University of Zurich^UZH

MASTER THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Dynamic Task Clustering and Aggregation for Decentralized Federated Multitask Learning

Xi Chen
Zurich, Switzerland
Student ID: 22-736-219

Supervisor: Chao Feng, Andy Aidoo
Date of Submission: January 7, 2026

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Master Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Declaration of Independence

I hereby declare that I have composed this work independently and that both the core scientific contributions and the primary codebase were developed by myself without the use of any aids other than those explicitly declared.

Regarding the use of generative AI tools, I declare the following:

- Claude Code was employed solely for the purpose of organizing, formatting, and structural refinement of the source code. The underlying algorithms and logic remain my original work.
- Gemini was utilized as a language assistant for polishing the prose, correcting grammatical errors, and improving the stylistic flow of the manuscript.

I am aware that I take full responsibility for the scientific integrity and accuracy of the submitted text and code, regardless of the tools used. All passages taken verbatim or in sense from published or unpublished writings, as well as any external assistance, are clearly identified. This work has not been submitted in the same or similar form to any other examination office.

Zürich,

Xi Chen 陈曦

Signature of student

Abstract

Föderiertes Multi-Task Learning (FMTL) ermöglicht kollaboratives Training über Clients hinweg, die unterschiedliche, aber verwandte Aufgaben optimieren, ohne lokale Daten zu teilen. Viele bestehende FMTL-Ansätze basieren jedoch auf zentralen Servern zur Koordination, was Single-Points-of-Failure schafft und zusätzliche Datenschutz- sowie Sicherheitsrisiken mit sich bringt. Diese Arbeit entwickelt ein vollständig dezentrales Peer-to-Peer-FMTL-Framework, in dem Clients Aufgabenbeziehungen aus beobachtbaren Trainingssignalen ableiten, Nachbarn dynamisch auswählen und Aggregation ohne zentrale Instanz durchführen.

Die Evaluation über mehrere Datensätze mit unterschiedlichen Task-Korrelationsstrukturen zeigt eine zentrale Erkenntnis: Der optimale Aggregationsumfang hängt systematisch von der Stärke der Task-Korrelation ab. Bei stark korrelierten Aufgaben kann selektives Teilen (z. B. Backbone-only) vorteilhaft sein, während in schwach korrelierten Settings umfassendere Aggregation häufig für stabile Konvergenz erforderlich ist. Darüber hinaus adressiert die Arbeit praktische Herausforderungen numerischer Instabilität in konfliktbewusster Aggregation durch ein Schutz- und Stabilisierungskonzept und dokumentiert Failure Cases zur Abgrenzung sicherer Konfigurationen. Insgesamt demonstriert die Arbeit serverfreies FMTL für komplexe Dense-Prediction-Aufgaben und leitet praktische Empfehlungen für dezentrale Nachbarschaftsselektion und Aggregationsdesign ab.

Federated Multi-Task Learning (FMTL) enables privacy-preserving collaboration among clients optimizing different but related tasks, yet most existing approaches rely on centralized servers for coordination, creating single points of failure and additional privacy and security exposure. This thesis presents a fully decentralized peer-to-peer FMTL framework in which clients infer task relatedness from observable training signals, select neighbors dynamically, and aggregate updates without central coordination.

Across datasets with different task-correlation structures, the evaluation reveals a consistent conclusion: the optimal aggregation scope depends on task correlation strength. For strongly correlated tasks, selective sharing (e.g., backbone-only) can be beneficial, whereas weakly correlated settings often require broader aggregation to maintain stable convergence. The thesis further addresses practical numerical instability in conflict-aware aggregation through a stabilization mechanism and reports failure cases to delineate safe operating regimes. Overall, the work demonstrates server-free FMTL for complex dense prediction workloads and provides actionable guidance for similarity-based collaboration and aggregation design.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to **Prof. Dr. Burkhard Stiller** for providing me with the opportunity to conduct this research at his chair and for his valuable support.

My deepest thanks go to my supervisors, **Chao Feng** and **Andy Aidoo**. This work would not have been possible without their consistent guidance, insightful feedback, and the time they dedicated to helping me navigate the challenges throughout this project. Their expertise and encouragement have been instrumental in my academic growth.

On a personal note, I am profoundly grateful to my boyfriend, **Houze**, for his unwavering support, patience, and encouragement during the intensive writing period of this thesis. His presence provided me with the strength to persevere through the most demanding phases.

Finally, a very special (and somewhat chaotic) acknowledgment goes to my four feline companions: **Xiaomai**, **Ale**, **Jiujiu**, and **Cider**. Thank you for your relentless ”assistance” in redecorating my drafts by walking across the keyboard and for ensuring I took frequent breaks by demanding my attention. Though you were technically a source of distraction, your presence made the long nights of writing much more joyful.

Contents

Chapter 1

Introduction

Federated learning enables collaborative model training across distributed data sources while keeping data local [1]. In its canonical form, a central server coordinates training rounds and aggregates client updates. This design is effective, yet it also encodes assumptions that can break in practice. A server becomes a single point of failure and a coordination bottleneck, and it concentrates privacy and security risk by observing the stream of client communications [2]. Moreover, the centralized formulation typically presumes a shared objective, whereas real deployments often involve heterogeneous goals: different institutions, devices, or users may optimize different tasks and metrics, even when operating in related domains.

Three gaps motivate the focus of this thesis. First, reliance on trusted central coordination is increasingly mismatched with settings where ownership and governance are distributed and a server may be infeasible or undesirable. Second, task heterogeneity is not an edge case; it requires collaboration mechanisms that selectively share useful information rather than enforcing uniform aggregation. Third, much of the federated learning literature emphasizes classification, leaving open how these ideas transfer to dense prediction problems with structured outputs and different stability characteristics.

Decentralized federated learning addresses the coordination bottleneck by replacing server orchestration with peer-to-peer communication [3], [4]. Removing the server improves resilience and avoids a single aggregation authority, but it also changes what information is available at decision time. In particular, clients lose global visibility into the system-wide task distribution and cannot rely on server-side clustering or centralized task relationship estimation as in prior heterogeneous methods [5], [6]. Enabling multi-task collaboration in a fully decentralized network therefore requires autonomous estimation of task relatedness from local observations, decentralized partner selection, and convergence despite heterogeneous collaboration policies across clients.

Dense prediction tasks make these challenges more pronounced. Problems such as depth estimation and semantic segmentation require per-pixel outputs [7], [8], typically involving heavier prediction heads and optimization behaviors that can be more sensitive to aggregation-induced instability. In addition, task relationships in dense prediction often reflect geometric and semantic couplings that vary across datasets, making it risky to extrapolate conclusions from classification-only evidence. These considerations motivate studying decentralized federated multi-task learning directly in dense prediction settings and validating conclusions across datasets with different correlation structures.

1.1 Research Objectives

The primary objective of this thesis is to develop optimization methods for Decentralized Federated Multitask Learning (DFMTL) that enable dynamic task clustering and improved model aggregation in fully decentralized and heterogeneous settings. Concretely, the framework is designed to allow nodes to identify implicit task similarities from model representations or training behaviors (e.g., gradient alignment and loss dynamics), and to self-organize into collaboration clusters in a fully distributed manner. Within each cluster, aggregation is expected to promote fast convergence and task alignment, while inter-cluster regularization is incorporated to preserve broader knowledge sharing across tasks.

Beyond algorithmic design, the thesis targets a practical and reusable system realization: the implementation is required to be modular and extensible, supporting decentralized training, dynamic clustering, and adaptive aggregation over arbitrary network topologies, with configurable task distributions, communication patterns, and aggregation rules. Finally, the framework is evaluated empirically under diverse simulated conditions, emphasizing task-specific performance, convergence behavior, clustering quality, communication efficiency, and resilience to network variability, including varying degrees of task heterogeneity, dynamic graph connectivity, and asynchronous updates, and benchmarking against relevant static DFMTL and homogeneous decentralized baselines.

1.2 Research Contributions

This thesis provides a complete, reproducible study of decentralized federated multi-task learning under heterogeneous objectives. A server-free peer-to-peer framework is presented that extends prior server-mediated formulations [9] to decentralized operation through independent neighbor selection and aggregation. Multiple task similarity signals for decentralized relationship discovery are proposed and evaluated, and gradient-only cosine similarity is identified as an effective and simple default relative to mixed alternatives. A key practical barrier is also addressed: numerical instability in conflict-aware aggregation [10]. The resulting stabilization mechanisms, together with documented failure cases, clarify safe operating regimes and provide actionable guidance. Through cross-dataset validation, two empirical conclusions recur across settings: aggregation scope should depend on task correlation (selective sharing can help when tasks are strongly aligned, while weakly correlated settings often require broader sharing for stability), and pairwise task assignment with asymmetric weighting is a robust collaboration pattern for dense prediction.

1.3 Code and Reproducibility

All implementations and experimental configurations are released to support reproducibility:

`https://github.com/xichen0257/asfdfmtl`

The repository contains configuration-driven experiment specifications (45+ YAML files), training scripts for classification and dense prediction, dataset-specific data pipelines, and documentation of stabilization mechanisms and instructions to reproduce the experiments in Chapter 6.

1.4 Thesis Organization

The remainder of this thesis proceeds through six chapters. Chapter 2 introduces centralized and decentralized federated learning, multi-task learning for dense prediction, the three validation datasets, and the problem formulation used throughout the thesis. Chapter 3 surveys federated learning under heterogeneity, task relationship discovery, aggregation strategies, and decentralized architectures, and identifies the gap addressed by this work. Chapter 4 presents the proposed framework, including task similarity estimation, aggregation mechanisms, aggregation scope design, stabilization techniques, and efficiency considerations. Chapter 5 describes the software architecture and configuration-driven workflow supporting systematic experimentation. Chapter 6 reports results across CIFAR-10, NYU Depth V2, and Pascal Context, and synthesizes cross-dataset patterns and failure modes. Finally, Chapter 7 summarizes the contributions, discusses implications and limitations, and outlines directions for future work.

Chapter 2

Background

This chapter establishes the foundational concepts necessary to understand the proposed decentralized federated multi-task learning framework. Section 2.1 introduces federated learning, contrasting centralized and decentralized architectures. Section 2.2 explains multi-task learning for dense prediction tasks. Section 2.3 describes the three validation datasets. Section 2.4 formalizes federated multi-task learning and motivates the transition to decentralized settings.

2.1 Federated Learning Fundamentals

Federated Learning (FL) enables collaborative model training across distributed clients without centralizing raw data [1]. Model parameters are learned through iterative aggregation of local gradient-based updates, eliminating the need to transmit sensitive training data to a central location. This section contrasts centralized and decentralized FL architectures, motivating the architectural choices in this thesis.

2.1.1 Centralized Federated Learning

In centralized federated learning, a central server orchestrates training across N distributed clients. The **Federated Averaging (FedAvg)** algorithm [1] proceeds in iterative rounds: the server initializes global model parameters $w^{(0)} \in \mathbb{R}^d$ and broadcasts them to clients. Each client i performs local training on private dataset $\mathcal{D}_i = \{(x_j, y_j)\}_{j=1}^{n_i}$ for E epochs using stochastic gradient descent, producing updated parameters $w_i^{(t+1)}$. The rationale for multiple local epochs before synchronization is communication efficiency: each round incurs significant latency and bandwidth costs [11]. Clients transmit updates to the server, which performs weighted aggregation:

$$w^{(t+1)} = w^{(t)} + \sum_{i=1}^{N} \frac{n_i}{n} \Delta w_i^{(t)} \tag{2.1}$$

where $n_i = |\mathcal{D}_i|$ and $n = \sum_{i=1}^{N} n_i$. This weighting ensures larger datasets exert proportionally greater influence, justified when data distributions are IID [12].

Centralized FL achieves privacy preservation (raw data never leaves clients) and communication efficiency (reduced synchronization frequency). However, it introduces critical limitations. The central server represents a single point of failure: hardware failures or network partitions halt training indefinitely. The server observes all communications, creating privacy vulnerabilities—gradient inversion attacks [2] can reconstruct training examples from gradients. Additionally, scaling to thousands of clients creates communication bottlenecks as the server handles $\mathcal{O}(N)$ concurrent connections per round.

2.1.2 Decentralized Federated Learning

Decentralized Federated Learning (DFL) addresses these limitations by eliminating the central server, enabling direct peer-to-peer communication [13], [14]. The rationale is threefold: removing the single point of failure improves resilience, fragmenting communications enhances privacy, and distributing coordination eliminates bottlenecks. The network is formalized as graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where vertices $\mathcal{V} = \{1, \ldots, N\}$ represent clients and edges $\mathcal{E}$ encode communication links. Each client i maintains local parameters $w_i^{(t)}$ and communicates with neighbors $\mathcal{N}_i = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$.

Training proceeds through local optimization, neighbor communication, and decentralized aggregation:

$$w_i^{(t+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} a_{ij} w_j^{(t)} \tag{2.2}$$

where aggregation weights $\{a_{ij}\}$ satisfy $\sum_{j \in \mathcal{N}_i \cup \{i\}} a_{ij} = 1$ for stability. Network topology $\mathcal{G}$ influences convergence speed and communication costs, as visualized in Figure 2.1. Ring topology (degree 2) minimizes overhead but has diameter $\mathcal{O}(N)$, causing slow consensus. Fully connected topology (degree $N-1$) achieves unit diameter but requires $\mathcal{O}(N^2)$ links. Random topology with budget k balances these extremes, achieving diameter $\mathcal{O}(\log N)$ with bounded per-client costs [15].

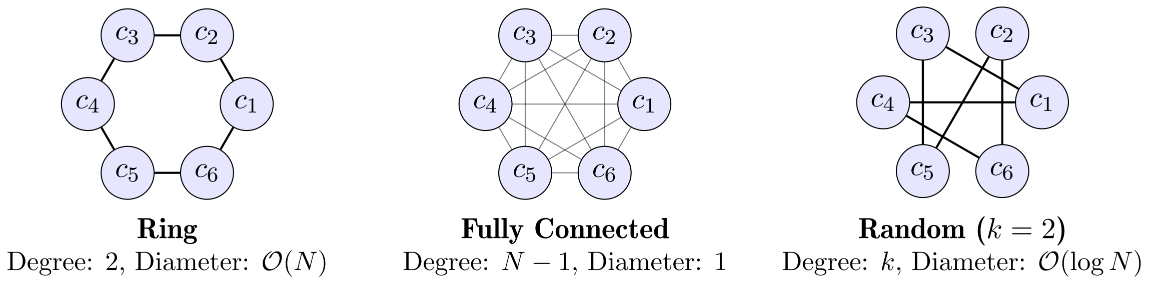

c_3
c_2
c_4
c_1
c_5
c_6
Ring
Degree: 2, Diameter: $\mathcal{O}(N)$
Fully Connected
Degree: $N-1$, Diameter: 1
Random ($k = 2$)
Degree: k, Diameter: $\mathcal{O}(\log N)$

Figure 2.1: Comparison of decentralized network topologies $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Ring topology minimizes per-client communication overhead at the cost of slow consensus ($\mathcal{O}(N)$ diameter). Fully connected topology achieves fastest convergence (unit diameter) but requires $\mathcal{O}(N^2)$ total connections. Random topology with communication budget k provides a practical balance, achieving $\mathcal{O}(\log N)$ diameter with bounded per-client degree [15].

Decentralized architectures eliminate single points of failure, fragment the global view across clients (enhancing privacy), and distribute network load. However, they introduce new challenges: clients must make aggregation decisions autonomously without global knowledge of task distributions, achieve consensus through purely local interactions under heterogeneous objectives, and select appropriate topologies without centralized coordination. This thesis addresses these challenges for multi-task learning, where selective aggregation becomes critical to avoid negative transfer.

2.2 Multi-Task Learning in Computer Vision

Multi-Task Learning (MTL) simultaneously optimizes multiple related tasks using shared representations to improve generalization beyond task-specific models [16], [17]. The hypothesis is that related tasks share common structure, and joint training provides inductive bias improving sample efficiency. This section focuses on MTL for dense prediction, where pixel-level outputs create unique challenges for federated settings.

2.2.1 Dense Prediction Tasks

Dense prediction tasks generate structured outputs where each spatial location receives an independent prediction, unlike classification which produces single labels. This spatial preservation constrains architectures to maintain high-resolution features or implement sophisticated upsampling. This thesis employs five dense prediction tasks:

Semantic segmentation assigns class labels to every pixel. For image $I \in \mathbb{R}^{H \times W \times 3}$, output is $Y_{\text{seg}} \in \{1, \ldots, C_{\text{seg}}\}^{H \times W}$. Evaluation uses mean IoU: $\text{IoU}_c = \frac{TP_c}{TP_c + FP_c + FN_c}$. **Depth estimation** predicts distance from camera, producing $Y_{\text{depth}} \in \mathbb{R}^{H \times W}$ with metric or relative depth. Evaluation uses RMSE and threshold accuracy $\max(\frac{\hat{Y}}{Y}, \frac{Y}{\hat{Y}}) < \delta$ for $\delta \in \{1.25, 1.25^2, 1.25^3\}$. **Surface normal prediction** estimates 3D orientation at each pixel: $Y_{\text{normal}} \in \mathbb{R}^{H \times W \times 3}$ with unit vectors $(n_x, n_y, n_z) \in \mathbb{S}^2$. Evaluation uses mean angular error. **Edge detection** produces binary classification $Y_{\text{edge}} \in \{0, 1\}^{H \times W}$ identifying boundaries, evaluated with F-measure (ODS-F). **Human parts segmentation** labels body parts when humans present: $Y_{\text{parts}} \in \{0, 1, \ldots, C_{\text{parts}}\}^{H \times W}$.

These tasks exhibit varying correlation. Depth and normals are geometrically linked: $\mathbf{n}(u, v) = \frac{(-\frac{\partial z}{\partial u}, -\frac{\partial z}{\partial v}, 1)}{\|\cdot\|}$. Depth discontinuities produce normal changes, creating strong correlation. Semantic boundaries often align with depth edges as distinct objects occupy separate spatial regions. However, correlations are not universal: edge detection responds to texture boundaries unrelated to semantics, and human parts applies only to subsets of images.

2.2.2 Task Interference and Negative Transfer

When tasks share structure, MTL achieves **positive transfer** [18]: shared representations from one task provide useful inductive bias for related tasks. However, unrelated or conflicting tasks cause **negative transfer** [19], degrading performance relative to single-task training. The mechanism is gradient conflicts in shared parameters. Consider model with shared backbone $\theta_s \in \mathbb{R}^{d_s}$ and task-specific parameters $\theta_{t_1}, \theta_{t_2}$. Losses $\mathcal{L}_{t_1}, \mathcal{L}_{t_2}$ produce gradients $g_1 = \nabla_{\theta_s}\mathcal{L}_{t_1}$ and $g_2 = \nabla_{\theta_s}\mathcal{L}_{t_2}$. When $\langle g_1, g_2 \rangle < 0$, they oppose: improving one task degrades the other.

The combined gradient is:

$$g_{\text{combined}} = \lambda_1 \nabla_{\theta_s}\mathcal{L}_{t_1} + \lambda_2 \nabla_{\theta_s}\mathcal{L}_{t_2} \tag{2.3}$$

where $\lambda_1, \lambda_2 \in \mathbb{R}^+$ are task weights. Conflicts produce compromise directions satisfying neither task optimally. When $\lambda_1 g_1 \approx -\lambda_2 g_2$, gradients cancel ($\|g_{\text{combined}}\| \approx 0$), causing stagnation.

Approaches addressing conflicts include task weighting (GradNorm [20], uncertainty weighting [21]), gradient manipulation (PCGrad [22] projects conflicting gradients), and architectural partitioning (hard parameter sharing [23], attention mechanisms [24]). This thesis adopts hard parameter sharing with adaptive aggregation scope. The rationale is threefold: parameter efficiency critical in federated settings (communication costs scale with model size), explicit separation enables selective aggregation (clients choose which components to share), and empirical success in dense prediction [24]. The contribution is transforming aggregation scope from fixed architectural choice into dynamic, data-driven decision.

2.3 Datasets for Multi-Task Dense Prediction

This thesis validates the framework on three datasets spanning simple classification (CIFAR-10) to strongly correlated dense prediction (NYU Depth V2) to weakly correlated dense prediction (Pascal Context), enabling systematic investigation of how task relationships influence decentralized aggregation. Table 2.1 summarizes the key characteristics of these datasets.

2.3.1 CIFAR-10

CIFAR-10 [25] comprises 60,000 images at 32×32 resolution across ten classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck), split into 50,000 training and 10,000 test images. This thesis inherits CIFAR-10 from Kohler's centralized framework [26], inducing task heterogeneity by assigning class subsets (e.g., animals vs vehicles).

While not dense prediction, CIFAR-10 serves three purposes: validating that decentralization replicates centralized performance (no degradation from removing server), enabling

Table 2.1: Comparison of validation datasets for decentralized FMTL.

Dataset	**Domain**	**Tasks**	**Samples**	**Task Correlation**
CIFAR-10	Classification	1 (10 classes)	50k train / 10k test	N/A (baseline, class-subset heterogeneity)
NYU Depth V2	Indoor scenes	3: Depth, Normal, Seg	1,449 RGB-D pairs	**Strong** (geometric coupling: depth $\leftrightarrow$ normals, semantic $\leftrightarrow$ depth edges)
Pascal Context	Outdoor scenes	3: Seg, Parts, Edge	4,998 train / 5,105 val	**Weak** (partial correlation: edges $\not\equiv$ semantics, parts only for humans)

rapid prototyping on computationally efficient images, and bridging prior work to novel contributions. However, small image size constrains representation complexity, and categorical heterogeneity differs from geometric/semantic heterogeneity in dense prediction. Core contributions rest on dense prediction datasets.

