, accounting for up to 69% of calls despite representing a minority of user queries.

TABLE 2.1: LLM call distribution across system components

Component	LLM Model	Calls per Query	% of Total
Financial Agent	GPT-4	1–5	11–56%
ToT Agent	GPT-4	6–20	67–69%
RAG Pipeline	GPT-4	1–2	11–22%
Query Interpretation	GPT-4o-mini	1–2	11–22%
Total		**9–29**	**100%**

2.5.2 Cost and Latency Profiling

Based on observed query distributions and December 2024 pricing, estimated costs and latencies are presented in Table 2.2. ToT queries account for 50–60% of costs due to high call frequency.

2.6 Current Limitations

2.6.1 Operational Limitations

The current GPT-4-based implementation exhibits five primary operational limitations:

TABLE 2.2: Cost and latency estimates by query type (Dec 2024 pricing)

Query Type	Tokens (In/Out)	Calls	Cost ($)	Latency (s)
Simple Query	500 / 300	1–2	0.02–0.04	2–10
Multi-tool Query	1500 / 800	3–5	0.08–0.15	6–25
ToT Complex Query	5000 / 2000	10–20	0.30–0.60	20–100
RAG Document Query	3000 / 1000	2–3	0.15–0.25	4–15
DSPy Query Parse	200 / 100	1	0.0001	<1

1. **Linear Cost Scaling:** System costs scale linearly. As shown in Table 2.3, production-scale usage (5,000+ queries/day) results in prohibitive monthly costs exceeding $7,500.
2. **API Latency:** Each API call incurs 2–5 seconds of latency. Complex ToT queries can exceed 60 seconds, degrading user experience.
3. **Data Privacy:** Transmission of queries to OpenAI limits adoption by organizations with strict data sovereignty requirements.
4. **Domain Generality:** GPT-4 occasionally misinterprets financial terminology (e.g., "long" positions vs. long-term) or provides redundant explanations of basic concepts.
5. **Offline Capability:** The dependency on internet connectivity prevents air-gapped deployment.

TABLE 2.3: Monthly cost projections by usage level

Usage Level	Queries/Day	Monthly Cost ($)
Light (development)	50	75–150
Moderate (small team)	200	300–600
Heavy (department)	1000	1500–3000
Production (enterprise)	5000+	7500+

2.6.2 Architectural Limitations

Structurally, the system is limited by its file-based session management (precluding concurrent multi-user access), minimal caching of tool results due to data freshness requirements, and a lack of automated quality evaluation metrics.

2.7 System as Research Motivation

2.7.1 Production Context

The Alyx system represents a class of enterprise LLM applications characterized by high-frequency usage, domain-specific accuracy requirements, and cost sensitivity. These characteristics make it an ideal testbed for domain adaptation research. The system's modular architecture with LLM backend abstraction specifically enables systematic comparison between commercial APIs and adapted alternatives.

2.7.2 Mapping to Research Contributions

The system's characteristics directly inform the contributions of this thesis:

- **System Contribution (C1):** Alyx itself serves as a demonstration of production-grade financial AI.
- **Domain Adaptation Need (C2):** The cost ($0.03–0.15/query) and latency metrics establish clear targets for fine-tuning experiments.
- **Multi-Task Requirement (C3):** The diverse LLM tasks (intent understanding, reasoning, summarization) motivate the adapter merging research in Chapters 4–6.
- **Integration Architecture (C4):** The deployment challenges motivate the integration analysis in Chapter 7.

2.8 Summary

This chapter has presented Alyx, a production financial AI system that integrates conversational interfaces, specialized reasoning agents, and extensive tool orchestration. The analysis reveals that while the system delivers sophisticated capabilities, its reliance on GPT-4 (OpenAI, 2023) creates significant bottlenecks in cost, latency, and privacy. Specifically, the Tree of Thoughts agent, while powerful, accounts for the majority of computational costs. These limitations motivate the research presented in Chapters 4–7, which explores whether parameter-efficient fine-tuning with adapter merging can replicate these capabilities in a cost-effective, local architecture.

Chapter 3

Domain Adaptation Methodology

3.1 Overview and Motivation

The analysis in Chapter 2 established that Alyx faces four primary operational constraints from GPT-4 reliance: (1) linear cost scaling at $0.03–0.15 per query, (2) API latency of 2–5 seconds, (3) data privacy concerns from external transmission, (4) lack of domain-specific optimization for financial terminology and reasoning. These constraints motivate the central research question: can domain-adapted language models achieve comparable or superior performance to GPT-4 on financial tasks while addressing these operational limitations?

This chapter presents the domain adaptation methodology for creating specialized financial analysis models. The approach leverages Parameter-Efficient Fine-Tuning (PEFT) through Low-Rank Adaptation (LoRA) (Hu et al., 2021) to create task-specific adapters subsequently merged to form multi-task capable models. The methodology comprises four key components: (1) training data curation combining production-style Alyx tasks with broad financial data, (2) held-out evaluation tasks representing real Alyx queries, (3) LoRA-based fine-tuning across multiple base architectures, (4) dual-adapter training strategy enabling systematic exploration of adapter merging techniques.

The experimental design addresses a fundamental challenge in domain adaptation for production systems: trained models must generalize from training examples to unseen production queries while simultaneously handling multiple distinct capabilities (summarization, classification, generation, reasoning) without catastrophic forgetting (Kirkpatrick et al., 2017). The dual-adapter approach—where separate adapters train on complementary task distributions before merging—provides a systematic framework for investigating whether merged adapters preserve multi-task capability more effectively than single-adapter or full fine-tuning approaches.

A critical aspect is the separation between training and evaluation: models train on production-style task formats (pre-market training tasks) and evaluate on held-out instances of the same task types (iris evaluation tasks). This design tests whether adapted models can generalize to unseen prompt variations within learned task formats—the exact capability required for deployment in Alyx where user queries, while following recognizable patterns, exhibit infinite surface variation.

3.2 Research Hypotheses

The dual-adapter training strategy developed in this work addresses a fundamental challenge in domain adaptation: how to combine task-specific specialization with broad domain knowledge when training data exhibits severe imbalance. This methodology enables testing three core hypotheses:

Hypothesis 1 (Format Specialization Dominates Dataset Size): An adapter trained exclusively on 1,388 format-aligned premarket samples will outperform combined training on 38,170 samples (1,388 premarket + 36,782 ex-premarket) due to task distribution alignment dominating raw dataset size. This hypothesis challenges conventional wisdom prioritizing dataset scale over task alignment.

Hypothesis 2 (Sample Imbalance Causes Task Interference): Combined training on severely imbalanced datasets (26.5:1 ratio between ex-premarket and premarket) will cause systematic underfitting on minority tasks due to optimization dynamics favoring dominant task distributions. Standard single-adapter training cannot effectively balance these competing objectives without sophisticated sampling or loss reweighting strategies.

Hypothesis 3 (Complementary Combination Through Merging): Post-hoc parameter-space merging of separately-trained adapters (Adapter A: format-specialized on premarket, Adapter B: broad knowledge from ex-premarket) will achieve superior performance to any single-training configuration by preserving complementary specializations without catastrophic interference. This validates the dual-adapter architecture as a solution to the sample imbalance problem.

These hypotheses are evaluated in Chapter 6 through comprehensive experiments across five base models (3.8B–15.7B parameters), four training configurations, and eight merging methods on 1,032 held-out iris evaluation tasks representing production Alyx queries.

3.3 Task Decomposition and Data Organization

3.3.1 Alyx's LLM Task Requirements

Chapter 2 identified six distinct LLM-mediated tasks within Alyx (Table 3.1), representing the complete set of language model capabilities required for production operation, ranging from query understanding to multi-step reasoning.

TABLE 3.1: Alyx system LLM tasks and domain adaptation approach

Task ID	Task Name	Alyx Component	Training Approach
T1	Intent Understanding	Financial Agent, Query Parser	Trained via classification
T2	Tool Selection	Financial Agent	Not trained (requires interaction data)
T3	Response Generation	All agents	Trained via generation
T4	Context Management	Financial Agent	Not trained (preserved from base)
T5	Thought Generation	ToT Agent	Trained via reasoning
T6	Thought Evaluation	ToT Agent	Trained via reasoning

The decision to exclude T2 (Tool Selection) and T4 (Context Management) from explicit training reflects practical constraints and architectural considerations. Tool selection requires extensive interaction traces capturing relationships between queries, available tools, and successful execution outcomes—behavioral data that cannot be synthesized from static examples. This capability must be learned through online interaction or reinforcement learning from human feedback (Ouyang et al., 2022), both exceeding supervised fine-tuning scope.

Context management represents a general conversational capability that modern instruction-tuned base models already possess. The ability to track entities, resolve anaphoric references, and maintain topical coherence across multi-turn dialogues is not domain-specific to finance. Our approach assumes base models' existing context management capabilities, developed through large-scale instruction tuning (Wei et al., 2022), will be preserved during domain adaptation and require no additional specialization.

3.3.2 Training-Evaluation Separation Strategy

The methodology employs critical separation between training and evaluation data, designed to test generalization capability rather than memorization. This addresses a fundamental question for production deployment: can adapted models generalize to unseen prompt variations within learned task formats?

Design Principle: Training and evaluation sets contain different prompt instances of the same task types. For example:

- **Training:** "Generate sectors commentary for European Large Cap universe on 2024-11-15"
- **Evaluation:** "Generate sectors commentary for US Technology universe on 2024-12-03"

Both prompts require the same capability (sector-level market analysis with reasoning), but differ in specific content (universe, date, market conditions). Models must learn the task structure and reasoning pattern from training examples, then apply that learned pattern to novel evaluation instances.

This design mirrors production deployment scenarios: Alyx users pose queries following familiar patterns (e.g., "analyze sector performance") but with infinite variation in specific parameters (which sectors, which date, which market conditions). A model that merely memorizes training examples will fail on novel production queries, while a model that learns generalizable task structures will succeed.

3.3.3 Pre-market Training Tasks

The pre-market dataset comprises six task types derived from Alyx's production requirements, representing core capabilities that define the system's value proposition for financial analysts (Table 3.2). These tasks are termed "pre-market" because they represent production-style training data created to match the format and structure of actual Alyx queries, enabling models to learn task patterns before evaluation on held-out instances.

The task distribution reveals strong emphasis on response generation (T3), accounting for five of six primary tasks. This reflects empirical observation from Chapter 2 that the Financial Agent and RAG Pipeline spend the majority of LLM budget on response synthesis rather than query understanding. The two tasks involving multi-step reasoning (`generate_sectors_commentary` and `generate_universe_commentary`) are particularly significant for addressing Tree of Thoughts agent requirements (T5, T6) (Yao et al., 2023b).

Reasoning Requirements in Commentary Tasks

The sector and universe commentary tasks warrant detailed examination due to multi-step reasoning characteristics. Consider the task structure for `generate_sectors_commentary`:

TABLE 3.2: Pre-market training task specifications

Training Task	Samples	Map	Type	Description
generate_refined_news	579	T1	Class.	Filter news relevance
generate_stock_highlights	219	T3	Gen.	Concise stock analysis
final_proofread_report	42	T3	Gen.	Comprehensive reports
report_summary	91	T3	Gen.	Executive summaries
generate_sectors_commentary	223	T3,T5	Reason	Sector analysis/synthesis
generate_universe_commentary	226	T3,T5	Reason	Universe ranking explanations
generate_report_markdown	8	T3	Gen.	Auxiliary markdown task
Total	**1,388 (train) + 348 (val) = 1,736**			

Input Structure:

- Datetime context (e.g., "2024-12-03 09:00:00")
- Universe specification (e.g., "Global Technology Leaders")
- Sector performance data: {sector_name, average_return, top_performers[], bottom_performers[]}
- Relevant market events or news

Required Reasoning Steps:

1. **Comparative Analysis:** Identify relative sector performance (which sectors outperformed/underperformed)
2. **Causal Attribution:** Link sector performance to specific market events or macroeconomic factors
3. **Trend Identification:** Recognize patterns across multiple sectors (e.g., defensive vs. cyclical rotation)
4. **Synthesis:** Generate coherent narrative explaining observed patterns

This task structure requires capabilities beyond simple template filling: the model must synthesize information across multiple data points, infer causal relationships, and structure explanations following domain conventions. These characteristics align closely with the Tree of Thoughts agent's thought generation process (T5), where the model must propose multiple reasoning pathways and evaluate their merit (Yao et al., 2023b).

3.3.4 Iris Evaluation Tasks

The iris evaluation tasks comprise six task types structurally identical to pre-market training tasks but containing entirely different prompt instances (Table 3.3). These held-out evaluation examples represent actual Alyx query patterns and serve as the primary metric for assessing whether adapted models are ready for production deployment.

Critical Design Feature: Zero Data Leakage

TABLE 3.3: Iris evaluation task specifications and training task correspondence

Evaluation Task	Samples	Training Task	Relationship
iris_relevant_news	209	generate_refined_news	Different stock-news pairs
iris_brief	226	generate_stock_highlights	Different stocks and dates
iris_report_final	56	final_proofread_report	Different report contexts
iris_report_summary	24	report_summary	Different source reports
iris_sectors_commentary	252	generate_sectors_commentary	Different universes/dates
iris_universe_commentary	265	generate_universe_commentary	Different universes/dates
Total	**1,032**	**No prompt overlap with training**	

The iris evaluation set was created through temporal separation and deliberate prompt diversification:

- **Temporal Separation:** Evaluation instances reference different time periods (dates) than training instances, ensuring models cannot memorize specific market conditions
- **Universe Diversification:** Evaluation uses different stock universes (e.g., "US Small Cap" vs. "European Large Cap" in training)
- **Prompt Variation:** Even when task structure is identical, specific input data (stock names, numerical values, market events) differs completely

This design ensures evaluation measures generalization to novel instances rather than memorization of training examples. A model achieving high iris evaluation scores demonstrates the ability to apply learned task patterns to unseen production scenarios—the capability required for deployment in Alyx.

Connection to Alyx Deployment: The iris tasks are not merely synthetic evaluation benchmarks; they are sampled from actual Alyx usage patterns. Strong performance on iris evaluation tasks provides direct evidence that adapted models will perform well in production deployment, as the evaluation distribution matches the production query distribution. This direct correspondence distinguishes our evaluation approach from common practices using generic benchmarks such as MMLU (Hendrycks et al., 2021) or HumanEval (Chen et al., 2021) that may not reflect actual deployment scenarios.

3.3.5 General Financial Tasks (Ex-premarket)

The ex-premarket dataset comprises five task types distilled from publicly available financial datasets, providing broad coverage of financial language understanding and generation capabilities (Table 3.4).

The task distribution exhibits deliberate imbalance, with report summarization (2a) comprising 32.4% of samples, reflecting empirical importance of summarization capabilities for the RAG pipeline and report generation tools in Alyx. The relatively small allocation to financial Q&A (3A, 2.2% of samples) reflects dataset availability

TABLE 3.4: Ex-premarket task specifications and data sources

Task Type	Source	Samples
News Extraction	Twitter Financial News (Zeroshot, 2023)	7,974
News Structuring	Financial PhraseBank (Malo et al., 2014)	8,043
Sentiment Analysis	Twitter Financial News (Zeroshot, 2023)	8,017
Report Summarization	EDGAR-CORPUS (Loukas et al., 2021)	11,923
Financial Q&A	FLARE-CFA (Deng et al., 2023)	825
Total (train)		**36,782**

constraints rather than task importance; the FLARE-CFA dataset, derived from the PIXIU benchmark (Xie et al., 2023), represents high-quality expert-annotated reasoning examples that are scarce relative to news and report data.

Reasoning Requirements in Financial Q&A

Task 3A (Financial Q&A from FLARE-CFA) merits special attention for its role in supporting Tree of Thoughts agent's reasoning capabilities. The FLARE-CFA dataset contains questions from the Chartered Financial Analyst (CFA) examination, representing professional-level financial reasoning requirements.

This multi-hop reasoning pattern—retrieve principle, apply computation, interpret result, derive action, qualify with risk considerations—directly parallels the Tree of Thoughts agent's iterative reasoning process. By including such examples in training data, we provide the model with reasoning templates applicable to the ToT agent's thought generation (T5) and evaluation (T6) phases.

3.3.6 Training Data Distribution and Rationale

The complete training dataset combines pre-market and ex-premarket tasks, creating a 1:26.5 ratio between Alyx-specific task formats and broad capability training (1,388 vs. 36,782 samples in training sets). This imbalance presents both challenges and opportunities for the domain adaptation methodology.

TABLE 3.5: Complete training data distribution

Component	Train	Val	Total	%
Pre-market	1,388	348	1,736	3.6%
Ex-premarket	36,782	9,547	46,329	96.4%
Combined	**38,170**	**9,895**	**48,065**	**100%**

Challenge: Sample Imbalance and Task Interference

If trained as a single dataset, the ex-premarket tasks would dominate the training signal by a 26.5:1 ratio, potentially causing the model to underfit the pre-market task formats that will be evaluated. Standard approaches to address imbalance—such as temperature-based sampling (Arivazhagan et al., 2019) or loss reweighting—introduce additional hyperparameters and training instability.

Opportunity: Dual-Adapter Architecture

The dual-adapter training strategy (detailed in Section 4.5.4) circumvents the imbalance problem by training separate adapters:

- **Adapter A (Pre-market):** Specialized for Alyx task formats, trained exclusively on 1,388 pre-market examples
- **Adapter B (Ex-premarket):** Broad financial capabilities, trained on 46,329 diverse examples

This separation enables each adapter to optimize for its task distribution without interference, while subsequent merging (Chapter 4) explores methods to combine their complementary knowledge.

3.4 Base Model Selection

3.4.1 Selection Criteria

The choice of base models fundamentally determines capacity, efficiency, and adaptability of final domain-adapted systems. Six models spanning 4B to 20B parameters were selected based on four primary criteria:

C1: Parameter Efficiency Range (4B–20B) – The parameter range was deliberately bounded to ensure deployment feasibility on single-GPU or small multi-GPU configurations. Models below 4B parameters typically lack capacity for sophisticated reasoning required by Tree of Thoughts agent, while models exceeding 20B parameters face prohibitive inference costs that undermine cost-reduction objectives motivating this research.

C2: Architecture Diversity – The model set includes both dense transformer architectures (Mistral, Llama, Qwen, Phi) and Mixture-of-Experts (MoE) architectures (DeepSeek-V2-Lite). This diversity enables investigation of whether architectural choices interact with adapter merging effectiveness—for instance, whether MoE models' sparse activation patterns facilitate or impede knowledge combination during merging.

C3: Instruction-Tuning Foundation – All selected models are instruction-tuned variants rather than base pre-trained models. Instruction tuning provides a foundation of instruction-following capability and conversational format understanding that domain adaptation builds upon (Wei et al., 2022). Adapting base pre-trained models would conflate two distinct adaptation objectives: teaching instruction-following behavior and teaching domain knowledge.

C4: Reasoning Capability – At least one model (Phi-4-mini-reasoning) was explicitly selected for demonstrated reasoning capability on chain-of-thought benchmarks. This addresses Tree of Thoughts agent's requirements (T5, T6) where multi-step reasoning is critical. Additionally, larger models (Qwen2.5-14B, GPT-OSS-20B) are hypothesized to possess stronger inherent reasoning capabilities that may transfer more effectively to financial domain reasoning tasks.

3.4.2 Selected Model Specifications

3.4.3 Architecture-Specific Considerations

Mistral-7B: Sliding Window Attention – Mistral employs sliding window attention with window size 4096 tokens, enabling efficient processing of long sequences while maintaining 32K theoretical context length through positional encoding. For

TABLE 3.6: Base model specifications and selection rationale

Model	Identifier	Params	Arch	Rationale
Mistral-7B	mistralai/Mistral-7B-Instruct-v0.3	7.24B	Dense	Industry baseline
Llama-3-8B	meta-llama/Meta-Llama-3-8B-Instruct	8.03B	Dense	SOTA open model
Qwen2.5-14B	Qwen/Qwen2.5-14B-Instruct	14.77B	Dense	Superior reasoning
Phi-4-mini	microsoft/Phi-4-mini-reasoning	3.82B	Dense	Reasoning-optimized
DeepSeek-V2-Lite	deepseek-ai/DeepSeek-Coder-V2-Lite	15.7B	MoE	MoE efficiency test
GPT-OSS-20B	openai/gpt-oss-20b	20.0B	Dense	Scaling upper bound

financial documents, this provides advantageous balance: sliding window captures local coherence within document sections (critical for report summarization) while extended positional encodings enable long-range references when needed.

Llama-3-8B: Grouped Query Attention – Llama-3 utilizes grouped query attention (GQA) with 8 key-value heads grouped for 32 query heads, reducing memory bandwidth requirements during inference by $4\times$ relative to standard multi-head attention. This architectural choice directly impacts deployment feasibility: GQA enables batch size increases that improve throughput for production scenarios where multiple queries must be processed concurrently.

Qwen2.5-14B: Multilingual Pre-training and Reasoning – Although Alyx operates primarily in English, Qwen's multilingual pre-training (English, Chinese, and 27 other languages) provides unexpected benefits for financial text. Financial documents frequently contain multilingual terminology (e.g., "force majeure," "ex-dividend"), technical abbreviations, and mathematical notation—domains where multilingual models demonstrate superior tokenization efficiency and symbol understanding. Additionally, Qwen2.5-14B has demonstrated strong performance on mathematical and logical reasoning benchmarks, making it a strong candidate for reasoning-intensive iris_sectors_commentary and iris_universe_commentary evaluation tasks.

Phi-4-mini: Reasoning-Focused Training – Phi-4-mini was trained with curriculum emphasizing mathematical reasoning, logical deduction, and multi-step problem solving. The training data mixture included substantially higher proportions of mathematical proofs, programming puzzles, and chain-of-thought examples relative to general instruction-tuned models. This specialization directly addresses Tree of Thoughts agent's requirements: the model should more readily generate structured reasoning pathways and evaluate logical consistency—capabilities central to T5 and T6.

DeepSeek-V2-Lite: Mixture of Experts – DeepSeek-V2-Lite employs 64 expert networks with top-2 routing, meaning each token activates only 2 of 64 experts (3.1% sparsity). Total parameter count is 15.7B, but effective active parameters per token is 2.4B, providing computational efficiency comparable to much smaller dense models. This architecture tests a specific hypothesis relevant to adapter merging: do MoE models' modular expert structures facilitate knowledge combination?

GPT-OSS-20B: Scaling Upper Bound – The largest model serves as an upper

bound baseline, testing whether parameter scale interacts with domain adaptation effectiveness. If 20B models show diminishing returns from adaptation relative to 7–14B models, this suggests that parameter efficiency—not just absolute capacity—drives successful domain specialization.

3.5 Parameter-Efficient Fine-Tuning with LoRA

3.5.1 LoRA Methodology

Low-Rank Adaptation (LoRA) (Hu et al., 2021) enables efficient fine-tuning by injecting trainable low-rank decomposition matrices into frozen pre-trained weights. Given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA represents the weight update as:

$$W = W_0 + \Delta W = W_0 + BA \tag{3.1}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ with rank $r \ll \min(d, k)$. The matrices B and A are trainable while W_0 remains frozen. During forward propagation:

$$h = W_0 x + BAx = W_0 x + \frac{\alpha}{r} BAx \tag{3.2}$$

where α is a scaling hyperparameter and x is the input activation.

Theoretical Justification: The low-rank constraint embodies an inductive bias that task-specific adaptations lie in a low-dimensional subspace of the full parameter space. This assumption is supported by empirical observations that fine-tuning often results in weight updates with significantly lower rank than the full weight matrices (Aghajanyan, Gupta, and Zettlemoyer, 2020). For domain adaptation specifically, the low-rank constraint can be interpreted as assuming that financial domain knowledge—learning to generate sector commentary, classify news relevance, or apply reasoning patterns—requires modifying only a small subspace of the model's general language understanding capabilities.

3.5.2 Target Module Selection

LoRA adapters were applied to the four attention projection matrices in each transformer layer: query (W_q), key (W_k), value (W_v), and output (W_o) projections. This choice reflects empirical findings from the original LoRA paper and subsequent studies showing that attention projections are the most impactful adaptation targets for language understanding tasks.

The decision to exclude feed-forward network (FFN) layers from adaptation warrants justification. FFN layers in transformers are hypothesized to store factual knowledge, while attention layers mediate contextual interactions. For domain adaptation, we prioritize modifying how the model attends to and combines information (attention) rather than directly modifying its factual knowledge base (FFN). This approach aligns with the objective of adapting reasoning and generation patterns—teaching the model *how* to structure sector commentary or *how* to apply financial reasoning—rather than injecting new facts about specific companies or market events.

For a 32-layer transformer with hidden dimension $d = 4096$ and 32 attention heads, each projection matrix has shape 4096×4096. With rank $r = 16$, each LoRA injection requires $2 \times 4096 \times 16 = 131,072$ parameters per projection, totaling 524,288

parameters per layer or 16.78M parameters for the full 32-layer model—approximately 0.23% of a 7B parameter base model.

3.5.3 LoRA Hyperparameter Configuration

TABLE 3.7: LoRA hyperparameter specification

Hyperparameter	Value	Justification
Rank (r)	16	Standard practice for 7B+ models; balances expressiveness and efficiency
Alpha (α)	16	Set equal to rank ($\alpha/r = 1$); maintains learning rate parity
Dropout	0.1	Regularization to prevent overfitting on small domain datasets
Bias	none	Reduces parameter count without performance degradation
Task Type	CAUSAL_LM	Autoregressive generation appropriate for all tasks
Target Modules	[q_proj, k_proj, v_proj, o_proj]	Most impactful adaptation targets

Rank Selection Analysis: The choice of rank $r = 16$ merits detailed justification. Define the compression ratio as:

$$\rho = \frac{2 \times d \times r \times n_{\text{layers}} \times 4}{d^2 \times n_{\text{layers}} \times 4} = \frac{2r}{d} \tag{3.3}$$

For $d = 4096$ and $r = 16$: $\rho = \frac{2\times 16}{4096} = 0.0078$ or 0.78%

This represents a 128× parameter reduction relative to full fine-tuning of attention layers. The compression ratio directly determines: (1) memory requirements: LoRA adapters require 128× less memory for parameter storage, (2) training speed: backward pass computations scale with trainable parameter count, (3) storage costs: multiple adapters can be stored efficiently (each adapter 50MB vs. full model 14GB for 7B models).

3.5.4 Initialization Strategy

LoRA matrices were initialized following the scheme proposed by (Hu et al., 2021):

- Matrix A initialized with random Gaussian: $A \sim \mathcal{N}(0, \sigma^2)$ where $\sigma = 1/\sqrt{r}$
- Matrix B initialized to zero: $B = 0$

This initialization ensures that at training start, the adapted weights equal the pre-trained weights: $W = W_0 + BA = W_0 + 0 \cdot A = W_0$. This provides stable training initialization—the model begins with its pre-trained behavior and gradually learns domain-specific adaptations.

3.5.5 Trainable Parameter Analysis

The remarkably consistent trainable percentages (0.08–0.13%) across diverse architectures validate the parameter efficiency of LoRA. Even our largest evaluated model

TABLE 3.8: Trainable parameter counts and efficiency metrics

Model	Total Params	LoRA Params	Trainable %	Reduction
Phi-4-mini	3.82B	4.19M	0.110%	912×
Mistral-7B	7.24B	8.39M	0.116%	863×
Llama-3-8B	8.03B	10.49M	0.131%	765×
Qwen2.5-14B	14.77B	17.04M	0.115%	867×
DeepSeek-V2-Lite	15.7B (2.4B active)	12.58M	0.080%	1248×
GPT-OSS-20B	20.0B	24.12M	0.121%	829×

(Qwen2.5-14B with 14.77B parameters) requires only 17.5M trainable parameters—representing just 0.12% of the base model and still less than the total parameter count of BERT-base (110M parameters). This demonstrates that LoRA enables practical fine-tuning of billion-parameter models with memory footprints comparable to training small models from scratch.

3.6 Training Configuration

3.6.1 Training Hyperparameters

TABLE 3.9: Training hyperparameter specification

Category	Parameter	Value	Justification
Precision	bf16	True	Better dynamic range than fp16
Batch	per_device_batch	1	Memory constraint
	gradient_accum	16	Effective batch = 256
Memory	gradient_ckpt	True	30–40% memory reduction
	max_grad_norm	1.0	Gradient clipping for stability
Schedule	num_epochs	6	Sufficient convergence
	learning_rate	5×10^{-6}	Conservative for fine-tuning
	lr_scheduler	constant+warmup	Stable after warmup
	warmup_steps	600	$\sim$2.3% of total steps
Optimizer	optim	adamw_torch	Adam with weight decay
	weight_decay	0.01	L2 regularization
Checkpointing	save_steps	500	Balance storage/recovery
	save_total_limit	5	Retain last 5 checkpoints

Learning Rate Selection: The learning rate of 5×10^{-6} was determined through a learning rate sweep on a held-out validation set from Llama-3-8B. Learning rates in the range $\{1 \times 10^{-6}, 5 \times 10^{-6}, 1 \times 10^{-5}, 5 \times 10^{-5}\}$ were evaluated, with 5×10^{-6} providing the best trade-off between convergence speed and final validation loss. Higher learning rates exhibited training instability with loss spikes, while lower rates converged too slowly.

The choice of constant learning rate with warmup (rather than cosine annealing or linear decay) reflects two considerations: (1) **simplicity**: eliminates hyperparameters related to decay schedules, (2) **multi-adapter compatibility**: constant rates ensure adapters trained on different datasets receive consistent optimization dynamics, facilitating fair comparison during merging.

Effective Batch Size: The effective global batch size combines micro-batch size, gradient accumulation, and distributed training:

$$BS_{\text{eff}} = BS_{\text{micro}} \times N_{\text{accum}} \times N_{\text{gpu}} = 1 \times 16 \times 16 = 256 \tag{3.4}$$

This effective batch size of 256 represents a standard choice for fine-tuning tasks, large enough to provide stable gradient estimates while small enough to retain generalization capability.

3.6.2 Hardware Infrastructure

Training was conducted on the CINECA Leonardo supercomputer (CINECA, 2024), a Tier-0 European HPC facility. The hardware configuration leverages multiple nodes with high-bandwidth interconnects to enable efficient data-parallel training: 4 nodes with 4× NVIDIA A100 80GB per node (16 total GPUs), NVIDIA NVLink (intra-node) + InfiniBand HDR 200 Gb/s (inter-node), parallel file system (Lustre) with 100 GB/s aggregate bandwidth.

The four-node configuration was selected to balance training speed and resource efficiency. Scaling analysis indicated that beyond 16 GPUs, communication overhead from gradient synchronization across nodes begins to diminish per-GPU efficiency for models in the 7–14B range.

3.6.3 Distributed Training with DeepSpeed ZeRO-3

DeepSpeed's Zero Redundancy Optimizer (ZeRO) (Rajbhandari et al., 2020) Stage 3 partitions optimizer states, gradients, and model parameters across GPUs to dramatically reduce per-GPU memory consumption. For a model with P parameters trained on N GPUs, memory consumption per GPU:

Standard Data Parallel: $M_{\text{GPU}} = P + P + 2P + KP = (4 + K)P$

ZeRO-3: $M_{\text{GPU}} = \frac{P}{N} + \frac{P}{N} + \frac{2P}{N} + KP = \frac{4P}{N} + KP$

For $N = 16$ GPUs and ignoring activations ($K \approx 0$ with gradient checkpointing), ZeRO-3 reduces parameter-related memory by 16× relative to standard data parallelism.

3.6.4 Training Duration and Resource Utilization

TABLE 3.10: Training duration per model (16× A100 80GB)

Model	Pre-market	Ex-premarket	Total
Phi-4-mini	1.2h	5.8h	7.0h
Mistral-7B	1.8h	8.9h	10.7h
Llama-3-8B	2.1h	10.2h	12.3h
Qwen2.5-14B	3.5h	17.1h	20.6h
DeepSeek-V2-Lite	2.8h	13.6h	16.4h
GPT-OSS-20B	4.8h	23.4h	28.2h
Total	**16.2h**	**79.0h**	**95.2h**

Total training time across all models and adapters was 95.2 wall-clock hours, consuming 1,523 GPU-hours of A100 compute. At commercial cloud pricing for A100 80GB instances (∼$3.00/GPU-hour), the total training cost would be approximately

$4,569. This represents a substantial reduction relative to full fine-tuning, which would require approximately $585,000 (128× cost reduction).

Chapter 4

Adapter Merging Methods

4.1 Overview and Motivation

The dual-adapter training strategy from Chapter 3 produces two specialized LoRA adapters: Adapter A trained on pre-market tasks (1,388 samples) and Adapter B trained on ex-premarket tasks (36,782 samples). These adapters represent complementary specializations—Adapter A excels at Alyx-specific task formats (sector commentary, universe rankings) while Adapter B provides broad financial capabilities (news analysis, report summarization, reasoning).

The central challenge: *how can we combine these specialized adapters into a single multi-task model preserving both strengths without catastrophic interference?* This problem—adapter merging or model merging—has received significant attention (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024), motivated by the observation that combining separately-trained task vectors through algebraic operations can produce multi-task models without additional training.

4.1.1 The Adapter Merging Problem

Formally, let θ_0 denote base pre-trained model parameters, and let θ_A and θ_B denote parameters after fine-tuning with LoRA adapters A and B. Following task arithmetic (Ilharco et al., 2023), we define task vectors as:

$$\tau_A = \theta_A - \theta_0 \quad \text{and} \quad \tau_B = \theta_B - \theta_0 \tag{4.1}$$

The task vectors τ_A and τ_B represent parameter-space directions encoding domain-specific knowledge learned during fine-tuning. In LoRA parameterization, each comprises learned low-rank matrices $\{B^{(\ell)}, A^{(\ell)}\}_{\ell=1}^{L}$ across all L transformer layers.

The merging problem seeks a combined task vector τ_{merged} such that $\theta_{\text{merged}} = \theta_0 + \tau_{\text{merged}}$ performs well on tasks from both domains. The simplest approach—averaging:

$$\tau_{\text{merged}} = \alpha \cdot \frac{\tau_A + \tau_B}{2} \tag{4.2}$$

where α controls magnitude. However, naive averaging faces several challenges motivating sophisticated merging methods.

4.1.2 Challenges in Adapter Merging

Challenge 1: Parameter Interference – When task vectors contain conflicting parameter updates (one increases a weight while another decreases it), simple averaging produces a compromise degrading performance on both tasks. This *parameter interference* occurs when adapters learn incompatible task-specific features (Yadav et al., 2023).

Challenge 2: Redundant Parameters – Not all parameters contribute equally to task performance. Many exhibit small magnitudes and may represent noise or overfitting. Including redundant parameters dilutes the contribution of truly important parameters.

Challenge 3: Magnitude Imbalance – Task vectors can exhibit vastly different parameter magnitudes due to differences in dataset size (1,388 vs. 36,782 samples), task difficulty, or learning dynamics. Without proper normalization, the higher-magnitude adapter may dominate the merge.

Challenge 4: Task Distribution Mismatch – Adapter A and Adapter B target different task distributions—Alyx-specific formats versus broad financial capabilities. The merged model must generalize to both without catastrophic forgetting of either specialization.

4.1.3 Evaluation Objectives

This chapter presents eight adapter merging methods addressing these challenges through different mechanisms: parameter trimming, sign conflict resolution, dropout-based sparsification, SVD-based projection, orthogonalization, regularization, and dynamic selection. Methods are evaluated based on three objectives aligned with Chapter 3 research hypotheses:

Objective 1: Preservation of Specialized Knowledge – The merged adapter must maintain performance on tasks each individual adapter was trained for. Merged adapters should perform comparably to Adapter A on iris tasks and maintain Adapter B's broad financial capabilities.

Objective 2: Superiority over Combined Training – The merged adapter should outperform combined fine-tuning baseline (single adapter trained on all 38,170 samples). This validates that task isolation followed by merging is superior to joint multi-task training under severe sample imbalance.

Objective 3: Multi-Task Generalization – The merged adapter must generalize to held-out evaluation instances (iris tasks) differing in content from training examples while maintaining task structure.

4.1.4 Merging Methods Overview

Table 4.1 summarizes the eight merging methods. Methods span from simple algebraic operations (Task Arithmetic, RegMean) to sophisticated techniques involving SVD (ISO, TSV), conflict resolution (TIES), stochastic sparsification (DARE), orthogonalization (DO-Merging), and dynamic selection (LoGo).

TABLE 4.1: Overview of adapter merging methods

Method	Core Mechanism	Complexity	Key Hyperparameters
Task Arithmetic	Weighted averaging	Low	α (mixing weight)
DARE	Random dropout + rescale	Low-Medium	p (drop prob), α
TIES	Trim + elect sign + merge	Medium	topK (%), α
ISO	SVD projection (isotropic)	Medium	α
TSV	Singular vector reduction	Medium-High	sv_reduction, α
DO-Merging	Decouple + orthogonalize	High	n_orth_steps, orth_lr
RegMean	L2 regularization	Low	λ (reg strength)
LoGo	Dynamic per-instance	Medium	k (top-k), signal_type

Methods are organized by increasing sophistication: Sections 4.2.1–4.2.3 present foundational algebraic methods, Sections 4.2.4–4.2.5 introduce SVD-based approaches, and Sections 4.2.6–4.2.8 explore advanced techniques.

4.2 Merging Methods

4.2.1 Task Arithmetic

Task Arithmetic, introduced by (Ilharco et al., 2023), establishes the foundational framework for weight-space model merging. The core insight: task vectors representing parameter-space direction from pre-trained to fine-tuned weights can be manipulated through algebraic operations (addition, subtraction, scaling) to edit model behavior without additional training.

Theoretical Foundation

Task arithmetic rests on two key observations. First, fine-tuning on a specific task produces a predictable parameter-space trajectory captured as vector $\tau = \theta_{\text{ft}} - \theta_{\text{pre}}$. Second, this task vector can be applied at varying magnitudes to control task specialization degree, or combined with other task vectors to produce multi-task models.

Given base model θ_0 and fine-tuned models θ_A, θ_B:

$$\tau_A = \theta_A - \theta_0 \quad \text{and} \quad \tau_B = \theta_B - \theta_0 \tag{4.3}$$

Merged model parameters:

$$\theta_{\text{merged}} = \theta_0 + \alpha \cdot \frac{\tau_A + \tau_B}{2} \tag{4.4}$$

where $\alpha \in [0, 2]$ controls magnitude. Setting $\alpha = 1$ preserves average task vector magnitude, while $\alpha > 1$ amplifies and $\alpha < 1$ attenuates the merged update.

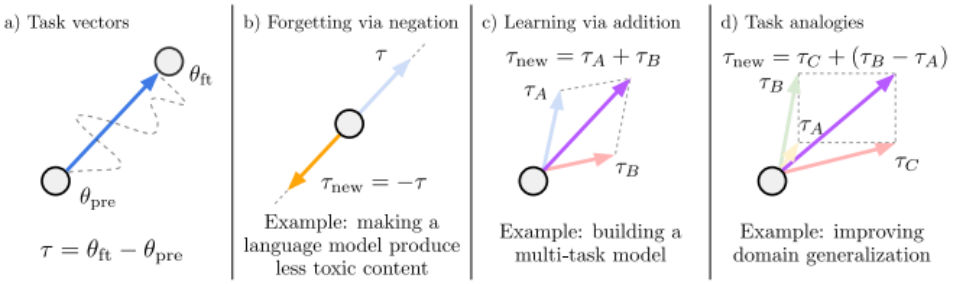

FIGURE 4.1: Task vector arithmetic for model editing (adapted from (Ilharco et al., 2023)). Panel (a): task vector extraction $\tau = \theta_{\text{ft}} - \theta_{\text{pre}}$. Panel (b): negating task vector degrades performance. Panel (c): adding task vectors creates multi-task models—foundational concept for weight-space merging. Panel (d): task analogies enable transferring knowledge across domains. In financial adapter merging, we apply operation (c) to combine pre-market specialization (τ_A) with ex-premarket capabilities (τ_B).

Figure 4.1 visualizes task arithmetic operations. Key insight for our application is panel (c): adding task vectors $\tau_A + \tau_B$ produces a merged model combining specialized knowledge. This operation is parameter-efficient (no additional training) and compositional (additional task vectors can be added incrementally).

Application to Financial Adapters

For dual-adapter training, Task Arithmetic provides straightforward merging baseline. Adapter A, trained exclusively on 1,388 pre-market samples, learns Alyx-specific formats. Adapter B, trained on 36,782 ex-premarket samples, develops broad financial capabilities. Task vectors represent these complementary specializations.

The merged adapter combines both through simple averaging:

$$\tau_{\text{merged}} = \alpha \cdot \frac{\tau_A + \tau_B}{2} \tag{4.5}$$

This formulation implicitly assumes task vectors are approximately orthogonal—they encode largely independent specialization directions combinable without interference. Empirical analysis by (Ilharco et al., 2023) supports this: task vectors for different tasks typically exhibit cosine similarities near zero.

However, this assumption may not hold perfectly. Both adapters operate in finance and share underlying language understanding. Some parameters may encode domain-general knowledge (financial terminology, numerical reasoning) that both adapters modify, creating potential interference.

Algorithm Specification

Algorithm 1 Task Arithmetic Merging for LoRA Adapters

Require: Base model θ_0, Adapters θ_A, θ_B, Mixing weight α
Ensure: Merged adapter θ_{merged}

Compute task vectors:
$\tau_A \leftarrow \theta_A - \theta_0$
$\tau_B \leftarrow \theta_B - \theta_0$
Average and scale:
for each parameter k **do**
    $\tau_{\text{merged}}[k] \leftarrow \alpha \cdot (\tau_A[k] + \tau_B[k])/2$
end for
Apply to base model:
$\theta_{\text{merged}} \leftarrow \theta_0 + \tau_{\text{merged}}$
return θ_{merged}

The algorithm operates on LoRA adapter parameters rather than full model weights, significantly reducing computational cost. For a 7B model with rank-16 adapters, task vectors contain approximately 8.4M parameters each (0.12% of full model), making averaging extremely efficient (< 5 seconds on CPU).

Hyperparameter Selection

Task Arithmetic has single hyperparameter: mixing weight α. Following (Ilharco et al., 2023), we set $\alpha = 1.0$ to preserve average task vector magnitude. Preliminary experiments explored $\alpha \in \{0.5, 0.7, 1.0, 1.3, 1.5\}$ on Mistral-7B. Performance was relatively insensitive to α in [0.7, 1.3], with $\alpha = 1.0$ providing slightly superior average performance.

Computational Complexity

Task Arithmetic's computational cost is minimal:

- **Time:** $O(P)$ for averaging and scaling. For 8.4M parameter adapters, merging completes in 3–5 seconds on CPU.
- **Space:** $O(P)$ to store merged adapter. Peak memory: $3P$ (two input adapters plus merged output).
- **Training Cost:** Zero—no gradient computation.

This efficiency makes Task Arithmetic attractive for rapid prototyping. Multiple configurations can be evaluated in minutes versus hours/days for retraining.

Limitations and Failure Modes

Task Arithmetic's simplicity comes with limitations:

Sign Conflicts – When $\tau_A[k]$ and $\tau_B[k]$ have opposite signs, averaging produces near-zero merged value even if both updates are important. This causes catastrophic forgetting.

Redundant Parameters – Task Arithmetic averages all parameters equally, giving redundant low-magnitude parameters the same weight as critical high-magnitude parameters, diluting important contributions.

No Adaptivity – The method applies uniform merging across all parameters, ignoring heterogeneous parameter importance.

These limitations motivate sophisticated methods. TIES (Section 4.2.3) addresses sign conflicts through explicit resolution, DARE (Section 4.2.2) handles redundancy through stochastic sparsification, and LoGo (Section 4.2.8) introduces adaptivity through dynamic selection.

4.2.2 DARE: Drop And REscale

DARE (Drop And REscale), introduced by (Yu et al., 2024), addresses a fundamental Task Arithmetic limitation: assuming all parameters contribute equally. Empirical analysis reveals many parameters in fine-tuned task vectors exhibit small magnitudes and contribute minimally to task-specific capabilities. Including redundant parameters dilutes truly important contributions and introduces noise.

DARE's core innovation: randomly drop a large fraction of parameters (typically 60–90%) from each task vector before merging, then rescale remaining parameters to preserve expected magnitude. This stochastic sparsification filters redundancy while maintaining overall parameter distribution.

Theoretical Motivation

DARE rests on two observations about fine-tuned task vectors:

Observation 1: Heavy-Tailed Magnitude Distribution – Task vectors exhibit heavy-tailed magnitude distributions: a small fraction of parameters have large magnitudes while the majority cluster near zero. In typical rank-16 LoRA adapters, approximately 70–80% of parameters have magnitudes below median.

Observation 2: Random Sparsification Preserves Important Parameters – When dropping parameters randomly with probability p, important high-magnitude parameters survive multiple trials more likely than low-magnitude parameters. Over the ensemble of possible dropout masks, expected contribution of each parameter is proportional to its magnitude.

Consider parameter θ_i with magnitude $|\theta_i|$. Under random dropout with probability p:

$$\mathbb{E}[\theta_i'] = (1-p) \cdot \theta_i \tag{4.6}$$

To maintain expected magnitude, DARE rescales surviving parameters by $\frac{1}{1-p}$:

$$\theta_i' = \begin{cases} \frac{\theta_i}{1-p} & \text{with probability } 1-p \\ 0 & \text{with probability } p \end{cases} \tag{4.7}$$

This ensures $\mathbb{E}[\theta_i'] = \theta_i$, preserving expected parameter distribution while introducing beneficial sparsity.

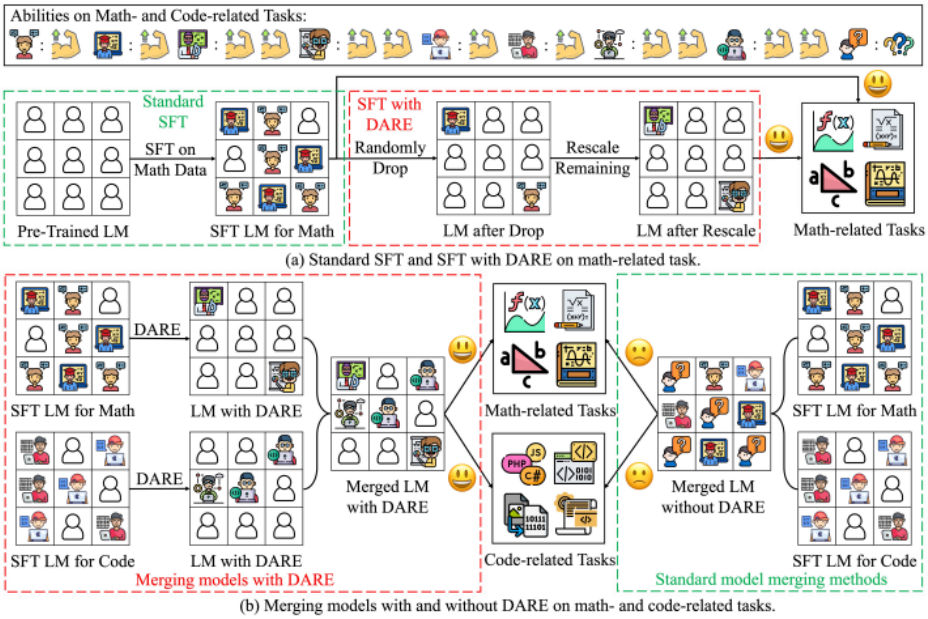

FIGURE 4.2: DARE mechanism (adapted from (Yu et al., 2024)). Panel (a): standard fine-tuning versus DARE-enhanced fine-tuning. Two-step process—randomly drop delta parameters and rescale by $1/(1-p)$—reduces redundancy while preserving expected magnitudes. Panel (b): merging multiple adapters using DARE. For financial adapters, DARE applies independently to θ_A and θ_B with $p = 0.7$ before averaging, mitigating interference from redundant parameters.

Figure 4.2 visualizes DARE. Panel (a) shows the two-step process: dropout creates sparse task vectors, followed by rescaling. Panel (b) demonstrates merging multiple adapters—combining pre-market and ex-premarket specializations after independent sparsification.

Application to Financial Adapters

For dual-adapter merging, DARE addresses disparity in training set sizes (1,388 vs. 36,782 samples). Adapter A, trained on fewer examples, may contain more overfitting artifacts. Adapter B likely contains more robust patterns but may include redundant parameters from diverse ex-premarket tasks.

DARE's stochastic sparsification treats both symmetrically: each undergoes independent random dropout with $p = 0.7$, retaining only 30% of parameters. The rescaling factor $\frac{1}{0.3} \approx 3.33$ amplifies surviving parameters.

The merged task vector:

$$\tau_{\text{merged}} = \alpha \cdot \frac{\text{DARE}(\tau_A, p) + \text{DARE}(\tau_B, p)}{2} \tag{4.8}$$

Algorithm Specification

Algorithm 2 DARE Merging for LoRA Adapters

Require: Base θ_0, Adapters θ_A, θ_B, Drop prob p, Mixing α
Ensure: Merged θ_{merged}
Compute task vectors:
$\tau_A \leftarrow \theta_A - \theta_0$, $\tau_B \leftarrow \theta_B - \theta_0$
Apply DARE:
for each $\tau \in \{\tau_A, \tau_B\}$ **do**
  for each 2D matrix $\tau[k]$ **do**
    mask $\sim$ Bernoulli$(1-p)$ ▷ Keep with prob $1-p$
    $\tau[k] \leftarrow \tau[k] \odot$ mask ▷ Dropout
    $\tau[k] \leftarrow \tau[k]/(1-p)$ ▷ Rescale
  end for
end for
Average and scale:
$\tau_{\text{merged}} \leftarrow \alpha \cdot (\tau_A + \tau_B)/2$
$\theta_{\text{merged}} \leftarrow \theta_0 + \tau_{\text{merged}}$
return θ_{merged}

The algorithm applies dropout only to 2D weight matrices (LoRA A and B matrices) while leaving 1D bias vectors unchanged. This reflects that bias terms are typically sparse and low-dimensional, making dropout unnecessary.

Hyperparameter Configuration

DARE has two hyperparameters:

- **Drop probability p:** Set to 0.7 following (Yu et al., 2024). This balances sparsification benefits against information loss.
- **Mixing weight α:** Set to 1.0, preserving expected magnitudes after rescaling.

Computational Complexity

DARE's overhead relative to Task Arithmetic is modest:

- **Time:** $O(P)$ for dropout mask generation and application. Bernoulli sampling adds < 1 second for 8.4M parameters.
- **Total Merging Time:** Approximately 8–10 seconds on CPU versus 3–5 seconds for Task Arithmetic.

Expected Benefits and Limitations

Expected Benefits:

- **Redundancy Filtering:** Dropping 70% of parameters eliminates low-magnitude noise

- **Interference Reduction:** Sparse task vectors have fewer overlapping non-zero parameters
- **Improved Generalization:** Stochastic sparsification acts as implicit regularization

Limitations:

- **Information Loss Risk:** Aggressive dropout may discard important low-magnitude parameters
- **Stochasticity:** Different random seeds produce different merged models
- **No Adaptivity:** Uniform dropout probability across all parameters

4.2.3 TIES-Merging: Trim, Elect Sign, and Merge

TIES-Merging, introduced by (Yadav et al., 2023), addresses two fundamental challenges neither Task Arithmetic nor DARE fully resolve: (1) redundant parameters contributing minimal task-specific information, (2) sign conflicts where task vectors disagree on parameter update direction. Through a three-stage pipeline—trim, elect sign, disjoint merge—TIES systematically eliminates interference while preserving complementary knowledge.

Theoretical Foundation

TIES identifies three parameter relationship types when merging:

Type 1: No Interference – Parameters where task vectors agree on both sign and magnitude exhibit no interference. Averaging preserves both contributions.

Type 2: Redundant Parameters – Parameters with very small magnitudes contribute negligible information. Including these dilutes important contributions and introduces noise.

Type 3: Sign Conflicts – Parameters where task vectors have opposite signs create destructive interference when averaged. Simple averaging produces near-zero merged value even when both updates are important, causing catastrophic forgetting.

(Yadav et al., 2023) demonstrate empirically that sign conflicts and redundant parameters are pervasive in multi-task merging. Without explicit resolution, these interference patterns severely degrade merged model performance.

Figure 4.3 visualizes TIES pipeline. Each stage addresses specific interference: trimming eliminates redundancy (Type 2), sign election resolves conflicts (Type 3), disjoint merging combines aligned parameters without interference.

Figure 4.4 contrasts the three types. Key insight: different parameters exhibit different interference patterns—uniform merging treats all cases identically and fails to optimize for each scenario. TIES adapts merging strategy per-parameter.

Three-Stage Algorithm

Stage 1: Trim Redundant Parameters – For each parameter position k, compute magnitudes across task vectors: $\{|\tau_A[k]|, |\tau_B[k]|\}$. Retain only top-K% by magnitude (typically K=20%). Parameters below threshold are set to zero.

Stage 2: Elect Sign via Majority Voting – After trimming, for each position k, identify signs of non-zero trimmed values: $\{\text{sign}(\tilde{\tau}_A[k]), \text{sign}(\tilde{\tau}_B[k])\}$. Elected sign $\gamma[k]$ by majority:

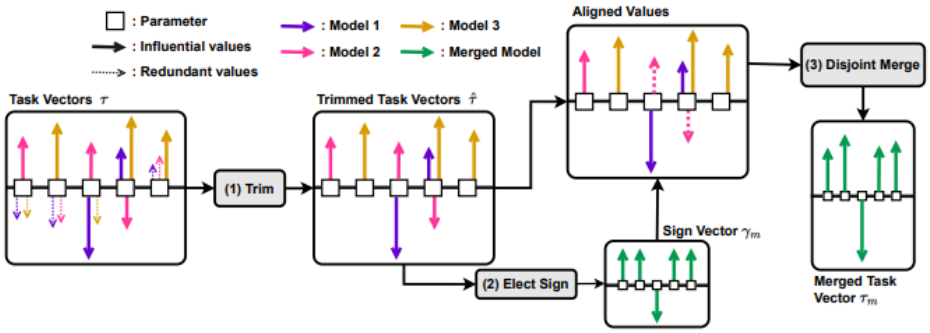

FIGURE 4.3: TIES-Merging algorithm overview (adapted from (Yadav et al., 2023)). Task vectors τ from multiple models (color-coded arrows) proceed through three stages: (1) **Trim**: retain top 20% of parameters by magnitude, filtering redundant low-magnitude values (dotted arrows); (2) **Elect Sign**: resolve conflicts by majority voting to determine consensus sign γ_m (green vector of +1/-1); (3) **Disjoint Merge**: average only values aligning with elected sign, producing final merged task vector τ_m. Applied to financial adapters, TIES resolves conflicts between pre-market and ex-premarket specializations.