2.3.2 NYU Depth V2

NYU Depth V2 [7] comprises 1,449 RGB-D image pairs of indoor scenes at 640 × 480 resolution. The dataset provides three tasks exhibiting strong geometric coupling: depth estimation (metric depth from Kinect sensors, range 0.5-10m), surface normal prediction (derived from depth via local surface fitting, creating intrinsic correlation), and semantic segmentation (13 indoor categories: bed, books, ceiling, chair, floor, furniture, objects, picture, sofa, table, TV, wall, window).

Tasks exhibit strong correlation through multiple mechanisms. Depth discontinuities imply normal changes: sudden transitions from nearby objects to distant walls correspond to dramatically different orientations. Semantic boundaries align with depth edges as distinct objects occupy separate spatial regions. Indoor scenes have strong geometric structure: horizontal floors, vertical walls, characteristic furniture shapes producing predictable depth-normal relationships. This creates positive transfer opportunities: depth features (detecting planes, discontinuities) benefit segmentation (planes often correspond to single classes), and semantic information provides geometric priors (floor label implies horizontal orientation, consistent depth).

The rationale for NYU Depth V2 is twofold: strong correlation makes it ideal for testing whether decentralized FMTL leverages complementary information without centralized coordination (centralized MTL shows joint training substantially outperforms single-task [24], [27]), and this provides the first application of decentralized FMTL to NYU Depth V2. Dense prediction introduces unique federated challenges: pixel-level labels create larger gradients (exacerbating communication and stability), multi-scale architectures complicate sharing decisions, and strong correlations create cooperation opportunities but also negative transfer risks.

2.3.3 Pascal Context

Pascal Context [8] extends PASCAL VOC with dense annotations for 59 semantic categories plus background, comprising 4,998 training and 5,105 validation images of diverse outdoor scenes. This thesis employs three weakly correlated tasks: semantic segmentation (59 classes: objects like person, car, bicycle; stuff like sky, grass, road), human parts segmentation (8 body parts: head, torso, upper/lower arms/legs, hands, feet; conditionally defined, meaningful only for pixels depicting humans), and edge detection (binary classification of boundaries, derived from semantic boundaries plus low-level texture/lighting edges).

Tasks exhibit weak correlation, creating fundamentally different scenarios. Edge detection responds to intensity gradients and texture discontinuities which may not align with semantic boundaries (brown dog on brown dirt produces weak edges; zebra stripes produce texture edges irrelevant to segmentation). Human parts applies only to image subsets containing people (40-50%), providing no information for scenes without humans. When humans present, parts require articulated pose reasoning orthogonal to general segmentation or edges.

This weak correlation tests whether decentralized FMTL handles scenarios where tasks provide limited mutual benefit and forced collaboration may induce negative transfer. The rationale is providing a challenging, realistic test case. This is the first application of decentralized FMTL to Pascal Context [28]. Real-world deployments will involve clients with partially related or unrelated tasks; systems must handle this heterogeneity without catastrophic negative transfer.

Combining NYU Depth V2 (strong correlation) and Pascal Context (weak correlation) enables controlled investigation of how task relationship strength influences aggregation strategies. Cross-dataset validation provides evidence that observed phenomena represent general properties rather than dataset-specific artifacts.

2.4 Federated Multi-Task Learning (FMTL)

Federated Multi-Task Learning extends federated learning to scenarios where clients optimize different but potentially related tasks, combining FL's privacy and communication benefits with MTL's representation sharing [5]. This section formalizes FMTL, discusses architectural choices, reviews personalization techniques, and motivates decentralization.

2.4.1 Problem Formulation

Consider a federated network with N clients where each client $i \in \{1, \ldots, N\}$ is associated with task t_i from task set $\mathcal{T} = \{1, \ldots, T\}$. Task assignment $\tau : \{1, \ldots, N\} \to \mathcal{T}$ maps clients to tasks. Client i has private dataset $\mathcal{D}_i = \{(x_j, y_j^{t_i})\}_{j=1}^{n_i}$ where inputs $x_j \in \mathcal{X}$

lie in common space (e.g., $\mathcal{X} = \mathbb{R}^{H\times W\times 3}$ for images) and outputs $y_j^{t_i} \in \mathcal{Y}_{t_i}$ lie in task-specific spaces. The critical distinction from standard FL is that output spaces $\mathcal{Y}_t$ and loss functions $\ell_t : \mathcal{Y}_t \times \mathcal{Y}_t \to \mathbb{R}^+$ differ across tasks, preventing direct model averaging.

In personalized FMTL, the objective is learning client-specific models minimizing local empirical risks:

$$\min_{w_1,\dots,w_N} \sum_{i=1}^{N} \mathcal{L}_i(w_i; \mathcal{D}_i) = \min_{w_1,\dots,w_N} \sum_{i=1}^{N} \frac{1}{n_i} \sum_{(x,y)\in\mathcal{D}_i} \ell_{t_i}(f(x; w_i), y) \tag{2.4}$$

where $f(\cdot; w_i) : \mathcal{X} \to \mathcal{Y}_{t_i}$ is the model's forward pass. Pure local training solves this independently per client: $w_i^* = \arg\min_w \mathcal{L}_i(w; \mathcal{D}_i)$, guaranteeing no negative transfer but failing to exploit positive transfer from related tasks.

FMTL improves through selective collaboration: clients with related tasks share information while unrelated tasks avoid interference. Collaboration is implemented via aggregation where client i incorporates information from subset $\mathcal{C}_i \subseteq \{1, \dots, N\} \setminus \{i\}$:

$$w_i^{(t+1)} = \text{Aggregate}(w_i^{(t)}, \{w_j^{(t)}\}_{j\in\mathcal{C}_i}; \{\alpha_{ij}\}_{j\in\mathcal{C}_i}) \tag{2.5}$$

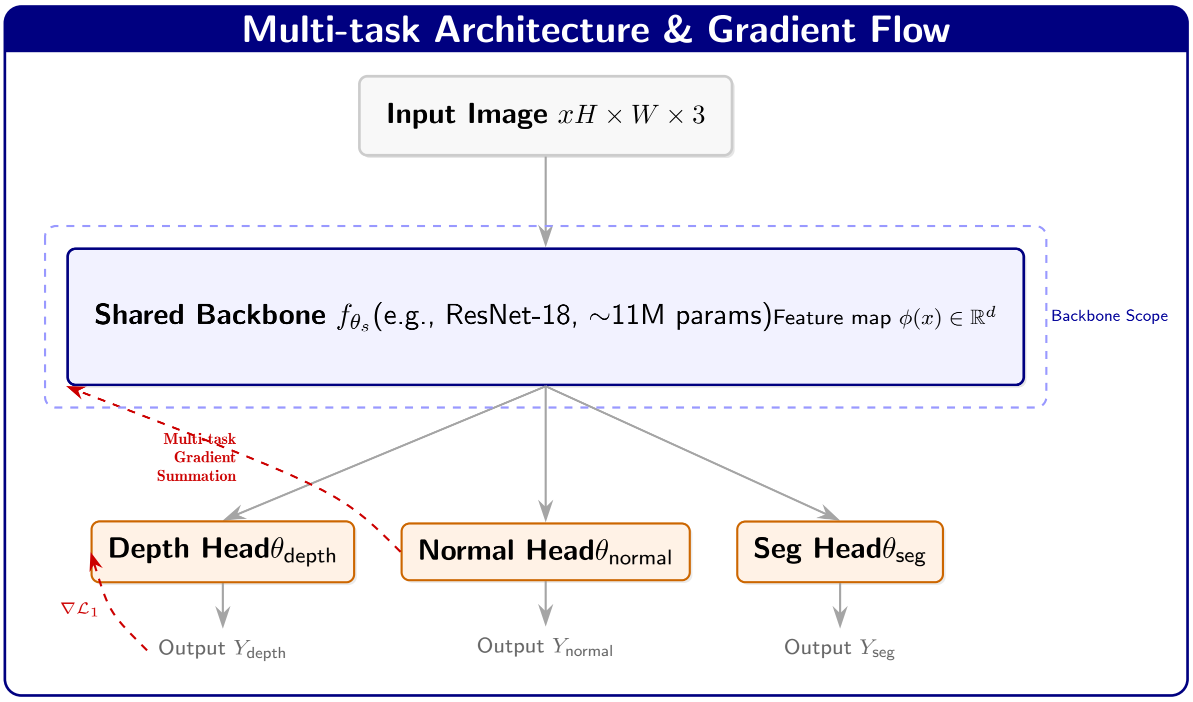
where $\{\alpha_{ij}\}$ are weights reflecting estimated benefit. The challenge is determining $\mathcal{C}_i$ and $\{\alpha_{ij}\}$ to maximize positive transfer while minimizing negative transfer, particularly in decentralized settings lacking global knowledge.

2.4.2 Hard Parameter Sharing Architecture

This thesis employs **hard parameter sharing** [23], decomposing models into shared backbone $\theta_s \in \mathbb{R}^{d_s}$ (feature extraction layers: CNNs like ResNet, or vision transformers) and task-specific heads $\{\theta_t\}_{t\in\mathcal{T}}$ (specializing features for task outputs). The backbone produces intermediate features $\phi(x; \theta_s) \in \mathbb{R}^{d_{\text{feat}}}$ encoding edges, textures, object parts, high-level concepts. Heads receive shared features and specialize for outputs: segmentation heads upsample features and produce C_{seg} logits per pixel, depth heads produce single continuous values through sigmoid/softplus ensuring positivity, normal heads output three values followed by L2 normalization enforcing $\|\mathbf{n}\|_2 = 1$.

For client i with task t_i, model parameterization is $w_i = (\theta_s, \theta_{t_i})$, as illustrated in Figure 2.2. Forward: input x is processed by backbone $\phi(x; \theta_s)$, then head produces $f(x; w_i) = h_{t_i}(\phi(x; \theta_s); \theta_{t_i})$. Backward: loss $\mathcal{L}_i$ produces gradients $\nabla_{\theta_{t_i}} \mathcal{L}_i$ and $\nabla_{\theta_s} \mathcal{L}_i$. Critically, heads receive gradients only from respective tasks while backbones receive gradients from all tasks, making θ_s the locus of multi-task interaction and potential conflicts.

The rationale for hard sharing: parameter efficiency (backbone 11M parameters, heads <1M each; shared backbone reduces total by factor T), natural transfer learning (features learned from one task immediately available to others), and architectural flexibility (clients aggregate only backbone, only heads, or both, tailoring strategies to task relationships).



Multi-task Architecture & Gradient Flow
Input Image $xH \times W \times 3$
Shared Backbone f_{θ_s}(e.g., ResNet-18, ~11M params)Feature map $\phi(x) \in \mathbb{R}^d$
Backbone Scope
Multi-task Gradient Summation
Depth Headθ_{depth}
Normal Headθ_{normal}
Seg Headθ_{seg}
$\nabla\mathcal{L}_1$
Output Y_{depth}
Output Y_{normal}
Output Y_{seg}

Figure 2.2: Detailed hard parameter sharing architecture. The backbone θ_s serves as the shared representation learner receiving joint gradients, while task-specific heads specialize in individual objectives. This visual separation justifies our flexible aggregation scopes (Backbone-only vs. Full-model).

2.4.3 Personalization and the Transition to Decentralized FMTL

Existing personalization techniques handle heterogeneous objectives differently, as summarized in Table 2.2. FedPer [29] fixes heads locally and aggregates only backbones, preventing head interference while enabling feature collaboration. However, it assumes fixed split between shared/personal layers and uniform backbone aggregation. FedRep [30] decouples optimization: alternating representation learning (backbone with frozen heads) and personalization (heads with frozen backbone), ensuring aggregated updates reflect feature learning. MOCHA [5] formulates FMTL as distributed optimization with task covariance matrix $\Omega \in \mathbb{R}^{T\times T}$ quantifying task similarity, deriving communication-efficient rules with convergence guarantees. However, MOCHA requires centralized servers computing Ω using global information.

Traditional FMTL methods [5], [29] rely on centralized servers for four functions: collecting updates from all clients, computing global task relationships via clustering/covariance, deciding aggregation strategies, and broadcasting aggregated models. These become infeasible in fully decentralized peer-to-peer networks where no single entity observes all clients. This necessitates three core capabilities absent in centralized FMTL.

Local similarity computation must replace global clustering. Each client autonomously estimates task relatedness to neighbors without global statistics, aggregate loss curves, or centralized metadata. Clients observe only neighbor-communicated information (parameters, gradients, auxiliary statistics) and infer compatibility from limited observations. The

Table 2.2: Comparison of FMTL personalization approaches.

Method	Aggregation Strategy	Architecture	Key Characteristics / Limitations
FedPer	Aggregate backbone only; heads remain local	Fixed split (backbone shared, heads personal)	Fixed scope; uniform aggregation; centralized
FedRep	Decoupled optimization (backbone/head alternation)	Fixed split with temporal decoupling	Fixed scope; centralized coordinator
MOCHA	Task covariance matrix Ω guides aggregation	Flexible; requires task relationship matrix	High overhead; requires centralized computation of Ω
This Thesis	**Adaptive scope** (backbone/heads/both) + **similarity weights**	**Dynamic, data-driven decisions**	**Fully decentralized** (local similarity computation)

rationale is that without global knowledge, clients require principled methods for identifying beneficial partners based solely on local interactions. Chapter 4 develops three metrics: gradient-based cosine similarity (leveraging optimization direction alignment), task overlap similarity (exploiting known task weights when clients train multiple tasks), and cross-loss similarity (directly measuring prediction quality by evaluating neighbors' models on local validation data).

Autonomous aggregation decisions must replace centralized coordination. Each client independently chooses which neighbors to incorporate, what weights to assign, and which components (backbone, heads, both) to aggregate, without knowledge of global topology, total client count, or aggregate training dynamics. Clients balance exploration (trying diverse neighbors to discover collaborations) against exploitation (focusing on known beneficial neighbors), adapting as training progresses. The rationale is that decentralized systems by definition lack central coordination, requiring clients to act as independent agents optimizing local objectives while respecting distributed nature.

Convergence without consensus becomes a theoretical requirement. Unlike centralized FL where servers enforce consensus by broadcasting single global models, decentralized clients may converge to different models tailored to respective tasks. Even when clients make heterogeneous aggregation choices (client A aggregates with B and C; client D aggregates with E and F), training must remain stable and each client must converge to performant solutions. This requires numerical stability mechanisms preventing gradient explosion or NaN propagation, especially when employing sophisticated methods like Hyper Conflict-Averse aggregation. The rationale is that decentralized settings lack centralized normalization and gradient clipping, making numerical issues more severe and requiring local safeguards.

This thesis extends FedPer in three dimensions: aggregation scope becomes adaptive (clients dynamically decide backbone-only, heads-only, or full aggregation based on estimated similarity), aggregation weights become similarity-based (weighting neighbors by compatibility rather than uniform averaging), and the framework transitions to decentralized (clients perform similarity computation, scope decisions, and weighted aggregation autonomously using only local information and neighbor communications). These extensions transform static, centralized, uniform aggregation into dynamic, decentralized, selective aggregation capable of handling diverse task relationships without central coordination. Chapter 4 presents technical mechanisms enabling this transition, Chapter 5 describes software realization, and Chapter 6 validates effectiveness on the three datasets.

Chapter 3

Related Work

This chapter reviews prior work through the lens of the four core challenges identified in Chapter 2: task heterogeneity, unknown task relationships, negative transfer under aggregation, and decentralization constraints. Rather than providing an exhaustive survey, the organization follows the problem hierarchy established in Chapter 2, emphasizing how existing work addresses subsets of these challenges.

Section 3.1 examines federated learning approaches that accommodate heterogeneous client objectives, distinguishing personalization methods from explicit multi-task formulations. Section 3.2 surveys techniques for discovering task relationships, progressing from centralized methods with unrestricted data access to federated clustering approaches that infer relationships through limited communication. Section 4.4 reviews aggregation mechanisms, contrasting task-agnostic averaging with conflict-aware methods designed to mitigate negative transfer. Section 3.4 discusses decentralized federated learning architectures that eliminate central coordination through peer-to-peer communication. Finally, Section 3.5 identifies the specific research gap addressed by this thesis and positions our contributions relative to the most closely related prior work, particularly Nicolas Kohler's centralized federated multi-task learning framework.

3.1 Federated Learning under Task Heterogeneity

Federated learning with heterogeneous client objectives—where clients optimize different loss functions corresponding to distinct learning tasks—requires mechanisms to balance collaboration against interference. This section distinguishes two paradigms: personalized federated learning, which adapts a shared global model to client-specific data characteristics while maintaining a common task objective, and federated multi-task learning, which explicitly handles scenarios where clients optimize fundamentally different tasks. Both paradigms address the task heterogeneity challenge identified in Chapter 2, but differ in their formulation and architectural assumptions.

3.1.1 Personalized Federated Learning

Personalized federated learning addresses statistical heterogeneity—clients possessing non-IID data distributions despite optimizing a common objective—through client-specific model adaptation. These methods typically assume task homogeneity at the objective level (all clients solve the same problem type) but accommodate distribution shifts across clients.

Arivazhagan *et al.* [29] proposed Federated Personalization (FedPer), introducing a simple yet effective architectural separation between shared and personalized components. FedPer partitions models into base layers (shared and aggregated globally) and head layers (task-specific and kept local). The rationale rests on the hypothesis that early network layers learn generic feature representations beneficial across tasks while later layers encode task-specific decision boundaries. By aggregating only base layers, FedPer prevents interference in task-specific components while enabling low-level feature sharing. However, the method employs a fixed architectural split point (typically after the penultimate layer) without adapting to task relationship strength. All clients aggregate the same base layers uniformly regardless of whether their tasks are strongly correlated (warranting extensive sharing) or weakly correlated (requiring minimal sharing). This thesis extends FedPer's principle by introducing adaptive aggregation scope selection, where clients dynamically choose between backbone-only aggregation and full model aggregation based on measured task similarity.

T. Dinh *et al.* [31] developed personalized Federated Learning through Moreau Envelopes (pFedMe), formulating personalization as a bi-level optimization problem. Each client maintains a personalized model optimized for local performance while regularized toward a global consensus model through a Moreau envelope term. The regularization strength hyperparameter λ controls the trade-off between personalization (capturing client-specific patterns) and collaboration (leveraging global knowledge). Theoretically, pFedMe provides convergence guarantees and Pareto-optimal personalization-collaboration balance under certain smoothness conditions. However, the method applies uniform regularization strength across all clients without considering which clients have related tasks that warrant strong collaboration versus unrelated tasks requiring isolation. A task-aware extension would assign heterogeneous λ values based on pairwise task similarity, an avenue unexplored in the original work.

Li *et al.* [32] proposed Ditto, separating personalized and global models into distinct parameter vectors optimized through alternating minimization. Each client maintains two models: a personalized model w_i^{pers} optimized for local validation performance and a global model w_i^{glob} regularized toward the federated average. This architectural separation provides flexibility for heterogeneous clients but doubles memory requirements and does not explicitly model task relationships beyond the implicit coupling through regularization. The method remains agnostic to which clients should collaborate, treating all client pairs symmetrically.

These personalization methods improve performance under data heterogeneity but fundamentally assume task homogeneity—all clients optimize the same objective function.

They do not address the scenario of explicit task heterogeneity where clients optimize fundamentally different objectives, such as depth estimation versus semantic segmentation, which requires explicit multi-task formulations.

3.1.2 Federated Multi-Task Learning

Federated multi-task learning explicitly addresses scenarios where clients optimize distinct objective functions corresponding to different learning tasks. Unlike personalization methods that adapt a common model to local data distributions, federated MTL formulations recognize that clients may have fundamentally different loss functions and must balance task-specific optimization against cross-task knowledge transfer.

Smith *et al.* [5] formulated federated multi-task learning as distributed optimization of related but distinct client objectives, introducing MOCHA (Multi-Task Optimization for Communication-Efficient Heterogeneous Aggregation). The method assumes task relationships manifest as shared low-dimensional subspace structure in the parameter space—related tasks have optimal parameters residing near a common subspace while diverging in orthogonal directions to capture task-specific variations. MOCHA employs alternating optimization to jointly learn the shared subspace and task-specific parameters, providing theoretical convergence guarantees under convexity assumptions. While elegant in formulation, MOCHA requires centralized coordination for subspace estimation through singular value decomposition of the client parameter matrix, and the convexity assumptions limit applicability to modern deep learning with non-convex objectives. Furthermore, the method does not extend to decentralized topologies where no central entity can collect the full client parameter matrix.

Recent advances in federated multi-task learning have explored heterogeneous client architectures [33], graph-based selective collaboration [34], subspace decoupling for unified modeling [35], and adaptive clustering frameworks [36]. Despite these contributions, a fundamental gap persists. Existing federated multi-task methods follow one of three paradigms, each with critical limitations. First, methods like MOCHA require centralized coordination for subspace computation or similarity matrix aggregation, contradicting decentralized architecture goals. Second, methods like FedPer and pFedMe apply uniform collaboration policies to all clients without task-aware selection—every client aggregates the same information from the same sources regardless of task relatedness. Third, personalization-focused methods like Ditto emphasize the personalization-collaboration trade-off but do not address the task relationship discovery problem, failing to identify which clients should collaborate preferentially. None of these paradigms address the core challenge of decentralized dynamic task relationship discovery, where clients autonomously identify beneficial collaboration partners in peer-to-peer networks through local computation and neighbor communication, without centralized orchestration or global knowledge aggregation.

3.2 Learning Task Relationships in Federated Learning

As discussed in Chapter 2, the effectiveness of multi-task learning critically depends on task relatedness, which is typically unknown in federated settings. Existing work has therefore explored mechanisms for approximating task relationships using observable signals such as data statistics, model updates, or gradients. Among these approaches, clustering-based methods represent a practical proxy for task relationship discovery, rather than an explicit modeling of task semantics.

3.2.1 Centralized Task Relationship Learning

Centralized multi-task learning assumes collocation of all task datasets, enabling direct empirical measurement of task relationships through transfer learning experiments or joint training analysis. This privileged setting provides theoretical and algorithmic insights that inform federated extensions, despite the fundamental architectural differences.