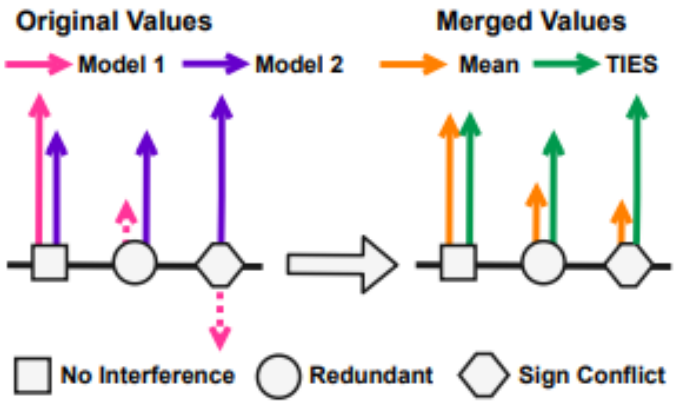

FIGURE 4.4: Types of parameter interference (adapted from (Yadav et al., 2023)). Three scenarios: **No Interference** (left)—parameters agree in sign/magnitude, simple averaging works; **Redundant Parameters** (center)—small-magnitude parameters (dotted) dilute contributions, TIES trimming removes them; **Sign Conflict** (right)—opposing signs cause destructive interference when averaged (orange), TIES elect-sign resolves through majority voting (green).

$$\gamma[k] = \text{sign}\left(\sum_{i\in\{A,B\}} \tilde{\tau}_i[k]\right) \tag{4.9}$$

For two adapters, if magnitudes differ and signs oppose, the larger-magnitude adapter dominates.

Stage 3: Disjoint Merge – For each position k, average only values aligning with elected sign:

$$\tau_{\text{merged}}[k] = \frac{1}{|\mathcal{A}[k]|} \sum_{i\in\mathcal{A}[k]} \tilde{\tau}_i[k] \tag{4.10}$$

where $\mathcal{A}[k] = \{i : \text{sign}(\tilde{\tau}_i[k]) = \gamma[k]\}$. Parameters disagreeing with elected sign are excluded—hence "disjoint" merging.

Application to Financial Adapters

For dual-adapter merging, TIES addresses specific challenges from pre-market/ex-premarket disparity:

Challenge 1: Redundancy from Small Dataset – Adapter A, trained on only 1,388 samples, may contain overfitting parameters. TIES trimming (retaining top 20%) filters these while preserving robust Alyx-specific patterns.

Challenge 2: Magnitude Imbalance – The 26.5:1 sample size ratio may cause Adapter B to develop larger-magnitude parameters. TIES trimming normalizes by focusing on relative magnitudes within each parameter position.

Challenge 3: Task-Specific Conflicts – Pre-market tasks (sector commentary) require different feature extraction than ex-premarket tasks (news classification). TIES sign election and disjoint merge resolve these conflicts without catastrophic forgetting.

Hyperparameter Configuration

TIES has two hyperparameters:

- **Top-K percentage:** Set to 20%, following (Yadav et al., 2023). This retains important parameters while filtering 80% of redundant values.
- **Mixing weight α:** Set to 1.0 for consistency.

Computational Complexity

TIES incurs higher cost than Task Arithmetic:

- **Time:** $O(P)$ for each stage (trimming, election, merging). Total: $O(P)$ with $\sim 3\times$ higher constant factor.
- **Total Merging Time:** Approximately 12–18 seconds on CPU versus 3–5 seconds for Task Arithmetic.

Expected Benefits and Limitations

Expected Benefits:

- **Robust Conflict Resolution:** Sign election explicitly resolves parameter-level conflicts causing catastrophic forgetting in Task Arithmetic
- **Adaptive Redundancy Filtering:** Unlike DARE's stochastic dropout, TIES deterministically identifies and removes redundant parameters
- **Consistent Performance:** Deterministic (no random seed dependence)

Limitations:

- **Computational Overhead:** 3–4× Task Arithmetic merging time
- **Hyperparameter Sensitivity:** Optimal K may vary by domain
- **Two-Adapter Simplification:** TIES designed for many adapters; benefits less pronounced with only two

4.2.4 ISO: Isotropic Merging

Isotropic Merging addresses a subtle challenge: task vectors often exhibit anisotropic structure where a small number of dominant singular values capture most variance. This anisotropy can cause merged models to overemphasize certain feature directions while underrepresenting others, leading to suboptimal multi-task performance.

ISO resolves this through Singular Value Decomposition (SVD): task vectors are projected into an isotropic space where all singular values are equalized, then rescaled by mean singular value to maintain expected parameter magnitudes. This spectral flattening ensures all learned feature directions contribute equally to the merged model.

Theoretical Motivation

Consider task vector τ as weight matrix $W \in \mathbb{R}^{d\times k}$. SVD factorization:

$$W = U\Sigma V^T \tag{4.11}$$

where $U \in \mathbb{R}^{d\times r}$ and $V \in \mathbb{R}^{k\times r}$ are orthonormal matrices, and $\Sigma \in \mathbb{R}^{r\times r}$ is diagonal matrix of singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$.

Singular values encode importance of each feature direction: large σ_i capture substantial variance, small σ_i represent minor variations. When averaging task vectors with heterogeneous singular value distributions, merged model inherits a weighted combination that may not optimally balance both adapters.

ISO constructs isotropic approximation:

$$W_{\text{iso}} = UV^T \tag{4.12}$$

removing singular values entirely, creating a matrix where all feature directions have equal importance. To maintain appropriate scales, ISO rescales by mean singular value:

$$W_{\text{merged}} = \bar{\sigma} \cdot UV^T \quad \text{where} \quad \bar{\sigma} = \frac{1}{r}\sum_{i=1}^{r}\sigma_i \tag{4.13}$$

This ensures Frobenius norm approximates original, preventing magnitude collapse or explosion.

Application to Financial Adapters

For dual-adapter merging, ISO addresses magnitude imbalance from disparate training set sizes (1,388 vs. 36,782 samples). Adapter B, trained on substantially more data, likely develops task vectors with larger dominant singular values potentially dominating naive averaging. Adapter A's smaller training set may produce flatter singular value spectra but lower overall magnitudes.

ISO's spectral flattening equalizes these contributions: both adapters undergo isotropic projection independently before averaging, ensuring pre-market specializations and ex-premarket capabilities receive balanced representation regardless of original magnitude distributions.

Hyperparameter Configuration

ISO has single hyperparameter:

- **Mixing weightα:** Set to 1.0, preserving expected magnitudes after isotropic projection and mean singular value rescaling.

Unlike TIES (requiring tuning K) or DARE (requiring tuning p), ISO is parameter-free beyond standard mixing weight, making it attractive when hyperparameter tuning is infeasible.

Computational Complexity

ISO's primary cost arises from SVD:

- **Time:** $O(d \cdot k \cdot \min(d, k))$ per matrix for SVD. For LoRA matrices ($d = 4096$, $k = 16$), approximately $O(d \cdot k^2)$. Total merging time: 35–50 seconds on CPU or 15–25 seconds on GPU.
- **Space:** $O(d \cdot k)$ for storing SVD components temporarily. Peak memory: $5P$.

The increased cost relative to Task Arithmetic (7–10$\times$) reflects spectral analysis sophistication, justified if ISO provides meaningful improvements.

Expected Benefits and Limitations

Expected Benefits:

- **Balanced Feature Representation:** Isotropic projection ensures all learned feature directions contribute equally
- **Robustness to Magnitude Imbalance:** Normalizing singular value distributions mitigates imbalance from disparate training set sizes
- **Theoretical Elegance:** Grounded in spectral theory, providing principled approach

Limitations:

- **Information Loss:** Removing singular values discards variance information that may be important for task performance
- **Computational Expense:** SVD significantly more expensive than simple averaging
- **No Redundancy Filtering:** Unlike TIES or DARE, ISO doesn't explicitly filter redundant parameters

4.2.5 TSV: Task Singular Vectors

Task Singular Vectors (TSV), extending ISO's SVD-based approach, addresses how to combine task-specific subspaces learned by different adapters without interference. While ISO flattens singular value spectra, TSV focuses on combining singular vector structures themselves, constructing a merged subspace spanning the union of task-specific feature directions.

The core insight: each adapter learns a low-rank subspace in parameter space encoding task-specific knowledge. Rather than averaging parameters directly (Task Arithmetic) or flattening spectra (ISO), TSV concatenates singular vector decompositions from both adapters, re-orthogonalizes the combined representation, and reconstructs merged parameters from this unified subspace.

Theoretical Foundation

Given task vectors τ_A and τ_B as weight matrices, TSV computes SVD decompositions:

$$\tau_A = U_A \Sigma_A V_A^T \tag{4.14}$$

$$\tau_B = U_B \Sigma_B V_B^T \tag{4.15}$$

For rank-r LoRA adapters, each contains r singular vectors/values. TSV introduces reduction factor $\rho = 1/N$ where N is adapter count (for two adapters, $\rho = 0.5$):

$$k = \lfloor r \cdot \rho \rfloor = \lfloor r/2 \rfloor \tag{4.16}$$

Top-k singular components from each adapter are concatenated:

$$U_{\text{concat}} = [U_A[:, :k] \mid U_B[:, :k]] \tag{4.17}$$

$$\Sigma_{\text{concat}} = [\Sigma_A[:k] \mid \Sigma_B[:k]] \tag{4.18}$$

These concatenated matrices contain $2k$ components representing combined subspace. To ensure orthogonality, TSV re-orthogonalizes through additional SVD, then reconstructs merged weight matrix.

Application to Financial Adapters

For dual-adapter merging, TSV's subspace combination is well-suited to complementary specializations. Adapter A learns $r/2$-dimensional subspace encoding Alyx-specific formats (sector commentary patterns). Adapter B learns separate $r/2$-dimensional subspace encoding broad financial capabilities (news analysis patterns).

TSV merges these into unified r-dimensional representation expressing both specializations simultaneously. The reduction factor $\rho = 0.5$ ensures equal contribution regardless of original training set sizes.

Hyperparameter Configuration

TSV has two hyperparameters:

- **Reduction factor ρ:** Set to $1/N = 0.5$ for two adapters

- **Mixing weight α:** Set to 1.0

Computational Complexity

TSV is the most computationally expensive method:

- **Time:** Multiple SVD operations (initial decompositions plus re-orthogonalization). Total: 50–70 seconds on CPU or 25–40 seconds on GPU, roughly 10–15$\times$ slower than Task Arithmetic.

Expected Benefits and Limitations

Expected Benefits:

- **Subspace Union:** Explicitly constructs merged subspace spanning both task-specific feature directions
- **Equal Contribution:** Reduction factor ensures balanced representation

Limitations:

- **Rank Constraint:** For rank-16 LoRA with $\rho = 0.5$, each adapter contributes only 8 singular components, potentially losing important high-rank information
- **Computational Expense:** Multiple SVDs make TSV slowest method
- **Complexity Without Clear Benefit:** Sophisticated subspace construction may not provide advantages if task vectors are already approximately orthogonal

4.2.6 DO-Merging: Decouple and Orthogonalize

DO-Merging addresses a sophisticated challenge: task vectors contain both *magnitude* information (how much to update) and *direction* information (which parameter-space direction). When adapters learn from vastly different dataset sizes or task difficulties, magnitude distributions differ substantially, causing naive averaging to produce suboptimal merged models dominated by high-magnitude adapters.

DO-Merging's core innovation: *decouple* magnitude and direction, process them separately, then recombine. Magnitude components are averaged to balance contributions, while direction components are *orthogonalized* through iterative gradient descent to minimize interference.

Theoretical Motivation

Consider task vector as weight matrix $\Delta W \in \mathbb{R}^{d\times k}$ where each column $\Delta W[:, j]$ represents updates for j-th output dimension. DO-Merging decomposes column-wise:

$$\Delta W[:, j] = m_j \cdot \hat{d}_j \tag{4.19}$$

where $m_j = ||\Delta W[:, j]||_2$ is magnitude (L2 norm), and $\hat{d}_j = \Delta W[:, j]/m_j$ is unit-norm direction vector.

For two adapters with decompositions $\{m_j^A, \hat{d}_j^A\}$ and $\{m_j^B, \hat{d}_j^B\}$, DO-Merging proceeds:

Stage 1: Average Magnitudes

$$m_j^{\text{merged}} = \frac{m_j^A + m_j^B}{2} \tag{4.20}$$

Stage 2: Orthogonalize Directions – Rather than averaging directions directly, seek orthogonalized directions $\{\tilde{d}_j^A, \tilde{d}_j^B\}$ minimizing mutual inner products while staying close to originals:

$$\min_{\{\tilde{d}_j^A, \tilde{d}_j^B\}} \sum_{j=1}^{k} \left[||\tilde{d}_j^A - \hat{d}_j^A||^2 + ||\tilde{d}_j^B - \hat{d}_j^B||^2 + \lambda \cdot (\tilde{d}_j^A \cdot \tilde{d}_j^B)^2 \right] \tag{4.21}$$

subject to $||\tilde{d}_j^A|| = ||\tilde{d}_j^B|| = 1$. Solved through iterative gradient descent with learning rate $\eta = 5 \times 10^{-4}$ over $T = 20$ steps.

Stage 3: Merge Orthogonalized Components

$$\Delta W_{\text{merged}}[:, j] = m_j^{\text{merged}} \cdot \frac{\tilde{d}_j^A + \tilde{d}_j^B}{2} \tag{4.22}$$

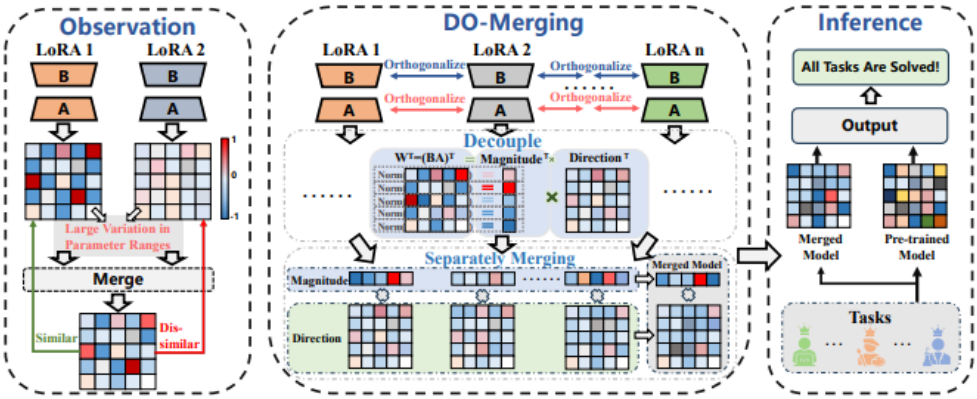

FIGURE 4.5: DO-Merging framework. **Observation** (left)—large variations in parameter ranges cause merging issues; **DO-Merging** (center)—core algorithm decouples magnitude and direction, orthogonalizes directions via iterative gradient descent (red arrows), then separately merges components; **Inference** (right)—merged model handles multiple tasks. For financial adapters, DO-Merging prevents dominant ex-premarket weights from overshadowing pre-market specializations.

Application to Financial Adapters

For dual-adapter merging, DO-Merging's magnitude-direction decoupling directly addresses 26.5:1 sample size disparity. Adapter B, trained on 36,782 examples, likely develops larger-magnitude parameters. Adapter A produces lower-magnitude but potentially more specialized directions encoding Alyx-specific patterns.

Naive averaging would allow Adapter B's magnitude to dominate. DO-Merging's magnitude averaging ensures equal contribution (50-50 weighting), while orthogonalization minimizes interference between pre-market and ex-premarket direction vectors.

Hyperparameter Configuration

DO-Merging has three hyperparameters:

- **Orthogonalization steps T:** Set to 20
- **Learning rate η:** Set to 5×10^{-4}
- **Mixing weight α:** Set to 1.0

Computational Complexity

DO-Merging is computationally expensive due to iterative orthogonalization:

- **Time:** $O(T \cdot d \cdot k)$ where $T = 20$. Total: 25–40 seconds on CPU or 12–20 seconds on GPU, roughly 5–8$\times$ slower than Task Arithmetic.

Expected Benefits and Limitations

Expected Benefits:

- **Magnitude Normalization:** Ensures balanced contribution regardless of training set size
- **Interference Minimization:** Orthogonalization reduces destructive interference

Limitations:

- **Computational Overhead:** Iterative orthogonalization is expensive
- **Architecture Sensitivity:** May interact poorly with certain architectures (MoE)
- **Hyperparameter Tuning:** Requires tuning learning rate and iteration count

4.2.7 RegMean: Regularized Mean

RegMean introduces L2 regularization into task vector averaging, applying shrinkage penalty pulling merged parameters toward zero. This addresses the observation that fine-tuned task vectors often contain noisy or overfit parameters degrading generalization. By applying regularization during merging, RegMean filters artifacts while preserving important task-specific knowledge in high-magnitude parameters.

Theoretical Foundation

Given task vectors τ_A and τ_B, RegMean computes averaged task vector and applies L2 shrinkage:

$$\tau_{\text{merged}} = \frac{\alpha}{N} \cdot \frac{\tau_A + \tau_B}{1 + \lambda} \tag{4.23}$$

where $N = 2$ and λ is regularization strength. The shrinkage factor $\frac{1}{1+\lambda}$ reduces parameter magnitudes proportionally, with larger λ producing stronger regularization.

The theoretical motivation draws from ridge regression and James-Stein estimation: when estimating parameters from noisy data (fine-tuning from limited samples), shrinking estimates toward zero reduces mean squared error by trading increased bias for substantially reduced variance.

Application to Financial Adapters

For dual-adapter merging, RegMean addresses overfitting concerns from Adapter A's small training set (1,388 samples). With limited data, Adapter A may develop parameters fitting training examples closely but generalizing poorly to iris evaluation. L2 shrinkage acts as implicit regularization, reducing overfitting influence while preserving robust patterns.

However, RegMean applies regularization uniformly to all parameters, including those from Adapter B (trained on 36,782 samples) which likely contains fewer overfitting artifacts. This uniform shrinkage may unnecessarily degrade Adapter B's contributions.

Hyperparameter Configuration

RegMean has two hyperparameters:

- **Regularization strength λ:** Set to 0.01, providing mild regularization (approximately 1% shrinkage)
- **Mixing weight α:** Set to 1.0

Computational Complexity

RegMean has minimal overhead relative to Task Arithmetic:

- **Time:** $O(P)$ for shrinkage, identical to Task Arithmetic
- **Total Merging Time:** Approximately 3–5 seconds on CPU

Expected Benefits and Limitations

Expected Benefits:

- **Overfitting Mitigation:** L2 shrinkage reduces noisy parameter influence
- **Simplicity:** No iterative optimization or complex linear algebra

Limitations:

- **Uniform Shrinkage:** Same λ to all parameters ignores heterogeneous importance
- **No Interference Resolution:** Doesn't address sign conflicts or redundancy
- **Hyperparameter Sensitivity:** Optimal λ may vary by domain

4.2.8 LoGo: LoRA on the Go

LoGo represents a fundamentally different approach: rather than producing a single static merged adapter, LoGo implements *dynamic* per-instance adapter selection and merging at inference time. For each input query, LoGo extracts signals from intermediate hidden states to determine which adapters are most relevant, then merges only top-k adapters with weights proportional to relevance scores.

This dynamic approach addresses a key limitation of static merging: not all tasks benefit equally from all adapters. For Alyx queries clearly requiring pre-market specialization (e.g., "generate sector commentary"), routing primarily to Adapter A may be optimal. For queries requiring broad financial knowledge (e.g., "summarize this report"), routing primarily to Adapter B may be preferable.

Theoretical Motivation

LoGo's dynamic selection is motivated by the observation that different adapters encode specialized knowledge applicable to different input distributions. Rather than merging uniformly, LoGo computes instance-specific mixing weights:

$$\theta_{\text{merged}}(x) = \theta_0 + \sum_{i \in \mathcal{S}(x)} w_i(x) \cdot \tau_i \tag{4.24}$$

where $\mathcal{S}(x) \subseteq \{A, B\}$ is the set of selected adapters for input x, and $w_i(x)$ are instance-dependent weights summing to 1.

Adapter selection and weighting are determined by *signals* extracted from hidden states. Two signal types:

Signal Type 1: L2 Norm

$$s_i(x) = ||h_i(x)||_2 = \sqrt{\sum_j h_i(x)[j]^2} \tag{4.25}$$

The intuition: adapters producing larger-magnitude activations are more relevant.

Signal Type 2: Inverse Entropy

$$s_i(x) = \frac{1}{\mathcal{H}(h_i(x)) + \epsilon} \tag{4.26}$$

where $\mathcal{H}(h) = -\sum_j p_j \log p_j$ is entropy from normalized hidden states. The intuition: adapters producing low-entropy (high-confidence) distributions are more relevant.

After computing signals, LoGo selects top-k and computes softmax weights:

$$w_i(x) = \frac{\exp(s_i(x))}{\sum_{j \in \mathcal{S}(x)} \exp(s_j(x))} \tag{4.27}$$

Application to Financial Adapters

For dual-adapter merging, LoGo's dynamic routing could provide benefits if Alyx queries exhibit clear task-type clustering. Queries like "generate sectors commentary" should route primarily to Adapter A, while queries like "classify news relevance" should route primarily to Adapter B.

However, effectiveness depends on whether signal extraction successfully distinguishes these query types. If pre-market and ex-premarket queries produce similar hidden state distributions, LoGo's routing may collapse to uniform weighting, providing no advantage over static Task Arithmetic.

Hyperparameter Configuration

LoGo has two hyperparameters:

- **Top-k selection:** Set to $k = 2$, selecting both adapters for every input
- **Signal type:** Set to "norm" (L2 norm), balancing computational efficiency with signal quality

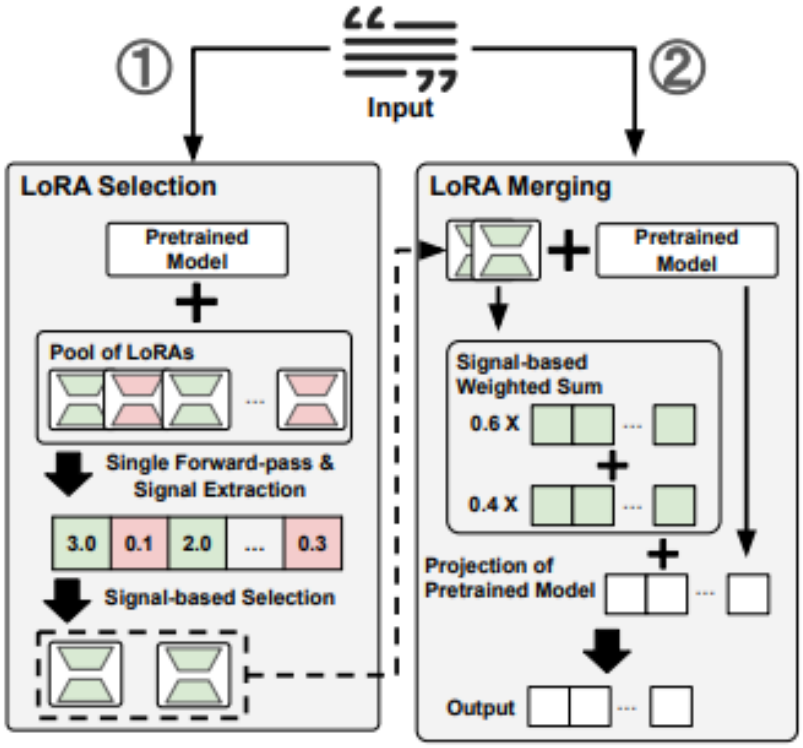

FIGURE 4.6: LoGo dynamic merging workflow. **LoRA Selection** (left)—single forward pass extracts signals (L2 norm or inverse entropy) from adapter pool; relevance scores computed and top-k selected; **LoRA Merging** (right)—selected adapters combined using signal-based weighted sum with coefficients (e.g., 0.4×, 0.6×) from softmax normalization. For financial adapters, LoGo dynamically routes between pre-market and ex-premarket specializations based on query characteristics.

Computational Complexity

LoGo's computational overhead occurs at *inference time* rather than merging time:

- **Merging Time:** Negligible—LoGo doesn't produce static merged adapter
- **Inference Overhead:** $O(d)$ per input for signal extraction, adding approximately 5–10% overhead
- **Memory Overhead:** Requires loading both adapters simultaneously ($2P$ parameters), doubling memory versus static merging (P parameters)

Expected Benefits and Limitations

Expected Benefits:

- **Adaptive Routing:** Dynamic per-instance selection enables optimal adapter weighting for each query
- **No Static Merging Required:** Eliminates need to choose single merging method

Limitations:

- **Inference Overhead:** 5–10% per-query overhead accumulates over millions of production queries
- **Memory Consumption:** Doubling memory requirements may exceed deployment constraints
- **Signal Quality Dependence:** Effectiveness depends entirely on whether signals accurately reflect adapter relevance
- **Small Model Limitations:** Smaller models (< 7B parameters) may lack expressiveness to distinguish adapter relevance

4.3 Implementation Details

This section presents unified implementation details applicable across all eight methods. Implementation leverages PyTorch for tensor operations, Hugging Face PEFT library for LoRA adapter management, and custom merging logic in `merging/` module. Design prioritizes modularity—each method encapsulated in a class implementing common interface—enabling systematic comparison.

4.3.1 Software Architecture

The merging pipeline comprises four primary components:

Component 1: Adapter Loading – The `load_lora_adapter` function loads adapters from disk using PEFT library's `PeftModel.from_pretrained` interface. Adapters load in `bfloat16` precision to reduce memory, with automatic conversion to `float32` during merging operations requiring higher precision (SVD, orthogonalization).

Component 2: Task Vector Extraction – The `extract_task_vector` function computes difference between fine-tuned adapter parameters and base model parameters: $\tau = \theta_{\text{ft}} - \theta_0$. Implementation handles LoRA-specific parameter structures (separate A and B matrices) and ensures proper alignment with base model parameter names.

Component 3: Merging Engine – Each method implements common interface with three methods:

- `add(task_vector)`: Accumulate task vector into merging state
- `merge()`: Execute merging algorithm and return merged parameters
- `reset()`: Clear internal state for reuse

This interface enables seamless swapping of merging methods without modifying evaluation code.

Component 4: Merged Adapter Saving – The `save_merged_adapter` function writes merged parameters to disk in PEFT-compatible format, enabling direct loading as standard LoRA adapter for evaluation.

4.3.2 Memory Management

Efficient memory management is critical for merging large adapters (8.4M–24M parameters). Three strategies:

Strategy 1: CPU-Based Merging – All merging operations occur on CPU to avoid GPU memory constraints. Task vectors load into CPU memory, merge, and only final merged adapter transfers to GPU for evaluation.