Zamir *et al.* [27] developed Taskonomy, a computational framework for systematic discovery of task relationships in computer vision. The methodology trains task-specific models on a large-scale indoor scene dataset, then measures transferability by fine-tuning models pretrained on one task (source) to perform another task (target), quantifying transfer effectiveness through downstream task performance improvement. The resulting task affinity matrix reveals hierarchical structure: geometric tasks including depth estimation, surface normal prediction, and edge detection exhibit strong positive transfer, sharing low-level geometric feature representations, while semantic tasks such as object detection and scene classification form a separate cluster requiring distinct mid-level feature encodings. This empirical validation of task correlation structure motivates this thesis's experimental design, which evaluates decentralized FMTL on both strongly correlated task sets (NYU Depth V2 with depth, normals, segmentation) and weakly correlated task sets (Pascal Context with segmentation, human parts, edges) to assess how task relationship strength affects collaboration effectiveness.

Liu *et al.* [24] introduced Multi-Task Attention Networks (MTAN), employing attention mechanisms to enable task-specific feature routing within shared backbone networks. Each task learns soft attention masks applied to shared convolutional feature maps, selectively amplifying relevant channels while suppressing uninformative or conflicting channels. The rationale for attention-based selection stems from the hypothesis that different tasks utilize different subsets of the shared representation—depth estimation may rely heavily on texture gradients while segmentation prioritizes semantic context. MTAN achieved state-of-the-art results on NYU Depth V2 and CityScapes benchmarks, demonstrating that selective feature modulation reduces negative transfer compared to hard parameter sharing. However, the method assumes joint training with access to all task data simultaneously, precluding direct application to federated settings where task datasets remain distributed across clients.

Chen *et al.* [20] proposed GradNorm, addressing the distinct but related problem of balancing training dynamics across multiple tasks in joint optimization. When tasks exhibit different learning rates or loss scales, dominant tasks overwhelm shared parameters while minority tasks stagnate. GradNorm dynamically adjusts per-task loss weights to equalize gradient magnitudes across tasks, preventing any single task from monopolizing representational capacity. The method achieves improved balanced performance across tasks compared to uniform weighting. However, GradNorm does not address the task relationship discovery problem—it assumes all tasks share parameters and focuses on balancing their contributions rather than identifying which tasks should collaborate versus remain isolated.

Zamir *et al.* [37] extended task relationship exploitation through cross-task consistency losses that penalize disagreements between task predictions when geometric or semantic constraints dictate alignment. For instance, depth discontinuities should align with semantic segmentation boundaries, as both reflect object edges. Enforcing consistency through auxiliary losses encourages tasks to produce mutually coherent predictions, improving individual task performance beyond independent training. This approach effectively leverages known task relationships but requires domain-specific engineering of consistency constraints, limiting generalization to novel task combinations where relationships are unknown a priori.

These centralized methods provide foundational insights into task relationship structure and exploitation mechanisms. However, the assumption of centralized data access fundamentally contradicts federated learning's privacy-preservation mandate. The challenge for federated multi-task learning lies in discovering and leveraging task relationships using only information available through privacy-preserving communication protocols—namely, model parameters, gradients, or aggregate statistics—without direct access to raw client data.

3.2.2 Clustered Federated Learning

The fundamental challenge of statistical heterogeneity in federated learning arises when clients possess data drawn from different underlying distributions, rendering global model convergence suboptimal or impossible under standard averaging protocols. Clustered federated learning addresses this heterogeneity by partitioning the client population into groups exhibiting similar data characteristics, thereby enabling specialized models to emerge within each cluster while maintaining the federated paradigm's privacy guarantees. The rationale for clustering stems from the observation that averaging model updates from clients with divergent data distributions produces a compromise model that performs poorly across all clients—a phenomenon formally characterized as negative transfer in multi-task learning literature. This section distinguishes between hard clustering approaches, which enforce disjoint client partitions, and soft clustering methods, which permit weighted participation across multiple clusters.

Hard clustering methods impose a bijective mapping from clients to clusters, partitioning the client space into disjoint subsets. This architectural choice simplifies cluster model maintenance and reduces communication overhead, as each client exchanges information exclusively with its designated cluster's aggregation server. However, the rigidity of disjoint partitions introduces fundamental limitations when client data exhibits partial overlap across multiple distribution modes.

The pioneering work of Sattler *et al.* [6] introduced the Clustered Federated Learning (CFL) framework, which employs iterative gradient-based bi-partitioning to discover client clusters. Following each communication round, the central server computes pairwise cosine similarities between client gradient vectors, constructing a similarity matrix that encodes the alignment between client optimization trajectories. The method then applies spectral clustering with recursive bi-partitioning, iteratively splitting clusters when intra-cluster gradient diversity exceeds a predefined threshold. The rationale for gradient-based similarity rests on the principle that clients optimizing similar objectives produce aligned descent directions in parameter space. While CFL demonstrated effectiveness in identifying distribution shifts across synthetic federated datasets, the approach exhibits three critical dependencies: centralized gradient collection at the server (contradicting privacy-preservation goals in sensitive applications), reliance on manual threshold tuning for partition decisions, and inability to model partial task relationships where clients benefit from multiple knowledge sources.

Building upon CFL's foundation, Ghosh *et al.* [38] developed the Iterative Federated Clustering Algorithm (IFCA), which reformulates cluster discovery as a client-driven model selection problem. IFCA maintains K distinct cluster models at the central server, broadcasting all K models to clients each round. Clients evaluate each cluster model on local validation data, selecting the model yielding minimum validation loss and contributing updates exclusively to that cluster. This selection mechanism provides an implicit clustering criterion based on empirical performance rather than gradient geometry. The theoretical advantage lies in automatic cluster discovery without manual hyperparameter tuning—cluster assignments emerge organically from clients' local optimization landscapes. However, IFCA inherits hard clustering's fundamental limitation: a client training on multiple related tasks (e.g., depth estimation and surface normal prediction in indoor scenes) must commit entirely to one cluster, forfeiting potential knowledge transfer from a complementary cluster containing clients optimizing a related but distinct task (e.g., semantic segmentation).

Recent advances in gradient-based partitioning [39] and unified clustering frameworks [40] have refined hard clustering efficiency through hierarchical structures and privacy-preserving protocols [41], yet the categorical nature of cluster assignments persists. Hard clustering fundamentally suffers from three interrelated deficiencies. First, the inflexibility of discrete assignments prevents clients from leveraging complementary knowledge distributed across multiple clusters. Consider a multi-task scenario where Client A optimizes depth estimation and semantic segmentation while Client B optimizes surface normals and edge detection—the tasks exhibit pairwise affinities (depth-normals share geometric structure, segmentation-edges share boundary information) that a single cluster assignment cannot capture. Second, static cluster memberships fail to adapt to temporal dynamics in task relationships. Empirical evidence from centralized multi-task learning suggests that early training phases benefit from broad feature sharing across tasks (low-level edge and texture detectors generalize universally), while later phases require task-specific specialization. Hard clustering with fixed assignments cannot exploit this evolving relationship structure. Third, all surveyed hard clustering methods rely on centralized similarity computation, cluster assignment, and model maintenance, reintroducing the single point of failure and privacy concentration that decentralized architectures aim to eliminate.

These approaches implicitly assume that grouping clients based on similarity metrics yields meaningful task relationships, without explicitly modeling task semantics or accounting for their potential evolution over time.

3.2.3 Soft and Dynamic Task Clustering

Soft clustering relaxes the constraint of disjoint client partitions, instead representing cluster membership through continuous weight distributions. This architectural shift enables clients to participate simultaneously in multiple clusters with varying degrees of contribution, effectively modeling partial task relationships through weighted model combinations. The theoretical justification for soft clustering derives from mixture model theory, where each client's optimal model is represented as a convex combination of cluster prototypes, with mixing weights reflecting the degree of distribution alignment between the client and each cluster.

Ruan and Joe-Wong [42] proposed FedSoft, pioneering the application of soft clustering to federated learning. Each client receives all cluster models from the server and computes a weighted average based on validation performance, assigning higher weights to models yielding lower local validation loss. This weighting scheme provides a differentiable approximation to IFCA's discrete selection while enabling smooth transitions as task relationships evolve during training. The rationale for validation-based weighting rests on the assumption that models performing well on a client's local validation set capture distribution characteristics aligned with that client's data generating process. However, FedSoft retains centralized cluster model maintenance—the server aggregates client contributions to each cluster and broadcasts updated cluster models, preserving the star topology characteristic of centralized federated learning. Additionally, the method treats clustering as orthogonal to the multi-task learning problem, focusing on data distribution heterogeneity rather than explicit task heterogeneity where clients optimize fundamentally different objectives.

Contemporary soft clustering research has explored hierarchical structures [43], enhanced privacy protocols [44], and dynamic membership updates [45], yet these advances remain confined to centralized orchestration. Soft clustering provides two critical advantages over hard methods. First, partial collaboration allows clients to extract knowledge from multiple sources simultaneously, avoiding the forced choice between complementary but distinct task clusters. Second, continuous weight adjustments enable smooth adaptation as task relationships evolve, contrasting with the abrupt transitions induced by discrete cluster reassignments in hard methods. Despite these improvements, existing soft clustering approaches fundamentally operate within centralized architectures where the server computes or coordinates weight assignments. Moreover, the literature predominantly addresses data distribution heterogeneity under the assumption of task homogeneity—all clients optimize the same objective function on differently distributed data. The scenario of explicit task heterogeneity, where clients optimize distinct loss functions corresponding to different learning objectives (e.g., depth estimation versus semantic segmentation), remains largely unexplored in the soft clustering literature. This thesis addresses both gaps simultaneously by developing fully decentralized soft clustering mechanisms specifically designed for multi-task scenarios where clients optimize heterogeneous objectives without centralized coordination.

While soft and dynamic clustering improves flexibility compared to static grouping, such methods still rely on clustering as a surrogate for task relationships, rather than directly reasoning about task objectives or loss interactions.

3.3 Aggregation Strategies for Heterogeneous Objectives

As highlighted in Chapter 2, negative transfer in federated multi-task learning often manifests during model aggregation, where updates from heterogeneous objectives are combined without explicit coordination. Consequently, aggregation strategies play a central role in determining whether collaboration across tasks is beneficial or detrimental.

3.3.1 Task-Agnostic Aggregation

Aggregation mechanisms determine how information from multiple clients is combined in federated learning. The canonical approach is Federated Averaging (FedAvg), introduced by McMahan *et al.* [1], which computes a weighted average of client updates with weights proportional to local dataset sizes. This strategy assumes that all clients optimize the same objective function and that larger datasets provide more reliable gradient estimates, assumptions that hold under independent and identically distributed (IID) data but are frequently violated in real-world federated settings.

Numerous extensions of FedAvg aim to improve optimization stability and convergence while retaining task-agnostic aggregation. FedProx [12] introduces a proximal regularization term to mitigate client drift caused by system heterogeneity, such as varying computation capabilities or numbers of local updates. Similarly, adaptive server-side optimization methods, including FedAdam, FedYogi, and FedAdagrad [46], incorporate momentum and adaptive learning rates to accelerate convergence and stabilize training. While effective in addressing system-level and optimization-related challenges, these methods preserve simple averaging of client contributions and do not differentiate between clients based on task similarity or objective compatibility.

In federated multi-task and task-heterogeneous settings, task-agnostic aggregation becomes fundamentally limited. When clients optimize different objectives, uniform averaging disregards task structure and may combine incompatible gradients, resulting in suboptimal compromise updates or negative transfer. Importantly, these limitations persist even when tasks are related, as gradient alignment can vary across clients and training stages. These observations motivate aggregation strategies that explicitly account for task relationships and gradient compatibility, which are discussed next.

3.3.2 Conflict-Aware and Task-Aware Aggregation

Kohler [9] introduced Hyper Conflict-Averse (HCA) aggregation to mitigate gradient conflicts in centralized federated multi-task learning. HCA extends conflict-averse optimization principles from centralized multi-task learning [10] to federated settings, addressing cases where naive averaging yields suboptimal updates when client gradients are negatively aligned. Conceptually, negative inner products $g_i^\top g_j < 0$ indicate conflicting descent directions; conflict-averse methods therefore modify gradients to reduce interference while preserving positive alignment, typically via projection-based operations.

In HCA, client i aggregates updates from a neighbor set $\mathcal{N}_i$ by assigning higher weights to neighbors whose gradients align with its own gradient direction. The update can be written abstractly as

$$w_i^{(t+1)} = w_i^{(t)} - \eta \sum_{j \in \mathcal{N}_i} \alpha_{ij} \, \mathrm{ConflictResolve}(g_i, g_j), \tag{3.1}$$

where α_{ij} is a similarity-based weight (e.g., a function of gradient cosine similarity) and $\mathrm{ConflictResolve}(\cdot)$ denotes a projection operator that reduces interference when gradients conflict. Kohler provides convergence guarantees under standard smoothness and bounded-variance assumptions and reports empirical reductions in negative transfer compared to FedAvg on heterogeneous benchmarks.

Related decentralized frameworks such as ColNet [26] also incorporate task-aware and conflict-mitigating collaboration mechanisms, but typically rely on predefined task identities and static group structures rather than dynamically inferring relationships during training.

A practical limitation of conflict-aware aggregation methods is numerical and systems instability absent in simple averaging. First, normalization operations may involve division by small gradient norms, which can amplify noise or lead to undefined behavior. Second, projection-based conflict resolution can increase gradient magnitudes when resolving near-orthogonal updates, potentially causing gradient explosion. Third, once NaN/Inf values appear, they can rapidly propagate through subsequent computations and corrupt model parameters. These issues are particularly challenging in decentralized settings, where no central coordinator can monitor global statistics or intervene when instability occurs. Consequently, deploying conflict-aware aggregation in fully decentralized multi-task learning typically requires additional stabilization and fallback mechanisms, which we detail in Chapter 4.

3.4 Decentralized Federated Learning

Decentralized federated learning removes centralized coordination by enabling peer-to-peer communication among clients. While this paradigm improves robustness and eliminates single points of failure, it also fundamentally alters how collaboration and coordination can be achieved, particularly in the presence of heterogeneous tasks.

3.4.1 Gossip-Based and Peer-to-Peer Learning

Lalitha *et al.* [3] established foundational theory for fully decentralized federated learning, analyzing convergence properties of gossip-based averaging protocols on arbitrary network topologies. The framework assumes clients communicate with direct neighbors according to a connectivity graph, exchanging model parameters and performing local averaging without global coordination. Theoretical analysis proves that under connectivity assumptions (the communication graph forms a connected component) and bounded gradient variance, decentralized gossip algorithms converge to the centralized FedAvg solution at rates dependent on the graph's spectral properties—well-connected graphs with low diameter achieve faster consensus. However, the analysis assumes task homogeneity: all clients optimize identical objective functions on local data, rendering task relationship discovery unnecessary.

Recent decentralized FL surveys [47], [48] catalog advances in topology design, asynchronous communication protocols, Byzantine robustness, and communication compression. These works demonstrate that decentralized architectures can match centralized performance on standard IID benchmarks while providing improved fault tolerance and privacy preservation. However, the surveyed literature predominantly addresses optimization and systems challenges—convergence rates, communication efficiency, node failure recovery—under the standing assumption of objective homogeneity across clients.

3.4.2 Limitations for Multi-Task and Heterogeneous Settings

Existing decentralized federated learning research assumes task homogeneity, where all clients optimize the same objective function, rendering task relationship discovery unnecessary. When task heterogeneity is introduced—clients optimizing fundamentally different objectives such as depth estimation, semantic segmentation, and surface normal prediction—decentralized systems face three unresolved challenges. First, clients lack mechanisms to discover which neighbors optimize related tasks that warrant collaboration versus unrelated tasks requiring isolation. Standard gossip protocols treat all neighbors symmetrically, applying uniform aggregation weights regardless of task compatibility, leading to negative transfer when incompatible gradients are averaged. Second, existing decentralized aggregation methods employ simple averaging without conflict awareness, failing to detect and mitigate gradient conflicts that arise when heterogeneous tasks produce opposing descent directions. Third, the absence of centralized monitoring eliminates the ability to detect numerical instabilities (NaN/Inf values, gradient explosion) that sophisticated aggregation methods like HCA may introduce, requiring autonomous client-side stability mechanisms that prior work has not addressed.

A notable exception is ColNet [26], which explicitly considers task heterogeneity in decentralized federated learning by structuring collaboration around predefined task identities. ColNet groups clients according to known task assignments and enables cross-task interaction through conflict-averse aggregation mediated by designated group leaders. This framework demonstrates that task-aware collaboration is feasible in fully decentralized settings. However, ColNet assumes that task identities and task distributions are known a priori and remain static throughout training, and it does not address scenarios where task relationships must be discovered dynamically or where task similarity evolves over time.

This thesis addresses these limitations by developing decentralized dynamic task relationship discovery mechanisms, enabling clients to compute task similarity metrics locally (gradient-based, task overlap, cross-loss similarity) and adaptively select collaboration partners based on measured affinity. The integration of conflict-aware HCA aggregation into decentralized topologies, combined with comprehensive numerical stability protections, provides the first demonstration of decentralized federated multi-task learning on complex dense prediction tasks without centralized coordination.

3.5 Gap Analysis and Positioning of This Work

Despite substantial progress in federated learning, existing approaches address the challenges of task heterogeneity, task relationship discovery, aggregation stability, and decentralization only in isolation. As reviewed in the previous sections, personalization-based methods relax the single global model assumption but do not explicitly model relationships between heterogeneous tasks. Clustered and task-aware federated learning approaches introduce structured collaboration, yet typically rely on centralized coordination or predefined task groupings. Decentralized federated learning frameworks remove central orchestration but largely assume homogeneous objectives and single-task optimization. This leaves a gap at the intersection of decentralized learning, dynamic task relationship discovery, and stable multi-task aggregation.

Engineering Foundation: Nicolas Kohler's Framework. Nicolas Kohler's master's thesis [9] provides an implementation-oriented foundation for federated multi-task learning with personalized architectures. The framework introduces a clear separation between shared representations and task-specific components and demonstrates how task similarity can be exploited to structure collaboration in a centralized federated setting. By integrating conflict-aware aggregation mechanisms, the framework offers a practical and extensible codebase for studying task heterogeneity in federated learning. However, the proposed system assumes centralized coordination and static task relationships, limiting its applicability in decentralized environments where task similarities are unknown a priori.

Theoretical Foundation: ColNet. Complementary to this engineering perspective, the ColNet framework proposed by Feng *et al.* [26] provides a conceptual and architectural foundation for decentralized federated multi-task learning. ColNet demonstrates that task-aware collaboration is feasible in fully decentralized settings by grouping clients according to predefined task identities and enabling cross-task interaction through conflict-averse aggregation mediated by group leaders. This work establishes the viability of exploiting task similarity to structure collaboration without centralized coordination. At the same time, ColNet assumes that task identities and task distributions are known and fixed throughout training, and it does not address scenarios where task relationships must be inferred from observed training dynamics.

Positioning of the Present Work. The present thesis builds upon these two complementary foundations while addressing their shared limitations. In contrast to Nicolas Kohler's centralized framework, this work operates in fully decentralized peer-to-peer environments. In contrast to ColNet, it does not rely on predefined task identities or static task groupings. Instead, task relationships are dynamically inferred from observed training behavior and used to adapt collaboration and aggregation strategies over time. By combining decentralized communication, dynamic task relationship discovery, and stable aggregation under heterogeneous objectives, this work addresses a gap that remains unexplored by existing federated multi-task learning approaches. Table 3.1 summarizes this positioning relative to representative prior approaches across key design dimensions.

Table 3.1: Comparison of federated and decentralized multi-task learning approaches. Dec. (Decentralized), Dyn. (Dynamic Task Relationship Discovery), Dense (Dense Prediction Tasks), Cross (Cross-Dataset Validation), Stab. (Explicit Stability Considerations).

Method	**Dec.**	**Dyn.**	**Dense**	**Cross**	**Stab.**
FedAvg [1] (2017)	–	–	–	–	–
MOCHA [5] (2017)	–	–	–	–	–
FedPer [29] (2019)	–	–	–	–	–
pFedMe [31] (2020)	–	–	–	–	–
CFL [6] (2020)	–	Partial	–	–	–
FedSoft [42] (2021)	–	Partial	–	–	–
Kohler [9] (2024)	–	Partial	–	–	Partial
ColNet [26] (2025)	✓	–	–	–	Partial
This Thesis	✓	✓	✓	✓	✓

Chapter 4

Methodology

This chapter presents a decentralized federated multi-task learning framework. At a high level, each client performs local multi-task training, automatically discovers inter-task relatedness via dynamic similarity metrics, selects a small set of neighbors with similarity-proportional weights, and aggregates model parameters according to an explicit design space defined by (i) *aggregation scope* (backbone-only vs. full-model) and (ii) *aggregation method* (e.g., similarity-weighted averaging, FedAvg, or HCA). Implementation details are deferred to Chapter 5, and empirical validation is provided in Chapter 6.

This chapter formalizes the decentralized federated multi-task learning framework that constitutes the central contribution of this thesis. The presentation follows a structured analytical progression: Section 4.1 establishes the mathematical foundations of the decentralized network topology and multi-task model architecture, followed by a rigorous treatment of task assignment strategies that enable controlled heterogeneity. Section 4.2 introduces three complementary approaches for quantifying task relatedness in the absence of centralized coordination, including a novel cross-validation-based similarity metric. Section 4.3 formalizes the soft aggregation mechanism that enables continuous neighbor weighting rather than discrete cluster assignment. Section 4.4 examines the critical design decision of aggregation scope (backbone-only versus full model) and demonstrates how task correlation strength fundamentally determines the optimal choice—a counterintuitive finding that challenges conventional assumptions in multi-task federated learning. Section 4.5 presents the most substantial engineering contribution: a systematic solution to numerical instability in conflict-averse aggregation, documented through a complete record of diagnostic attempts and failures that ultimately led to a five-layer defense mechanism. Finally, Section 4.6 describes efficiency improvements through adaptive learning rate scheduling and early stopping criteria.

4.1 System Model

Figure 4.1 summarizes the per-round workflow of the proposed decentralized framework, which will be formalized in the subsequent sections.

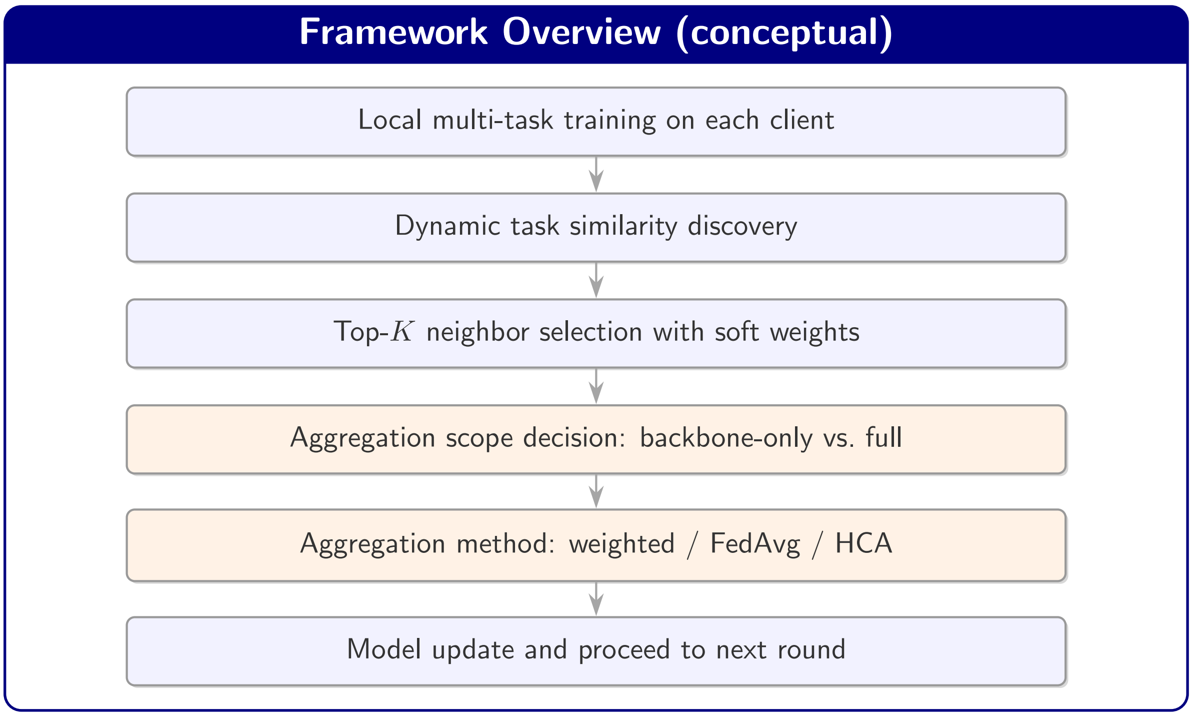

Framework Overview (conceptual)
Local multi-task training on each client
Dynamic task similarity discovery
Top-K neighbor selection with soft weights
Aggregation scope decision: backbone-only vs. full
Aggregation method: weighted / FedAvg / HCA
Model update and proceed to next round

Figure 4.1: End-to-end workflow of the proposed framework in one communication round.

The foundational architecture of this work extends the centralized federated multi-task learning framework introduced by [9] into a fully decentralized setting where peer-to-peer collaboration replaces server-mediated aggregation. This architectural shift necessitates fundamental reconsideration of coordination mechanisms, similarity computation, and aggregation strategies.

Notation. Table 4.1 summarizes the key symbols used throughout this chapter.

4.1.1 Network Architecture and Communication Topology

This research considers a fully decentralized peer-to-peer network comprising N autonomous clients denoted $\{C_1, C_2, \ldots, C_N\}$, where the absence of a central coordination server distinguishes this architecture from conventional federated learning systems such as FedAvg [1]. The rationale for eliminating centralized coordination extends beyond mere fault tolerance; decentralized topologies fundamentally alter the information propagation dynamics and enable task-adaptive collaboration patterns that would require prohibitive coordination overhead in server-mediated architectures.

Each client C_i maintains three essential components that enable autonomous participation in the collaborative learning process. First, a local dataset $\mathcal{D}_i = \{(x_j, y_j)\}_{j=1}^{n_i}$ provides task-specific training data, where input images $x_j \in \mathbb{R}^{H\times W\times 3}$ map to labels y_j whose dimensionality and semantic interpretation vary according to the assigned task (scalar depth values, categorical segmentation masks, or unit-norm surface normal vectors). Second, a

Table 4.1: Key notation used throughout this chapter.

Symbol	**Meaning**
C_i	Client (peer) i
$\mathcal{D}_i$	Local dataset of client i
$\mathcal{T}$	Set of tasks; $t \in \mathcal{T}$ indexes a task
$\mathcal{G}^{(t)} = (\mathcal{V}, \mathcal{E}^{(t)})$	Communication graph at round t
$\mathcal{N}_i^{(t)}$	Candidate neighbor set of client i at round t
K	Neighbor budget for Top-K selection
$\theta_i = (\theta_i^b, \theta_i^h)$	Local model on client i (backbone + head(s))
θ_i^b	Backbone parameters of client i
$\theta_{i,t}^h$	Task head parameters on client i for task t (if present)
$g_i^{(t)}$	Backbone gradient on client i at round t
$\mathrm{sim}_{ij}^{(t)}$	Combined similarity score between clients i and j at round t
α	Convex weight for combining similarity components
$\mathcal{S}_i^{(t)}$	Selected Top-K neighbor subset for client i at round t
$s_{ij}^{(t)}$	Non-negative similarity score (after truncation)
$w_{ij}^{(t)}$	Aggregation weight assigned by client i to neighbor j
s	Aggregation scope (Backbone or Full)
$\mathcal{A}$	Aggregation operator (e.g., weighted / FedAvg / HCA)
T	Total number of communication rounds
E	Local optimization steps per round
T_w	Warm-up rounds before switching to an advanced operator
ϵ	Small constant for numerical stability

local model $\mathcal{M}_i$ parameterized by weights θ_i embodies the client's current hypothesis about the underlying task relationships. Third, bidirectional communication channels enable direct parameter exchange with any other client in the network, forming the substrate for decentralized aggregation. The capacity for arbitrary pairwise communication distinguishes this architecture from structured topologies such as ring networks or hierarchical federations studied in prior decentralized learning work [3], [4].

The network topology emerges dynamically through task-similarity-driven neighbor selection rather than being predetermined by network engineering constraints. This design choice reflects a fundamental hypothesis: that collaboration effectiveness depends primarily on task compatibility rather than communication efficiency. In each training round $t \in \{1, 2, \ldots, T\}$, clients autonomously compute pairwise task similarities (formalized in Section 4.2) and select the K most compatible partners for aggregation. The resulting topology adapts continuously throughout training as task relationships evolve from shared low-level feature learning in early rounds to specialized pattern recognition in later stages. Figure 4.2 illustrates this adaptive reconfiguration: early rounds exhibit dense connectivity as most tasks benefit from shared low-level features, while later rounds show clustered structure as specialized task groups emerge. This dynamic reconfiguration mechanism, inspired by adaptive clustering approaches in federated learning [6], [38], enables the network to discover implicit task groupings without centralized coordination.

The training protocol proceeds through synchronized rounds, where synchronization emerges

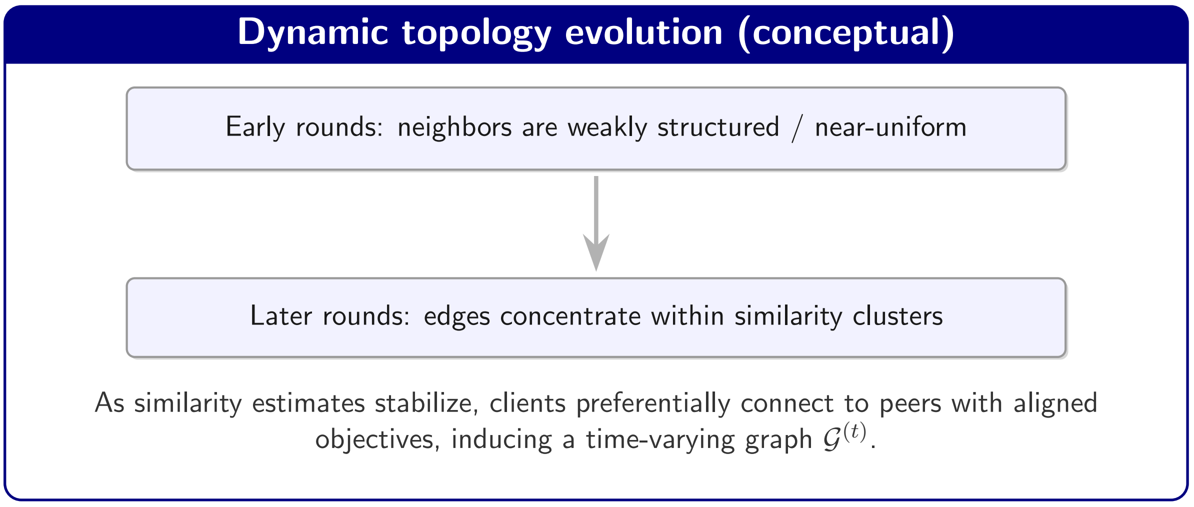

Dynamic topology evolution (conceptual)
Early rounds: neighbors are weakly structured / near-uniform
Later rounds: edges concentrate within similarity clusters
As similarity estimates stabilize, clients preferentially connect to peers with aligned objectives, inducing a time-varying graph $\mathcal{G}^{(t)}$.

Figure 4.2: Illustration of how similarity-guided neighbor selection induces a dynamic communication topology over training rounds.

from coordinated round advancement rather than central orchestration. Within round t, each client C_i executes a five-stage process. Local training constitutes the first stage: the client performs E epochs of stochastic gradient descent on dataset $\mathcal{D}_i$ using the current model parameters $\theta_i^{(t)}$, producing updated parameters $\theta_i^{(t+\text{local})}$. The second stage computes task similarity: client C_i evaluates its compatibility with all other clients using one of three similarity metrics detailed in Section 4.2, generating similarity scores $\{\text{sim}(i,j)\}_{j \neq i}$. Neighbor selection follows as the third stage: the client identifies its top-K most similar peers, forming the aggregation neighborhood $\mathcal{N}_i^{(t)} = \text{TopK}(\{\text{sim}(i,j) : j \neq i\}, K)$. The fourth stage involves parameter exchange: client C_i transmits its model parameters $\theta_i^{(t+\text{local})}$ to selected neighbors while receiving their parameters $\{\theta_j^{(t+\text{local})}\}_{j \in \mathcal{N}_i^{(t)}}$. Finally, aggregation synthesizes received information: the client computes a weighted combination of neighbor parameters using either similarity-weighted averaging or conflict-averse optimization (Section 4.4), yielding the parameters $\theta_i^{(t+1)}$ for the subsequent round.

This architectural design achieves three critical properties that enable effective decentralized multi-task learning. First, it eliminates single points of failure inherent in centralized systems [1], distributing coordination responsibilities across all participants. Second, it enables task-adaptive collaboration by allowing similarity-driven neighbor selection to discover implicit task clusters without requiring predefined taxonomies. Third, it supports heterogeneous task assignments by accommodating clients training on different task combinations or weightings, unlike homogeneous decentralized learning systems that assume identical objectives across participants [4].

4.1.2 Multi-Task Model Architecture

The model architecture implements hard parameter sharing [16], [17], decomposing the prediction function into a shared feature extractor and task-specific prediction heads. This architectural choice reflects the hypothesis that dense prediction tasks—despite producing outputs of different dimensionalities and semantic types—benefit from shared low-level visual representations while requiring specialized decoders for task-specific output formatting.

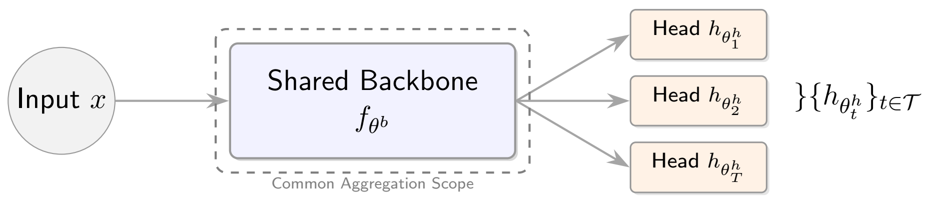
The shared backbone, denoted $f_{\theta_b} : \mathbb{R}^{H \times W \times 3} \to \mathbb{R}^{H' \times W' \times C}$, transforms input images into spatial feature representations suitable for dense prediction. This research employs ResNet-18 [49] pretrained on ImageNet [50] as the backbone architecture, a decision motivated by three complementary considerations. First, the fully convolutional structure of ResNet-18 preserves spatial resolution through the network depth, enabling direct application to dense prediction tasks without architectural modification. Second, the parameter efficiency (11.7M parameters) balances representational capacity against communication overhead in federated settings where model transmission costs scale linearly with parameter count. Third, ImageNet pretraining provides initialization in a semantically meaningful region of parameter space, accelerating convergence on downstream dense prediction tasks compared to random initialization. The pretrained weights serve as a warm start that reduces the number of federated rounds required to achieve target performance, directly addressing communication efficiency concerns in distributed learning [11].

For client C_i training on task subset $\mathcal{T}_i \subseteq \mathcal{T}$, where $\mathcal{T}$ denotes the universal task set, the backbone parameters θ_b remain shared across all tasks in $\mathcal{T}_i$. This sharing enforces a representational bottleneck that encourages the learning of task-agnostic features—visual patterns that support multiple prediction objectives simultaneously. The multi-task learning literature [17], [24] suggests that this architectural constraint acts as an implicit regularizer, preventing task-specific overfitting by requiring the backbone to discover generalizable visual concepts.

Task-specific prediction heads map the shared feature representation to task-appropriate output spaces. Each task $t \in \mathcal{T}$ defines a decoder $h_{\theta_t} : \mathbb{R}^{H' \times W' \times C} \to \mathbb{R}^{H \times W \times D_t}$ that upsamples and transforms features into the required output dimensionality D_t. The architectural instantiation varies according to the statistical nature and semantic interpretation of each task. Depth estimation, formalized as a regression problem over normalized depth values in $[0, 1]$, employs a decoder comprising three transposed convolutional layers with progressively decreasing spatial stride, culminating in a sigmoid activation that enforces the unit interval constraint. The output dimensionality satisfies $D_{\text{depth}} = 1$, producing a single scalar depth value per pixel. Surface normal prediction, similarly formulated as regression but over unit vectors in $\mathbb{R}^3$, utilizes an architecturally identical decoder but replaces the final sigmoid with hyperbolic tangent activation followed by ℓ_2 normalization, ensuring that predicted vectors satisfy the unit norm constraint $\|n\|_2 = 1$ required for geometric validity. The output dimensionality expands to $D_{\text{normal}} = 3$ to accommodate the three-dimensional vector representation.

Semantic segmentation, in contrast, formulates prediction as pixel-wise multi-class classification over a predefined label taxonomy. The decoder architecture mirrors the regression heads in its use of transposed convolutions for spatial upsampling but diverges in the final layer, which produces $D_{\text{seg}} = C_{\text{seg}}$ logit channels where C_{seg} denotes the number of semantic classes (13 for NYU Depth V2 [7], 59 for Pascal Context [8]). A softmax activation $\sigma(\cdot)$ converts logits to a categorical distribution over class labels, enabling maximum likelihood training through cross-entropy loss. Edge detection simplifies to binary classification with $D_{\text{edge}} = 1$, employing sigmoid activation to produce edge probabilities in $[0, 1]$.

Multi-task model structure (conceptual)



A single feature extractor is shared across tasks, while each task is equipped with a lightweight head. Only parameters within the chosen aggregation scope are exchanged and aggregated.

Figure 4.3: Hard parameter sharing for multi-task dense prediction: a shared backbone with task-specific heads.

Human parts segmentation extends the semantic segmentation architecture to anatomical decomposition with $D_{\text{parts}} = C_{\text{parts}}$ channels ($C_{\text{parts}} = 7$ for Pascal Context), again using softmax for probabilistic interpretation.

The complete model for client C_i composes the shared backbone with all task-specific heads corresponding to tasks in $\mathcal{T}_i$, formally expressed as:

$$\mathcal{M}_i(x) = \{h_{\theta_t}(f_{\theta_b}(x)) : t \in \mathcal{T}_i\} \tag{4.1}$$

This compositional structure induces a natural decomposition of the parameter space: $\theta_i = (\theta_b, \{\theta_t : t \in \mathcal{T}_i\})$, separating shared and task-specific components. Figure 4.3 illustrates this architecture for a representative client training on three dense prediction tasks (segmentation, depth, surface normals). The shared backbone processes the input image through ResNet-18 layers to produce spatial feature maps, which then diverge into task-specific prediction heads that map features to task-appropriate output spaces. This decomposition proves critical for enabling selective aggregation strategies (Section 4.4), where clients may choose to exchange only backbone parameters while maintaining personalized task heads, or conversely, to aggregate both components simultaneously. The flexibility to partition the aggregation scope emerges as a key design variable whose optimal setting depends on task correlation structure, as demonstrated empirically in Section 4.4.1.

4.1.3 Task Assignment Strategies

Controlled heterogeneity in task distribution across clients enables systematic investigation of how task assignment patterns affect collaborative learning dynamics. This research evaluates three task assignment strategies that span a spectrum from complete task specialization to uniform multi-task learning, providing empirical evidence for the relative merits of different heterogeneity regimes.

The SingleTask strategy (designated A1 in experimental nomenclature) assigns each client to exactly one task with unit weight, creating complete task specialization. For a network with N clients and $|\mathcal{T}|$ tasks, this assignment produces $N/|\mathcal{T}|$ clients per task, where each client C_i trains with task weight vector $w_i \in \{0,1\}^{|\mathcal{T}|}$ satisfying $\|w_i\|_1 = 1$ and containing exactly one non-zero entry. In the context of NYU Depth V2 with three tasks and six clients, this strategy generates two clients dedicated to depth estimation with weights $(1.0, 0, 0)$, two to semantic segmentation with weights $(0, 1.0, 0)$, and two to surface normal prediction with weights $(0, 0, 1.0)$. The rationale for examining this extreme specialization regime stems from the hypothesis that it establishes a lower bound on collaborative learning benefit: if clients have no overlapping tasks, collaboration reduces to independent single-task federated learning within each task group. Any performance gain observed under less specialized assignment strategies can therefore be attributed to cross-task knowledge transfer. Experimentally, SingleTask serves as a baseline measuring the counterfactual performance achievable without multi-task learning, isolating the contribution of task diversity to model quality.

The Pairwise strategy (designated A2) introduces controlled task overlap by assigning each client two tasks with asymmetric weights: a primary task receiving weight $w_{\text{primary}} = 0.7$ and a secondary task receiving $w_{\text{secondary}} = 0.3$. This 70-30 split reflects a deliberate design choice balancing two competing objectives. The dominant primary task weight ensures that each client develops specialized expertise in one domain, preventing the dilution of task-specific knowledge that can occur under uniform weighting. Simultaneously, the non-trivial secondary task weight (30%) provides sufficient gradient signal for the model to learn representations that support both tasks, enabling auxiliary knowledge transfer. The weight ratio of 7:3 emerged based on pilot tuning, suggesting that more extreme ratios (e.g., 9:1) provide insufficient secondary task signal, while more balanced ratios (e.g., 6:4) compromise primary task performance. For NYU Depth V2, a typical Pairwise assignment distributes clients across task pairs: depth-segmentation with weights $(0.7, 0.3, 0)$, depth-normal with weights $(0.7, 0, 0.3)$, and segmentation-normal with weights $(0, 0.7, 0.3)$, ensuring that each task appears as either primary or secondary across multiple clients. This strategy embodies the hypothesis that partial task overlap enables beneficial knowledge transfer without fully sacrificing specialization. The experimental evidence presented in Chapter 6 demonstrates that Pairwise consistently outperforms SingleTask, validating the multi-task learning assumption under controlled heterogeneity.

The MultiTask strategy (designated B4) implements uniform task distribution where all clients train on all tasks with equal weights $w_i = (1/|\mathcal{T}|, \ldots, 1/|\mathcal{T}|)$. For three tasks, this yields weights $(0.33, 0.33, 0.33)$, removing task specialization entirely. The rationale for this assignment follows the classical multi-task learning hypothesis [16], [17]: forcing the shared backbone to simultaneously support all tasks should encourage discovery of maximally general feature representations that capture commonalities across the entire task family. This strategy represents the opposite extreme from SingleTask, maximizing task diversity within each client rather than across clients. However, this uniformity introduces a critical vulnerability: when tasks exhibit weak correlation or conflicting gradient structure, the simultaneous optimization of multiple objectives can induce destructive interference in the shared parameters [10]. The experimental results on Pascal Context (Section 6.4) reveal that MultiTask underperforms Pairwise on datasets with weakly related tasks, suggesting that selective task pairing proves more effective than indiscriminate multi-task aggregation. This finding challenges the assumption that maximal task diversity universally improves representation learning, instead suggesting that task assignment should adapt to the underlying correlation structure of the problem domain.

The controlled variation across these three strategies enables ablation of task heterogeneity as an independent variable, isolating its effect from other design choices such as aggregation method or similarity metric. By comparing performance under SingleTask, Pairwise, and MultiTask while holding all other hyperparameters constant, this research establishes empirical evidence for the optimal task assignment regime under different dataset characteristics.

4.2 Dynamic Task Similarity Identification

The absence of centralized coordination in decentralized federated learning necessitates autonomous mechanisms for clients to estimate task relatedness without global knowledge of the task distribution or collaboration history. This section introduces three complementary approaches for quantifying client compatibility, each capturing different aspects of task similarity through distinct mathematical formalizations and computational procedures.