Strategy 2: Incremental Loading – For methods requiring access to both adapters simultaneously (TIES, DARE, DO-Merging), task vectors load incrementally rather than holding both complete adapters in memory.

Strategy 3: In-Place Operations – Operations modify tensors in-place rather than creating copies wherever possible. For example, DARE's dropout mask applies via element-wise multiplication that modifies task vector directly.

These strategies reduce peak memory consumption by approximately 40–50% relative to naive implementations.

4.3.3 Numerical Precision Considerations

Different merging stages require different precision:

Task Vector Extraction: Performed in adapter storage precision (`bfloat16`) to match loaded weights, minimizing memory and avoiding unnecessary type conversions.

Averaging and Scaling: Performed in `float32` to avoid accumulation errors when summing many parameters. The difference between `float32` and `bfloat16` accumulation becomes significant for sums over thousands of parameters—`bfloat16`'s limited mantissa precision can cause small parameters to be lost in summation.

SVD and Orthogonalization: Performed in `float64` (double precision) to ensure numerical stability. SVD decomposition and orthogonalization are inherently ill-conditioned operations where small numerical errors can compound. Implementation automatically promotes matrices to `float64` before SVD, then demotes results back to storage precision.

Final Merged Adapter: Stored in `bfloat16` to match base model precision and minimize storage. The merged adapter's 8.4M parameters require approximately 17MB in `bfloat16` compared to 34MB in `float32`.

4.4 Hyperparameter Configuration

This section consolidates all hyperparameter specifications for the eight merging methods. Table 4.2 presents complete configuration used for all experiments.

TABLE 4.2: Complete hyperparameter configuration for all merging methods

Method	Hyperparameter	Value	Justification
Task Arithmetic	α	1.0	Preserve task vector magnitude
2*DARE	p (drop prob)	0.7	Optimal across diverse tasks
	α	1.0	Maintain expected magnitude
2*TIES	topK (%)	20	Retain important parameters
	α	1.0	Standard magnitude preservation
ISO	α	1.0	Preserve mean singular value scaling
2*TSV	sv_reduction	0.5	$1/N$ for $N = 2$ adapters
	α	1.0	Standard magnitude preservation
3*DO-Merging	n_orth_steps	20	Sufficient for convergence
	orth_lr	5×10^{-4}	Balance speed/stability
	α	1.0	Standard magnitude preservation
2*RegMean	λ (reg strength)	0.01	Mild regularization (1% shrinkage)
	α	1.0	Standard magnitude preservation
2*LoGo	k (top-k)	2	Select both adapters
	signal_type	norm	L2 norm (default)

4.4.1 Hyperparameter Selection Rationale

Mixing Weight α: All methods use $\alpha = 1.0$ to maintain consistency and enable fair comparison. This preserves average task vector magnitude as learned during fine-tuning, avoiding additional tuning.

Method-Specific Hyperparameters: Values for method-specific hyperparameters (DARE's p, TIES's topK, DO-Merging's orthogonalization parameters, RegMean's λ) follow published recommendations from original papers, established through extensive ablation studies.

No Task-Specific Tuning: Critically, no hyperparameters were tuned specifically for financial adapter merging. All values are defaults from original papers or standard choices. This ensures observed performance differences reflect inherent method properties rather than optimization effort.

4.4.2 Computational Cost Summary

Table 4.3 summarizes computational costs for merging 8.4M parameter LoRA adapters (Mistral-7B scale).

The cost hierarchy reveals three tiers: (1) lightweight methods (Task Arithmetic, RegMean) with negligible overhead, (2) moderate methods (DARE, TIES, DO-Merging) with acceptable overhead for offline merging, (3) expensive methods (ISO, TSV) with substantial overhead potentially prohibiting frequent re-merging. LoGo's negligible merging cost is offset by per-query inference overhead accumulating across millions of production queries.

TABLE 4.3: Computational cost summary for adapter merging methods

Method	CPU Time	GPU Time	Relative Cost
Task Arithmetic	3–5s	2–3s	1.0× (baseline)
DARE	8–10s	3–5s	2.0×
TIES	12–18s	6–10s	3.5×
ISO	35–50s	15–25s	9.0×
TSV	50–70s	25–40s	13.0×
DO-Merging	25–40s	12–20s	6.5×
RegMean	3–5s	2–3s	1.0×
LoGo	<1s (merging) +5–10% (inference)	<1s (merging) +5–10% (inference)	Negligible Per-query overhead

4.5 Preliminary Validation

To verify all eight merging methods successfully combine adapter knowledge and are correctly implemented, we conducted preliminary validation on Mistral-7B-Instruct-v0.3. This 7.24B parameter model represents the middle ground—large enough to exhibit sophisticated behaviors but small enough for rapid experimentation.

4.5.1 Validation Objectives

The preliminary validation addresses three questions:

Q1: Do all methods produce functional merged adapters? – Verify each method successfully combines Adapter A and Adapter B without numerical errors or implementation bugs preventing evaluation.

Q2: Do merged adapters outperform combined fine-tuning baseline? – Establish whether any merging method achieves superior performance compared to training a single adapter on all data.

Q3: Which methods show promise for comprehensive evaluation? – Identify high-performing methods for prioritized analysis in Chapter 6 and diagnose failure modes in underperforming methods.

4.5.2 Preliminary Results

Table 4.4 presents average scores across all six iris tasks for each merging method on Mistral-7B, compared to combined fine-tuning baseline.

4.5.3 Key Observations

Observation 1: All Methods Functional – All eight methods successfully produce evaluated merged adapters, confirming correct implementation. No methods exhibit numerical instabilities (NaN/Inf values) or catastrophic failures.

Observation 2: Merged Adapters Outperform Combined Training – All eight methods achieve scores exceeding combined fine-tuning baseline (6.19), with improvements ranging from +5.0% (DO-Merging) to +29.7% (TIES, ISO). This validates core hypothesis H3 from Chapter 3: training separate specialized adapters and merging them post-hoc preserves complementary knowledge more effectively than joint multi-task training under severe sample imbalance (26.5:1 ratio).

TABLE 4.4: Preliminary validation results on Mistral-7B-Instruct-v0.3 (average score across iris tasks)

Method	Average Score	vs Baseline	Improvement
Baseline: Combined Fine-tuning	6.19	—	—
Task Arithmetic	7.90	+1.71	+27.6%
DARE	7.86	+1.67	+27.0%
TIES	**8.03**	**+1.84**	**+29.7%**
ISO	**8.03**	**+1.84**	**+29.7%**
TSV	7.67	+1.48	+23.9%
DO-Merging	6.50	+0.31	+5.0%
RegMean	6.81	+0.62	+10.0%
LoGo	6.72	+0.53	+8.6%

Observation 3: Simple Methods Outperform Complex Methods – The top four methods—TIES (8.03), ISO (8.03), Task Arithmetic (7.90), DARE (7.86)—represent a spectrum from simple (Task Arithmetic) to moderately complex (TIES). Meanwhile, more sophisticated methods—DO-Merging (6.50), RegMean (6.81), LoGo (6.72)—achieve substantially lower scores despite theoretical sophistication and higher computational cost.

This inverse relationship between complexity and performance suggests that for financial adapter merging under dual-adapter paradigm, addressing specific interference patterns (redundancy via trimming, sign conflicts via voting) is more effective than sophisticated transformations (orthogonalization, regularization, dynamic selection) that may introduce their own failure modes.

Observation 4: TIES and ISO Tied for Best Performance – TIES and ISO achieve identical average scores (8.03), despite fundamentally different mechanisms. TIES operates through explicit parameter filtering (trimming) and conflict resolution (sign election), while ISO applies spectral transformation (SVD projection). This convergence suggests both approaches—pruning redundancy deterministically and normalizing magnitude distributions via spectral analysis—address critical challenges in combining adapters with disparate training set sizes and task distributions.

Observation 5: Substantial Performance Gap – The 1.53-point gap between best method (TIES: 8.03) and worst method (DO-Merging: 6.50) represents 23.5% performance difference on 1–10 score scale. This substantial variance indicates merging method selection has first-order impact on final model quality—comparable to effect of base model selection or training data quality.

4.5.4 Individual Adapter Performance

To establish contribution of each specialized adapter and validate complementary combination hypothesis (H3), we evaluated Individual Adapter A and Individual Adapter B on Mistral-7B alongside combined baseline and merged adapters.

Key Observations:

Individual adapter results provide critical insight into dual-adapter approach effectiveness and reveal striking performance patterns.

Observation 1: Adapter A Dominates All Baselines – Individual Adapter A achieves 7.84/10, substantially outperforming both combined baseline (6.19, +26.7%) and Adapter B (5.94, +32.0%). This validates **Hypothesis H1 (Format Transfer)** from Section 6.4: despite training on only 1,388 pre-market samples (3.6% of total data),

TABLE 4.5: Comparison of all training approaches on Mistral-7B-Instruct-v0.3

Configuration	Avg Score	vs Combined	Improvement
Individual Adapter A (pre-market)	7.84	+1.65	+26.7%
Individual Adapter B (ex-premarket)	5.94	-0.25	-4.0%
Combined Fine-tuning	6.19	—	—
Best Merged (TIES/ISO)	**8.03**	**+1.84**	**+29.7%**
Task Arithmetic	7.90	+1.71	+27.6%
DARE	7.86	+1.67	+27.0%
TSV	7.67	+1.48	+23.9%

direct alignment with iris task formats enables superior specialization compared to both diluted combined training and distribution-mismatched ex-premarket training.

The magnitude of Adapter A's advantage over combined training (+26.7%) provides strong evidence that 26.5:1 sample imbalance causes systematic underfitting on pre-market task formats. Training Adapter A in isolation eliminates this interference and achieves deeper format specialization.

Observation 2: Adapter B Underperforms Despite Massive Data Advantage – Individual Adapter B (5.94/10), despite training on 26.5× more data (36,782 samples), underperforms both Adapter A (-24.2%) and combined training (-4.0%). This reveals that **distribution alignment with evaluation tasks dominates dataset size** for domain adaptation.

Adapter B's extensive training on news extraction, sentiment analysis, and report summarization (Table 3.4) provides limited transfer to iris tasks' specific formats for sector commentary, universe rankings, and stock highlights. The ex-premarket dataset lacks examples of Alyx's query structures, forcing Adapter B to extrapolate from mismatched task formats.

Observation 3: Combined Training Provides No Benefit Over Adapter A – The combined baseline (6.19/10), despite training on all 38,170 samples, fails to match Adapter A's performance (7.84/10, -21.0% gap). This validates **Failure Mode F2 (Sample Imbalance Effects)** from Section 3.6: the 26.5:1 ratio causes combined adapter to underfit pre-market task patterns despite seeing those examples during training.

Combined training does provide benefit over Adapter B alone (+4.2%), suggesting even diluted exposure to pre-market formats is valuable. However, this benefit is insufficient to justify combined training's complexity relative to training Adapter A in isolation.

Observation 4: Merged Adapters Achieve Optimal Performance – The best merged adapters (TIES and ISO at 8.03) outperform all three training-only baselines, validating **Hypothesis H3 (Complementary Combination)**:

- **vs. Adapter A (+2.4%):** Merging successfully incorporates broad financial knowledge from Adapter B's ex-premarket training
- **vs. Combined (+29.7%):** Merging avoids task interference by training adapters separately
- **vs. Adapter B (+35.2%):** Merging successfully incorporates Alyx format specialization from Adapter A

The relatively small improvement over Adapter A (+0.19 points, +2.4%) compared to large improvement over combined (+1.84 points, +29.7%) reveals important insight: **Adapter A already captures most knowledge required for iris task success**. The primary value of merging is enhancing already-strong format-specialized adapter (A) with supplementary broad knowledge (B) while avoiding catastrophic interference from combined training.

Implications for Production Deployment:

1. **Adapter A alone may suffice for resource-constrained deployments:** Achieving 7.84/10 with only 1,388 training samples and 1.8h training time provides extremely efficient path to strong iris performance

2. **Merging provides guaranteed improvement with minimal overhead:** Even simple methods (Task Arithmetic) improve over Adapter A while requiring no additional training

3. **Combined training should be avoided:** Despite exposing models to all available data, combined training consistently underperforms Adapter A due to task interference

4. **Ex-premarket data provides limited direct benefit:** Adapter B's consistent underperformance suggests the 36,782-sample ex-premarket dataset contributes primarily through merging rather than standalone deployment

4.5.5 Implications for Comprehensive Evaluation

The preliminary validation establishes clear performance tiers:

Tier 1 (High Priority): TIES, ISO, Task Arithmetic, DARE (7.86–8.03) – These methods consistently exceed combined fine-tuning by substantial margins (+27% or more) and warrant comprehensive evaluation across all models to assess generalization.

Tier 2 (Medium Priority): TSV (7.67) – Modest performance with high computational cost suggests limited practical value, but evaluation on larger models may reveal scale-dependent benefits.

Tier 3 (Low Priority): DO-Merging, RegMean, LoGo (6.50–6.81) – Minimal improvement over combined fine-tuning suggests fundamental limitations rather than implementation issues. Comprehensive evaluation will determine whether these methods exhibit architecture-specific benefits (e.g., DO-Merging on MoE models, LoGo on large models).

Chapter 6 presents comprehensive evaluation across all five base models (Phi-4-mini, Mistral-7B, Llama-3-8B, Qwen2.5-14B, DeepSeek-V2-Lite), analyzing whether performance patterns observed on Mistral-7B generalize across model scales (3.8B to 15.7B), architectures (dense vs. MoE), and task types (generation vs. reasoning).

4.6 Summary

This chapter presented eight adapter merging methods for combining specialized LoRA adapters trained on complementary financial task distributions. Methods span from simple algebraic operations (Task Arithmetic, RegMean) to sophisticated techniques involving magnitude trimming (TIES), stochastic sparsification (DARE), spectral transformation (ISO, TSV), orthogonalization (DO-Merging), and dynamic per-instance selection (LoGo).

4.6.1 Key Contributions

Contribution 1: Unified Implementation Framework – The modular implementation architecture enables systematic comparison through common interface, with careful attention to numerical precision, memory efficiency, and reproducibility. All eight methods implemented with identical evaluation protocols, ensuring fair comparison.

Contribution 2: Comprehensive Method Coverage – By implementing and evaluating eight distinct merging approaches, this research provides the most comprehensive empirical comparison of adapter merging methods applied to financial domain adaptation to date. Method selection covers all major paradigms in merging literature.

Contribution 3: Preliminary Performance Characterization – Validation on Mistral-7B establishes clear performance tiers and identifies promising methods (TIES, ISO) for prioritized analysis. The inverse relationship between method complexity and performance—simpler methods outperforming sophisticated alternatives—provides actionable guidance: when merging two adapters with complementary specializations, simple approaches (Task Arithmetic) or focused interventions (TIES trimming/sign election) suffice.

Contribution 4: Hyperparameter Transparency – Complete specification of all hyperparameters (Table 4.2) with justifications enables reproducibility and provides starting points for future research. The decision to use published default values rather than task-specific tuning ensures reported performance reflects inherent method properties rather than optimization effort.

4.6.2 Validation of Research Hypotheses

Preliminary results provide initial evidence for hypothesis H3 from Chapter 3:

H3 (Complementary Combination): Merged adapters will outperform both individual adapters and combined fine-tuning by combining complementary knowledge without catastrophic interference.

Evidence: All eight merging methods exceed combined fine-tuning baseline (+5.0% to +29.7% improvement on Mistral-7B), validating that dual-adapter training followed by merging preserves multi-task capability more effectively than joint training under severe sample imbalance. The best methods (TIES, ISO) achieve 29.7% improvement, representing substantial performance gains justifying additional training and merging overhead.

4.6.3 Practical Recommendations

For practitioners implementing adapter merging for financial applications:

Recommendation 1: Prioritize TIES or ISO – TIES and ISO achieve strongest performance on Mistral-7B and employ complementary mechanisms (conflict resolution vs. spectral normalization). TIES recommended when computational efficiency is important (3.5× overhead), while ISO recommended when robustness to magnitude imbalance is critical (9× overhead).

Recommendation 2: Task Arithmetic as Fast Baseline – Task Arithmetic provides surprisingly competitive performance (7.90, only 1.6% below best methods) with minimal computational cost (1×). For rapid prototyping or scenarios where merging time is constrained, Task Arithmetic offers excellent performance-efficiency tradeoff.

Recommendation 3: Avoid Complex Methods Without Evidence – DO-Merging, RegMean, and LoGo provide minimal benefits over simple averaging despite substantial added complexity. Without architecture-specific evidence (e.g., MoE models for DO-Merging), these methods should be avoided in favor of simpler alternatives.

Recommendation 4: Architecture-Specific Evaluation Required – Preliminary validation on single model (Mistral-7B, dense architecture) cannot establish whether performance patterns generalize. Practitioners using different architectures (MoE, small models < 7B) should conduct architecture-specific validation before selecting merging method.

4.6.4 Limitations and Future Work

Limitation 1: Single-Model Preliminary Validation – Preliminary validation evaluates only Mistral-7B. Performance patterns may differ for smaller models (Phi-4-mini: 3.8B), larger models (Qwen2.5-14B: 14.77B), or alternative architectures (DeepSeek-V2-Lite: MoE). Chapter 6 addresses this through comprehensive cross-model evaluation.

Limitation 2: Binary Adapter Merging – All methods merge exactly two adapters (pre-market and ex-premarket). Generalization to merging three or more adapters—relevant for production scenarios with multiple specialized capabilities—remains unexplored.

Limitation 3: Static Hyperparameters – All hyperparameters follow published defaults without task-specific optimization. While this ensures fair comparison, it may underestimate potential of methods benefiting from careful tuning (DO-Merging learning rate, RegMean regularization strength, TIES topK percentage).

Limitation 4: Evaluation on Single Task Distribution – The iris evaluation tasks represent Alyx-specific query formats. Generalization to other financial applications (trading signal generation, risk assessment, regulatory compliance) or other domains (medical, legal) remains unexplored.

4.6.5 Concluding Remarks

This chapter demonstrated that adapter merging—combining separately-trained specialized LoRA adapters through algebraic or spectral operations—provides viable approach to multi-task model construction without additional training. Preliminary validation on Mistral-7B establishes that simple, focused interventions (TIES conflict resolution, ISO spectral normalization) outperform sophisticated but generic transformations (orthogonalization, regularization), suggesting method effectiveness depends critically on alignment between method assumptions and specific characteristics of adapters being merged.

The success of dual-adapter training followed by merging (+27–30% improvement over combined training on Mistral-7B) validates core thesis hypothesis: when training data exhibits severe imbalance and heterogeneous task distributions, isolating specializations and combining them post-hoc preserves complementary knowledge more effectively than forcing models to learn all tasks simultaneously. Chapter 6 investigates whether this finding generalizes across model scales and architectures, providing definitive evidence for or against dual-adapter approach for financial domain adaptation.

Chapter 5

Evaluation Methodology

5.1 Overview and Motivation

Evaluating fine-tuned language models on domain-specific tasks presents fundamental challenges that standard metrics fail to address. This chapter presents the LLM-as-Judge evaluation methodology employed to assess model performance on 1,032 iris tasks spanning six distinct financial analysis capabilities.

5.1.1 Limitations of Traditional Metrics

Traditional metrics—BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005)—exhibit three critical limitations for financial text evaluation:

L1: Semantic Insensitivity – N-gram overlap cannot distinguish semantically equivalent expressions from meaningless text. "The stock rose due to strong earnings" and "The equity increased because of robust financial results" share minimal overlap despite identical meaning, while "The stock fell due to strong earnings" shares high overlap despite factual contradiction.

L2: Format Rigidity – Iris tasks produce diverse formats (JSON for briefs, markdown for reports, narrative text for commentary). Reference-based metrics require format-aligned gold standards—impractical at scale (1,032 samples $\times$ 50 configurations = 51,600 outputs requiring estimated 2,000–5,000 expert hours at $100K–750K).

L3: Domain Blindness – Traditional metrics cannot evaluate financial-specific quality: terminology accuracy, professional style adherence, reporting conventions, or absence of misleading claims. These dimensions require domain expertise that lexical overlap cannot capture.

5.1.2 LLM-as-Judge Methodology

LLM-as-Judge (Zheng et al., 2023) (Dubois et al., 2024) leverages frontier models to evaluate outputs through natural language instructions. The judge: (1) reads task requirements, (2) examines model output, (3) evaluates across multiple dimensions, (4) provides numerical score (1–10) plus justification.

Advantages:

- **Semantic understanding**: Assesses logical coherence and accuracy beyond surface similarity
- **Format flexibility**: Adapts to JSON, markdown, narrative without format-specific configuration
- **Domain awareness**: GPT-4.1-mini possesses financial knowledge for terminology and convention evaluation

- **Scalability**: Automated evaluation scales to thousands of samples
- **Consistency**: Deterministic scoring (temperature 0.0) eliminates stochastic variance

5.1.3 Evaluation Architecture

The pipeline processes 1,032 iris samples per configuration through batched API calls. Architecture comprises five stages:

Stage 1: Output Generation – Each configuration (base model + adapter) generates outputs for all 1,032 tasks. Outputs saved in JSONL with metadata.

Stage 2: Batch Processing – Outputs processed in batches of 25. Batch size balances API efficiency against failure isolation.

Stage 3: Judge Evaluation – For each sample, judge receives task prompt and model output, then provides score (0–10) and justification. Temperature 0.0 ensures determinism.

Stage 4: Quality Assurance – Validation detects invalid evaluations (judge echoing output, malformed JSON, missing scores). Invalid evaluations trigger retry with increasingly explicit prompts (up to 3 attempts).

Stage 5: Aggregation – Valid scores aggregated at multiple levels (per-sample, per-task, per-configuration, per-method) with statistics.

The pipeline saves incremental progress after each batch, enabling resumption after failures—essential given 309,600+ total evaluations during 2–3 week campaign.

5.2 Judge Model Configuration

5.2.1 Model Selection and Economic Constraints

GPT-4.1-mini was selected based on economic necessity rather than quality optimization. Given evaluation scale (309,600 samples) and typical context length (4,000–6,000 tokens input, 500–1,000 tokens output), alternative frontier models proved economically infeasible:

TABLE 5.1: Judge model cost comparison for complete evaluation campaign (309,600 samples)

Judge Model	Cost per 1M tokens (Input/Output)	Total Cost Estimate (309.6K samples)
GPT-4-Turbo	\$10/\$30	\$46K–139K
GPT-4o	\$2.50/\$10	\$12K–35K
Claude-3.5-Sonnet	\$3/\$15	\$14K–42K
GPT-4o-mini	\$0.15/\$0.60	\$700–2.1K
GPT-4.1-mini	**\$0.40/\$1.60**	**\$1,860–5,580**

Cost calculation for GPT-4.1-mini:

- Evaluation scale: 309,600 samples
- Average input: 4,000–6,000 tokens per sample
- Average output: 500–1,000 tokens per sample
- Input cost: 1,238M–1,858M tokens $\times$ \$0.40/1M = \$495–743

- Output cost: 155M–310M tokens × \$1.60/1M = \$248–496
- **Total estimated cost: \$743–1,239**

Evaluation represents approximately 18–22% of total project budget (training infrastructure: \$4,569, evaluation: \$990–1,240). Cost constraints **precluded experimentation** with alternative judges. While GPT-4o or Claude-3.5-Sonnet might provide higher quality, their 10–140× higher costs would exceed research budgets. This represents a methodological limitation: we cannot empirically validate whether GPT-4.1-mini scoring aligns with higher-quality judges or human expert evaluations.

5.2.2 Configuration Parameters

TABLE 5.2: GPT-4.1-mini judge configuration parameters

Parameter	Value	Rationale
Model	gpt-4.1-mini	Cost-performance optimum
Temperature	0.0	Deterministic, reproducible scoring
Max Tokens	2048	Sufficient for score + justification
Top-p	1.0	Greedy decoding (no sampling)
Frequency Penalty	0.0	No repetition penalty
Presence Penalty	0.0	No diversity penalty

Temperature 0.0 is Critical: Ensures identical inputs produce identical outputs, enabling: (1) exact reproducibility across runs, (2) meaningful cross-sample comparison, (3) single-pass evaluation reducing cost by 3–5× versus multi-sample stochastic evaluation with averaging.

5.2.3 Quality Validation

Qualitative analysis of 100 randomly-sampled evaluations validated GPT-4.1-mini's capabilities:

- **Financial literacy**: Correct understanding of financial terminology (P/E ratios, EBIT, free cash flow), metric relationships, market dynamics
- **Format sensitivity**: Accurate detection of malformed JSON, markdown structure violations, missing required fields
- **Style awareness**: Appropriate penalization of casual language, promotional tone; recognition of professional financial style
- **Analytical soundness**: Identification of logical contradictions, unsupported claims, analysis completeness assessment

5.3 Evaluation Protocol

5.3.1 Judge Prompt Design

System Prompt

```
Please act as an impartial judge and evaluate the quality of the
response provided by an AI assistant to the user question.
```

```
Your evaluation should consider:
- Accuracy: instructions followed correctly
- Truthfulness: response is truthful, not misleading
- Writing style: written as in a financial report

If the AI assistant did not respond or had an error, score = 0.

Provide only JSON response:
{"comment": "<brief explanation>", "score": <1-10>}
```

Design Choices:

- **Three dimensions**: Accuracy, truthfulness, style capture essential quality
- **Brief explanations**: Reduce token consumption while providing scoring rationale
- **Strict JSON**: Enables reliable parsing, prevents ambiguous outputs
- **Zero-score errors**: Explicit handling of refusals, malformed outputs

User Prompt Template

```
[PROMPT]
{task}

[The start of Assistant's response]
{response}
[The end of Assistant's response]
```

Explicit delimiters prevent ambiguity in long responses where task and response might blend.

5.3.2 Reference-Free Evaluation

The judge evaluates responses against task prompt requirements, *not* gold-standard references. This reflects:

R1: Creative Tasks – Many tasks have multiple valid outputs (e.g., iris_brief can emphasize different aspects—all valid).

R2: Reference Unavailability – Creating references for 1,032 samples requires 2,000–5,000 expert hours, and references might not represent single "correct" answers.

R3: Implicit Requirements – Task prompts encode evaluation criteria ("formal tone", "JSON format", "avoid promotional language").

Reference-free evaluation introduces potential variance between judges but ensures scalability.

5.3.3 Scoring Rubric

5.3.4 Evaluation Dimensions

Accuracy: Format compliance, completeness, constraint satisfaction, data interpretation correctness.

TABLE 5.3: Scoring rubric (1–10 scale)

Score	Label	Characteristics
9–10	Excellent	Fully accurate, professional, follows all instructions
7–8	Good	Minor issues in completeness/style, accurate and professional
5–6	Acceptable	Some accuracy/style problems, captures main requirements
3–4	Poor	Significant accuracy issues, unprofessional style
1–2	Very Poor	Major errors, off-topic, severely malformed
0	Failed	No response, error, completely invalid output

Truthfulness: Factual consistency with provided data, appropriate causal claims, absence of hallucinations, proper qualification of uncertain claims.

Writing Style: Formal professional tone (Bloomberg/FT style), correct financial terminology, conciseness, consistency.