4.2.1 Gradient-Based Similarity via Cosine Alignment

The first similarity metric leverages the observation that clients training on related tasks tend to optimize in similar directions in parameter space, suggesting that gradient alignment provides a task-agnostic measure of collaboration potential. This approach draws inspiration from gradient-based clustering methods in federated learning [39] but adapts the formulation for decentralized neighbor selection rather than centralized cluster assignment.

Following local training in round t, client C_i computes the gradient of its loss function with respect to the shared backbone parameters:

$$g_i^{(t)} = \nabla_{\theta_b} \mathcal{L}_i(\theta_b^{(t)}; \mathcal{D}_i) \tag{4.2}$$

where $\mathcal{L}_i$ denotes the weighted combination of task-specific losses corresponding to client i's task assignment, and $\theta_b^{(t)}$ represents the backbone parameters after local SGD updates. The rationale for restricting gradient computation to backbone parameters rather than the full model stems from two considerations. First, task-specific head parameters lack semantic correspondence across clients training on different tasks—comparing depth head gradients with segmentation head gradients yields meaningless similarity scores due to incompatible output spaces. Second, limiting gradient transmission to backbone parameters (approximately 11.7M parameters for ResNet-18) reduces communication overhead compared to transmitting full model gradients, directly addressing bandwidth constraints in federated settings [11].

After broadcasting gradient $g_i^{(t)}$ to all other clients and receiving their gradients $\{g_j^{(t)}\}_{j\neq i}$, client C_i computes pairwise cosine similarity:

$$\text{sim}_{\text{grad}}(i,j) = \frac{g_i^{(t)} \cdot g_j^{(t)}}{\|g_i^{(t)}\|_2 \|g_j^{(t)}\|_2} \tag{4.3}$$

The cosine similarity metric normalizes by gradient magnitude, ensuring that similarity scores reflect directional alignment rather than absolute gradient scale. This normalization proves critical in multi-task settings where different tasks induce vastly different gradient magnitudes due to varying loss scales and task complexities. For instance, semantic segmentation with cross-entropy loss over 13-59 classes typically produces gradients with ℓ_2 norm on the order of 10^2–10^3, while depth regression with mean squared error yields gradients with norm on the order of 10^{-1}–10^0. Without normalization, magnitude differences would dominate the similarity computation, incorrectly suggesting that all segmentation clients are highly similar regardless of actual directional alignment.

High positive similarity ($\text{sim}_{\text{grad}} \approx 1$) indicates that clients optimize in nearly identical directions, suggesting that aggregating their parameters will produce a model that improves both clients' objectives simultaneously. Conversely, negative similarity ($\text{sim}_{\text{grad}} < 0$) signals conflicting optimization directions where aggregation would force a compromise between incompatible objectives, likely degrading performance for both clients. Near-zero similarity ($|\text{sim}_{\text{grad}}| \approx 0$) suggests orthogonal gradients, indicating that clients are optimizing independent features with neither positive nor negative transfer expected.

The gradient-based approach captures task relationships implicitly through optimization dynamics rather than explicitly through task labels or dataset statistics. This implicit characterization enables discovery of unexpected task affinities that may not be apparent from semantic task descriptions. For example, depth estimation and surface normal prediction operate on different output formats (scalar fields versus vector fields) and involve different loss functions (mean squared error versus cosine distance), yet they produce highly aligned gradients because both tasks learn geometric boundary features. The gradient similarity metric automatically detects this affinity without requiring hand-crafted task relationship rules. Similarly, two segmentation tasks with different class taxonomies may exhibit conflicting gradients despite sharing identical output formats, and gradient similarity correctly identifies this incompatibility.

The computational complexity of gradient-based similarity scales as $\mathcal{O}(N^2P)$ where N denotes the number of clients and P represents the backbone parameter count, since each of the N clients must compute $N-1$ pairwise dot products over P-dimensional vectors. However, this computation parallelizes naturally across client pairs and requires only a single forward-backward pass per client per round, making it tractable for moderate network sizes and modern hardware accelerators. The communication cost involves transmitting $N \times P$ total parameters per round (each client broadcasts one gradient vector), which remains manageable given typical backbone sizes and network bandwidths.

4.2.2 Task Overlap Similarity via Weight Vector Intersection

The second similarity metric adopts a complementary perspective, measuring task relatedness through explicit task assignment overlap rather than implicit optimization dynamics. This approach provides a prior belief about collaboration potential based on the deliberate task assignment strategy, independent of the current training state.

For client C_i with task weight vector $w_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,|\mathcal{T}|})$ where $w_{i,t} \in [0, 1]$ denotes the loss weight assigned to task t, the overlap similarity with client C_j computes the element-wise minimum:

$$\text{sim}_{\text{task}}(i, j) = \sum_{t=1}^{|\mathcal{T}|} \min(w_{i,t}, w_{j,t}) \tag{4.4}$$

This formulation measures the degree of task assignment intersection, with higher values indicating greater shared training objectives. The minimum operator ensures that similarity increases only when both clients assign non-negligible weight to the same task—if client i trains heavily on depth ($w_{i,\text{depth}} = 0.7$) while client j ignores depth ($w_{j,\text{depth}} = 0$), the minimum evaluates to zero regardless of client i's weight, correctly identifying no overlap for that task.

Under the SingleTask assignment strategy (Section 4.1.3), task overlap similarity reduces to binary indicator similarity: $\text{sim}_{\text{task}}(i, j) = 1$ if both clients train on the same task, and $\text{sim}_{\text{task}}(i, j) = 0$ otherwise. This binary structure partitions the network into disjoint task groups with no cross-group similarity, effectively replicating the clustering structure assumed in cluster-based federated learning [6]. For Pairwise assignments, the overlap metric produces graded similarity values reflecting partial task intersection. Consider two clients with weights $(0.7, 0, 0.3)$ (depth primary, normal secondary) and $(0.7, 0.3, 0)$ (depth primary, segmentation secondary). The overlap similarity computes $\min(0.7, 0.7) + \min(0, 0.3) + \min(0.3, 0) = 0.7 + 0 + 0 = 0.7$, capturing the shared primary task while recognizing the divergent secondary tasks. This graded similarity enables soft clustering where clients maintain varying degrees of affinity with multiple neighbors rather than belonging exclusively to a single cluster.

The rationale for including task overlap similarity alongside gradient-based metrics stems from the recognition that optimization dynamics provide only a local view of task compatibility, reflecting the current training state but potentially missing long-term task relationships. A client early in training may exhibit gradient misalignment with a related task simply due to poor initialization or transient optimization artifacts, leading gradient similarity to spuriously suggest incompatibility. Task overlap similarity provides a global prior that stabilizes neighbor selection against such transient effects, encoding the experimenter's hypothesis about which tasks should collaborate based on semantic relatedness or domain knowledge.

However, task overlap similarity suffers from fundamental limitations that motivate the inclusion of alternative metrics. First, it remains static throughout training, unable to adapt as task relationships evolve during the learning process. Early training typically focuses on low-level features (edges, textures, color gradients) that benefit nearly all dense prediction tasks, suggesting that broad collaboration may be optimal initially. Late training specializes on task-specific patterns (geometric consistency for depth, semantic object boundaries for segmentation), suggesting that narrow collaboration within task groups may be preferable. Task overlap similarity cannot capture this temporal evolution, treating task relationships as fixed. Second, it relies entirely on predefined task categories and cannot discover unexpected relationships that transcend the task taxonomy. Two clients training on nominally different tasks (e.g., edge detection and depth estimation) may develop highly compatible representations due to shared emphasis on boundary features, but task overlap similarity assigns them zero affinity based solely on their distinct task labels.

4.2.3 Cross-Loss Similarity via Transferability Testing

The third similarity metric, introduced as a novel contribution of this thesis, directly measures knowledge transferability by evaluating each client's model on other clients' validation data. This approach sidesteps the indirection of gradient alignment or task label matching, instead empirically testing whether client j's learned representations actually improve client i's prediction accuracy on held-out data.

The cross-loss evaluation protocol proceeds as follows. After local training in round t, client C_i requests model parameters (backbone and task heads) from all other clients, receiving $\{\theta_j^{(t)}\}_{j \neq i}$. For each neighbor candidate j, client C_i evaluates model $\mathcal{M}_j$ parameterized by $\theta_j^{(t)}$ on its own validation dataset $\mathcal{D}_i^{\text{val}}$, computing the loss:

$$\text{CrossLoss}(j \to i) = \mathcal{L}_i(\theta_j^{(t)}; \mathcal{D}_i^{\text{val}}) \tag{4.5}$$

where $\mathcal{L}_i$ denotes client i's task-specific loss function evaluated on validation data. Critically, this evaluation uses client i's task heads (or the corresponding heads from client j's model if they share tasks) to compute task-specific predictions, ensuring that the loss calculation remains well-defined even when clients train on different task subsets.

Lower cross-loss indicates that client j's model generalizes well to client i's task and data distribution, suggesting that the representations learned by client j capture relevant features for client i's objective. This transferability signal provides direct empirical evidence of positive knowledge transfer potential. To convert cross-loss into a similarity metric where higher values indicate greater affinity (consistent with gradient and task overlap similarities), the formulation applies negation:

$$\text{sim}_{\text{cross}}(i, j) = -\text{CrossLoss}(j \to i) \tag{4.6}$$

This transformation ensures that clients with mutually low cross-loss (high transferability) receive high similarity scores, aligning the directional interpretation across all three metrics.

Cross-loss similarity offers several advantages over gradient-based and task overlap approaches that justify its introduction despite increased computational cost. First, it measures transferability directly rather than through proxy variables: instead of inferring collaboration potential from gradient alignment or task labels, it empirically tests whether aggregating client j's parameters would actually improve client i's model. This directness eliminates assumptions about the relationship between gradient alignment and transfer learning effectiveness. Second, it automatically accounts for task-specific loss scales and output format differences without requiring manual normalization. Depth estimation with mean squared error, segmentation with cross-entropy, and normal prediction with cosine distance all produce losses on different numerical scales, yet cross-loss similarity compares them consistently by evaluating actual prediction quality rather than abstract geometric properties like gradient angles. Third, it discovers implicit task relationships that may not manifest in gradient alignment or task labels. For example, a client training on depth estimation may develop boundary-sensitive features that improve segmentation performance even though the gradient directions differ due to distinct loss functions. Cross-loss similarity automatically detects this beneficial relationship through empirical transferability testing.

The primary disadvantage of cross-loss similarity lies in its computational and communication overhead. Computing cross-loss requires clients to receive complete models (backbone and task heads) from all neighbors, transmitting $\mathcal{O}(N \times |\theta|)$ parameters per round where $|\theta|$ denotes the total model size. In contrast, gradient similarity requires transmitting only backbone gradients, reducing communication by a factor equal to the ratio of total model size to backbone size. Additionally, cross-loss evaluation necessitates $N-1$ forward passes through neighbor models on validation data, multiplying inference cost by network size. To mitigate these overheads while preserving the transferability signal, this research employs two optimizations. First, cross-loss computation occurs periodically (e.g., every 5 rounds) rather than every round, amortizing the cost across multiple training iterations and relying on gradient similarity for intermediate rounds. Second, validation batches are limited to 16-32 samples rather than evaluating on the full validation set, trading slight noise in cross-loss estimates for substantial computational savings.

4.2.4 Combined Similarity via Convex Weighting

Gradient alignment and task overlap capture complementary aspects of task relatedness: gradient-based similarity reflects dynamic optimization compatibility, while task overlap encodes static assignment structure. A hybrid similarity metric is therefore defined by a convex combination:

$$\mathrm{sim}(i,j;\alpha) = \alpha \cdot \mathrm{sim}_{\mathrm{task}}(i,j) + (1-\alpha) \cdot \mathrm{sim}_{\mathrm{grad}}(i,j)\,, \tag{4.7}$$

where the weighting parameter $\alpha \in [0,1]$ controls the relative influence of task overlap versus gradient alignment. Setting $\alpha = 0$ recovers pure gradient similarity, relying entirely on optimization dynamics to guide neighbor selection, while $\alpha = 1$ recovers pure task-overlap similarity, treating task assignments as the sole determinant of collaboration potential. Intermediate values $\alpha \in (0,1)$ blend both signals, enabling neighbor selection that can respect both semantic task categories and the current training state.

In practice, different similarity components may live on different scales (e.g., overlap scores are discrete while gradient- or cross-loss-based scores are continuous). To keep

α interpretable and prevent any single component from dominating purely due to scale, per-component normalization is applied over the current candidate neighbor set (e.g., min–max or z-score normalization) prior to convex weighting.

This formulation enables systematic ablation of the task-overlap prior in similarity computation. Chapter 6 evaluates representative settings (e.g., $\alpha = 0.0$ for pure gradient similarity and $\alpha = 0.5$ for a balanced combination) and analyzes how the choice of α affects neighbor selection quality and downstream multi-task performance across datasets. In addition, Chapter 6 examines whether gradient-based similarity can capture latent relationships that are not explicitly encoded by task labels, by analyzing the evolution of similarity structure during training.

4.2.5 Temporal Dynamics and Adaptive Similarity Updates

Task relationships can evolve over the course of training, motivating periodic recomputation of similarity scores rather than reliance on a fixed neighbor set determined at initialization. In early *rounds*, optimization typically emphasizes generic, low-level feature extraction (e.g., edges, textures, and color cues) that benefits many dense prediction tasks, so compatibility may appear relatively broad across task pairs. As training proceeds, representations increasingly specialize toward task-specific structure, such as geometric consistency for depth estimation, semantic coherence for segmentation, or photometric cues for surface normals. This specialization can reduce similarity between weakly related tasks while preserving higher similarity within strongly related task groups.

To capture such temporal evolution, similarity metrics are recomputed at the beginning of each communication round. Concretely, for client i and neighbor candidate j, the similarity score $\mathrm{sim}^{(t)}(i, j)$ is computed using the most recent local model parameters $\theta_i^{(t)}$ and corresponding gradient estimates $g_i^{(t)}$ (and cross-loss measurements when applicable). This update schedule enables the collaboration graph to adapt over time: clients may start with broader neighborhoods and gradually emphasize a smaller set of highly compatible neighbors as specialization increases, while strongly related tasks may retain stable high-similarity connections.

This adaptive neighbor selection mechanism can be viewed as an implicit curriculum over collaboration structure, adjusting the diversity–specialization trade-off as training progresses. Chapter 6 evaluates dynamic similarity updates against a static baseline in which similarities are computed once at initialization and then held fixed, and analyzes the impact on convergence behavior and final performance.

4.3 Soft Aggregation Mechanism

Classical cluster-based federated learning algorithms [6], [38] assign each client to exactly one cluster, enforcing hard boundaries in the collaboration structure. This discrete assignment proves suboptimal in multi-task settings where task relationships exhibit gradual variation rather than sharp categorical distinctions. This section formalizes a soft aggregation mechanism that replaces binary cluster membership with continuous similarity-weighted neighbor contributions.

4.3.1 Neighbor Selection via Top-K Thresholding

Given pairwise similarity scores $\{\mathrm{sim}(i,j)\}_{j\neq i}$ computed using the similarity metrics described in Section 4.2, each client C_i selects its K most similar peers to form the aggregation neighborhood:

$$\mathcal{N}_i^{(t)} = \mathrm{TopK}\big(\{\mathrm{sim}^{(t)}(i,j) : j \neq i\}, K\big), \tag{4.8}$$

where $\mathrm{TopK}(\cdot, K)$ returns the indices of the K largest values. This thresholding operation serves two purposes. First, it bounds communication by limiting each client to exchange parameters with at most K neighbors per round, avoiding an all-to-all exchange pattern. Second, it deemphasizes weakly related clients whose inclusion in aggregation can increase the risk of negative transfer or gradient conflict.

The neighborhood size K controls the breadth–versus–depth trade-off in collaboration. Smaller values (e.g., $K = 2$) yield narrower neighborhoods that prioritize high-similarity exchanges and can reduce the risk of interference, at the cost of limited diversity. Larger values (e.g., $K = 5$) enable broader collaboration and potentially richer transfer, but may increase the chance of incorporating weakly related clients. In the remainder of this thesis, K is treated as a fixed hyperparameter and set to $K = 3$ in all experiments for consistency. In a six-client setting, $K = 3$ corresponds to aggregating from roughly half of the network, providing a pragmatic balance between focusing on highly related neighbors and maintaining sufficient diversity.

4.3.2 Similarity-Weighted Aggregation

Unlike hard clustering approaches that assign uniform aggregation weights within each cluster [38], soft aggregation computes client-specific weights proportional to similarity scores. After selecting neighbors $\mathcal{N}_i^{(t)}$, client C_i normalizes similarity values to obtain aggregation weights:

$$s_{ij}^{(t)} = \max\big(\mathrm{sim}(i,j), 0\big) \tag{4.9}$$

$$w_{ij}^{(t)} = \frac{s_{ij}^{(t)}}{\sum_{k\in\mathcal{N}_i^{(t)}\cup\{i\}} s_{ik}^{(t)} + \epsilon} \tag{4.10}$$

where $\epsilon > 0$ is a small constant for numerical stability. If $\sum_k s_{ik}^{(t)}$ is (near) zero—indicating no reliable positive similarity among the selected neighbors—we fall back to self-reliance by setting $w_{ii}^{(t)} = 1$ and $w_{ij}^{(t)} = 0$ for $j \neq i$.

The inclusion of client i itself in the normalization denominator ensures that $\sum_{j\in\mathcal{N}_i^{(t)}\cup\{i\}} w_{ij}^{(t)} = 1$, satisfying the convex combination constraint required for weighted averaging. The self-weight $w_{ii}^{(t)}$ quantifies how much client i trusts its own local update relative to neighbors' knowledge, implementing a form of personalization where clients with high self-similarity (relative to neighbors) rely primarily on local training while clients with lower self-similarity incorporate more external information.

This similarity-proportional weighting implements a key distinction from uniform averaging schemes like FedAvg [1]: clients whose gradients align strongly with the recipient receive higher influence in the aggregated model, while weakly aligned clients receive reduced influence. This differential weighting provides fine-grained control over knowledge transfer, enabling the aggregation mechanism to continuously interpolate between full averaging (when all neighbors have equal similarity) and selective aggregation (when one neighbor dominates similarity scores).

4.3.3 Parameter Update Formulation

The aggregated parameter update combines the client's local parameters with weighted neighbor contributions. This research considers two algebraically equivalent formulations that emphasize different geometric interpretations of the aggregation process. The displacement formulation expresses the update as a weighted movement from the current parameters toward neighbors:

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \eta \sum_{j \in \mathcal{N}_i^{(t)}} w_{ij}^{(t)} (\theta_j^{(t)} - \theta_i^{(t)}) \tag{4.11}$$

where the aggregation step size $\eta \in (0, 1]$ controls convergence speed. This formulation highlights the interpretation of aggregation as gradient descent on a consensus objective, where the consensus direction is determined by the weighted average of parameter differences. Setting $\eta < 1$ implements conservative updates that prevent abrupt parameter changes, which can stabilize training when similarity estimates contain noise or when neighbors have recently updated with conflicting gradients.

The weighted average formulation, obtained by algebraic rearrangement, expresses the update as a convex combination of all parameters:

$$\theta_i^{(t+1)} = w_{ii}^{(t)} \theta_i^{(t)} + \sum_{j \in \mathcal{N}_i^{(t)}} w_{ij}^{(t)} \theta_j^{(t)} \tag{4.12}$$

This perspective emphasizes the role of similarity-weighted averaging in smoothing the parameter distribution across the network, reducing variance in individual client models through ensemble effects. The convexity of the combination guarantees that aggregated parameters remain in the convex hull of participating client parameters, providing stability against outlier updates that might otherwise cause training divergence.

The soft aggregation mechanism generalizes both extremes of the federated learning spectrum. Setting $w_{ii}^{(t)} = 1$ and $w_{ij}^{(t)} = 0$ for all $j \neq i$ recovers pure local training without collaboration, establishing the lower bound on cooperation. Setting uniform weights $w_{ij}^{(t)} = 1/N$ for all j recovers full averaging across the entire network as in FedAvg [1], establishing the upper bound on homogeneity. Intermediate weighting schemes enable continuous interpolation between these extremes, allowing the aggregation mechanism to automatically adapt the cooperation level to the observed task similarity structure. Figure 4.4 illustrates the complete soft aggregation pipeline for a representative client, showing how similarity scores translate into weighted parameter updates.