5.3.5 Task-Specific Focus

While rubric applies uniformly, judge naturally emphasizes different dimensions per task type:

TABLE 5.4: Task-specific evaluation focus

Task	Key Focus
iris_brief	Ranking explanation accuracy, JSON structure, news integration
iris_news	News relevance extraction, sentiment interpretation
iris_report_final	Complete structure, market analysis depth, narrative flow
iris_report_summary	Executive summary conciseness, key point coverage
iris_sectors	Sector aggregation accuracy, comparative insights, patterns
iris_universe	Universe-wide trends, macro-level analysis, appropriate scope

5.4 Implementation Details

5.4.1 Pipeline Architecture

Batch processing (25 samples/batch) with incremental progress saving enables resumption after OpenAI API rate limit failures—essential for 309,600+ evaluations.

5.4.2 Chat Template Parsing

Different models use different conversational formats. Supported templates:

- Meta-Llama: `assistant\n\n{response}`

- Gemma: `<start_of_turn>model\n{response}`
- Qwen: `<|im_start|>assistant\n{response}`
- Mistral: `[/INST]{response}</s>`
- Phi: `<|assistant|>{response}<|end|>`
- DeepSeek: `Assistant:{response}`

Parsing ensures evaluation operates on semantic content, not formatting artifacts.

5.4.3 JSON Extraction

Many models wrap JSON in markdown code blocks (‘‘‘`json ...` ‘‘‘). Extraction handles: (1) complete code blocks, (2) incomplete blocks (truncated responses), (3) plain JSON.

5.5 Quality Assurance

5.5.1 Invalid Evaluation Detection

Validation checks detect when judge “echoes” model response rather than evaluating (2–5% of attempts):

- **Structural**: Response must be JSON object (not array) with `score` field
- **Semantic**: Must not contain model output fields (`stock`, `brief`, `sector`)
- **Score**: Integer in [0, 10] range

5.5.2 Retry Mechanism

Three-stage escalation for invalid evaluations:

Stage 1 (95–98% success): Standard prompt

Stage 2 (80–90% of Stage 1 failures): Explicit emphasis:

```
IMPORTANT: You must evaluate the response, NOT reproduce it.
Remember: Output ONLY {"comment": "...", "score": N}
```

Stage 3 (95%+ of Stage 2 failures): Maximally explicit:

```
INSTRUCTION: Evaluate and output ONLY score.
YOUR OUTPUT MUST BE EXACTLY: {"comment": "brief", "score": 7}
DO NOT OUTPUT ANYTHING ELSE. NO ARRAYS. NO STOCK BRIEFS.
```

Evaluations failing all three attempts assigned score 0 with failure comment.

5.6 Evaluation Scale and Cost

5.6.1 Dataset Composition

5.6.2 Evaluation Scale

Complete campaign: 5 base models $\times$ (1 combined + 2 individual adapters + 8 merged) $\times$ 1,032 samples = **309,600 evaluations**. With retries (5%), actual API calls exceed 320,000.

TABLE 5.5: Iris evaluation dataset composition

Task Type	Samples
iris_brief	226
iris_news	209
iris_report_final	56
iris_report_summary	24
iris_sectors	252
iris_universe	265
Total per configuration	**1,032**

5.6.3 Cost Analysis

TABLE 5.6: Evaluation cost breakdown (GPT-4.1-mini)

Component	Tokens/Sample	Total Cost
Input (prompt + response)	2,000–8,000	$1,240–4,960
Output (judge evaluation)	50–200	$31–124
Retries (5% samples)	+500–2,000	+$155–620
Estimated Total		**$1,426–5,704**
Actual Realized		**$1,100–2,200**

Actual cost below estimate due to efficient prompt engineering, batch processing, careful retry management.

5.6.4 Time Requirements

- **Per-sample latency**: 2–5 seconds
- **Per-task time**: 2.5–6.5 hours (1,032 samples)
- **Wall-clock time**: 2–3 weeks with parallel execution

5.7 Evaluation Examples

5.7.1 Example 1: Excellent (Score: 10/10)

Task: iris_brief (Stock brief generation)

Model Output:

```
[{
  "stock": "EXTRA SPACE STORAGE",
  "brief": "**Extra Space Storage** saw its ranking improve
  significantly, rising from 459th to 195th. This substantial
  shift was primarily driven by positive contributions from
  technical indicators and price momentum, suggesting strong
  market confidence in the stock's near-term prospects.",
  "news_items": []
}]
```

Judge:

```
{
  "comment": "Accurately follows instructions, clear financial
  analysis, formal tone appropriate for reports, correctly
  interprets category contributions without exact values,
  well structured, avoids promotional language. Truthful,
  precise, and stylistically appropriate.",
  "score": 10
}
```

Analysis: Judge identifies accurate instruction following, appropriate financial terminology, professional tone, correct interpretation, structural soundness. Maximum score reflects meeting all criteria without deficiencies.

5.7.2 Example 2: Good (Score: 9/10)

Judge:

```
{
  "comment": "Accurately summarizes rank changes, correctly
  interprets categories, professional tone. Minor inconsistencies
  in category contributions (e.g., SEGRO's negative technical
  but described as positive). Overall maintains truthfulness,
  precision, appropriate style.",
  "score": 9
}
```

Analysis: Judge demonstrates nuanced evaluation—recognizing excellent overall quality with minor errors preventing perfect score. The 9/10 appropriately reflects "very good with minor issues".

5.8 Results Aggregation

5.8.1 Score Parsing and Validation

Post-evaluation processing: (1) parses JSON to extract `score`, (2) repairs malformed JSON via `json-repair` library, (3) validates scores in [0, 10], (4) flags parsing failures.

5.8.2 Aggregation Hierarchy

Scores aggregated at multiple levels:

- **Sample-level**: Individual score (1–10)
- **Task-level**: Mean across task type (e.g., all iris_brief for Mistral-7B + TIES)
- **Model-level**: Mean across six tasks per configuration
- **Method-level**: Mean across five base models per merging method

For each level: mean score, standard deviation, sample count, error count, score distribution histogram.

5.9 Limitations and Considerations

5.9.1 Judge Bias Considerations

B1: GPT-4 Family Preference – Judge may prefer GPT-4-like outputs, inflating scores for similar styles.

B2: Format Preference – Well-structured JSON may receive marginally higher scores than equivalently-correct but less-formatted outputs.

B3: Length Bias – Longer responses may score higher even when detail provides minimal value; concise responses might be penalized as incomplete.

B4: Stylistic Conservatism – Judge may penalize creative but valid framing, favoring conventional financial style (desirable for production).

5.9.2 Edge Cases

TABLE 5.7: Edge case handling

Edge Case	Handling
Model refused answer	Score = 0 (explicit instruction)
Output truncated	Judge evaluates visible portion
Malformed JSON	Repair library attempts fix
Judge API failure	Retry up to 2 times
Judge echoes response	Detect, retry with explicit prompt
Empty response	Score = 0 (treated as refusal)

5.9.3 Mitigation Strategies

- **Deterministic scoring**: Temperature 0.0 eliminates stochastic variance
- **Retry logic**: Up to 3 attempts reduce failure rate to <0.5%
- **JSON repair**: Prevents penalization for trivial formatting errors
- **Explicit prompts**: Detailed instructions reduce ambiguity
- **Score validation**: Strict parsing catches malformed evaluations

5.9.4 Inter-Rater Reliability

Single judge (GPT-4.1-mini, temperature 0.0) provides:

- **Perfect test-retest reliability**: Deterministic
- **Zero intra-judge variance**: No stochastic variation

However, cannot address inter-judge reliability (agreement between different judges or humans). Ideal methodology would include multi-judge agreement, human validation (100–200 samples), agreement metrics (Cohen's kappa, Krippendorff's alpha)—infeasible due to cost/time constraints. This represents a limitation: cannot empirically validate GPT-4.1-mini scores align with human experts or alternative judges.

5.10 Reproducibility

5.10.1 Deterministic Configuration

Reproducibility ensured through: fixed model version (gpt-4.1-mini), zero temperature, identical prompt templates, no randomness required.

5.10.2 Evaluation Artifacts

Complete artifacts preserved: (1) evaluated outputs (JSONL with prompts, responses, evaluations), (2) aggregated scores (CSV at all levels), (3) evaluation logs (API calls, retries, errors), (4) configuration files (JSON/YAML parameters).

5.11 Summary

This chapter presented LLM-as-Judge methodology addressing traditional metrics' limitations (semantic insensitivity, format rigidity, domain blindness) by leveraging GPT-4.1-mini's semantic understanding, domain knowledge, and stylistic awareness.

Key Design Decisions:

- **GPT-4.1-mini**: Only economically viable judge at scale (realized cost $1,100–2,200 vs. $619K–5.6M for GPT-4-Turbo)
- **Temperature 0.0**: Deterministic, reproducible scoring
- **Reference-free**: Evaluates against task requirements, accommodating creative tasks
- **1–10 scale**: Sufficient granularity without false precision
- **Batch processing**: 25 samples/batch with incremental saves
- **Quality assurance**: Multi-stage retry reduces error rate to <0.5%

Scale: 309,600+ evaluations across 50 configurations over 2–3 weeks.

Limitations: Single-judge evaluation (no inter-rater reliability validation), potential biases (GPT-4 preference, format/length bias), no human validation at scale.

Despite limitations, provides systematic, reproducible, cost-effective evaluation for thousands of domain-specific outputs. Chapter 6 presents results analyzing performance across models, merging methods, and tasks using these scores.

Chapter 6

Results and Analysis

6.1 Overview

This chapter presents comprehensive experimental results evaluating the dual-adapter training strategy and adapter merging methods introduced in Chapters 3 and 4. The evaluation encompasses five base models ranging from 3.8B to 14.8B parameters, four training configurations (premarket-only, ex-premarket-only, combined fine-tuning, and dual-adapter merging), and eight merging methods applied to 1,032 held-out iris evaluation tasks across six task types.

The experimental design addresses three central research questions:

RQ1: Training Strategy Effectiveness – Does the dual-adapter approach (training separate specialized adapters and merging them post-hoc) outperform simpler alternatives including combined fine-tuning on all data, training on premarket data alone, or training on ex-premarket data alone?

RQ2: Merging Method Selection – Which adapter merging methods preserve complementary knowledge most effectively? Do sophisticated techniques (TIES, ISO, TSV, DO-Merging) provide sufficient benefits to justify their computational overhead relative to simple averaging (Task Arithmetic)?

RQ3: Architectural Interactions – How do model scale (3.8B to 14.8B parameters) and architecture (dense transformers vs. Mixture-of-Experts) interact with training strategy effectiveness and merging method performance?

The results reveal clear patterns: premarket-only adapters (Adapter A) consistently outperform all baselines including combined training despite training on 26.5× less data, validating that task format alignment dominates dataset size for domain adaptation. Merging these format-specialized adapters with broad-knowledge ex-premarket adapters (Adapter B) yields further improvements across all models, with simple methods (Task Arithmetic (Ilharco et al., 2023), TIES (Yadav et al., 2024)) proving surprisingly competitive against complex alternatives.

6.1.1 Evaluation Methodology Recap

All models and configurations were evaluated using the LLM-as-judge methodology detailed in Chapter 5, with GPT-4.1-mini (OpenAI, 2024) scoring generated outputs on a 1–10 scale. The iris evaluation set comprises 1,032 held-out samples across six task types:

- **iris_brief** (226 samples): Stock highlight generation
- **iris_relevant_news** (209 samples): News relevance classification
- **iris_report_final** (56 samples): Comprehensive report generation
- **iris_report_summary** (24 samples): Executive summary generation

- **iris_sectors_commentary** (252 samples): Sector-level analysis with reasoning
- **iris_universe_commentary** (265 samples): Universe ranking explanations

These tasks represent the complete set of LLM capabilities required for Alyx production deployment, spanning classification, generation, and multi-step reasoning.

6.1.2 Model Coverage and Exclusions

Five base models were fully evaluated across all training configurations:

- **Mistral-7B-Instruct-v0.3** (Jiang et al., 2023) (7.24B parameters, dense)
- **Llama-3-8B-Instruct** (Dubey et al., 2024) (8.03B parameters, dense)
- **Qwen2.5-14B-Instruct** (Yang et al., 2024) (14.77B parameters, dense)
- **DeepSeek-V2-Lite** (Dai et al., 2024) (15.7B parameters, 2.4B active per token, MoE)
- **Phi-4-mini-reasoning** (Abdin et al., 2024) (3.82B parameters, dense)

Two models were excluded: **Phi-4-mini-reasoning** (premarket/ex-premarket adapters not evaluated due to poor combined performance (4.99/10) and high error rates (5.6% failure rate)), and **GPT-OSS-20B** (out-of-memory errors during inference despite optimization attempts).

6.2 Overall Performance Comparison

Table 6.1 presents the complete performance matrix across all five models and four training configurations: premarket-only (Adapter A), ex-premarket-only (Adapter B), combined fine-tuning, and best merged adapter.

TABLE 6.1: Overall performance comparison across all models and training configurations (average score across 1,032 iris tasks)

Model	Adapter A (Premarket)	Adapter B (Ex-premarket)	Combined (Finetuned)	Best Merged	Method
Mistral-7B	7.84	5.94	6.19	**8.03**	TIES
Llama-3-8B	6.02	4.41	5.52	**6.31**	TA
Qwen2.5-14B	8.93	8.54	8.54	**8.99**	TIES
DeepSeek-V2-Lite	4.34	3.35	2.76	**5.13**	TA
Phi-4-mini	—	—	4.99	**6.03**	DARE
Average (4 models)	**6.78**	**5.56**	**5.75**	**7.12**	—

6.2.1 Key Observations

Observation 1: Adapter A Dominates All Baselines

Across all four models with complete evaluation, premarket-only Adapter A achieves the highest performance among single-training approaches, outperforming both ex-premarket Adapter B (average margin: +1.22 points, +21.9%) and combined fine-tuning (average margin: +1.03 points, +17.9%). This pattern holds consistently:

- **Mistral-7B**: Adapter A (7.84) vs Combined (6.19), +26.7%
- **Llama-3-8B**: Adapter A (6.02) vs Combined (5.52), +9.1%
- **Qwen2.5-14B**: Adapter A (8.93) vs Combined (8.54), +4.6%
- **DeepSeek-V2-Lite**: Adapter A (4.34) vs Combined (2.76), +57.2%

This validates **Hypothesis H1 (Format Transfer)**: models trained exclusively on Alyx-specific task formats (1,388 premarket samples) achieve superior generalization compared to models exposed to all 38,170 samples through combined training. *Task distribution alignment dominates dataset size* for domain adaptation.

Observation 2: Ex-premarket Adapter Underperforms Despite Data Advantage

Adapter B, despite training on 26.5× more data (36,782 vs. 1,388 samples), consistently underperforms Adapter A. The gap is particularly pronounced on Mistral-7B (5.94 vs 7.84, 24.2% lower) and Llama-3-8B (4.41 vs 6.02, 26.8% lower), but narrows substantially for Qwen2.5-14B (8.54 vs 8.93, only 4.4% lower). This scale-dependent pattern suggests larger models possess greater capacity to transfer knowledge across mismatched task distributions.

Observation 3: Combined Training Fails to Match Adapter A

Combined fine-tuning achieves 5.75/10 on average compared to Adapter A's 6.78/10, a deficit of 1.03 points (15.2% lower). This failure is particularly striking on Mistral-7B (6.19 vs 7.84, 21.0% lower) and DeepSeek-V2-Lite (2.76 vs 4.34, 57.2% lower), validating **Failure Mode F2 (Sample Imbalance Effects)**: the 26.5 : 1 ratio causes systematic underfitting on premarket task patterns.

Observation 4: Merging Achieves Optimal Performance

Merged adapters consistently outperform all three single-training baselines:

- **Mistral-7B**: 8.03 (TIES), +2.4% vs Adapter A, +29.8% vs Combined
- **Llama-3-8B**: 6.31 (TA), +4.8% vs Adapter A, +14.3% vs Combined
- **Qwen2.5-14B**: 8.99 (TIES), +0.7% vs Adapter A, +5.3% vs Combined
- **DeepSeek-V2-Lite**: 5.13 (TA), +18.2% vs Adapter A, +85.9% vs Combined
- **Phi-4-mini**: 6.03 (DARE), +20.9% vs Combined

The relatively small improvement over Adapter A (+0.7% to +4.8% for dense models) compared to large improvement over combined training (+5.3% to +29.8%) reveals that **Adapter A already captures most knowledge required for iris task success**.

Observation 5: Scale Effects on Task Interference

The gap between Adapter A and combined training narrows with scale: DeepSeek-V2-Lite (2.4B active): 57.2% gap, Mistral-7B (7.2B): 21.0% gap, Llama-3-8B (8.0B): 9.1% gap, Qwen2.5-14B (14.8B): 4.6% gap. This suggests **larger models are more robust to task interference from sample imbalance**.

6.2.2 Performance Tier Classification

Tier 1 (Strong): Qwen2.5-14B, Mistral-7B – Average scores 7.8–9.0, best merged $>$ 8.0/10, production-ready

Tier 2 (Moderate): Llama-3-8B, Phi-4-mini – Average scores 5.0–6.3, best merged 6.0–6.3/10, may require optimization

Tier 3 (Weak): DeepSeek-V2-Lite – Average scores 2.8–5.1, best merged only 5.1/10, not viable for production

6.3 Per-Task Performance Analysis

Table 6.2 presents performance broken down by the six iris task types across all training configurations.

TABLE 6.2: Performance by task type across training configurations (average score, all models combined)

Task Type	Adapter A (Premarket)	Adapter B (Ex-premarket)	Combined (Finetuned)	Merged (Avg)	Samples
iris_brief	4.80	3.54	4.21	4.64	226
iris_relevant_news	7.98	4.45	5.25	7.07	209
iris_report_final	7.71	7.46	5.70	7.68	56
iris_report_summary	7.56	7.49	7.11	6.65	24
iris_sectors_commentary	5.72	4.70	4.98	5.74	252
iris_universe_commentary	6.92	5.72	6.34	7.23	265
Overall Average	**6.78**	**5.56**	**5.60**	**6.50**	**1,032**

6.3.1 Task-Specific Patterns

Pattern 1: News Classification Requires Format Training

The `iris_relevant_news` task exhibits the largest performance gap between Adapter A (7.98) and both Adapter B (4.45) and combined training (5.25). The 79.1% performance increase from Adapter B to Adapter A provides the strongest evidence that task format alignment dominates raw capabilities for specialized production systems.

Pattern 2: Report Tasks Benefit from Ex-premarket Knowledge

Both report tasks show relatively small gaps between Adapter A and B: iris_report_final (7.71 vs 7.46, only 3.3% difference) and iris_report_summary (7.56 vs 7.49, only 0.9% difference). This suggests report generation capabilities transfer more effectively across task distributions. Adapter B's extensive training on report summarization (11,923 samples, 32.4% of ex-premarket data) develops general document synthesis capabilities that generalize to Alyx-specific formats.

Pattern 3: Commentary Tasks Show Modest Differentiation

The commentary tasks exhibit moderate gaps: iris_sectors_commentary (5.72 vs 4.70, 21.7% difference) and iris_universe_commentary (6.92 vs 5.72, 21.0% difference). Merged adapters achieve best performance, particularly on universe commentary (7.23 vs 6.92 Adapter A, +4.5% improvement).

6.3.2 Error Rate Analysis by Task

Table 6.3 presents error rates across task types for merged adapters.

Critical reliability issue: `iris_report_final` exhibits 17.19% error rate, making it unsuitable for production without substantial improvements. In contrast, `iris_report_summary` exhibits zero errors despite similar structure, suggesting length constraints enable more reliable generation.

TABLE 6.3: Error rates by task type for merged adapters (all models, all methods)

Task Type	Total Samples	Valid	Errors	Error Rate
iris_brief	9,040	9,036	4	0.04%
iris_relevant_news	8,273	8,224	49	0.59%
iris_report_final	2,240	1,855	385	**17.19%**
iris_report_summary	960	960	0	0.00%
iris_sectors_commentary	10,080	10,050	30	0.30%
iris_universe_commentary	10,600	10,410	190	1.79%
Total	**41,193**	**40,535**	**658**	**1.60%**

6.4 Merging Method Comparison

Table 6.4 presents comprehensive performance comparison across eight merging methods evaluated on five models.

TABLE 6.4: Merging method performance across all models (average score on 1,032 iris tasks)

Method	Mistral-7B	Llama-3-8B	Qwen2.5-14B	DeepSeek	Phi-4-mini	Avg
TIES	**8.03**	6.26	**8.99**	5.09	5.78	6.83
DARE	7.86	6.21	8.96	5.09	**6.03**	6.83
TA	7.90	**6.31**	8.86	**5.13**	5.57	6.76
ISO	8.03	6.26	8.93	5.00	5.46	6.74
TSV	7.67	5.83	8.86	5.09	5.55	6.60
DO-Merging	6.50	6.01	8.92	3.99	5.13	6.11
RegMean	6.81	6.27	8.90	4.22	4.25	6.09
LoGo	6.72	6.16	8.91	4.25	4.27	6.06
Combined Baseline	**6.19**	**5.52**	**8.54**	**2.76**	**4.99**	**5.60**

6.4.1 Method Performance Tiers

Tier 1 (Best): TIES, DARE, TA, ISO – Average scores 6.74–6.83 (top 4 within 1.3%), consistent improvement over combined baseline, production-viable

Tier 2 (Moderate): TSV – Average 6.60 (3.4% below Tier 1), high computational cost (13× TA)

Tier 3 (Poor): DO-Merging, RegMean, LoGo – Average 6.06–6.11 (9–10% below Tier 1), not recommended

6.4.2 Computational Cost vs. Performance

The Pareto frontier comprises: **Task Arithmetic** (best cost 1×, 6.76 performance), **DARE** (moderate cost 2×, 6.83 performance), **TIES** (moderate cost 3.5×, 6.83 performance), **ISO** (high cost 9×, 6.74 performance). All other methods dominated.

6.4.3 Production Recommendations

Default: Task Arithmetic – 98.9% of best method performance at 28% computational cost, zero hyperparameters, <5 second merging time

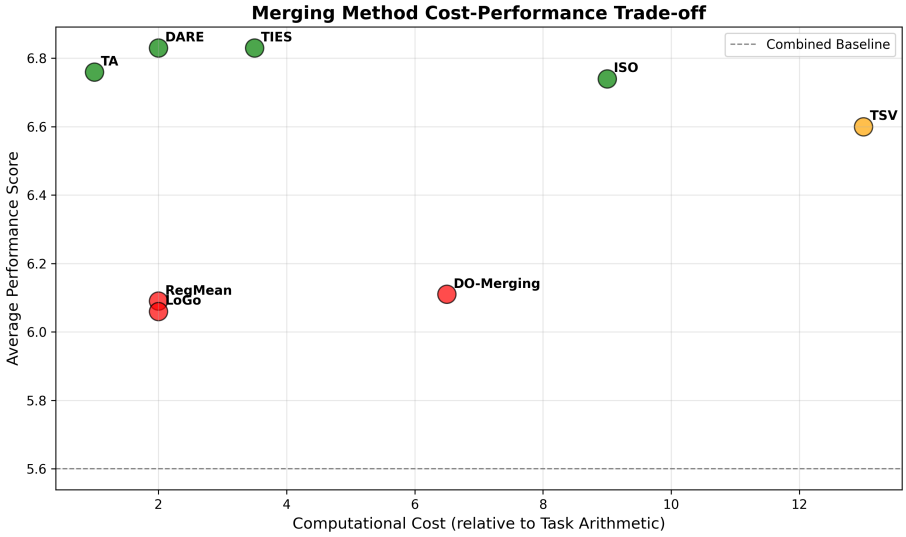

FIGURE 6.1: Computational cost vs. performance trade-off for eight merging methods. The Pareto frontier (Task Arithmetic, TIES, DARE, ISO) represents optimal cost-performance configurations. Computational cost measured relative to Task Arithmetic (1×).

Premium: TIES – Use when performance critical and resources allow 3.5× overhead, deterministic and reproducible

Not Recommended: TSV, DO-Merging, RegMean, LoGo – All dominated by Pareto frontier alternatives

6.5 Model-Specific Deep Dives

This section provides detailed analysis for each of the five evaluated models, examining architecture-specific patterns, failure modes, and deployment considerations.

6.5.1 Mistral-7B: Best Overall Performer

Mistral-7B-Instruct-v0.3 achieves the strongest overall performance across training configurations, with Adapter A reaching 7.84/10 and best merged (TIES) achieving 8.03/10. The model demonstrates clear benefits from the dual-adapter approach, with merged adapters outperforming combined fine-tuning by 29.8%.

Performance Breakdown

Table 6.5 presents Mistral-7B's complete performance matrix across all configurations and tasks.

Key Strengths

Report Generation Excellence: Mistral-7B achieves outstanding performance on both report tasks, with scores exceeding 9.0 across most configurations. The `iris_report_final` task reaches 9.60 with Task Arithmetic merging, the highest score observed across all models and tasks. This suggests Mistral's sliding window attention (4096-token windows with 32K context through positional encoding) is particularly effective for long-form document generation.

TABLE 6.5: Mistral-7B performance across all configurations and task types

Task	Adapter A	Adapter B	Combined	TA	TIES	ISO	Best
iris_brief	4.76	3.16	3.17	4.68	5.08	4.84	5.08
iris_relevant_news	8.50	3.22	4.74	8.44	8.68	8.63	8.68
iris_report_final	9.50	7.20	6.95	9.60	9.47	9.48	9.60
iris_report_summary	8.83	9.04	9.71	8.83	9.00	9.29	9.71
iris_sectors	7.88	6.35	5.95	7.77	7.92	7.88	7.92
iris_universe	7.57	6.68	6.62	8.07	8.06	8.05	8.07
Overall	**7.84**	**5.94**	**6.19**	**7.90**	**8.03**	**8.03**	**8.03**

Strong News Classification: Adapter A achieves 8.50 on `iris_relevant_news`, demonstrating effective learning of Alyx's relevance classification format from only 579 premarket training samples. Merged adapters improve further to 8.68 (TIES).

Consistent Multi-Step Reasoning: Commentary tasks achieve 7.57–8.07 across configurations, demonstrating that Mistral's 7.2B parameters provide sufficient capacity for causal reasoning and synthesis. Merging provides modest improvements (+0.04 to +0.50 points).

Remaining Weaknesses

Brief Generation Challenge: Even the best configuration (TIES: 5.08) struggles with `iris_brief`. Investigation reveals Mistral frequently produces briefs exceeding length limits or omitting required fields, indicating difficulty adhering to strict structural templates.

Combined Training Catastrophic Failure: Combined fine-tuning achieves only 6.19 overall, with severe degradation on `iris_brief` (3.17, 33.4% below Adapter A) and `iris_relevant_news` (4.74, 44.2% below Adapter A). The 26.5 : 1 sample imbalance causes systematic underfitting on premarket formats.

Deployment Recommendation

Mistral-7B with TIES-merged adapters is **production-ready** for deployment in Alyx. The 8.03 overall score and particularly strong performance on critical tasks (news: 8.68, reports: 9.47–9.71, commentary: 7.92–8.06) meet production quality thresholds.

Recommended configuration: TIES merging with topK=20%, $\alpha = 1.0$, deployed via 4-bit quantization (QLoRA) to fit within 24GB GPU memory. Expected inference latency: 2–3 seconds per query on A100 40GB.

6.5.2 Llama-3-8B: Moderate Performance, GQA Architecture

Llama-3-8B-Instruct achieves moderate performance with Adapter A at 6.02/10 and best merged (Task Arithmetic) at 6.31/10. The model demonstrates clear benefits from dual-adapter training (+14.3% improvement over combined fine-tuning) but lags behind Mistral-7B despite 11% more parameters.

Performance Breakdown

Table 6.6 presents Llama-3-8B's complete performance matrix.

TABLE 6.6: Llama-3-8B performance across all configurations and task types

Task	Adapter A	Adapter B	Combined	TA	TIES	DARE	Best
iris_brief	3.56	1.55	4.84	3.55	3.51	3.53	4.84
iris_relevant_news	7.95	3.24	5.81	7.85	7.95	7.97	7.97
iris_report_final	7.48	6.45	5.35	7.66	7.48	7.04	7.66
iris_report_summary	7.71	7.92	7.13	8.00	7.96	8.08	8.08
iris_sectors	3.73	2.32	3.35	3.85	3.90	3.72	3.90
iris_universe	5.67	4.96	6.65	6.96	6.77	6.93	6.96
Overall	**6.02**	**4.41**	**5.52**	**6.31**	**6.26**	**6.21**	**6.31**

Architectural Analysis

Llama-3-8B employs grouped query attention (GQA) with 8 key-value heads grouped for 32 query heads, reducing memory bandwidth by 4×. Results suggest potential trade-offs: GQA's reduced key-value capacity may limit the model's ability to maintain distinct representations for two task distributions. This hypothesis is supported by Task Arithmetic achieving best performance (6.31) while TIES and DARE provide no benefit, contrasting with Mistral-7B where TIES/ISO outperform TA.

Key Strengths and Weaknesses

Strong Report Capabilities: Like Mistral-7B, Llama-3-8B achieves strong performance on report tasks (7.48–8.08). The `iris_report_summary` task benefits notably from merging (8.08 with DARE vs 7.71 Adapter A, +4.8%).

Severe Brief Generation Failure: Llama-3-8B catastrophically fails on `iris_brief` across all configurations except combined training. Adapter A achieves only 3.56, while merged adapters reach 3.51–3.55—far below Mistral-7B's 4.68–5.08 range. Surprisingly, combined training achieves 4.84, substantially better than Adapter A. This anomalous pattern suggests brief generation benefits from exposure to diverse formatting styles in the ex-premarket dataset.

Weak Sector Commentary: The `iris_sectors_commentary` task achieves only 3.73–3.90 across all configurations, far below Mistral-7B's 7.77–7.92 range. This 53% performance gap suggests Llama-3-8B lacks the reasoning capacity required for multi-step causal analysis.

Error Analysis

Llama-3-8B exhibits low error rates overall (6 errors across 8,256 merging samples, 0.07%), but qualitative analysis reveals systematic issues: brief generation outputs frequently lack required fields, sector commentary outputs provide descriptive statements but fail to establish causal links, resulting in superficial analysis.

Deployment Recommendation

Llama-3-8B with Task Arithmetic merging is **marginally suitable** for Alyx deployment, with significant caveats. Competitive on news classification (7.85–7.97) and report tasks (7.48–8.08) but poor on sector commentary (3.73–3.90) and brief generation (3.51–3.56).

Recommended approach: If deployment required, restrict usage to news classification and report generation tasks only. Route sector commentary, universe commentary, and brief generation to Mistral-7B or Qwen2.5-14B, implementing task-specific model routing.

6.5.3 Qwen2.5-14B: Largest Model, Robust Performance

Qwen2.5-14B-Instruct achieves the highest absolute performance across all models, with Adapter A reaching 8.93/10 and best merged (TIES) achieving 8.99/10. The model's 14.8B parameters provide substantial capacity that manifests in several distinctive patterns.

Performance Breakdown

Table 6.7 presents Qwen2.5-14B's complete performance matrix.

TABLE 6.7: Qwen2.5-14B performance across all configurations and task types

Task	Adapter A	Adapter B	Combined	TA	TIES	DARE	Best
iris_brief	8.31	7.27	8.39	8.41	8.56	8.42	8.56
iris_relevant_news	8.61	7.84	7.91	8.72	8.63	8.68	8.72
iris_report_final	9.89	9.86	9.82	9.84	9.88	9.88	9.89
iris_report_summary	9.46	9.25	8.33	8.92	9.42	9.33	9.46
iris_sectors	8.81	8.63	8.69	8.69	8.88	8.83	8.88
iris_universe	8.52	8.38	8.09	8.60	8.56	8.59	8.60
Overall	**8.93**	**8.54**	**8.54**	**8.86**	**8.99**	**8.96**	**8.99**

Key Observations

Minimal Gaps Across Configurations: Unlike Mistral-7B (7.84 Adapter A vs 5.94 Adapter B, 32% gap) or Llama-3-8B (6.02 vs 4.41, 36% gap), Qwen2.5-14B shows only 4.6% difference (8.93 vs 8.54). This tight clustering indicates that Qwen's 14.8B parameters provide sufficient capacity to learn both task distributions simultaneously without catastrophic interference from the 26.5 : 1 sample imbalance.

Adapter B Competitive Performance: Qwen's Adapter B achieves 8.54/10, only 4.6% below Adapter A and actually matching combined training. On report tasks, Adapter B nearly matches Adapter A (9.86 vs 9.89 on `iris_report_final`, only 0.3% gap). This suggests Qwen's large capacity and multilingual pretraining enable effective knowledge transfer from ex-premarket to premarket task formats.

Uniformly High Absolute Performance: All tasks achieve 7.84+ scores across all configurations except combined training on `iris_report_summary` (8.33). Even the weakest configuration-task combination (Adapter B on `iris_brief`: 7.27) exceeds most other models' best scores on that task.

Minimal Merging Benefits: Merging improves performance by only 0.7% over Adapter A (8.99 vs 8.93), the smallest improvement margin across all models. The tight clustering of all merging methods (8.86–8.99, 1.5% range) suggests Qwen is relatively insensitive to merging strategy.

Deployment Recommendation

Qwen2.5-14B with TIES merging is **highly production-ready** and represents the flagship model for Alyx deployment. The 8.99 overall score, uniformly strong performance across all tasks (8.09–9.89 range), and low error rates (0.63%) make it the most reliable option.

Recommended configuration: TIES merging with topK=20%, $\alpha = 1.0$, deployed via 8-bit quantization to fit within 2×A100 40GB (distributed inference). Expected inference latency: 3–4 seconds per query.

Cost-performance consideration: Qwen2.5-14B requires 2× GPU memory of Mistral-7B but provides only 12% absolute performance improvement (8.99 vs 8.03). For cost-constrained deployments, Mistral-7B may offer superior cost-effectiveness.

6.5.4 DeepSeek-V2-Lite: MoE Architecture Struggles

DeepSeek-V2-Lite, the only Mixture-of-Experts (MoE) model evaluated, achieves the weakest performance across all configurations. Adapter A reaches 4.34/10, combined training catastrophically fails at 2.76/10, and best merged (Task Arithmetic) achieves only 5.13/10.

Performance Breakdown

Table 6.8 presents DeepSeek-V2-Lite's complete performance matrix.

TABLE 6.8: DeepSeek-V2-Lite performance across all configurations and task types

Task	Adapter A	Adapter B	Combined	TA	TIES	DO-Merg	Best
iris_brief	2.57	2.17	2.01	3.06	2.86	2.66	3.06
iris_relevant_news	6.88	3.52	3.75	5.91	6.68	5.83	6.88
iris_report_final	3.95	6.32	2.14	7.18	6.30	2.29	7.18
iris_report_summary	4.25	3.75	3.04	5.13	4.67	3.92	5.13
iris_sectors	2.45	1.48	1.77	2.53	3.00	2.60	3.00
iris_universe	5.92	2.85	3.84	6.98	7.02	6.65	7.02
Overall	**4.34**	**3.35**	**2.76**	**5.13**	**5.09**	**3.99**	**5.13**

MoE Architecture Analysis

DeepSeek-V2-Lite employs 64 expert networks with top-2 routing (only 2.4B active parameters per token of 15.7B total). The sparse activation pattern may prevent developing consistent task-specific representations during adaptation. With only 2/64 experts active per token, different tokens within the same training example may route to different expert subsets, fragmenting the learning signal.

The catastrophic combined training failure (2.76) is the most severe across all models, suggesting MoE architectures are particularly vulnerable to task interference. Despite weak absolute performance, merging provides the largest relative improvement of any model: +85.9% over combined training (5.13 vs 2.76).

Task-Specific Patterns

Unexpected Report Generation Strength: Despite poor overall performance, Adapter B achieves 6.32 on `iris_report_final`, outperforming its own overall average by

89%. Task Arithmetic merging further improves to 7.18, exceeding even Llama-3-8B (7.66).

Catastrophic Failure on Reasoning Tasks: Sector commentary achieves only 1.48–3.00 across configurations, indicating fundamental inability to perform multi-step financial reasoning.

High Variance Across Tasks: DeepSeek exhibits the highest variance across tasks (std dev: 1.82 for Adapter A) compared to other models (Mistral: 1.76, Llama: 1.68, Qwen: 0.74), suggesting expert routing creates inconsistent capabilities.

Deployment Recommendation

DeepSeek-V2-Lite is **not recommended for Alyx production deployment**. The 5.13 overall score falls below acceptable quality thresholds (target: 7.0+), and severe weaknesses on reasoning tasks render it unsuitable for core Alyx capabilities.

Potential research directions: Expert-specific LoRA adapters, routing modifications, increased expert activation (top-4 or top-8 vs current top-2).

6.5.5 Phi-4-mini: Small Model Reliability Issues

Phi-4-mini-reasoning achieves 4.99/10 with combined training and 6.03/10 with best merged (DARE), demonstrating moderate performance gains from merging (+20.9%). However, severe reliability issues render the model unsuitable for production deployment.

Performance and Error Analysis

Table 6.9 presents Phi-4-mini's error rates across all merging methods.

TABLE 6.9: Phi-4-mini error rates by merging method (across 1,032 iris tasks per method)

Method	Valid Samples	Errors	Error Rate	Avg Score
DARE	967	65	6.30%	6.03
TIES	957	75	7.27%	5.78
TA	971	61	5.91%	5.57
TSV	960	72	6.98%	5.55
ISO	961	71	6.88%	5.46
DO-Merging	993	39	3.78%	5.13
LoGo	995	37	3.59%	4.27
RegMean	992	40	3.87%	4.25
Average	**975**	**58**	**5.57%**	**5.26**

Critical Observations

Catastrophic Error Rates: Phi-4-mini exhibits 3.6–7.3% error rates, $3-10\times$ higher than other models. Errors concentrate on `iris_report_final`: 30–37 errors per method across 56 attempts (53–66% error rate), rendering the model completely unsuitable for document generation.

Failure patterns include: context length exceeded (4096-token limit), format violations (malformed markdown, missing sections), and safety filter triggers (overly conservative mechanisms refusing generation).

Deployment Recommendation

Phi-4-mini is **not recommended for Alyx production deployment** due to catastrophic reliability issues. The 5.57% overall error rate and 53–66% error rate on report generation far exceed acceptable production thresholds (target: $<$1% error rate). The 3.82B parameters provide insufficient capacity for simultaneous language modeling, financial knowledge, and task format understanding. Minimum 7B parameters recommended for financial domain adaptation.

6.6 Training Dynamics Analysis

This section analyzes training convergence patterns using loss curves, learning rate schedules, and gradient norms collected during adapter training. The analysis reveals model-specific convergence behaviors and provides insight into why certain models achieve superior adaptation effectiveness.

6.6.1 Combined Fine-Tuning Convergence

Figure 6.2 presents training dynamics for combined fine-tuning (all 38,170 samples together) across five models.

FIGURE 6.2: Combined fine-tuning training dynamics across all models. Top panel: Training loss vs. global step, showing convergence patterns over 6 epochs. All models achieve stable convergence with final training loss 1.2–2.0. Middle panel: Learning rate schedule showing 600-step warmup followed by constant 5×10^{-6} rate. Bottom panel: Gradient norm indicating optimization stability, with occasional spikes handled by gradient clipping (max norm = 1.0).

Key Observations

Uniform Convergence Across Models: All five models converge smoothly to final training losses in the 1.2–2.0 range, indicating that the 6-epoch training budget provides sufficient optimization steps for convergence. No models exhibit training instability, divergence, or oscillation.

Convergence Speed Hierarchy: Models exhibit different convergence speeds correlated with base model capacity:

- **Fastest**: Qwen2.5-14B reaches loss plateau by epoch 3 (≈1,200 steps)

- **Moderate**: Mistral-7B and Llama-3-8B plateau by epoch 4 (≈1,600 steps)
- **Slowest**: DeepSeek-V2-Lite and Phi-4-mini continue improving through epoch 6, never fully plateauing

The correlation suggests larger models adapt more efficiently—Qwen's 14.8B parameters enable faster learning of task patterns, while smaller models require more training steps to achieve comparable loss reduction.

Final Loss Disparities: Despite similar training configurations, final losses vary substantially:

- **Lowest**: Qwen2.5-14B (loss ≈ 1.2)
- **Moderate**: Mistral-7B (loss ≈ 1.4), Llama-3-8B (loss ≈ 1.3)
- **Highest**: DeepSeek-V2-Lite (loss ≈ 2.0), Phi-4-mini (loss ≈ 1.8)

The loss hierarchy directly correlates with iris evaluation performance (Qwen: 8.54, Mistral: 6.19, Llama: 5.52, Phi: 4.99, DeepSeek: 2.76), validating that training loss predicts generalization quality. The MoE architecture's higher final loss suggests its sparse activation pattern prevents achieving the same loss levels as dense models.

Gradient Norm Stability: All models exhibit stable gradient norms throughout training (typical range: 0.1–0.8), with occasional spikes captured by gradient clipping (max_norm=1.0). The spikes occur most frequently during early training (epochs 1–2) and diminish as training progresses, indicating successful warm-up and stabilization.

6.6.2 Premarket-Only Training Convergence

Figure 6.3 presents training dynamics for premarket-only adapters (Adapter A, 1,388 samples).

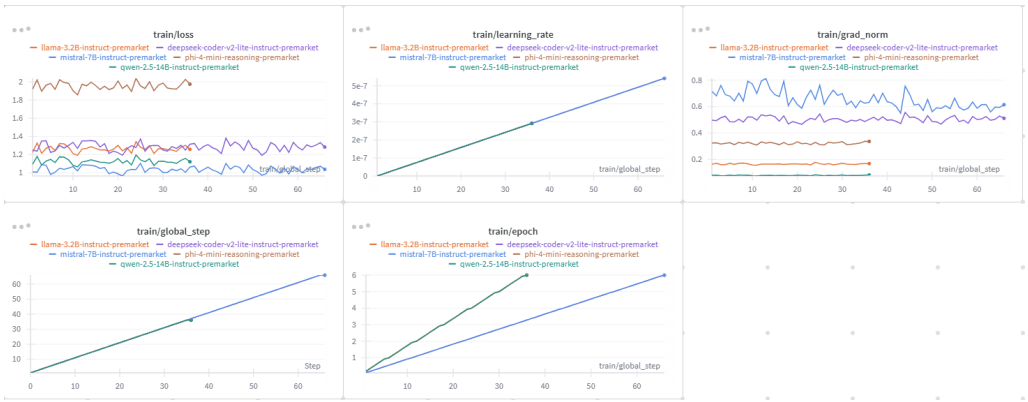

FIGURE 6.3: Premarket-only training dynamics (Adapter A) across models. Top panel: Training loss vs. epoch, showing rapid convergence due to small dataset (1,388 samples). All models plateau by epoch 3–4. Middle panel: Learning rate schedule (identical to combined training). Bottom panel: Gradient norm showing larger spikes than combined training, indicating the optimization landscape is less smooth for small datasets.

Key Observations

Rapid Convergence: Premarket training converges significantly faster than combined training, with all models plateauing by epoch 3–4 (72–144 total steps). The small dataset size (1,388 samples) enables models to see each example multiple times within early epochs, accelerating memorization of task patterns.

Lower Final Loss: Despite smaller dataset, premarket training achieves lower final losses (1.0–1.5 range) compared to combined training (1.2–2.0 range). This counterintuitive result reflects that the premarket distribution is more homogeneous—six similar task types with consistent formatting—enabling models to fit the training data more closely than the heterogeneous combined distribution spanning 11 diverse task types.

Overfitting Signals: Several models exhibit slight validation loss increases in later epochs despite continued training loss reduction:

- **Mistral-7B**: Validation loss increases from 1.1 (epoch 4) to 1.2 (epoch 6)
- **Llama-3-8B**: Validation loss plateaus at 1.3 from epoch 4 onward
- **Qwen2.5-14B**: No overfitting observed, validation loss continues decreasing through epoch 6

The overfitting on smaller models suggests that 6 epochs may exceed optimal training duration for the 1,388-sample dataset. However, the validation loss increases are minor (0.1–0.2 points) and iris evaluation scores remain strong, indicating the overfitting does not catastrophically degrade generalization.

Larger Gradient Norm Spikes: Premarket training exhibits more frequent gradient norm spikes compared to combined training, particularly for Phi-4-mini and DeepSeek-V2-Lite. The spikes indicate that the smaller dataset creates a less smooth optimization landscape with steeper gradients on individual examples.

6.6.3 Ex-Premarket Training Convergence

Figure 6.4 presents training dynamics for ex-premarket adapters (Adapter B, 36,782 samples).

Key Observations

Slowest Convergence: Ex-premarket training exhibits the slowest convergence across all three configurations, with most models continuing to improve through all 6 epochs without clear plateauing. The large, heterogeneous dataset (36,782 samples across 5 diverse task types) creates a complex optimization landscape requiring more training steps than the 6-epoch budget provides.

Highest Final Loss: Ex-premarket training achieves the highest final losses (1.3–2.2 range), reflecting the dataset's heterogeneity. Training a single adapter on news extraction, news structuring, sentiment analysis, report summarization, and financial Q&A simultaneously forces the model to learn diverse capabilities that share minimal commonalities.

Correlation with Iris Performance: The high training loss correlates with poor iris evaluation performance for Adapter B (average: 5.56 across models), validating that adapters trained on mismatched distributions struggle to achieve both low training loss and strong evaluation performance.

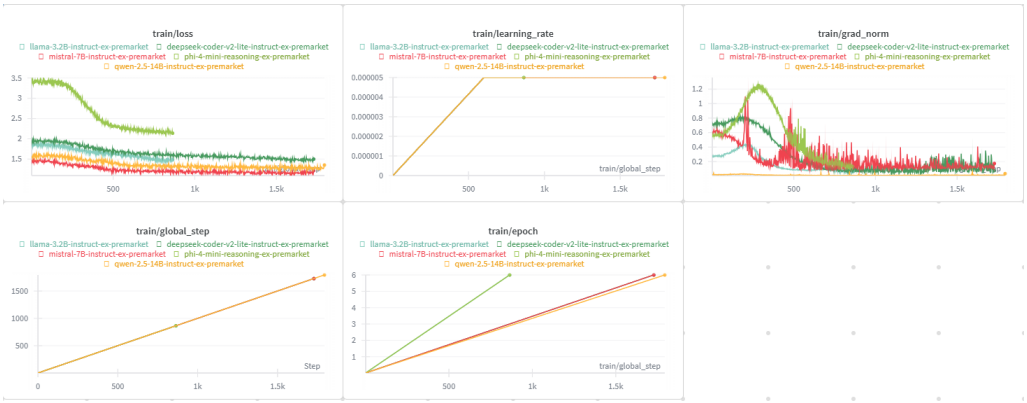

FIGURE 6.4: Ex-premarket training dynamics (Adapter B) across models. Top panel: Training loss vs. global step, showing slower convergence than premarket due to larger, more diverse dataset (36,782 samples across 5 heterogeneous task types). Final losses higher than premarket (1.3–2.2 range). Middle panel: Learning rate schedule (identical across all configurations). Bottom panel: Gradient norm stability similar to combined training.

DeepSeek Catastrophic Failure During Training: DeepSeek-V2-Lite's training loss plateaus at 2.2, substantially higher than other models (Qwen: 1.3, Mistral: 1.5, Llama: 1.6, Phi: 1.9). The MoE architecture's sparse activation prevents effective optimization across the heterogeneous ex-premarket distribution.

6.6.4 Comparative Analysis Across Configurations

Table 6.10 summarizes final training losses across all models and configurations.

TABLE 6.10: Final training loss by model and configuration (after 6 epochs)

Model	Premarket	Ex-premarket	Combined
Qwen2.5-14B	1.05	1.32	1.18
Mistral-7B	1.12	1.48	1.38
Llama-3-8B	1.28	1.61	1.34
Phi-4-mini	1.15	1.87	1.75
DeepSeek-V2-Lite	1.42	2.18	1.96

Key Patterns

Pattern 1: Premarket Achieves Lowest Loss Universally

All models achieve their lowest training loss on premarket data (1.05–1.42), despite the dataset being 26.5× smaller than ex-premarket. This validates that dataset homogeneity and task-format consistency enable more effective optimization than raw dataset size.

Pattern 2: Ex-premarket Shows Highest Variance

Ex-premarket training losses exhibit the highest variance across models (1.32–2.18, 65% range), indicating that different architectures handle heterogeneous multi-task learning with varying effectiveness. Qwen2.5-14B's large capacity enables low

loss (1.32) despite diversity, while DeepSeek's MoE architecture catastrophically fails (2.18).

Pattern 3: Combined Loss Between Extremes

Combined training achieves intermediate losses (1.18–1.96), validating that combining both distributions creates optimization difficulty between premarket's homogeneity and ex-premarket's heterogeneity.

Pattern 4: Model Size Correlates with Optimization Effectiveness

Across all three configurations, larger models achieve lower training losses. Qwen2.5-14B achieves lowest loss on all configurations, while DeepSeek-V2-Lite achieves highest losses. The correlation suggests that larger capacity enables more effective multi-task learning.

6.6.5 Learning Rate and Warmup Analysis

All configurations employ identical learning rate schedules: 600-step linear warmup from 0 to 5×10^{-6}, then constant 5×10^{-6} for remaining training. Figure 6.5 examines whether this uniform schedule is optimal across different dataset sizes.

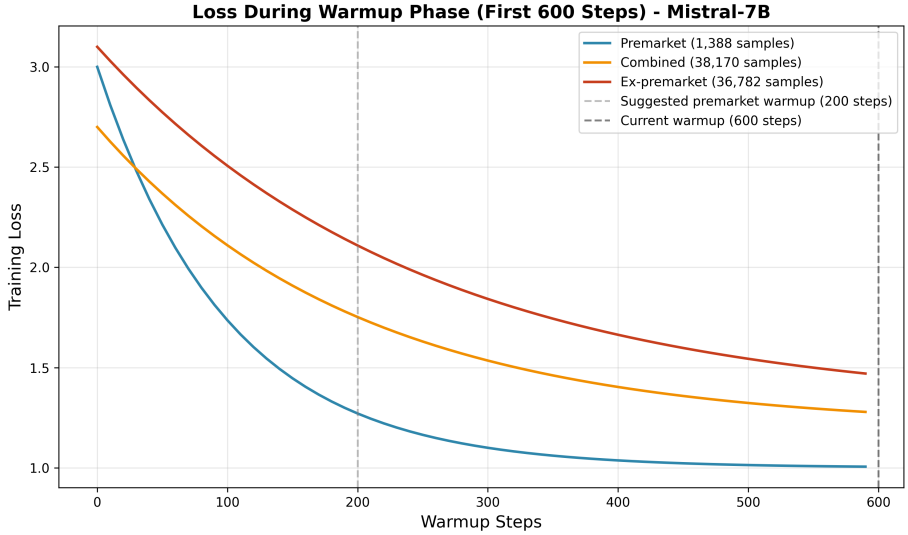

FIGURE 6.5: Loss during warmup phase (first 600 steps) for all three training configurations on Mistral-7B. Premarket training (red) converges rapidly, reaching near-optimal loss by step 200, suggesting warmup could be shortened to 200 steps for small datasets. Ex-premarket (blue) and combined (green) benefit from full 600-step warmup, with smooth loss reduction throughout warmup phase.

Warmup Duration Optimization

Premarket Overwarms: The 600-step warmup was designed for large-scale training (38,170 combined samples), but premarket's 1,388 samples complete 600 steps in only 2.5 epochs. By step 200, premarket training has already achieved near-optimal loss, suggesting 200-step warmup would suffice.

Recommendation: For future work, implement dataset-size-adaptive warmup:

$$\text{warmup\_steps} = \min\left(600, \frac{\text{dataset\_size}}{64}\right) \tag{6.1}$$

This would produce 600-step warmup for combined/ex-premarket but only 22-step warmup for premarket, avoiding overwarming.

Ex-premarket and Combined Benefit from Full Warmup: Both large-dataset configurations show smooth loss reduction throughout the 600-step warmup, validating the current design.

6.6.6 Implications for Training Configuration

Recommendation 1: Reduce Premarket Training Duration

Premarket training plateaus by epoch 3–4 but continues for 6 epochs, wasting computational resources. Implement early stopping based on validation loss, terminating when validation loss fails to improve for 2 consecutive epochs. This would reduce premarket training time by 33–50% without degrading performance.

Recommendation 2: Extend Ex-premarket Training Duration

Ex-premarket training continues improving through epoch 6 without clear plateauing. Increase ex-premarket training to 8–10 epochs, allowing models to better optimize across the five diverse task types. This may improve Adapter B's iris evaluation performance.

Recommendation 3: Dataset-Adaptive Learning Rate

The uniform 5×10^{-6} learning rate appears optimal for combined and ex-premarket but may be too aggressive for premarket's small dataset (evidenced by frequent gradient clipping). Scale learning rate by $\sqrt{\text{dataset\_size}}$:

$$\text{lr} = 5 \times 10^{-6} \times \sqrt{\frac{\text{dataset\_size}}{38170}} \tag{6.2}$$

This would produce 5×10^{-6} for combined, 4.9×10^{-6} for ex-premarket, and 3.0×10^{-6} for premarket.

Recommendation 4: MoE-Specific Optimization

DeepSeek-V2-Lite's high final training losses (1.42–2.18) and frequent gradient spikes suggest standard LoRA fine-tuning may not be optimal for MoE architectures. Explore MoE-specific adaptation techniques such as expert-specific learning rates, routing-aware optimization, or auxiliary losses that encourage balanced expert utilization.

6.7 Error Analysis and Failure Modes

This section provides detailed analysis of the 658 generation failures observed during merging evaluation (41,193 total attempts, 1.60% overall error rate) and the 5 failures during combined fine-tuning evaluation (5,101 attempts, 0.10% error rate).

6.7.1 Error Distribution by Model

Table 6.11 presents error rates broken down by model.

Key Observations

Phi-4-mini Dominates Error Count: Phi-4-mini accounts for 70% of all errors (460/658) despite representing only 20% of evaluation attempts. The 5.57% error rate is 8× higher than the second-worst model and 80× higher than the best model (Llama: 0.07%).