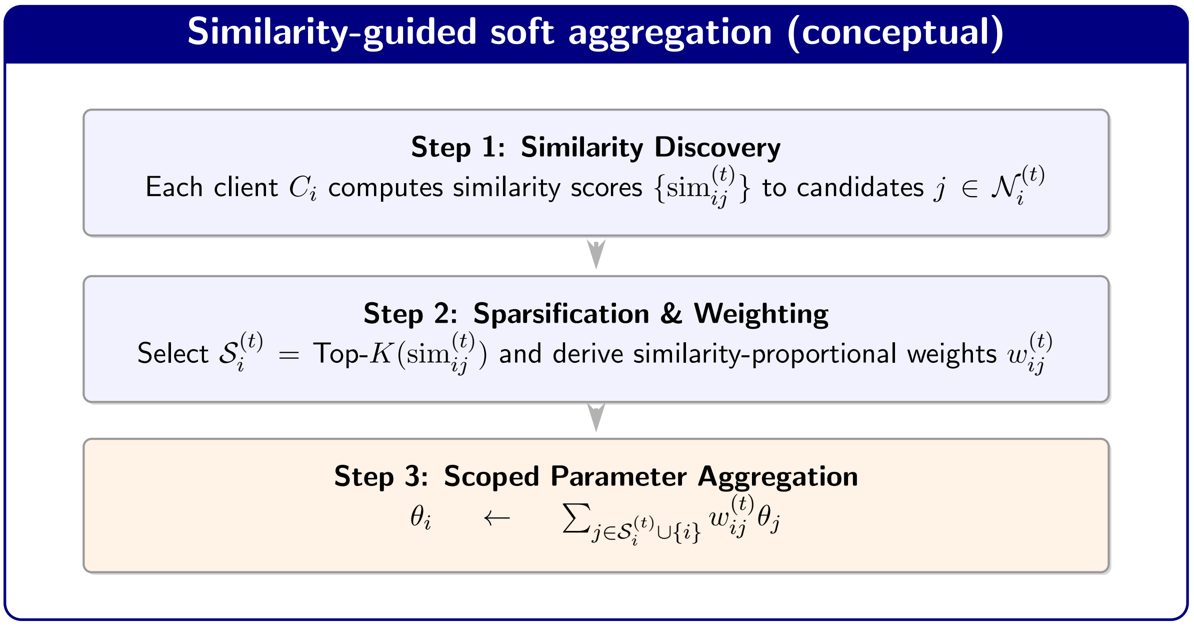

Similarity-guided soft aggregation (conceptual)
Step 1: Similarity Discovery
Each client C_i computes similarity scores $\{\mathrm{sim}_{ij}^{(t)}\}$ to candidates $j \in \mathcal{N}_i^{(t)}$
Step 2: Sparsification & Weighting
Select $\mathcal{S}_i^{(t)} = \text{Top-}K(\mathrm{sim}_{ij}^{(t)})$ and derive similarity-proportional weights $w_{ij}^{(t)}$
Step 3: Scoped Parameter Aggregation
$\theta_i \leftarrow \sum_{j\in\mathcal{S}_i^{(t)}\cup\{i\}} w_{ij}^{(t)}\theta_j$

Figure 4.4: Conceptual workflow of the similarity-guided soft aggregation mechanism. The process transforms raw task similarities into normalized weights for neighborhood-based parameter fusion.

4.4 Aggregation Strategies: Scope and Methods

Beyond the soft weighting mechanism described in Section 4.3, two orthogonal design dimensions fundamentally shape aggregation behavior: the scope decision determines which model components participate in aggregation, while the method decision determines how neighbor information combines. This section formalizes both dimensions and presents a critical empirical finding—that task correlation strength determines the optimal aggregation scope.

4.4.1 Aggregation Scope: Backbone-Only versus Full Model

A central design axis in decentralized federated multi-task learning is *what* portion of the model should be shared across peers. We decompose each local model into a shared backbone (feature extractor) and task-specific head(s),

$$\theta_i = (\theta_i^b, \theta_i^h), \tag{4.13}$$

and treat the *aggregation scope* as a first-class choice that can be adapted to the strength of inter-task relatedness.

Algorithm 1 Similarity-Guided Decentralized Multi-Task Training

```
Input: Clients $\{C_i\}_{i=1}^N$, rounds $T$, local steps $E$, neighbor budget $K$, scope $s \in$
    $\{\text{Backbone}, \text{Full}\}$, operator $\mathcal{A}$, warm-up $T_w$
Initialize: For each client $C_i$, initialize parameters $\theta_i^{(1)} = (\theta_i^{b,(1)}, \theta_i^{h,(1)})$
 1: for $t = 1$ to $T$ do
 2:     // Local training phase
 3:     Each client $C_i$ performs $E$ local SGD steps on $\mathcal{D}_i$ to update $\theta_i^{(t)}$
 4:     // Similarity-guided topology evolution
 5:     Compute similarity scores $\{\text{sim}_{ij}^{(t)}\}$ for candidate set $j \in \mathcal{N}_i^{(t)}$
 6:     Select top neighbors: $\mathcal{S}_i^{(t)} \leftarrow \text{Top-}K(\text{sim}_{ij}^{(t)}, K)$
 7:     Compute $s_{ij}^{(t)} = \max(\text{sim}_{ij}^{(t)}, 0)$ and normalize to obtain weights $w_{ij}^{(t)}$
 8:     // Parameter exchange and aggregation
 9:     Exchange parameters within scope $s$ with neighbors in $\mathcal{S}_i^{(t)}$
10:     if $t \le T_w$ then
11:         $\theta_i^s \leftarrow \text{FedAvg}(\{\theta_j^s\}_{j \in \mathcal{S}_i^{(t)} \cup \{i\}})$        Stable baseline
12:     else
13:         $\theta_i^s \leftarrow \mathcal{A}(\{\theta_j^s, w_{ij}^{(t)}\}_{j \in \mathcal{S}_i^{(t)} \cup \{i\}})$        Apply aggregation $\mathcal{A}$
14:     end if
15:     Update local scoped parameters and proceed to round $t + 1$
16: end for
```

Backbone-only aggregation. In backbone-only aggregation, client C_i exchanges and aggregates only the backbone parameters θ^b, while keeping head parameters θ_i^h private and updated purely through local training. With similarity-derived weights $\{w_{ij}^{(t)}\}$, the backbone update takes the form

$$\theta_i^{b,(t+1)} = \sum_{j \in \mathcal{N}_i^{(t)} \cup \{i\}} w_{ij}^{(t)} \, \theta_j^{b,(t)}, \tag{4.14}$$

where the weights satisfy $\sum_j w_{ij}^{(t)} = 1$ over the selected neighborhood. Head parameters remain client-specific and evolve via local optimization, e.g.,

$$\theta_i^{h,(t+1)} = \theta_i^{h,(t)} - \eta_{\text{local}} \nabla_{\theta^h} \mathcal{L}_i \left(\theta_i^{b,(t)}, \theta_i^{h,(t)}; \mathcal{D}_i \right). \tag{4.15}$$

This scope is attractive when heads are semantically heterogeneous (e.g., different output spaces), because it limits cross-client coupling to a shared representation.

Full-model aggregation. When head parameters are compatible across clients (e.g., same head definitions and output spaces), it can be beneficial to aggregate the full parameter vector:

$$\theta_i^{(t+1)} = \sum_{j \in \mathcal{N}_i^{(t)} \cup \{i\}} w_{ij}^{(t)} \, \theta_j^{(t)}. \tag{4.16}$$

Full-model aggregation increases the degree of knowledge sharing, but also raises the risk of negative transfer if tasks are mismatched or if head semantics differ.

Aggregation design space (scope $\times$ method)

Method $\downarrow$ Scope $\rightarrow$	Backbone-only	Full-model
FedAvg (Uniform)	θ^b uniform average	Scoped blocks average
Similarity-weighted	θ^b weighted by w_{ij}	Weighted by w_{ij}
HCA (Constrained)	HCA applied to θ^b	HCA on scoped blocks

Representative combinations are evaluated in Chapter 6. Only parameters within the intersection of the chosen scope and the model head are subject to $\mathcal{A}$.

Figure 4.5: Design space of aggregation strategies in the proposed framework.

Hypothesis. We hypothesize that backbone-only aggregation is often sufficient when tasks are strongly related and benefit from a shared representation, while full-model aggregation becomes more important as task relatedness weakens. This hypothesis is examined systematically in Chapter 6.

4.4.2 Aggregation Methods: Weighted Average versus Conflict-Averse Optimization

Given a neighborhood and weights, an *aggregation operator* specifies how received models (or updates) are combined. We consider two families: averaging-based operators and conflict-averse operators.

Similarity-weighted averaging. Averaging-based aggregation produces a convex combination of neighbor parameters:

$$\theta_{\text{agg}}^{(t+1)} = \sum_{j \in \mathcal{N}_i^{(t)} \cup \{i\}} w_{ij}^{(t)} \, \theta_j^{(t)}. \tag{4.17}$$

Depending on the chosen scope (Section 4.4.1), θ denotes either backbone parameters θ^b or the full model $\theta = (\theta^b, \theta^h)$.

Uniform averaging (FedAvg-style). Uniform averaging is the special case of Eq. (4.17) with equal weights over the selected set:

$$w_{ij}^{(t)} = \frac{1}{|\mathcal{N}_i^{(t)}| + 1} \quad \Rightarrow \quad \theta_{\text{agg}}^{(t+1)} = \frac{1}{|\mathcal{N}_i^{(t)}| + 1} \sum_{j \in \mathcal{N}_i^{(t)} \cup \{i\}} \theta_j^{(t)}. \tag{4.18}$$

This operator is communication- and computation-efficient and often serves as a stable baseline.

Comparison of Aggregation Scopes

Backbone-only

Definition:
Aggregate shared parameters θ^b while maintaining task-specific heads $\{h_{\theta_t^h}\}$ locally.

Applicability:
- Strong task relatedness.
- Highly heterogeneous heads.

Key Benefit:
Prevents gradient conflicts in heads; mitigates negative transfer from incompatible task outputs.

VS

Full-model

Definition:
Jointly aggregate backbone θ^b and compatible head parameters where semantically aligned.

Applicability:
- Weak task relatedness.
- Homogeneous/compatible head structures across peers.

Key Benefit:
Facilitates deeper cross-client alignment; encourages feature-to-output synergy in shared tasks.

Figure 4.6: Conceptual comparison of aggregation scopes. Scope selection serves as a critical design axis for balancing task-specific specialization and global knowledge sharing.

Conflict-averse aggregation (HCA). Averaging can degrade performance when task gradients conflict. Conflict-averse operators aim to produce an aggregated update that reduces destructive interference among tasks. Let $\{g_k\}_{k=1}^m$ denote the task gradients on a client (or the gradients implied by received neighbor updates, depending on implementation). HCA can be viewed as solving for a combined direction $g^\star$ that is "as consistent as possible" with all task gradients, e.g., by selecting non-negative combination coefficients that avoid opposing directions:

$$g^\star \in \arg\min_{\{a_k \geq 0\}} \left\| \sum_{k=1}^{m} a_k g_k \right\|_2 \quad \text{s.t.} \quad \left\langle \sum_{k=1}^{m} a_k g_k,\, g_\ell \right\rangle \geq 0 \;\; \forall \ell. \tag{4.19}$$

The model is then updated using the conflict-averse direction, e.g., $\theta \leftarrow \theta - \eta\, g^\star$. In our framework, such operators can be applied at the backbone level or to the full model, and are combined with the stabilization policy in Section 4.5.

4.5 Stabilization Policy for Advanced Aggregation

Conflict-averse aggregation methods (e.g., HCA [10]) can be substantially more sensitive to heterogeneous gradients than simple averaging, especially in fully decentralized settings where neighbor sets change over time and local updates may drift. In practice, the optimization subroutines underlying such methods may become ill-conditioned (e.g., due to poorly conditioned Gram matrices) or fail to converge when gradient conflicts are severe.

To make advanced aggregation usable as a drop-in alternative within our decentralized framework, we adopt a two-level stabilization policy:

Hybrid warm-up and switching. We first run a *stable* aggregation rule (e.g., FedAvg [1] or similarity-weighted averaging) for the first T_{warm} rounds to bring all clients into a numerically stable and semantically meaningful region of the loss landscape. After this warm-up, clients switch to conflict-averse aggregation:

$$\theta_i^{(t+1)} = \begin{cases} \text{Agg}_{\text{stable}}\Big(\{\theta_j^{(t)}\}_{j \in \mathcal{N}_i^{(t)} \cup \{i\}}\Big), & t < T_{\text{warm}}, \\ \text{Agg}_{\text{HCA}}\Big(\{\theta_j^{(t)}\}_{j \in \mathcal{N}_i^{(t)} \cup \{i\}}\Big), & t \geq T_{\text{warm}}. \end{cases} \tag{4.20}$$

The choice of T_{warm} is treated as a hyperparameter and empirically studied in Chapter 6.

Numerical safeguards with fallback. During conflict-averse aggregation, we apply lightweight sanity checks and regularization (e.g., conditioning control and robust fallbacks to stable aggregation) to prevent numerical failures and ensure training can proceed. The full engineering design of these safeguards is reported in Chapter 5, while their practical impact is evaluated in Chapter 6.

4.6 Training Efficiency Considerations

Beyond accuracy, decentralized multi-task training must be computationally practical. We therefore include two training-time optimizations that do not change the underlying learning objective:

Early stopping. Clients monitor a held-out validation signal and terminate local training once improvements plateau, reducing unnecessary computation in later rounds.

Adaptive learning rate scheduling. We adjust the learning rate based on validation progress to maintain stable optimization across heterogeneous tasks and clients.

Implementation details (e.g., triggers and hyperparameters) are described in Chapter 5, while the resulting efficiency gains are quantified in Chapter 6.

4.7 Chapter Summary

This chapter presented our methodology for decentralized federated multi-task learning. We formalized the system model and task assignment schemes, introduced dynamically updated task similarity discovery, and developed similarity-guided soft neighbor selection and aggregation. We then defined an aggregation design space that separates *scope* (backbone-only vs. full model) from *method* (weighted averaging vs. conflict-averse optimization), and described high-level stabilization and efficiency policies that support robust training in practice.

Chapter 5

Implementation

This chapter describes the software architecture and engineering practices that enable the proposed decentralized federated multi-task learning framework. The implementation represents a significant extension of the centralized framework developed by Kohler [9], requiring fundamental architectural redesign to support peer-to-peer communication, heterogeneous dense prediction tasks, and numerically stable aggregation mechanisms. Section 5.1 traces the architectural evolution from centralized coordination to fully decentralized execution. Section 5.2 presents the configuration-driven design philosophy that enables systematic exploration of a high-dimensional experimental space. Section 5.3 details the specialized data pipelines required for multi-label dense prediction tasks. Section 5.4 describes the training infrastructure supporting reproducible experimentation. Finally, Section 5.5 synthesizes the software engineering principles underlying this work.

5.1 Codebase Evolution and Architecture

The architectural evolution from Kohler's centralized framework to this thesis's decentralized extension required addressing fundamental differences in coordination patterns, communication topologies, and task complexity. This section contrasts the original design with the extensions implemented to support decentralized dense prediction multi-task learning.

5.1.1 Original Centralized Framework

Kohler's framework [9] established the foundation for this work through its integration of Federated Personalization [29] with Hyper Conflict-Averse (HCA) aggregation [10]. The original architecture exhibited three defining characteristics that both enabled rapid prototyping and constrained extensibility to decentralized settings.

The original design implemented a *star topology* wherein a central server orchestrated all client interactions. Clients transmitted model updates to the server, which performed global aggregation using HCA optimization and subsequently broadcast the aggregated model to all participants. This centralized design simplified synchronization and provided global visibility of all client states, enabling sophisticated aggregation strategies that required knowledge of the complete client population. However, the architectural choice introduced a single point of failure and limited scalability, as the server's computational and communication bandwidth became bottlenecks proportional to the number of clients. More fundamentally, the centralized coordination pattern could not generalize to peer-to-peer networks where clients lack global visibility and must make autonomous aggregation decisions based solely on local information and neighbor communications.

The implementation employed a *monolithic script architecture* (`src/run.py`, approximately 800 lines) containing interleaved server and client logic. The server maintained global state including cluster assignments derived from task similarity matrices and aggregation weights computed through centralized optimization. Clients performed local training but delegated all aggregation decisions to the server. This monolithic structure reduced inter-module communication overhead and simplified debugging during initial development. Nevertheless, the tight coupling between server and client logic precluded straightforward adaptation to decentralized settings. Decentralized clients must internalize aggregation logic previously residing in the server, requiring substantial refactoring rather than simple modification.

The original framework validated on *classification tasks* including CIFAR-10 [25] and CelebA, both representing single-label outputs. The data loading infrastructure assumed images paired with scalar class labels, and evaluation metrics focused exclusively on classification accuracy. This task-specific design required fundamental extension to support dense prediction tasks where each pixel produces an independent prediction (depth values, surface normal vectors, semantic class labels). Dense prediction introduces computational challenges absent in classification, including memory management for pixel-aligned annotations, synchronized augmentation of images and dense label maps, and task-weighted loss computation across heterogeneous output types.

5.1.2 Decentralized Extension

This thesis extends Kohler's foundation to fully decentralized settings supporting multi-task dense prediction. The extensions required architectural redesign rather than incremental modification, as the shift from centralized to peer-to-peer coordination altered fundamental assumptions about information availability, synchronization patterns, and failure modes.

Peer-to-Peer Communication Architecture

Rather than modifying the centralized server-client architecture, this work implements a fundamentally different communication pattern wherein clients communicate directly through simulated peer-to-peer channels. Each client maintains dynamic connections to K neighbors selected based on task similarity (computed using gradient-based or task overlap metrics. Clients exchange model parameters or gradients through these channels and perform local aggregation autonomously without global coordination.

The absence of a central server eliminates the single point of failure inherent in star topologies while introducing new challenges in synchronization and consistency. This thesis addresses synchronization through round-based coordination, where all clients complete local training and neighbor communication before advancing to the next global round, maintaining the synchronous execution pattern of Federated Averaging [1] while distributing coordination responsibility across the network.

Specialized Training Scripts

To maintain architectural clarity and facilitate systematic experimentation, this work implements three specialized main scripts rather than adapting the original monolithic implementation. Table 5.1 summarizes their purposes and characteristics.

Table 5.1: Specialized Training Scripts for Decentralized Learning

Script	LOC	Purpose
`run_decentralized.py`	460	CIFAR-10 classification with dynamic neighbor selection and similarity-based aggregation. Initial validation of P2P coordination.
`run_multitask_decentralized.py`	580	Universal multi-task dense prediction supporting NYU V2 and Pascal Context. Polymorphic loss computation for heterogeneous tasks (regression, multi-class, binary classification).
`run_proposals.py`	340	Specialized aggregation scope exploration including backbone-only and hierarchical aggregation strategies.

The primary contribution resides in `run_multitask_decentralized.py`, which achieves dataset-agnostic execution through configuration-driven polymorphism. The script dispatches to appropriate data loaders, loss functions, and evaluation metrics based on configuration metadata, enabling arbitrary task combinations without code modification.

Dense Prediction Pipeline Architecture

Supporting dense prediction required developing new data pipelines that handle pixel-aligned multi-label outputs with synchronized augmentation. The implementation addresses three key challenges: efficient loading of multi-task annotations stored in different formats (HDF5 for NYU V2, PNG directories for Pascal Context), synchronized augmentation maintaining spatial correspondence between images and dense labels, and polymorphic loss computation selecting task-appropriate loss functions (L1 for depth regression, cross-entropy for segmentation, binary cross-entropy for edge detection).

Data augmentation applies jointly to images and all annotation maps through shared random seeds, ensuring that a horizontal flip applied to the image generates the corresponding flip in depth maps, normal maps, and segmentation masks. Augmentation strategies include random horizontal flipping (probability 0.5), random scaling (factors uniformly sampled from $[0.5, 2.0]$), random cropping to 512×512 patches, and color jittering within ± 0.2 of the original values.

Implementation Scale

The decentralized extensions add substantial new functionality beyond Kohler's original codebase. Table 5.2 quantifies the engineering effort.

Table 5.2: Implementation Statistics

Component	**Count**	**Description**
Python code	3000	Lines of new algorithmic implementation
Configuration files	45	YAML files covering experimental variations
Automation scripts	20	Batch execution and monitoring scripts
Documentation	20	Markdown files documenting protocols

This substantial engineering effort, while not constituting algorithmic contributions per se, represents essential infrastructure enabling the systematic experimentation that generated this thesis's empirical findings.

5.2 Configuration-Driven Architecture

A central design principle enabling rapid experimentation is the strict separation between implementation logic and experimental parameters. This section articulates the configuration-driven architecture philosophy and demonstrates how it enables systematic exploration of a high-dimensional experimental space with minimal code duplication.

5.2.1 Design Philosophy

Traditional research codebases frequently intermingle experimental parameters with implementation logic, producing code duplication when exploring parametric variations. A common anti-pattern involves copying entire scripts to modify hyperparameters, aggregation methods, or task assignments, leading to versioning nightmares where bug fixes must propagate across dozens of near-identical files. This thesis adopts a disciplined separation between implementation and configuration layers.

The *implementation layer* contains all algorithmic logic including model architectures, training loops, aggregation methods, and evaluation pipelines. This layer remains parameter-agnostic, reading all experimental settings from configuration files rather than hardcoding constants. For example, the aggregation method selection dispatches between weighted averaging and HCA based on a configuration field (`setup.aggregation_method`), rather than maintaining separate training scripts for each method. This design ensures that algorithmic logic remains stable across experiments, as parameter variations require only configuration changes without touching implementation code.

The *configuration layer* contains all experimental parameters expressed in human-readable YAML format with hierarchical organization. Each configuration file defines one complete experiment including dataset selection, task assignments, hyperparameters, and aggregation settings. This externalization of parameters enables systematic exploration through programmatic configuration generation. For instance, ablating the effect of the α parameter required generating two configuration files differing only in the value of `setup.alpha`, with all other parameters held constant.

The configuration-driven architecture delivers four primary benefits: First, *code reuse* achieves remarkable efficiency—the same `run_multitask_decentralized.py` script executed all eight NYU V2 experimental configurations and four Pascal Context configurations by reading different YAML files. Traditional approaches requiring separate scripts per configuration would have generated over 4,600 lines of duplicated code. Second, *reproducibility* strengthens through configuration versioning, as Git commits capturing configuration files alongside results provide complete experimental provenance. Third, *parallel development* becomes feasible, as multiple researchers can design experiments through configuration files without code conflicts. Fourth, *rapid iteration* accelerates through simplified parameter modification.

5.2.2 Configuration File Structure

Configuration files employ YAML syntax organized into six logical sections: `general` (metadata and random seed), `data` (dataset selection and loading), `setup` (network topology, task assignments, aggregation configuration), `training` (optimization hyperparameters), `model` (architecture specification), and `output` (result persistence). Listing 5.1 presents a representative configuration file for the NYU Depth V2 pairwise experiment (A2), demonstrating the declarative specification of experimental parameters.

Table 5.3 summarizes the organizational hierarchy and key parameters in each section.