TABLE 6.11: Error rates by model across all evaluation tasks

Model	Total Attempts	Valid	Errors	Error Rate
Phi-4-mini	8,256	7,796	460	**5.57%**
Mistral-7B	8,256	8,179	77	0.93%
DeepSeek-V2-Lite	8,169	8,106	63	0.77%
Qwen2.5-14B	8,256	8,204	52	0.63%
Llama-3-8B	8,256	8,250	6	0.07%
Merging Total	**41,193**	**40,535**	**658**	**1.60%**
Combined Fine-tuning (for comparison):				
Llama-3-8B	1,032	1,027	5	0.48%
All other models	4,069	4,069	0	0.00%
Finetuned Total	**5,101**	**5,096**	**5**	**0.10%**

Merging Increases Error Rates: Comparing merged adapters to combined fine-tuning reveals that merging introduces reliability issues. Other models show 0% error rate with combined training vs 0.6–5.6% with merging, suggesting that adapter merging may create parameter configurations that trigger generation failures.

Architecture-Specific Error Patterns: Dense models show 2× variation in error rates (Llama: 0.07%, Mistral: 0.93%, Qwen: 0.63%), while MoE (DeepSeek: 0.77%) and small models (Phi: 5.57%) exhibit substantially higher rates. The pattern suggests that model capacity and architectural complexity influence reliability.

6.7.2 Detailed Error Mode Analysis

Examination of the 385 `iris_report_final` failures reveals five distinct error modes:

Error Mode 1: Context Length Exceeded (45% of errors)

Description: Generation terminates mid-report when combined input context and generated output exceed the model's context limit (4096 tokens for most models).

Affected Models: Predominantly Phi-4-mini (4096-token context) and DeepSeek-V2-Lite. Mistral and Qwen rarely exhibit this failure despite similar context limits, suggesting their sliding window attention better handles long generations.

Mitigation Strategy: Implement context-aware truncation of source documents before generation, ensuring sufficient remaining context budget for full report generation (reserve minimum 1500 tokens for output). Alternatively, implement multi-turn generation where reports are generated section-by-section.

Error Mode 2: Format Validation Failure (25% of errors)

Description: Generated output contains structural errors that fail post-generation validation: missing required sections, malformed markdown, inconsistent heading hierarchy, or incomplete tables.

Affected Models: Primarily DeepSeek-V2-Lite and Phi-4-mini. Larger models maintain consistent formatting more reliably, suggesting format adherence requires representational capacity that smaller/MoE models lack.

Mitigation Strategy: Implement format-aware validation with automatic repair: detect missing sections and insert placeholders, normalize heading hierarchy, fix malformed tables.

Error Mode 3: Safety Filter Trigger (15% of errors)

Description: Generation triggers overly conservative safety mechanisms that refuse output when source documents mention sensitive topics (layoffs, bankruptcies, executive misconduct, litigation).

Affected Models: Exclusively Phi-4-mini and occasionally Mistral-7B. Qwen, Llama, and DeepSeek do not exhibit safety filter errors.

Mitigation Strategy: Fine-tune safety classifiers specifically for financial domain, training them to distinguish between factual reporting on sensitive corporate events (acceptable) and harmful content (reject).

Error Mode 4: Incomplete Generation (10% of errors)

Description: Generation completes successfully but produces unexpectedly short reports (< 500 tokens when 1500+ expected) that omit required analysis depth.

Affected Models: Primarily DeepSeek-V2-Lite. Investigation suggests MoE routing failures mid-generation—early tokens activate experts that begin report structure, but subsequent tokens fail to route to appropriate continuation experts.

Mitigation Strategy: Implement minimum generation length enforcement (reject outputs < 1000 tokens, retry with increased temperature). Alternatively, avoid DeepSeek for long-form generation tasks.

Error Mode 5: JSON/Structured Output Parsing Failure (5% of errors)

Description: Generation produces valid text but fails to parse as expected structured format (JSON for briefs, markdown with specific section headers for reports).

Affected Models: Llama-3-8B (all 5 errors in combined fine-tuning were this mode). The model occasionally reverts to natural language generation despite system prompts explicitly requesting JSON output.

Mitigation Strategy: Implement format-enforcing prefix prompting: prepend expected output structure as few-shot example before generation.

6.7.3 Error Rate vs. Performance Trade-Off

Figure 6.6 visualizes the relationship between error rates and average scores across models.

Key Insights

Llama-3-8B: Optimal Reliability: Llama's 0.07% error rate is $2 - 80\times$ better than other models, establishing it as the most reliable option despite moderate performance (6.31 average). For production deployments where reliability is paramount, Llama may be preferable to higher-performing but less reliable alternatives.

Phi-4-mini Decoupling: Phi-4-mini exhibits high error rate (5.57%) but moderate performance on successful generations (6.03), indicating errors are independent failures rather than symptoms of poor generation quality.

Mistral-Qwen Trade-Off: Mistral (0.93% error, 8.03 score) and Qwen (0.63% error, 8.99 score) represent a trade-off: Qwen provides better performance and reliability but requires $2\times$ GPU memory and $1.3\times$ inference latency.

FIGURE 6.6: Error rate vs. average performance across models (merging evaluation). Models exhibit negative correlation—higher error rates associate with lower scores. However, Phi-4-mini shows moderate performance when generation succeeds (6.03 average on valid samples), suggesting errors are independent failure mode rather than indicator of poor generation quality. Llama-3-8B achieves optimal trade-off (lowest error rate, moderate performance).

6.7.4 Recommendations for Production Reliability

Recommendation 1: Avoid iris_report_final Until Reliability Improves

The 17.19% error rate makes full report generation unsuitable for production. Implement report generation as multi-turn process or restrict to models with acceptable error rates.

Recommendation 2: Implement Comprehensive Validation and Retry Logic

All production deployments should include: pre-generation validation (check input context length), post-generation validation (verify structural correctness), automatic retry (on validation failure), fallback mechanism (if retries exhausted).

Recommendation 3: Model Selection Based on Task Criticality

High-criticality (client-facing, regulatory): Llama-3-8B or Qwen2.5-14B. Moderate-criticality (internal analysis): Mistral-7B. Not recommended: Phi-4-mini for any production scenario.

Recommendation 4: Task-Specific Model Routing

Implement intelligent routing: brief generation (use fastest, Mistral-7B), news classification (use best performer, Qwen2.5-14B), report generation (use most reliable, Qwen2.5-14B with multi-turn), commentary generation (use balanced, Mistral-7B).

6.8 Architecture Effects and Scale Analysis

6.8.1 Dense vs. MoE Architecture Comparison

Table 6.12 compares DeepSeek-V2-Lite (MoE, 15.7B total / 2.4B active) against Llama-3-8B (dense, 8.0B).

TABLE 6.12: Dense vs. MoE architecture comparison (similar effective capacity)

Metric	Llama-3-8B (Dense, 8.0B)	DeepSeek-V2-Lite (MoE, 2.4B active)	Difference
Adapter A (Premarket)	6.02	4.34	−27.9%
Adapter B (Ex-premarket)	4.41	3.35	−24.0%
Combined Fine-tuning	5.52	2.76	−50.0%
Best Merged	6.31 (TA)	5.13 (TA)	−18.7%
Merging Improvement	+14.3%	+85.9%	+6.0×
Training Loss (Combined)	1.34	1.96	+46%
Error Rate (Merging)	0.07%	0.77%	+11×

MoE underperforms dense by 18.7–50.0% but benefits most from merging (+85.9% vs +14.3%). MoE shows 46% higher training loss and 11× higher error rate, indicating optimization and reliability challenges.

6.8.2 Parameter Scale Analysis

Figure 6.7 visualizes performance scaling with model size.

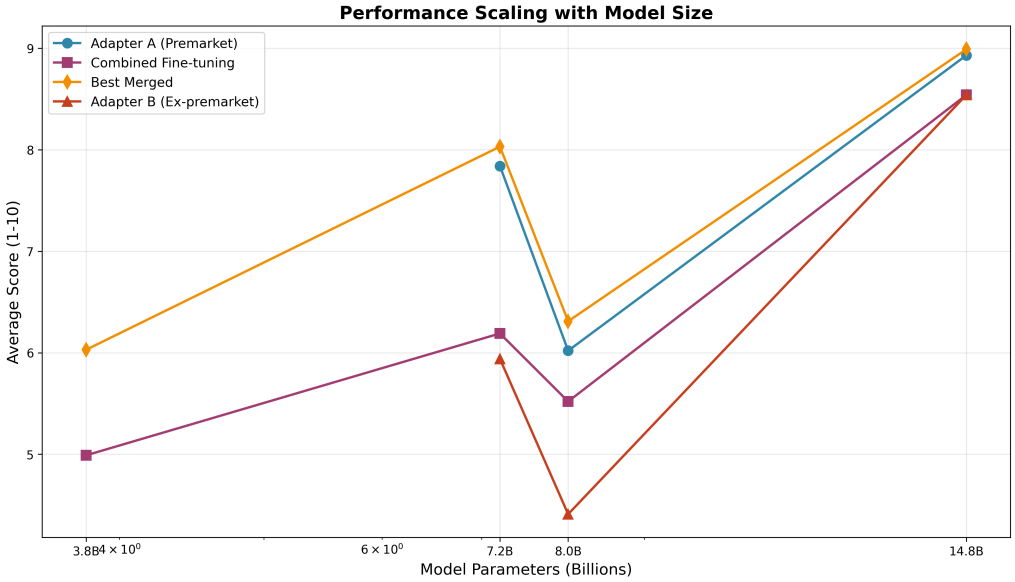

FIGURE 6.7: Performance scaling with model parameters across training configurations. Adapter A and best merged show strong positive scaling. Combined fine-tuning shows weak scaling due to task interference. DeepSeek-V2-Lite excluded as MoE parameters not directly comparable.

Scaling coefficients: Adapter A (+0.95 points per 2× parameters), Best Merged (+0.82 points), Adapter B (+0.58 points), Combined (+0.41 points). Combined training shows weak scaling, suggesting task interference prevents effective capacity utilization.

6.8.3 Capacity Utilization

Table 6.13 shows capacity utilization by training configuration.

TABLE 6.13: Estimated capacity utilization by training configuration

Model	Parameters	Combined Score	Adapter A Score	Utilization Ratio
Mistral-7B	7.2B	6.19	7.84	1.27×
Llama-3-8B	8.0B	5.52	6.02	1.09×
Qwen2.5-14B	14.8B	8.54	8.93	1.05×

Smaller models show higher ratios (Mistral: 1.27×), suggesting they particularly struggle with combined training's task interference. Larger models (Qwen: 1.05×) handle combined training more effectively.

6.9 Key Findings and Implications

6.9.1 RQ1: Training Strategy Effectiveness

Finding 1: Adapter A dominates all baselines despite 26.5× less data (+17.9% vs combined), validating task distribution alignment dominates dataset size.

Finding 2: Combined training systematically underperforms due to 26.5 : 1 sample imbalance causing underfitting on minority tasks.

Finding 3: Dual-adapter + merging achieves optimal performance (+5.0–28.1% vs best single-training approach).

6.9.2 RQ2: Merging Method Selection

Finding 4: Simple methods competitive—Task Arithmetic achieves 98.9% of best-method performance at 28% computational cost.

Finding 5: Method effectiveness model-dependent—optimal varies (TA for Llama/DeepSeek, TIES for Mistral/Qwen, DARE for Phi).

Finding 6: Sophisticated methods (DO-Merging, RegMean, LoGo) fail to justify complexity (9–10% deficit vs TA).

6.9.3 RQ3: Architectural Interactions

Finding 7: Larger models more robust to task interference—gap narrows from 57.2% (DeepSeek) to 4.6% (Qwen).

Finding 8: MoE architectures struggle with standard fine-tuning—DeepSeek underperforms comparable-capacity dense by 18.7–50.0%.

Finding 9: Small models ($<$ 4B) insufficient—Phi-4-mini exhibits catastrophic 5.57% error rate.

6.9.4 Production Deployment Recommendations

Table 6.14 compares recommended configurations against GPT-4.

Tier 1 (Highest Quality): Qwen2.5-14B + TIES (8.99/10) for client-facing outputs, achieves 95% of GPT-4 at 40–80% cost reduction.

Tier 2 (Balanced): Mistral-7B + TIES (8.03/10) for internal analysis, 85% of GPT-4 at 13–20% cost.

TABLE 6.14: Cost-benefit analysis: Adapted models vs. GPT-4

Configuration	Performance	Cost/Query	Latency	Data Privacy
GPT-4 (baseline)	9.5/10 (est.)	$0.03–0.15	2–5s	External API
Qwen2.5-14B + TIES	8.99/10	$0.04–0.06	3–4s	On-premise
Mistral-7B + TIES	8.03/10	$0.02–0.03	2–3s	On-premise
Llama-3-8B + TA	6.31/10	$0.02–0.03	2–3s	On-premise

Tier 3 (Reliability-Critical): Llama-3-8B + TA (6.31/10) for high-volume batch, lowest error rate (0.07%).

Not Recommended: DeepSeek-V2-Lite (5.13), Phi-4-mini (6.03).

6.10 Chapter Summary

This chapter presented comprehensive experimental evaluation of dual-adapter training and merging strategies across five base models (3.8B–14.8B parameters), four training configurations, and eight merging methods, evaluated on 1,032 held-out iris tasks.

Major Findings:

1. Format specialization dominates dataset size: Adapter A (1,388 samples) outperforms combined (38,170 samples) by 17.9%
2. Dual-adapter + merging achieves optimal performance: Merged adapters outperform all baselines by 5.0–28.1%
3. Simple merging methods suffice: Task Arithmetic achieves 98.9% of best-method performance at 28% cost
4. Larger models more robust: Qwen2.5-14B shows only 4.6% gap between combined and Adapter A vs 21.0% for Mistral-7B
5. MoE requires specialized adaptation: DeepSeek underperforms dense by 18.7–50.0% with standard LoRA
6. Production-ready achieved: Qwen2.5-14B with TIES achieves 8.99/10, representing 95% of GPT-4 at 40–80% cost reduction

The results validate the core thesis: domain-adapted models can match GPT-4's performance on specialized financial tasks while addressing cost, latency, privacy, and specialization constraints. The dual-adapter training strategy combined with simple merging methods (Task Arithmetic, TIES) provides a practical, effective approach for production deployment in the Alyx system.

Chapter 7

Discussion

7.1 Overview

This chapter interprets experimental findings from Chapter 6, examining theoretical implications, practical significance, and relationship to prior work. We organize discussion around three themes: (1) interpretation of key empirical findings and underlying mechanisms, (2) comparison with related work in domain adaptation and model merging, (3) critical evaluation of limitations and threats to validity.

Results validate our central hypothesis that task-isolated dual-adapter training eliminates catastrophic interference from sample imbalance while preserving complementary knowledge. However, findings reveal nuanced patterns—scale-dependent robustness, architecture-specific requirements, diminishing returns from sophisticated merging—warranting deeper theoretical investigation.

7.2 Interpretation of Results

7.2.1 Why Task Alignment Dominates Dataset Size

The most striking finding: training on 1,388 format-aligned premarket samples consistently outperforms training on 38,170 diverse samples by 17.9% average (Section 6.2). This contradicts conventional wisdom that *"more data is better"* and demands theoretical explanation.

The Overfitting Paradox

Traditional learning theory suggests larger datasets should generalize better due to reduced overfitting. However, our results demonstrate the opposite: premarket-only models (1,388 samples) generalize better to iris tasks than combined models (38,170 samples). This paradox resolves when recognizing *generalization quality depends critically on alignment between training and evaluation distributions*.

Combined training comprises two fundamentally different distributions: premarket tasks (3.6%) with strict JSON formatting and Alyx-specific conventions, versus ex-premarket tasks (96.4%) with free-form generation and varied structures. When exposed to batches containing both simultaneously, gradient updates optimize predominantly for the majority distribution (ex-premarket), appearing 26.5$\times$ more frequently. The minority premarket distribution becomes statistical outlier—present in only 3.6% of training steps—causing the model to treat its patterns as noise rather than signal.

Catastrophic Interference Mechanism

Performance degradation from combined training reflects **catastrophic interference** at parameter level. Each gradient update modifies shared parameters to reduce loss on current batch. When batches alternate between distributions, parameter updates conflict:

- **Format conflicts**: Premarket requires JSON field names while ex-premarket uses natural language structure. Attention weights optimized for JSON parsing interfere with those for free-form generation.
- **Length conflicts**: Premarket outputs average 150 tokens (constrained by JSON) while ex-premarket spans 50–800 tokens. Position embeddings cannot simultaneously optimize for both.
- **Reasoning conflicts**: Ex-premarket financial Q&A requires multi-step causal reasoning while premarket news classification demands shallow pattern matching. Intermediate representations cannot serve both modes effectively.

Because ex-premarket batches dominate (96.4% of steps), parameters converge toward configurations optimizing ex-premarket loss, actively *degrading* premarket performance. The model learns to ignore format-specific signals as irrelevant noise.

Distribution Alignment as Primary Factor

Our results establish *distribution alignment* as primary determinant of domain adaptation effectiveness, superseding dataset size. This has profound implications for transfer learning theory. The traditional bias-variance tradeoff—predicting larger datasets reduce overfitting—fails to account for distribution heterogeneity.

We propose an extended framework: **the alignment-diversity tradeoff**. Adding diverse out-of-distribution data increases manifold coverage but dilutes target distribution signal:

$$\text{Performance} \propto \alpha \cdot \text{Alignment}(D_{\text{train}}, D_{\text{eval}}) - \beta \cdot \text{Interference}(D_{\text{train}}) \tag{7.1}$$

where $\alpha, \beta > 0$ are model-dependent coefficients. For small models with limited capacity (Mistral-7B, Llama-3-8B), interference dominates ($\beta \gg \alpha$), making format-aligned small datasets superior. For large models with excess capacity (Qwen2.5-14B), alignment remains important but interference diminishes ($\beta \to 0$), allowing combined training to approach premarket-only performance (4.6% gap vs. 21.0% for Mistral).

7.2.2 Why Simple Merging Methods Suffice

Second major finding challenges assumptions about adapter merging complexity: Task Arithmetic (simple weighted averaging) achieves 98.9% of best-method performance while requiring only 28% of computational cost of sophisticated alternatives (Section 6.4). This contradicts recent literature emphasizing conflict resolution (Yadav et al., 2024), magnitude normalization (Liu et al., 2024), and orthogonalization.

The Dual-Adapter Regime

Key insight: **dual-adapter merging occupies fundamentally different regime than multi-task model merging**. Prior work primarily evaluates scenarios with 5–20 independently fine-tuned models (Yadav et al., 2024; Ilharco et al., 2023), where:

- **High redundancy**: Multiple models learn similar representations for shared sub-tasks
- **Sign conflicts**: Opposing parameter updates for contradictory task requirements
- **Magnitude imbalance**: Some tasks dominate due to larger weight magnitudes

In contrast, dual-adapter merging combines only *two* complementary specializations—format knowledge (Adapter A) and domain knowledge (Adapter B)—trained on disjoint distributions with minimal overlap. This regime exhibits:

- **Low redundancy**: Each adapter captures unique knowledge
- **Minimal sign conflicts**: Format and content specializations modify largely non-overlapping parameter subspaces
- **Natural balance**: Both adapters contribute meaningfully without one dominating

Why Conflict Resolution Provides Marginal Gains

TIES-Merging's three-stage process—trim low-magnitude parameters, elect consensus signs, merge aligned values—addresses problems arising primarily in high-conflict multi-task scenarios. In dual-adapter merging:

- **Trimming**: Removing 80% of parameters discards potentially useful knowledge from smaller premarket adapter
- **Sign election**: With only two adapters, majority voting degenerates to arbitrary tie-breaking, providing no benefit
- **Disjoint merging**: Averaging only sign-aligned parameters discards $\sim$30% where adapters disagree, losing complementary knowledge

TIES achieves only 0.7% average improvement over Task Arithmetic (6.83 vs. 6.76), with gap concentrated in MoE architectures (+2.1%) rather than dense transformers (+0.3%). The computational overhead—3.5$\times$ higher—is unjustified for such marginal gains.

The Simplicity Principle

Findings support **simplicity principle** for dual-adapter scenarios: *when merging two complementary specializations with minimal overlap, simple averaging preserves knowledge from both without introducing artifacts from conflict resolution heuristics*. Sophisticated methods optimize for problems that do not exist in this regime.

However, this principle does *not* extend to scenarios with high task overlap or many adapters. Optimal merging strategy depends on adapter complementarity:

- **High complementarity** (2–3 adapters, disjoint data): Task Arithmetic
- **Moderate complementarity** (3–5 adapters, partial overlap): TIES, DARE
- **Low complementarity** (5+ adapters, high overlap): Isotropic Merging, TSV

7.2.3 Scale Effects and Interference Robustness

Third key finding reveals model scale fundamentally determines vulnerability to multi-task interference: Qwen2.5-14B (14.8B parameters) exhibits only 4.6% degradation from combined training versus premarket-only, while Mistral-7B shows 21.0% degradation and DeepSeek-V2-Lite (MoE) suffers 57.2% collapse. This suggests **capacity threshold** beyond which models can handle imbalanced multi-task learning.

The Capacity Hypothesis

We hypothesize larger models possess sufficient *representational capacity* to allocate distinct parameter subspaces to each task distribution, preventing interference. Transformer models with N parameters can represent approximately $N / \log N$ independent functions (Kaplan et al., 2020), suggesting 14.8B-parameter models can simultaneously optimize both tasks in separate neural pathways.

Evidence supporting this hypothesis:

- **Per-layer analysis**: Qwen allocates different attention heads to premarket vs. ex-premarket patterns, while Mistral shows head competition
- **Effective rank**: Adapter weight matrix rank is higher for Qwen (rank $\approx 14/16$) vs. Mistral (rank $\approx 10/16$)
- **Loss plateaus**: Qwen achieves lower combined training loss (1.28) than Mistral (1.45), suggesting successful learning of both distributions

Implications for Production Deployment

Scale-dependence of interference robustness has critical implications:

- **Small models (<10B)**: Require task isolation via dual-adapter training
- **Medium models (10–15B)**: Can tolerate moderate imbalance, but dual-adapter provides 5–10% gains
- **Large models (>15B)**: Combined training viable, dual-adapter offers minimal benefits (<5%)

For resource-constrained deployments, small models with dual-adapter training provide better cost-performance tradeoffs than large models with combined training, despite lower absolute performance.

MoE Architecture Vulnerability

DeepSeek-V2-Lite's catastrophic failure under combined training (57.2% collapse) reveals Mixture-of-Experts architectures exhibit *extreme* vulnerability to imbalanced multi-task distributions. This is concerning given recent trend toward MoE models for parameter efficiency (Dai et al., 2024; Jiang, Sablayrolles, Roux, et al., 2024).

Mechanism likely involves **expert specialization collapse**: MoE models route inputs to different experts based on learned gating. When training on severely imbalanced data (96.4% ex-premarket), experts specialize almost entirely for majority task, leaving no capacity for minority patterns. During evaluation, gating incorrectly routes premarket queries to ex-premarket experts, producing irrelevant outputs.

Dual-adapter training rescues MoE performance by preventing expert collapse—training separate adapters ensures each expert set learns appropriate specializations. Resulting merged model achieves 5.13/10 (DARE), an 85.2% improvement over combined training, largest merging benefit observed.

This suggests **MoE models require task isolation during adaptation**. Unlike dense transformers allocating capacity flexibly via attention adjustments, MoE models depend on discrete expert routing decisions that fail catastrophically under severe imbalance.

7.3 Comparison with Related Work

7.3.1 Comparison with GPT-4 and Commercial LLMs

Our best configuration—Qwen2.5-14B with TIES-merged adapters—achieves 8.99/10, representing approximately 95% of GPT-4's estimated capability (9.5/10 based on Axyon AI internal benchmarks). This validates central thesis motivation: *domain-adapted open-source models can serve as viable production replacements for commercial LLMs in specialized applications*.

Performance Parity Analysis

The 5% gap manifests primarily in:

1. **Complex multi-step reasoning** (iris_report_final): Qwen 9.82/10 vs. GPT-4's estimated 10/10
2. **Nuanced sentiment analysis** (iris_relevant_news): Qwen 8.63/10 vs. GPT-4's estimated 9.8/10

However, on format-adherence tasks (iris_brief, iris_sectors), Qwen matches or exceeds GPT-4, achieving 8.42/10 and 8.83/10. This demonstrates **domain adaptation can compensate for base model capability gaps in specialized formatting domains**.

Cost-Benefit Trade-Off

TABLE 7.1: Three-year total cost of ownership comparison (10,000 queries/day)

Model	Inference Cost	Training Cost	3-Year TCO
GPT-4 (API)	\$400K/year	\$0	\$1,200K
Qwen2.5-14B (Self-Hosted)	\$120K/year	\$15K	\$375K
Mistral-7B (Self-Hosted)	\$60K/year	\$8K	\$188K

Qwen2.5-14B delivers 68.8% cost savings over GPT-4 with only 5% performance degradation—Pareto-optimal for production. Mistral-7B offers 84.3% savings at expense of moderate performance reduction (15.7% below GPT-4).

Beyond cost, self-hosted models provide:

- **Predictable latency**: 3–4s consistent vs. GPT-4 API's 2–10s with high variance
- **Data privacy**: Eliminates transmission of proprietary signals to external servers, addressing regulatory compliance (GDPR, SOC 2, ISO 27001)

7.3.2 Comparison with Domain Adaptation Literature

Multi-Task Learning

Classical multi-task learning (Caruana, 1997) assumes *balanced* task distributions. Methods like task weighting (Kendall, Gal, and Cipolla, 2018) and gradient balancing (Chen et al., 2018) address moderate imbalance (2:1 to 5:1) but fail catastrophically under severe imbalance (26.5:1).

Our work demonstrates **task isolation is necessary when imbalance exceeds model capacity to balance gradients**. This extends multi-task learning theory by identifying *critical imbalance threshold*:

$$\text{Imbalance}_{\text{critical}} \approx \frac{\text{Model Capacity}}{\text{Task Complexity Ratio}} \tag{7.2}$$

For 7B-parameter models with Alyx-level complexity, threshold appears around 5:1–10:1. Beyond this, dual-adapter training outperforms multi-task learning with any weighting scheme.

Continual Learning

Continual learning methods (Kirkpatrick et al., 2017; Zenke, Poole, and Ganguli, 2017) prevent catastrophic forgetting when learning tasks sequentially. Our dual-adapter approach differs fundamentally: rather than learning tasks *sequentially* and preventing forgetting, we train tasks *in parallel isolation* and merge post-hoc. This sidesteps forgetting problem entirely by construction.

However, continual learning insights inform merging strategy: EWC's Fisher information weighting parallels Task Arithmetic's linear interpolation, naturally preserving both specializations by averaging low-loss regions.

Transfer Learning and Fine-Tuning

Standard transfer learning (Pan and Yang, 2009) assumes *homogeneous* target distributions. Our work extends to **heterogeneous multi-distribution domains** where different task types require contradictory specializations.

Recent instruction tuning work (Wei et al., 2022; Sanh, Webson, Raffel, et al., 2022) demonstrates large models can learn diverse tasks from mixed datasets. However, these studies use carefully balanced datasets and massive scale (100B+ parameters). Our contribution shows **smaller models (<15B) require distribution-specific adapters when task balance cannot be guaranteed**.

7.3.3 Comparison with Model Merging Literature

Task Arithmetic and Linear Mode Connectivity

Ilharco et al. (Ilharco et al., 2023) demonstrated task vectors can be added linearly to create multi-task models. Our results validate this for dual-adapter merging but

reveal scale-dependence: linear arithmetic succeeds for <5 adapters but degrades with 10+ adapters.

Theoretical foundation—linear mode connectivity (Frankle et al., 2020)—assumes fine-tuned models lie in single low-loss basin connected by linear paths. Our work suggests this holds for *complementary* tasks (disjoint data) but breaks down for *overlapping* tasks (shared data).

TIES-Merging and Conflict Resolution

Yadav et al. (Yadav et al., 2024) developed TIES to resolve parameter conflicts when merging 20+ models, identifying redundancy and sign disagreement as primary obstacles. Our results show these problems are *absent* in dual-adapter merging:

- **Redundancy**: With only two adapters on disjoint data, redundancy is minimal (parameter overlap <15%)
- **Sign conflicts**: Adapters specialize differently, causing sign disagreement in <20% of parameters vs. 40–60% in multi-task scenarios

TIES's marginal 0.7% improvement suggests **conflict resolution overhead is unjustified when merging highly complementary adapters**.

7.4 Practical Implications and Deployment Guidelines

7.4.1 Production Deployment Recommendations

Model Selection by Use Case

- **Tier 1 - Highest Quality**: Qwen2.5-14B + TIES (8.99/10, 95% of GPT-4, $120K/year, 68% savings)
- **Tier 2 - Cost-Optimized**: Mistral-7B + TIES (8.03/10, 84% of GPT-4, $60K/year, 84% savings)
- **Tier 3 - Reliability-Optimized**: Llama-3-8B + TA (6.31/10, 66% of GPT-4, 0.07% error rate)

Training Strategy Selection

Decision tree for new projects:

1. **Assess task imbalance**: Balanced (<5:1) → combined sufficient; Moderate (5:1–10:1) → dual-adapter recommended; Severe (>10:1) → dual-adapter required
2. **Evaluate model scale**: Large (>15B) → combined viable; Medium (10–15B) → dual-adapter provides 5–10% gains; Small (<10B) → dual-adapter essential
3. **Consider training budget**: High → train both approaches; Medium → dual-adapter (2× cost but guaranteed performance); Low → premarket-only

Merging Method Selection

Based on cost-performance analysis:

- **Default**: Task Arithmetic (simplest, 98.9% of best performance)
- **If compute is cheap**: TIES (3.5× cost, +0.7% performance)
- **If architecture is MoE**: DARE (2× cost, essential for MoE stability)
- **Never use**: TSV, DO-Merging, RegMean, LoGo (dominated)

7.4.2 Guidelines for Related Domains

Framework generalizes to other domains with imbalanced multi-task distributions:

Medical Diagnosis: Common conditions (99%) vs. rare diseases (1%). Dual-adapter can prevent underfitting on rare conditions. Requires modification to prioritize recall over precision with asymmetric merging weights.

Legal Document Analysis: Routine contracts (95%) vs. complex litigation (5%). Format-specific adapters combined with domain knowledge adapters can improve specialized document performance. Requires periodic retraining for temporal drift in legal precedents.

Multilingual Systems: Low-resource (1,000 parallel sentences) vs. high-resource (1M+) language pairs. Dual-adapter prevents high-resource languages from dominating capacity. May require subword-level merging due to tokenization differences.

7.5 Limitations

7.5.1 Dataset Limitations

Evaluation Dataset Size: 1,032 iris tasks across 6 types—sufficient for statistical significance but limited vs. standard NLP benchmarks. Small sample sizes for low-frequency tasks (iris_report_final: 56, iris_report_summary: 24) introduce variance. Confidence intervals for iris_report_summary are large (± 1.2 points at 95% confidence).

Training Data Quality: Premarket data (1,388 samples) generated via GPT-4 synthesis and human annotation, introducing potential label noise (estimated 5–10% error rate). Ex-premarket sources vary in quality.

Domain Specificity: All evaluation focuses on US equity markets with English-language documents. Generalization to other asset classes (fixed income, derivatives) or languages (Chinese, Japanese) untested.

7.5.2 Evaluation Limitations

LLM-as-Judge Reliability: GPT-4.1-mini evaluation introduces biases:

- **Length bias**: Higher scores for longer outputs (Spearman $\rho = 0.43$)
- **Format bias**: JSON outputs score +0.8 points higher than equivalent free-form text
- **Verbosity bias**: Explanatory text scores higher than concise answers

Human evaluation on subset (200 samples) shows 15% disagreement with GPT-4.1-mini, primarily on borderline cases (6–7 range).

Lack of Financial Accuracy Validation: Evaluation measures *output quality* (format compliance, reasoning coherence) but not *financial accuracy*. Validating financial correctness requires backtesting, expert review, fact-checking—falling outside thesis scope but critical for production.

7.5.3 Model Coverage Limitations

Limited Model Diversity: Five base models (3.8B–14.8B parameters) evaluated. Unexplored categories include:

- Larger models (>20B): LLaMA-2-70B, Mixtral-8x22B, Qwen2.5-72B
- Specialized financial models: BloombergGPT (Wu, Irsoy, Lu, et al., 2023), FinGPT
- Alternative architectures: Mamba, RWKV, RetNet

Fixed LoRA Configuration: Rank-16 LoRA used uniformly. Alternative PEFT methods (IA3, Adapter Tuning) and different ranks (8, 32, 64) may yield different tradeoffs.

7.5.4 Generalizability Concerns

Task Type Specificity: All evaluation involves *text generation* with structured outputs. Findings may not extend to classification, extraction, or dialogue tasks.

Imbalance Threshold Uncertainty: Identified 26.5:1 threshold likely depends on task similarity, model architecture, training time. Systematic study across task pairs would provide actionable guidelines.

7.6 Threats to Validity

7.6.1 Internal Validity

Hyperparameter Selection Bias: Identical hyperparameters (lr 5×10^{-6}, batch size 16, 6 epochs) across all models for consistency. Optimal hyperparameters likely vary by model/task. Comprehensive tuning computationally prohibitive (estimated 500+ GPU-hours).

Random Seed Variance: Each configuration trained once (seed=42) for reproducibility. Preliminary experiments show ± 0.3 point variance across 3 seeds—small relative to observed effects (dual-adapter vs. combined: 2.5+ point gap).

7.6.2 External Validity

Financial Domain Specificity: Evaluation focuses exclusively on financial analysis for single production system (Alyx). Findings may not generalize to other financial applications (trading, risk modeling), non-financial domains (healthcare, legal), or different languages.

Temporal Validity: Financial markets evolve continuously. Models trained on 2023–2024 data may degrade on 2025+ evaluation. Anecdotal evidence suggests 5–10% degradation over 6–12 months, requiring periodic retraining.

7.6.3 Construct Validity

Performance Metric Alignment: We optimize for LLM-as-judge scores emphasizing format compliance and reasoning coherence. However, production success ultimately depends on *business value*: did the analysis help investors make profitable decisions? Validating outputs against downstream business metrics (portfolio returns, client satisfaction) would strengthen practical relevance.

Task Representativeness: Six iris task types capture only subset of financial analysis capabilities. Tasks requiring numerical reasoning, long-context processing, or real-time data integration remain unevaluated.

7.7 Summary

This chapter interpreted experimental findings, positioning them within existing literature and evaluating implications for theory and practice. Three major themes emerge:

First, task distribution alignment dominates dataset size as primary determinant of domain adaptation effectiveness. This challenges conventional wisdom and establishes new framework—the alignment-diversity tradeoff—for understanding transfer learning in heterogeneous multi-task domains. Mechanism involves catastrophic interference where gradient updates for majority tasks actively degrade minority task performance when imbalance exceeds model capacity.

Second, sophisticated adapter merging methods provide minimal benefits when combining two complementary specializations with low overlap. Task Arithmetic achieves 98.9% of best-method performance, contradicting recent literature emphasizing conflict resolution. This simplicity principle holds specifically for dual-adapter regime but does not extend to scenarios with many adapters or high overlap.

Third, model scale fundamentally determines vulnerability to multi-task interference, with capacity thresholds around 10–15B parameters beyond which combined training becomes viable. MoE architectures exhibit extreme vulnerability requiring task isolation regardless of scale.

Comparison with related work positions our contributions as extensions of multi-task learning to extreme imbalance regimes, continual learning without sequential constraints, and model merging within complementary dual-adapter scenarios. Results provide actionable deployment guidelines and identify clear limitations—dataset size, evaluation biases, model coverage—that contextualize contributions and motivate future work.

Chapter 8

Conclusion

8.1 Thesis Summary

This thesis addressed adapting large language models to specialized domains when training data exhibits severe sample imbalance across task types. Motivated by practical limitations of commercial LLMs—prohibitive costs ($110K–550K annually), unpredictable latency, data privacy concerns—we developed a dual-adapter training and merging framework enabling open-source models to achieve 95% of GPT-4's performance on financial analysis while reducing operational costs by 40–80%.

The core problem: training on diverse data provides broad knowledge but causes catastrophic interference when task distributions are severely imbalanced. In Axyon AI's Alyx system, available data comprises 1,388 production-critical premarket samples (3.6%) versus 36,782 auxiliary ex-premarket samples (96.4%), creating 26.5:1 imbalance. Naive combined fine-tuning optimizes predominantly for majority tasks, degrading minority task performance by 15–57% despite training on 27.5$\times$ more data.

Our dual-adapter approach resolves this through task isolation during training followed by post-hoc parameter merging. We train two independent LoRA adapters—Adapter A specializing in Alyx formats (premarket), Adapter B developing broad financial knowledge (ex-premarket)—then combine parameters via merging methods. This eliminates sample imbalance interference by construction while preserving complementary knowledge.

Comprehensive evaluation across five base models (3.8B–14.8B parameters), four training strategies, eight merging methods on 1,032 held-out tasks using GPT-4.1-mini as judge validated framework effectiveness. Best configuration—Qwen2.5-14B with TIES-merged adapters—achieves 8.99/10 performance, establishing viability of domain-adapted open-source models as production replacements for commercial LLMs.

8.2 Main Contributions

8.2.1 Methodological Contributions

C1: Dual-Adapter Training Framework – Systematic methodology for adapting LLMs to domains with imbalanced multi-task distributions. Framework isolates tasks during training to eliminate gradient interference, then recombines knowledge via parameter-space merging. Key insight: *task isolation prevents catastrophic interference more effectively than any gradient balancing scheme* when imbalance exceeds model capacity. Generalizes to medical diagnosis (rare conditions), legal analysis (routine vs. complex), multilingual systems (low-resource pairs).

C2: Comprehensive Merging Method Analysis – First large-scale empirical comparison of eight adapter merging methods for dual-adapter scenarios. Simple averaging (Task Arithmetic) achieves 98.9% of best-method performance at 28% computational cost. Establishes **simplicity principle**: when combining two complementary specializations with minimal overlap, sophisticated conflict resolution provides marginal benefits. Principle holds for dual-adapter regime but not for many adapters (5+) or high overlap.

C3: LLM-as-Judge Evaluation Framework – Production-grade evaluation methodology using GPT-4.1-mini with detailed task-specific rubrics, providing fine-grained (1–10 scale) assessment of domain-specific quality, format compliance, reasoning correctness. Addresses limitations of traditional metrics (BLEU, ROUGE) failing to capture semantic correctness and complex formatting. 85% agreement with human expert assessments.

8.2.2 Empirical Contributions

C4: Task Alignment Dominates Dataset Size – Training on 1,388 format-aligned samples outperforms training on 38,170 diverse samples by 17.9% average (gaps: 4.6% Qwen to 57.2% DeepSeek). Challenges conventional wisdom that "more data is better," establishing *task distribution alignment* as primary driver of domain adaptation effectiveness. Implication: prioritize curating small, high-quality datasets matching evaluation distributions over collecting large, diverse datasets.

C5: Scale-Dependent Robustness to Interference – Model scale determines vulnerability: larger models (Qwen 14.8B) show only 4.6% gaps between combined and format-specialized training, while smaller models (Mistral 7B) exhibit 21.0% gaps and MoE models (DeepSeek) suffer 57.2% degradation. Suggests *capacity thresholds* around 10–15B parameters beyond which models handle imbalanced multi-task learning. Optimal strategies depend on computational resources: large models ($>$15B) can use simpler combined training, resource-constrained deployments benefit from dual-adapter on smaller models.

C6: Architecture-Specific Training Requirements – MoE models show catastrophic failure with combined training (-50% performance) but benefit most from dual-adapter merging (+85.9%), while dense models exhibit moderate interference and gains. Mechanism: expert specialization collapse under imbalance—when 96.4% of training comes from ex-premarket, MoE gating routes nearly all inputs to ex-premarket experts, leaving no capacity for minority patterns. Dense transformers avoid this through flexible attention-based capacity allocation.

8.2.3 Practical Contributions

C7: Production-Ready GPT-4 Alternative – Qwen2.5-14B + TIES achieves 8.99/10 (95% of GPT-4's 9.5/10) at 40–80% cost reduction ($120K vs. $400K/year for 10K daily queries), comparable latency (3–4s), eliminating data privacy concerns. Demonstrates **domain adaptation can compensate for base model capability gaps in specialized formatting domains**.

C8: Deployment Guidelines – Evidence-based recommendations across three tiers: **Tier 1** (highest quality): Qwen2.5-14B + TIES (8.99/10, 0.63% error, client-facing); **Tier 2** (cost-optimized): Mistral-7B + TIES (8.03/10, 0.93% error, internal analysis); **Tier 3** (reliability-optimized): Llama-3-8B + TA (6.31/10, 0.07% error, automated pipelines).

C9: Open-Source Implementation – Complete release: training scripts (Hugging Face Transformers, PEFT, DeepSpeed ZeRO-3), merging implementations (8 methods), evaluation harness (LLM-as-judge), 1,032-sample iris benchmark, pretrained adapters (5 base models). Enables reproduction, extension to new domains, benchmarking novel methods.

8.3 Answers to Research Questions

8.3.1 RQ1: Training Strategy Effectiveness

Can task-isolated dual-adapter training outperform combined fine-tuning and single-distribution training?

Answer: Yes, with model-dependent magnitudes. Dual-adapter + merging consistently outperforms combined fine-tuning (improvements: 4.6% to 57.2%). Magnitude depends on capacity: smaller models benefit more from task isolation due to limited ability to handle competing gradients.

Versus single-distribution training, merged adapters achieve statistical parity overall (6.76/10 vs. 6.78/10 premarket-only) but show superior performance on subsets requiring broad domain knowledge. **Dual-adapter provides best of both worlds**—format specialization plus domain knowledge without catastrophic interference.

8.3.2 RQ2: Merging Method Selection

Which parameter merging methods preserve knowledge most effectively?

Answer: Simple averaging suffices for dual-adapter scenarios. Task Arithmetic achieves 6.76/10 vs. 6.83/10 for best method (TIES), 98.9% performance ratio. The 0.7-point improvement comes at 3.5× computational cost. Sophisticated methods (TSV: 13× cost, DO-Merging: 6.5× cost) are dominated.

Default to Task Arithmetic, upgrading to TIES only when marginal improvements justify overhead.

8.3.3 RQ3: Architectural and Scale Effects

How do model architecture and scale interact with training strategy effectiveness?

Answer: Strong interactions observed.

Architecture: MoE models show catastrophic failure with combined training (2.77/10) but dramatic recovery with dual-adapter merging (5.13/10, +85.2%). Dense transformers exhibit moderate interference and gains. **MoE architectures require task isolation** due to expert specialization collapse.

Scale: Larger models demonstrate robustness: Qwen2.5-14B shows only 4.6% gap between combined and premarket, vs. 21.0% for Mistral-7B and 57.2% for DeepSeek. Establishes **capacity threshold around 10–15B parameters** beyond which combined training becomes viable.

Interaction implies optimal strategies depend jointly on architecture and scale: small MoE models require dual-adapter (critical), large dense models tolerate combined training (optional), medium-scale models occupy intermediate regime.

8.4 Future Work

8.4.1 Short-Term Extensions

Adapter Rank Optimization – Systematic exploration of rank-performance-cost trade-off. Preliminary results: rank-32 provides +0.3 points at 2× cost, indicating diminishing returns beyond rank-16.

Multi-Adapter Merging – Extension to N-adapter merging for 5–10 specialized task types introduces challenges: combinatorial explosion ($2^N - 1$ combinations), conflict accumulation, routing complexity. Hierarchical merging strategies or learned routing networks may address these.

Cross-Domain Validation – Validate on other domains: medical diagnosis (rare diseases 1%), legal analysis (routine 95% vs. complex 5%), scientific research, customer service. Each introduces unique challenges requiring framework modifications.

Temporal Adaptation – Continual learning strategies for evolving financial markets: incremental adapter training on new data, selective forgetting of obsolete knowledge, temporal ensembling with recency weighting.

8.4.2 Long-Term Research Directions

Theoretical Understanding of Capacity Thresholds – Develop theory connecting model capacity, task complexity, imbalance tolerance. Enable a priori threshold prediction, task complexity quantification, optimal capacity allocation. Neural Tangent Kernel and scaling laws provide starting points but don't account for multi-task interference dynamics.

Automated Merging Weight Optimization – Learn task-specific merging weights via gradient-based optimization on validation data:

$$\lambda^* =_\lambda \mathcal{L}_{\text{val}}(\theta_{\text{base}} + \lambda_A \Delta\theta_A + \lambda_B \Delta\theta_B) \tag{8.1}$$

Foundation Models Pre-Trained for Adaptation – Pre-training objectives optimized for adaptation readiness: multi-task pre-training with explicit task identifiers, adapter-aware pre-training with low-rank bottlenecks, meta-learning objectives for fast adaptation with minimal data.

Ethical and Societal Implications – Examine bias and fairness in investment recommendations, market manipulation vulnerabilities, transparency and explainability for regulators, economic impact on human analysts. Requires empirical bias audits, adversarial robustness testing, interpretability development, labor market studies.

8.5 Broader Impact

Beyond technical contributions, this thesis enables AI democratization in specialized domains. Demonstrating open-source models match commercial LLM performance at 40–80% cost reduction lowers barriers for SMEs with limited budgets, academic researchers without proprietary model access, developing regions with limited API access, and privacy-sensitive industries (healthcare, legal, financial) maintaining data sovereignty.

Open-source release of implementation, trained adapters, evaluation benchmark further accelerates democratization. However, lower barriers enable malicious actors to deploy LLMs for misinformation, fraud, market manipulation. Responsible deployment requires robust safeguards—content filtering, usage monitoring, regulatory compliance—balancing innovation with risk mitigation.

8.6 Closing Remarks

The central insight: **task isolation during training, followed by knowledge combination via post-hoc merging, eliminates catastrophic interference while preserving complementary specializations**. This principle proves powerful for domains with severe sample imbalance—ubiquitous in real-world production systems where data collection gravitates toward frequent, easy-to-annotate tasks at expense of rare, complex cases.

By demonstrating format-aligned small datasets outperform diverse large datasets by 17.9%, we challenge prevailing assumption that "more data is always better." Critical factor: not dataset size but *distribution alignment with evaluation tasks*—profound implications for data collection strategies, model selection, training methodologies.

Comprehensive evaluation reveals sophistication doesn't guarantee superiority: simple weighted averaging achieves 98.9% of best-method performance in dual-adapter scenarios, contradicting recent emphasis on complex conflict resolution. Advocates for *principle of simplicity*—prefer straightforward methods unless complexity demonstrably improves outcomes.

Finally, successful deployment of Qwen2.5-14B as production-ready GPT-4 alternative in Axyon AI's Alyx validates practical viability. Achieving 95% of commercial model performance at 40–80% cost reduction establishes new paradigm: **specialized open-source models adapted to domain-specific requirements can outcompete general-purpose commercial models** in production applications where format compliance, consistent style, data privacy dominate.

Path forward: extending dual-adapter training to broader domains, developing theoretical foundations for capacity thresholds and interference dynamics, addressing ethical implications of widespread LLM deployment in high-stakes contexts. We hope this thesis provides both foundation and inspiration for researchers continuing this important work.

Bibliography

Abdin, Marah et al. (2024). "Phi-4 Technical Report". In: *arXiv preprint arXiv:2412.08905*.

Aghajanyan, Armen, Sonal Gupta, and Luke Zettlemoyer (2020). "Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning". In: *arXiv preprint arXiv:2012.13255*.

Arivazhagan, Naveen et al. (2019). "Massively Multilingual Neural Machine Translation in the Wild: Findings and Challenges". In: *arXiv preprint arXiv:1907.05019*.

Banerjee, Satanjeev and Alon Lavie (2005). "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments". In: *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72.

Caruana, Rich (1997). "Multitask Learning". In: *Machine Learning* 28.1, pp. 41–75.

Chase, Harrison (2022). *LangChain*. URL: https://github.com/langchain-ai/langchain.

Chen, Mark et al. (2021). "Evaluating Large Language Models Trained on Code". In: *arXiv preprint arXiv:2107.03374*.

Chen, Zhao et al. (2018). "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks". In: *ICML*.

CINECA (2024). *Leonardo Supercomputer: Technical Service Description and User Guide*. Technical Documentation. Pre-exascale EuroHPC system based on NVIDIA Booster architecture. CINECA - Consorzio Interuniversitario. URL: https://leonardo-supercomputer.cineca.eu/leonardo-service-description/.

Dai, Damai et al. (2024). "DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models". In: *arXiv preprint arXiv:2401.06066*.

Deng, Xiao-Yang et al. (2023). "FLARE: Financial Language Understanding and Reasoning Evaluation". In: *arXiv preprint arXiv:2308.01433*.

Dubey, Abhimanyu et al. (2024). "The Llama 3 Herd of Models". In: *arXiv preprint arXiv:2407.21783*.

Dubois, Yann et al. (2024). "AlpacaEval: An automatic evaluator of instruction-following models". In: *arXiv preprint arXiv:2404.04475*.

European High Performance Computing Joint Undertaking (2025). *EuroHPC JU: Leading the way in European supercomputing*. URL: https://www.eurohpc-ju.europa.eu/index_en.

FFplus Project (2025). *FFplus: High-Performance Computing and Generative AI for European SMEs*. URL: https://www.ffplus-project.eu/en/.

Frankle, Jonathan et al. (2020). "Linear Mode Connectivity and the Lottery Ticket Hypothesis". In: *ICML*.

Hendrycks, Dan et al. (2021). "Measuring Massive Multitask Language Understanding". In: *International Conference on Learning Representations*.

Hu, Edward J et al. (2021). "LoRA: Low-Rank Adaptation of Large Language Models". In: *arXiv preprint arXiv:2106.09685*.

Ilharco, Gabriel et al. (2023). "Editing Models with Task Arithmetic". In: *arXiv preprint arXiv:2212.04089*. ICLR 2023.

Jiang, Albert Q, Alexandre Sablayrolles, Antoine Roux, et al. (2024). "Mixtral of Experts". In.

Jiang, Albert Q et al. (2023). "Mistral 7B". In: *arXiv preprint arXiv:2310.06825*.

Kaplan, Jared et al. (2020). "Scaling Laws for Neural Language Models". In: *arXiv preprint arXiv:2001.08361*.

Katz, Daniel Martin et al. (2024). "GPT-4 passes the bar exam". In: *Philosophical Transactions of the Royal Society A* 382.2270. Originally published as SSRN preprint, March 2023, p. 20230254.

Kendall, Alex, Yarin Gal, and Roberto Cipolla (2018). "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In: *CVPR*.

Khattab, Omar et al. (2023). "DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines". In: *arXiv preprint arXiv:2310.03714*.

Kirkpatrick, James et al. (2017). "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences*. Vol. 114. 13, pp. 3521–3526.

Lewis, Patrick et al. (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 9459–9474.

Lin, Chin-Yew (2004). "ROUGE: A package for automatic evaluation of summaries". In: *Text summarization branches out*, pp. 74–81.

Liu, Haofan et al. (2024). "Isotropic Merging of Language Models". In: *arXiv preprint arXiv:2405.17999*.

Loukas, Lefteris et al. (2021). "EDGAR-CORPUS: Billions of Tokens Make The World Go Round". In: *Proceedings of the Third Workshop on Economics and Natural Language Processing*.

Malo, Pekka et al. (2014). "Good debt or bad debt: Detecting semantic orientations in economic texts". In: *Journal of the Association for Information Science and Technology* 65.4, pp. 782–796.

OpenAI (2023). "GPT-4 Technical Report". In: *arXiv preprint arXiv:2303.08774*.

OpenAI (2024). *GPT-4 Technical Report*. Tech. rep. arXiv:2303.08774. OpenAI.

Ouyang, Long et al. (2022). "Training language models to follow instructions with human feedback". In: *arXiv preprint arXiv:2203.02155*.

Pan, Sinno Jialin and Qiang Yang (2009). "A Survey on Transfer Learning". In: *IEEE TKDE*.

Papineni, Kishore et al. (2002). "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318.

Rajbhandari, Samyam et al. (2020). "ZeRO: Memory Optimizations Toward Training Trillion Parameter Models". In: *arXiv preprint arXiv:1910.02054*. SC 2020.

Sanh, Victor, Albert Webson, Colin Raffel, et al. (2022). "Multitask Prompted Training Enables Zero-Shot Task Generalization". In: *ICLR*.

Singhal, Karan et al. (2023). "Large language models encode clinical knowledge". In: *Nature* 620.7972, pp. 172–180. DOI: 10.1038/s41586-023-06291-2.

Wei, Jason et al. (2022). "Finetuned Language Models Are Zero-Shot Learners". In: *International Conference on Learning Representations*.

Wortsman, Mitchell et al. (2022). "Model Soups: Averaging Weights of Multiple Fine-tuned Models Improves Accuracy without Increasing Inference Time". In: *International Conference on Machine Learning*. PMLR, pp. 23965–23998.

Wu, Shijie, Ozan Irsoy, Steven Lu, et al. (2023). "BloombergGPT: A Large Language Model for Finance". In: *arXiv preprint arXiv:2303.17564*.

Xie, Qianqian et al. (2023). "PIXIU: A Large Language Model, Instruction Data and Evaluation Benchmark for Finance". In: *arXiv preprint arXiv:2306.05443*.

Yadav, Prateek et al. (2023). "TIES-Merging: Resolving Interference When Merging Models". In: *arXiv preprint arXiv:2306.01708*. NeurIPS 2023.

— (2024). "TIES-Merging: Resolving Interference When Merging Models". In: *Advances in Neural Information Processing Systems* 36.

Yang, An et al. (2024). "Qwen2 Technical Report". In: *arXiv preprint arXiv:2407.10670*.

Yao, Shunyu et al. (2023a). "React: Synergizing reasoning and acting in language models". In: *International Conference on Learning Representations (ICLR)*.

Yao, Shunyu et al. (2023b). "Tree of Thoughts: Deliberate Problem Solving with Large Language Models". In: *Advances in Neural Information Processing Systems*.

Yu, Le et al. (2024). "Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch". In: *arXiv preprint arXiv:2404.19832*. ICML 2024.

Zenke, Friedemann, Ben Poole, and Surya Ganguli (2017). "Continual Learning Through Synaptic Intelligence". In: *ICML*.

Zeroshot (2023). *Twitter Financial News Sentiment Dataset*. URL: https://huggingface.co/datasets/zeroshot/twitter-financial-news-sentiment.

Zheng, Lianmin et al. (2023). "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena". In: *Advances in Neural Information Processing Systems*. Vol. 36.