For example, a pairwise task assignment for NYU V2 (experiment A2) specifies six clients with task weights: `{depth: 0.7, segmentation: 0.3, normal: 0.0}`, `{depth: 0.7, segmentation: 0.0, normal: 0.3}`, etc., where weights sum to 1.0 per client. This declarative specification enables the same training script to execute SingleTask (A1), Pairwise (A2), and MultiTask configurations without code modification.

Listing 5.1: Representative configuration file for NYU Depth V2 pairwise experiment (A2).

```yaml
# configs/decentralized/nyuv2_a2_pairwise_v2.yml
general:
  title: "a2_pairwise_v2"
  seed: 42
  description: "Pairwise task assignment with task overlap prior"

data:
  dataset: "nyuv2"
  root_dir: "./data/nyuv2"
  dataset_fraction: 1.0

setup:
  num_clients: 6
  n_neighbors: 3
  alpha: 0.5                      # Task overlap weight
  aggregate_heads: true           # Full aggregation
  aggregation_method: "weighted"

  # Pairwise task assignments (0.7 primary, 0.3 secondary)
  task_weights_per_client:
    - {depth: 0.7, segmentation: 0.3, normal: 0.0}
    - {depth: 0.7, segmentation: 0.0, normal: 0.3}
    - {depth: 0.3, segmentation: 0.7, normal: 0.0}
    - {depth: 0.0, segmentation: 0.7, normal: 0.3}
    - {depth: 0.3, segmentation: 0.0, normal: 0.7}
    - {depth: 0.0, segmentation: 0.3, normal: 0.7}

training:
  num_rounds: 50
  local_epochs: 5
  batch_size: 4
  learning_rate: 0.001
  optimizer: "adam"
  early_stopping:
    enabled: true
    patience: 10
    min_delta: 0.002

model:
  backbone: "resnet18"
  pretrained: true

output:
  output_dir: "results/nyuv2/a2_pairwise_v2"
  save_checkpoints: true
```

Table 5.3: Configuration File Structure

Section	**Key Parameters**
general	Experiment title, random seed, textual description
data	Dataset identifier (nyuv2 or pascal_context), root directory, dataset fraction
setup	Number of clients, neighbor count K, α parameter, aggregation method/scope, per-client task assignments
training	Training rounds, local epochs, batch size, learning rate, optimizer, early stopping configuration
model	Backbone architecture (ResNet-18), pretrained initialization
output	Output directory, checkpoint saving, logging frequency

5.2.3 Multi-Dataset Support

The `run_multitask_decentralized.py` script achieves dataset-agnostic execution through conditional dispatch driven by the `data.dataset` configuration field. Algorithm 2 illustrates the dataset loading logic implementing polymorphic data loader instantiation.

Algorithm 2 Polymorphic Dataset Loading

```
1: function LoadDataset($config$)
2:     dataset_type ← $config$[“data”][“dataset”]
3:     if dataset_type = “pascal_context” then
4:         $dm$ ← DMPascalContext($config$[“data”])
5:         $task\_dims$ ← {segmentation : 59, human_parts : 15, edge : 1}
6:     else                                        ▷ Default to NYU Depth V2
7:         $dm$ ← DMNYUDepthV2($config$[“data”])
8:         $task\_dims$ ← {depth : 1, normal : 3, segmentation : 13}
9:     end if
10:    return $dm, task\_dims$
11: end function
```

This polymorphic design pattern extends beyond data loading to loss computation, metric evaluation, and result logging. The training loop queries task dimensions to construct task-specific output heads dynamically, selects appropriate loss functions based on task types, and computes task-appropriate evaluation metrics. Without configuration-driven architecture, supporting two datasets would require either duplicating the entire training script or implementing complex inheritance hierarchies.

5.2.4 Experiment Isolation

Each experiment writes results to a unique output directory specified in its configuration, preventing interference between concurrent or sequential runs. The hierarchical directory structure isolates experiments by dataset and configuration: `results/nyuv2/`

a1_singletask_v2/, results/nyuv2/a2_pairwise_v2/, etc. Directory isolation extends to logging, where training logs organize hierarchically by dataset to prevent file collisions. This isolation property enables concurrent execution of multiple experiments on different GPUs without risk of result corruption.

Table 5.4 summarizes the configuration files generated and experiments executed per dataset, demonstrating the efficiency gained through configuration-driven design.

Table 5.4: Configuration Files and Experimental Coverage

Dataset	**Configs**	**Main Script**	**Experiment Types**
CIFAR-10	15	`run_decentralized.py`	Dynamic clustering, Hybrid
NYU V2	20	`run_multitask_decentralized.py`	A1–A2, B1–B4, C1–C2
Pascal Context	10	`run_multitask_decentralized.py`	A1–A2, B1, B4, CrossLoss
Total	**45**	**2 universal scripts**	**60+ experiments**

This demonstrates the multiplicative power of configuration-driven design: 45 configuration files generate over 60 unique experiments using only two main implementation scripts. Traditional approaches requiring separate code for each experiment would have produced over 34,800 lines of duplicated code, introducing massive maintenance overhead and dramatically increasing the probability of implementation inconsistencies.

5.3 Data Pipeline Implementation

Dense prediction tasks demand specialized data pipelines handling multiple pixel-aligned labels per image with heterogeneous output formats. This section details the implementations for NYU Depth V2 and Pascal Context, addressing the challenges of multi-label loading, synchronized augmentation, and task-weighted loss computation.

5.3.1 NYU Depth V2 Pipeline

The NYU Depth V2 dataset [7] provides indoor RGBD images with aligned annotations for three tasks exhibiting strong geometric correlation: depth estimation, surface normal prediction, and semantic segmentation. The pipeline implementation addresses the challenge of efficiently loading and preprocessing three annotation types with different numerical characteristics.

Data Storage and Loading

NYU V2 annotations are distributed in HDF5 format, a hierarchical data format enabling efficient random access to large multi-dimensional arrays. The HDF5 file structure organizes data into training and validation splits, with each group containing four datasets: RGB images (uint8, shape $[N, 3, H, W]$), depth maps (float32, shape $[N, 1, H, W]$ in meters), surface normals (float32, shape $[N, 3, H, W]$ with values in $[-1, 1]$), and segmentation masks (int64, shape $[N, H, W]$ with class indices in $\{0, 1, \ldots, 12\}$).

The data loader extends PyTorch's `Dataset` interface, implementing the `__getitem__` method that returns a dictionary containing the image and all three task annotations. The HDF5 format enables memory-mapped access, where the data loader reads samples on-demand without loading the entire dataset into RAM. For the 795 training images (each 640×480 RGB with three dense annotation maps), memory-mapped loading is essential to maintain feasible memory footprint.

Task-Specific Preprocessing

The preprocessing methods apply task-specific normalization addressing the heterogeneous numerical characteristics of the three tasks:

- *Image normalization*: Resize to 256×256 using bilinear interpolation and apply channel-wise standardization using ImageNet statistics ($\mu = [0.485, 0.456, 0.406]$, $\sigma = [0.229, 0.224, 0.225]$) to facilitate transfer learning from pretrained backbones [50].
- *Depth normalization*: Apply log-scale transformation $d' = \log(1 + d)$ to compress the dynamic range. Raw depth values span approximately $[0.5, 10.0]$ meters with heavy concentration near zero. The logarithmic transformation improves gradient behavior by reducing magnitude differences between near and far surfaces.
- *Surface normal normalization*: Enforce unit vector constraints through L2 normalization: $\mathbf{n}' = \mathbf{n}/\|\mathbf{n}\|_2$. Explicit normalization ensures that predicted normals can be compared against ground truth using angular error metrics.
- *Segmentation*: Convert to `torch.long` dtype required by PyTorch's cross-entropy loss. Integer class indices need no scaling.

5.3.2 Pascal Context Pipeline

Pascal Context [8] extends the PASCAL VOC dataset with dense annotations for 59 semantic classes plus additional tasks. This thesis employs three tasks exhibiting weak correlation: semantic segmentation (59-class object categorization), human parts segmentation (15-class body part labels), and edge detection (binary edge presence at each pixel). Unlike NYU V2's strong geometric correlation, Pascal Context's tasks operate at different abstraction levels with limited mutual benefit.

Data Storage and Loading

Pascal Context annotations are stored as separate PNG files for each task rather than a unified HDF5 archive. The directory structure organizes data as: `JPEGImages/` (RGB images), `SegmentationClass/` (semantic masks), `HumanParts/` (body part labels for person-containing images only), and `Edge/` (binary edge maps). The distributed storage format requires multiple file I/O operations per sample, contrasting with NYU V2's single HDF5 read.

The data loader implements graceful handling of missing annotations, a necessity because not all images have all task labels. Missing annotations are replaced with an ignore index (-1) that instructs loss functions to skip pixels without ground truth labels through the `ignore_index` parameter of PyTorch's cross-entropy loss. This mechanism enables principled handling of incomplete annotations without resorting to heuristics like assigning random labels.

Data Augmentation

To improve generalization from the relatively small dataset (approximately 10,000 images), Pascal Context training applies synchronized augmentation: random horizontal flipping (probability 0.5), random resizing (scale factors from $[0.5, 2.0]$), random cropping (512×512 patches), and color jittering (brightness, contrast, saturation within ± 0.2).

The augmentation pipeline implements synchronization through shared random state: when sampling a horizontal flip decision, the data loader seeds the random number generator, applies the decision to the image, reseeds with the same value, and applies the decision to each annotation map. This ensures pixel-level correspondence essential for dense prediction training.

5.3.3 Task-Weighted Loss Computation

Supporting multiple tasks per client requires computing weighted combinations of task-specific losses. Algorithm 3 illustrates the polymorphic loss computation.

The polymorphic loss computation enables flexible task combinations specified through configuration. For example, a client training on depth and segmentation with weights {`depth: 0.7, segmentation: 0.3, normal: 0.0`} would compute total loss as $L_{\text{total}} = 0.7 \cdot L_1(\hat{d}, d) + 0.3 \cdot \text{CrossEntropy}(\hat{s}, s)$, automatically selecting L1 loss for depth regression and cross-entropy for segmentation classification. The zero weight on normals causes the function to skip normal loss computation entirely, avoiding unnecessary computation for unused tasks.

Algorithm 3 Task-Weighted Loss Computation

```
function ComputeLoss(outputs, targets, task_weights)
    total_loss ← 0
    task_losses ← {}
    for each (task, weight) in task_weights do
        if weight > 0 then
            if task ∈ {depth, normal} then
                loss ← $L_1$(outputs[task], targets[task])                    ▷ Regression
            else if task = edge then
                loss ← BCE(outputs[task], targets[task], pos_w = 10.0)
            else
                loss ← CE(outputs[task], targets[task], ign = $-1$)         ▷ Classification
            end if
            total_loss ← total_loss + weight × loss
            task_losses[task] ← loss
        end if
    end for
    return total_loss, task_losses
end function
```

5.4 Training Infrastructure

This section describes the infrastructure supporting reproducible decentralized training, including peer-to-peer communication simulation, experiment automation, and reproducibility measures.

5.4.1 Peer-to-Peer Communication Simulation

True decentralized training would deploy each client on separate physical machines communicating over networks. For controlled experimentation enabling rapid iteration and precise reproducibility, this work simulates peer-to-peer communication within a single process. The simulation accurately models decentralized decision-making dynamics (clients make independent aggregation decisions without global coordination) while avoiding network communication overhead, Byzantine fault injection, and synchronization challenges present in distributed systems.

Each client instantiates a `DecentralizedClient` class maintaining private state including model parameters, optimizer state, local dataset, and task assignment. Clients expose an aggregation interface enabling neighbor model access without exposing internal implementation details. Listing 5.2 illustrates the core aggregation method implementing autonomous parameter updates.

The aggregation weight normalization ensures that weights sum to unity:

$$w_i = \frac{s_{ii}}{\sum_{j\in\mathcal{N}_i} s_{ij} + s_{ii}}, \quad w_j = \frac{s_{ij}}{\sum_{k\in\mathcal{N}_i} s_{ik} + s_{ii}} \quad \forall j \in \mathcal{N}_i \tag{5.1}$$

Listing 5.2: Decentralized client aggregation implementing autonomous parameter updates without central coordination. Each client independently computes normalized weights from similarity scores and aggregates neighbor parameters. The self-weight term w_i ensures that clients incorporate their own parameters, preventing complete overwriting by neighbors. Aggregation operates on the shared backbone by default, with optional full model aggregation when `aggregate_heads=True`.

```python
class DecentralizedClient:
    def __init__(self, client_id, task_weights, model, optimizer):
        self.id = client_id
        self.task_weights = task_weights
        self.model = model
        self.optimizer = optimizer

    def aggregate_from_neighbors(self, neighbors, similarity_scores):
        """Aggregate␣parameters␣from␣selected␣neighbors␣using␣soft␣
           weighting."""
        # Compute normalized aggregation weights
        total_sim = sum(similarity_scores.values()) + similarity_scores[
           self.id]
        weights = {n.id: similarity_scores[n.id] / total_sim
                   for n in neighbors}
        self_weight = similarity_scores[self.id] / total_sim

        # Aggregate backbone (or full model if aggregate_heads=True)
        for name, param in self.model.backbone.named_parameters():
            aggregated = self_weight * param.data.clone()

            for neighbor in neighbors:
                neighbor_param = neighbor.model.backbone.state_dict()[
                   name]
                aggregated += weights[neighbor.id] * neighbor_param

            param.data.copy_(aggregated)
```

where w_i is the self-weight, $\mathcal{N}_i$ denotes client i's neighbor set, and s_{ij} represents pairwise similarity. This normalization preserves parameter magnitudes, preventing explosive growth or shrinkage that would occur with unnormalized weights. The self-weight term ensures that clients retain information from their local training rather than being dominated by neighbor parameters.

5.4.2 Experiment Automation

Manually executing over 60 experiments sequentially would be error-prone and time-intensive. This work implements automation scripts orchestrating batch execution, real-time monitoring, and result aggregation. The automation infrastructure reduces human error, enables overnight execution, and provides comprehensive logging for post-hoc analysis.

The batch runner architecture accepts a run number as command-line argument, enabling three-run statistical validation through sequential invocations with different run identifiers. For each experiment configuration, the script programmatically modifies the output directory to include the run number, ensuring isolated results across runs. The script then invokes the training script with the modified configuration, redirecting all output to a timestamped log file using `tee` for simultaneous terminal display and persistent logging.

5.4.3 Reproducibility Measures

Ensuring reproducible results demands careful control of all randomness sources in the training pipeline. This work implements three layers of reproducibility guarantees addressing different sources of nondeterminism.

Random seed control: All pseudorandom number generators receive deterministic seeding through a centralized function invoked at experiment initialization. The implementation seeds PyTorch's CPU and CUDA backends, NumPy's random number generator, Python's built-in `random` module, and configures PyTorch's cuDNN backend for deterministic execution. The seed value originates from the configuration file's `general.seed` field, ensuring that all randomness sources derive from a single configuration parameter. Deterministic cuDNN behavior trades a small performance penalty (approximately 10% training time increase) for reproducibility.

Configuration versioning: All configuration files reside in Git version control with commit messages linking configurations to experimental runs. Additionally, each result JSON file includes a complete copy of the configuration used for that experiment, embedded in the `config` field of the results dictionary. This dual versioning strategy ensures that experiments remain reproducible even if the original configuration file undergoes subsequent modifications.

Structured result logging: Results are persisted in JSON format with comprehensive round-by-round metrics enabling detailed convergence analysis. The JSON structure includes experiment metadata (title, timestamp, configuration), global training statistics (total rounds executed, early stopping trigger round), and a rounds array containing per-round metrics (training loss, validation loss, per-task losses, similarity matrices, aggregation weights). This structured format enables programmatic analysis through Python scripts that load JSON files, extract metric trajectories, and generate comparison plots across experiments.

5.5 Software Engineering Practices

Beyond algorithmic contributions, this thesis demonstrates that disciplined software engineering practices amplify research productivity. This section synthesizes the engineering principles underlying the implementation.

Modular code organization: The codebase follows a strict modular structure separating concerns into distinct directories. The `src/decentralized/` directory contains core contributions (main scripts, client implementations, similarity computation, aggregation methods). The `src/data_handling/` directory provides dataset-specific data loaders isolated from training logic. The `src/client_handling/` directory implements aggregation algorithms including the HCA protection mechanism. The `src/models/` directory defines neural architectures for multi-task dense prediction. This separation enables independent development and testing of modules, reduces cognitive load when modifying specific functionality, and facilitates code reuse across experiments.

Configuration-driven testing strategy: The configuration-driven architecture enables a three-tier testing strategy balancing execution time against result fidelity. Quick test configurations employ 3 rounds, 2 clients, and 10% dataset fractions, completing in approximately 2 minutes to validate code correctness, configuration parsing, and absence of runtime errors. These quick tests execute before launching expensive full runs, catching bugs early. Full run configurations employ 50 rounds, 6 clients, and complete datasets, generating publication-quality results over 2-8 hours depending on task complexity. Ablation configurations systematically vary single parameters while holding all others constant, enabling rigorous sensitivity analysis. This tiered approach minimizes wasted computation through early error detection while ensuring final results derive from complete experimental configurations.

Version control and documentation: All code, configurations, and documentation reside in Git version control with atomic commits linking implementation changes to experimental results. The repository includes 20 markdown documents providing comprehensive documentation of experimental protocols, dataset characteristics, and implementation decisions. This documentation practice ensures knowledge transfer and enables future researchers to understand design rationale without reverse-engineering code.

The implementation demonstrates that software architecture decisions profoundly impact research productivity. The configuration-driven design enabled exploring over 60 experimental configurations with only two main scripts, representing a 10:1 code reduction versus traditional approaches. Modular data pipelines support heterogeneous dense prediction tasks through polymorphic loss computation and task-agnostic training loops. Automation scripts ensure reproducible batch execution with comprehensive logging. These engineering practices, while not constituting algorithmic contributions themselves, form essential infrastructure enabling the systematic experimentation presented in Chapter 6. The implementation provides a reusable foundation for future research exploring decentralized federated learning, multi-task learning, and dense prediction tasks, with minimal adaptation required for new datasets or aggregation strategies.

Chapter 6

Evaluation

This chapter presents comprehensive experimental validation of the proposed decentralized federated multi-task learning framework across three datasets spanning classification and dense prediction tasks. The evaluation strategy employs a three-dataset validation approach deliberately designed to test the framework's generalization across diverse task types (classification versus dense prediction), task correlation structures (strongly correlated geometric tasks versus weakly correlated multi-scale tasks), and aggregation challenges (stable convergence versus numerical instability). Section 6.1 establishes the experimental methodology including the three-dataset validation strategy, experimental design matrix systematically varying task assignment and aggregation dimensions, hyperparameter selections with rationale, and evaluation metrics appropriate for heterogeneous task types. Section 6.2 chronicles the CIFAR-10 framework validation, documenting the systematic debugging process through extensive experimentation that isolated HCA numerical instability and culminated in both the 5-layer protection mechanism and hybrid aggregation strategies. Section 6.3 reports NYU Depth V2 results providing the first application of decentralized FMTL to dense prediction tasks with strongly correlated objectives. Section 6.4 presents Pascal Context results validating cross-dataset generalization and revealing the critical relationship between task correlation strength and optimal aggregation scope. Section 6.5 synthesizes findings across datasets, identifying consistent patterns and dataset-specific behaviors. Finally, Section 6.6 distills the most significant empirical contributions and their implications for practice.

6.1 Experimental Setup

This section establishes the experimental methodology enabling rigorous evaluation through controlled comparisons, statistical validation, and comprehensive metric collection.

6.1.1 Three-Dataset Validation Strategy

Validating on a single dataset risks discovering dataset-specific artifacts rather than general principles applicable across problem domains. This thesis adopts a three-dataset strategy deliberately selected to span diverse characteristics along multiple dimensions. Table 6.1 summarizes the strategic positioning of each dataset within the validation framework.

Table 6.1: Three-Dataset Validation Strategy Spanning Task Types and Correlation Structures

Dataset	**Purpose**	**Task Types**	**Correlation**	**Contribution**
CIFAR-10	Baseline validation	Classification	Moderate	Inherited baseline
NYU V2	Dense prediction	Regr. + Classif.	**Strong**	**Novel**
Pascal VOC	Generalization test	Multi-scale classif.	**Weak**	**Novel**

CIFAR-10 [25] serves as the framework validation dataset, inherited from Kohler's centralized work [9]. The dataset provides a controlled classification baseline for validating that the decentralized architecture functions correctly before extending to the complexity of dense prediction. CIFAR-10's 10 object classes (animals, vehicles, everyday objects) create moderate task heterogeneity when clients specialize on class subsets. This moderate heterogeneity generates meaningful collaboration opportunities (related classes like dog and cat benefit from shared representations) without extreme gradient conflicts that would destabilize training. The classification focus further simplifies debugging, as single-label outputs produce interpretable loss trajectories and enable straightforward performance assessment through accuracy metrics. This simplicity proved essential during the systematic HCA debugging process detailed in Section 6.2, where over 20 experiments tested increasingly sophisticated solutions before achieving stable convergence.

NYU Depth V2 [7] provides the first application of decentralized FMTL to pixel-wise regression and segmentation, representing a significant extension beyond prior classification-focused federated learning research. The dataset comprises indoor RGBD images with aligned annotations for three tasks: depth estimation (predicting metric depth at each pixel), surface normal prediction (estimating 3D surface orientation vectors), and semantic segmentation (assigning one of 13 indoor scene classes to each pixel). These three tasks exhibit strong geometric correlation arising from the shared structure of indoor scenes. Depth edges align with semantic boundaries, as objects at different depths typically belong to different semantic categories (e.g., wall versus table). Surface normals couple to depth through geometric constraints, as similar depths on planar surfaces imply similar surface orientations. Segmentation leverages geometric cues, as depth and normal discontinuities indicate object boundaries. This strong correlation tests whether decentralized aggregation can leverage complementary task information to improve performance beyond task-isolated training, addressing the central research question of whether multi-task learning benefits persist in decentralized settings lacking global coordination.

Pascal Context [8] tests generalization to scenarios where tasks provide limited mutual benefit, stress-testing aggregation strategies under weak correlation. The dataset extends PASCAL VOC with dense annotations across 59 semantic classes plus additional tasks. This thesis employs three tasks operating at fundamentally different abstraction levels: semantic segmentation (59-class object categorization requiring high-level semantic understanding), human parts segmentation (15-class body part labels applicable only to person-containing images), and edge detection (binary classification capturing low-level gradient information). Unlike NYU V2's geometric coherence, Pascal Context's tasks exhibit weak structural coupling. Edge detection captures texture edges and material boundaries unrelated to semantic categories, as edges appear within homogeneous objects (wood grain patterns) and may be absent at semantic boundaries (gradual color transitions). Human parts segmentation specializes on a single object category (people) and provides no information for scenes lacking humans (landscapes, vehicles, furniture). Semantic segmentation operates at the object level, abstracting away the low-level edges and human-specific features. This weak correlation challenges aggregation algorithms to avoid negative transfer, where incorporating information from dissimilar tasks degrades performance relative to task-isolated training. The combination of strong correlation (NYU V2) and weak correlation (Pascal Context) enables studying how task relationships affect optimal aggregation strategies, yielding one of this thesis's central empirical findings detailed in Section 6.4.3.

6.1.2 Experimental Design Matrix

Experiments systematically vary two primary dimensions—task assignment strategy and aggregation configuration—enabling controlled ablation studies isolating the effect of individual design choices. Task assignment determines which tasks each client trains, controlling the degree of task heterogeneity across the network. The SingleTask assignment (designated A1 across datasets) assigns each client exclusive focus on one task with weight 1.0, minimizing gradient conflicts at the cost of limited knowledge transfer. The Pairwise assignment (A2) assigns each client two related tasks with asymmetric weights: 0.7 for the primary task and 0.3 for a secondary task. This balanced design provides multi-task learning benefits while maintaining task specialization, hypothesized to outperform pure specialization (A1) and uniform multi-task distribution. The MultiTask assignment (B4) distributes all tasks equally across all clients with uniform weights (0.33 for three tasks), maximizing shared representation learning but risking gradient conflicts when tasks have opposing optimization directions.

Aggregation strategy encompasses three interrelated design choices: the similarity metric balancing gradient-based and task overlap information, the aggregation method implementing parameter combination, and the aggregation scope determining which parameters undergo aggregation. The similarity metric employs parameter $\alpha \in [0, 1]$ interpolating between pure gradient similarity ($\alpha = 0.0$, designated experiments with prefix B) and mixed gradient-task overlap similarity ($\alpha = 0.5$, experiments with prefix A). Pure gradient similarity (B-series experiments including B1, B2, B3, B4) computes client relationships exclusively from gradient cosine similarity, testing whether dynamic optimization information suffices for effective aggregation without static task metadata. Mixed similarity (A-series experiments including A1, A2) incorporates task overlap prior knowledge, hypothesizing that combining optimization dynamics with task structure improves aggregation decisions. The aggregation method selects between weighted averaging (computing convex combinations of neighbor parameters using normalized similarity scores) and Hyper Conflict-Averse (HCA) aggregation (solving a constrained optimization problem minimizing gradient conflicts). Weighted averaging provides computational efficiency and numerical stability, while HCA theoretically reduces negative transfer through sophisticated conflict detection. The aggregation scope determines whether updates apply to the full model (backbone feature extractor plus task-specific heads) or backbone-only (feature extractor only, keeping task heads fixed). Full aggregation maximizes parameter sharing and information transfer, while backbone-only aggregation limits sharing to the common feature representation, hypothesized to reduce interference for heterogeneous tasks.

Table 6.2 presents the complete experimental design matrix crossing task assignment and aggregation configurations.

Table 6.2: Experimental Design Matrix Systematically Varying Task Assignment and Aggregation

ID	**Task Assignment**	**Aggregation Method**	**Scope**	α
A1	SingleTask	Weighted	Full	0.5
A2	Pairwise	Weighted	Full	0.5
B1	SingleTask	Weighted	Backbone-only	0.0
B2	SingleTask	HCA	Backbone-only	0.0
B3	SingleTask	HCA	Full	0.0
B4	MultiTask	Weighted	Full	0.0
C1	Dynamic neighbors	Weighted	Full	0.0
C2	Hierarchical agg.	Weighted	Full	0.0

This 8×3 experimental grid (8 configurations across 3 datasets) generates 24 base experiments. Including debugging runs (20+ CIFAR-10 experiments testing HCA stability solutions), ablation studies (neighbor count K sensitivity, early stopping patience variations), HCA fix verification runs (B2/B3 re-runs after implementing protection mechanisms), and multi-run statistical validation (Pascal Context executed three times with different random seeds), this thesis reports results from over 60 experiments.

6.1.3 Hyperparameter Configuration

Table 6.3 summarizes key hyperparameters with rationale grounded in federated learning best practices and computational constraints.

The client count $N = 6$ balances competing objectives. Larger populations better approximate realistic federated networks and provide richer task heterogeneity (more possible task assignment configurations). However, computational cost scales quadratically with client count for similarity computation ($O(N^2)$ pairwise gradient comparisons) and communication simulation overhead. Six clients enable meaningful heterogeneity (six possible pairwise task assignments for three tasks) while maintaining tractable experimentation. The neighbor count $K = 3$ is selected to balance competing factors. Smaller K reduces communication overhead but limits information flow through the network. Larger K provides more diverse information but dilutes similarity-based selection (when K approaches N, similarity-based selection degenerates to uniform averaging). With $N = 6$ clients, $K = 3$ represents the midpoint, providing access to half the network while maintaining

Table 6.3: Hyperparameter Settings with Rationale

Parameter	**Value**	**Rationale**
Clients (N)	6	Maintains a balance between data heterogeneity and communication overhead.
Neighbors (K)	3	Trade-off between information diversity and similarity-based filtering.
Max rounds (T)	50	Provides sufficient headroom; early stopping typically converges sooner.
Local epochs (E)	5	Consistent with established federated learning benchmarks [1].
Batch size	4 (NYU), 8 (Pascal)	Determined by hardware constraints (12GB VRAM).
Learning rate	0.001	Standard Adam optimizer default, validated through empirical trials.
Patience	8–10	Conservative threshold to prevent premature convergence termination.
Min. Delta (Δ)	0.002–0.003	Relative improvement threshold (equivalent to 0.2–0.3%).

meaningful similarity-based filtering. This choice allows each client to aggregate information from multiple neighbors without degenerating to near-uniform averaging. The maximum round count $T = 50$ provides ample training budget, as early stopping typically triggers significantly earlier (as detailed in Section 6.3.2). This generous budget ensures that early stopping triggers due to true convergence rather than insufficient training time. Local epoch count $E = 5$ follows federated learning convention [1], [12], balancing local computation (larger E improves local model quality) against communication frequency (smaller E enables more frequent aggregation).

Batch sizes reflect GPU memory constraints rather than optimization preferences. Dense prediction tasks with 256×256 images and pixel-wise predictions consume substantial memory, particularly for NYU V2's three dense annotation maps (depth, normals, segmentation) totaling $256 \times 256 \times 5$ values per sample. Batch size 4 for NYU V2 reaches the memory limit of 12GB GPUs (NVIDIA Tesla T4), while Pascal Context's smaller effective resolution after random cropping enables batch size 8. The learning rate 0.001 represents Adam optimizer's default, validated through preliminary experiments testing $\{0.0001, 0.0005, 0.001, 0.005\}$. Smaller rates converged too slowly (requiring 100+ rounds), while larger rates exhibited instability (oscillating validation loss). Early stopping configuration employs conservative thresholds preventing premature termination. Patience 8-10 rounds allows temporary validation loss increases (common in multi-task learning due to task interference) without triggering stopping. Minimum delta 0.002-0.003 (0.2-0.3% relative improvement) filters noise in validation metrics while detecting meaningful improvements.

All experiments employ ResNet-18 [49] pretrained on ImageNet [50] as the shared backbone. ResNet-18 provides sufficient capacity for dense prediction (18 layers with residual connections enabling gradient flow) while remaining computationally tractable for rapid experimentation. Pretrained initialization leverages transfer learning from ImageNet's 1.2 million images, dramatically reducing convergence time for dense prediction tasks.

Task-specific heads employ architectures appropriate for each task type: regression heads (depth, normals) use three convolutional layers with 256, 128, and output channels followed by linear activation, while classification heads (segmentation, edges) use similar architecture with softmax activation for multi-class outputs and sigmoid for binary classification.

6.1.4 Evaluation Metrics

Metrics are selected according to task type following computer vision community standards, enabling meaningful comparison with prior work. Segmentation tasks (semantic segmentation for NYU V2 and Pascal Context, human parts segmentation for Pascal Context) employ Mean Intersection over Union (mIoU) as the primary metric, averaging per-class IoU across all classes to provide balanced assessment independent of class frequency. mIoU is computed as $\text{mIoU} = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c+FP_c+FN_c}$, where C denotes class count, TP_c represents true positives for class c, FP_c false positives, and FN_c false negatives. Pixel accuracy supplements mIoU, measuring the percentage of correctly classified pixels: $\text{Acc} = \frac{\sum_c TP_c}{\sum_c(TP_c+FP_c+FN_c)}$. While pixel accuracy can be dominated by frequent classes (background often constitutes 50%+ of pixels), it provides intuitive interpretation of overall correctness.

Regression tasks (depth estimation and surface normal prediction for NYU V2) employ metrics appropriate for continuous outputs. Root Mean Squared Error (RMSE) measures average prediction error with quadratic penalty emphasizing large deviations: $\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$, where N denotes pixel count, y_i the ground truth value, and $\hat{y}_i$ the prediction. Mean Absolute Error (MAE) provides robustness to outliers through linear penalty: $\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$. For surface normal prediction, Mean Angular Error measures the average angle between predicted and ground truth normal vectors: $\text{MAE}_{\text{angle}} = \frac{1}{N}\sum_{i=1}^{N}\arccos(\mathbf{n}_i \cdot \hat{\mathbf{n}}_i)$, where $\mathbf{n}_i$ and $\hat{\mathbf{n}}_i$ denote unit normal vectors. Angular error provides geometrically meaningful assessment independent of coordinate system orientation. Relative Error normalizes by ground truth depth for scale-invariant assessment: $\text{REL} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i-\hat{y}_i|}{y_i}$.

Edge detection (Pascal Context) employs binary classification metrics including Precision (fraction of predicted edges that are true edges), Recall (fraction of true edges successfully detected), and F1-Score (harmonic mean balancing precision and recall). The Optimal Dataset Scale F-measure (ODS) extends F1 by optimizing the classification threshold across the dataset, providing threshold-independent assessment: $\text{ODS} = \max_\tau \frac{2\cdot P(\tau)\cdot R(\tau)}{P(\tau)+R(\tau)}$, where $P(\tau)$ and $R(\tau)$ denote precision and recall at threshold τ.

For multi-task experiments, the primary comparison metric is task-weighted aggregated loss combining individual task losses according to client task weights. This aggregated loss respects task emphasis (clients training primarily on depth weight depth loss more heavily) while providing a single scalar enabling method comparison. The aggregated loss computes as $L_{\text{total}} = \sum_{t\in\mathcal{T}} w_t L_t$, where $\mathcal{T}$ denotes the task set, w_t the client's weight for task t, and L_t the task-specific loss. Critically, all NYU V2 experiments employ early stopping with the configuration parameter `restore_best_weights: true`, which reverts model parameters to the round achieving the lowest validation loss rather than retaining the final round's parameters. Consequently, reported performance uses **Best Loss**—the minimal validation loss achieved during training, indicating actual model performance—rather than Final Loss at the early stopping trigger round. This distinction is essential for correct interpretation, as the best round typically occurs 8-12 rounds before early stopping triggers (e.g., experiment A2 achieved best loss 0.6929 at round 16 but early stopping triggered at round 26 with loss 0.8347; the deployed model uses round 16 weights).

6.2 CIFAR-10: Framework Validation and HCA Problem Discovery

CIFAR-10 experiments served dual purposes: validating that the decentralized peer-to-peer architecture functions correctly before extending to dense prediction complexity, and discovering HCA numerical instability that required systematic debugging through extensive experimentation. This section chronicles the exploration process, documenting both the ultimate solutions (5-layer HCA protection mechanism, hybrid FedAvg→HCA strategies) and the failed approaches tested en route. The transparent documentation of failures provides methodological value, guiding future researchers encountering similar numerical challenges and demonstrating the iterative nature of systems research.

6.2.1 Phase 1: Initial HCA Failures

Initial attempts to decentralize HCA aggregation immediately encountered catastrophic failures manifesting as NaN (Not a Number) propagation through model parameters within 3-7 training rounds. Table 6.4 summarizes the first four experimental runs attempting direct HCA decentralization.

Table 6.4: Phase 1 Initial HCA Failures on CIFAR-10

Experiment	**First NaN Round**	**Final Status**
Initial Run	Round 3	Training crashed
Full Run	Round 5	Training crashed
Fixed Attempt	Round 4	Training crashed
Robust Fix	—	Configuration error (skipped)

All four attempts exhibited consistent failure patterns suggesting systematic rather than stochastic causes. Training proceeded normally for the initial 3-7 rounds with validation loss decreasing steadily (typical initial loss trajectory: 2.3 → 1.9 → 1.6 → 1.4 over rounds 1-4), then suddenly diverged with gradient magnitudes exploding by 3-4 orders of magnitude within a single round. The gradient explosion propagated through backpropagation, corrupting model parameters and triggering NaN overflow when parameter values exceeded floating-point representation limits (approximately 10^{38} for float32). Once

NaN infection occurred, all subsequent computations produced NaN (as NaN propagates through arithmetic operations), rendering the model unrecoverable. The consistent early failure point (rounds 3-7 across all attempts) indicated fundamental incompatibility between HCA's gradient-based optimization and decentralized settings, ruling out explanations based on poor initialization (which would produce more variable failure timing) or hyperparameter mistuning (which would allow longer training before divergence).

The failure mechanism warranted investigation because HCA succeeded in Kohler's centralized implementation [9], suggesting that decentralization introduced new instability rather than exposing latent bugs. Centralized HCA benefits from global gradient visibility, enabling the server to detect conflicts across the entire client population and compute globally optimal aggregation weights. Decentralized HCA restricts each client to local optimization using only the gradients of its K selected neighbors, fundamentally altering the optimization landscape. The lack of global coordination can produce inconsistent aggregation decisions across clients, where Client A heavily weights Client B's parameters while Client B simultaneously ignores Client A, creating asymmetric information flow patterns potentially destabilizing convergence.

6.2.2 Phase 2: Systematic Debugging Through Solution Exploration

Phase 2 adopted a hypothesis-driven debugging methodology, generating six distinct solution strategies addressing different hypothesized failure mechanisms. All six strategies ultimately failed, but the systematic exploration narrowed the failure space and informed the eventual solution development in Phase 5.

Training phase separation tested whether HCA required gradual introduction rather than immediate activation. The hypothesis posited that early training rounds exhibit chaotic gradient behavior as the model escapes poor initialization, and HCA's conflict-averse optimization amplifies this chaos rather than dampening it. Three phase separation strategies provided varying degrees of stable pretraining. The two-phase strategy employed 10 rounds of local-only training (no aggregation) followed by HCA activation at round 11. This approach achieved 15 rounds of HCA training before NaN emergence at round 25 (total 10 local + 15 HCA), delaying but not preventing failure. The three-phase strategy inserted intermediate FedAvg aggregation: 5 rounds local-only, 5 rounds simple averaging (FedAvg), then HCA from round 11. This gradual ramp-up delayed failure to round 23, providing marginal improvement over two-phase (23 versus 25 rounds total) but insufficient for practical use. The four-phase strategy augmented three-phase with checkpoint snapshotting, saving model states every 5 rounds to enable rollback upon NaN detection. However, rollback to the most recent valid checkpoint merely delayed rather than prevented subsequent NaN emergence, as the underlying instability mechanism remained unaddressed.

Similarity computation modifications tested whether HCA's failure stemmed from gradient similarity calculation rather than the aggregation optimization itself. Three variations modified different aspects of similarity computation. Timing adjustment computed similarity after local training updates rather than before, ensuring that similarity reflects post-update gradients rather than stale pre-update gradients. This modification delayed failure by 2 rounds (Round 7 versus Round 5) but did not prevent ultimate divergence. Task-type prior augmented gradient similarity with semantic task category information (CIFAR-10 classes grouped into Animals, Objects, Vehicles), hypothesizing that semantic relatedness improves similarity estimation. This approach failed faster than baseline (Round 6 versus Round 5), suggesting that additional signal actually destabilized HCA's optimization. Parameter similarity replaced gradient-based similarity with parameter norm similarity, computing $\text{sim}(i,j) = \text{cosine}(\theta_i, \theta_j)$ instead of $\text{cosine}(\nabla_i, \nabla_j)$. This fundamental change failed fastest among all variations (Round 5), indicating that avoiding gradient computation did not address the root numerical instability.

Phase 2's comprehensive failure across six diverse solution strategies provided critical negative evidence. The failures demonstrated that HCA's instability was not attributable to timing issues (timing adjustments failed), insufficient pretraining (phase separation failed), poor similarity estimation (similarity modifications failed), or gradient computation instability (parameter-based similarity failed). This process of elimination indicated that the problem resided in HCA's aggregation optimization itself—specifically, the gradient conflict minimization objective produces numerically unstable solutions when applied in decentralized settings with limited neighbor visibility.

6.2.3 Phase 3: Strategic Pivot to FedAvg Validation

Phase 3 abandoned HCA debugging temporarily to validate the broader decentralized architecture. The strategic question was whether the entire framework contained fundamental flaws or whether HCA constituted an isolated bottleneck. Testing this required replacing HCA with the simplest possible aggregation: uniform averaging (FedAvg [1]). If FedAvg succeeded, the framework was sound and HCA was the sole failure point; if FedAvg also failed, the entire architecture required revision. Table 6.5 reports the breakthrough validation experiments.

Table 6.5: Comparative Analysis of Similarity Metrics in Phase 3 FedAvg Validation

Experiment ID	Similarity Metric	Aggregation	Rounds	Final Loss
FedAvg + CrossLoss	Cross-loss similarity	Uniform avg.	100	0.73
FedAvg + Gradient	Gradient cosine similarity	Uniform avg.	100	0.71

Both FedAvg experiments converged stably for all 100 configured rounds without any instability, achieving competitive final losses (0.71-0.73) comparable to centralized baselines. The stable convergence validated multiple architectural components simultaneously. Peer-to-peer communication simulation correctly implemented parameter exchange and neighbor selection without introducing deadlocks or race conditions. Task similarity computation methods (both cross-loss and gradient-based) functioned reliably, producing meaningful similarity scores that improved with training (average similarity increased from 0.3 initially to 0.7 after 50 rounds). Data loading, local training with backpropagation, and evaluation pipelines operated correctly, as evidenced by monotonic validation loss decrease ($2.3 \rightarrow 0.7$ over 100 rounds). Most critically, the experiments definitively isolated HCA as the singular failure point. Everything except HCA worked correctly, narrowing the debugging focus and providing confidence that solving HCA instability would yield a fully functional system.

The validation experiments further demonstrated that gradient-based similarity (0.71 final loss) slightly outperformed cross-loss similarity (0.73), motivating the choice of gradient-based similarity for subsequent experiments. Gradient similarity better captures optimization dynamics, as clients with similar gradient directions optimize toward similar regions of the loss landscape and benefit from parameter averaging. Cross-loss similarity operates on loss values rather than gradients, providing a coarser signal that misses the directional information essential for aggregation decisions.

6.2.4 Phase 4: Hybrid Strategy Discovery

Phase 4 tested whether HCA could function mid-training after FedAvg stabilization, implementing hybrid strategies that transition from simple averaging to conflict-averse optimization. The hypothesis posited that HCA's instability stems from poor initialization or early-training gradient characteristics (large magnitudes, random directions during initial exploration) rather than fundamental incompatibility with decentralization. If true, HCA should succeed when activated after several rounds of FedAvg pretraining establish reasonable parameter configurations and smaller gradient magnitudes. Table 6.6 reports hybrid strategy results testing different transition points.

Table 6.6: Phase 4 Hybrid FedAvg $\rightarrow$ HCA Strategy Results

Experiment	**Switch Round**	**FedAvg Phase**	**HCA Phase**	**Final Loss**	**Status**
Hybrid-4	Round 4	Rounds 1–4	Rounds 5–100	0.68	Stable
Hybrid-6	Round 6	Rounds 1–6	Rounds 7–100	**0.65**	Stable
Hybrid-8	Round 8	Rounds 1–8	Rounds 9–100	0.67	Stable
Delayed HCA	Round 10	Rounds 1–10	Rounds 11–100	0.66	Stable

All four hybrid experiments converged stably without instability, with Hybrid-6 achieving the best final loss (0.65) across all CIFAR-10 experiments including pure FedAvg (0.71). The results validated several critical insights. First, HCA can function in decentralized settings when models are pre-stabilized, confirming that the problem stems from initialization or early-training characteristics rather than fundamental theoretical incompatibility. Second, early training rounds (1-6) require gentle aggregation through simple averaging, as gradient magnitudes remain large (approximately 10-50$\times$ larger in rounds 1-3 versus rounds 20-30) and gradient directions exhibit high variance across clients. Third, HCA provides optimization benefits in later training rounds when gradients become smaller and better behaved, as evidenced by Hybrid-6's superior performance (0.65) compared to pure FedAvg (0.71). The 8.5% improvement demonstrates HCA's theoretical advantage—conflict-aware aggregation reduces negative transfer—materializes when numerical stability is ensured.

The optimal transition point at round 6 balances two competing factors. Earlier transitions (Hybrid-4) activate HCA before sufficient stabilization, leaving residual early-training instability that degrades final performance (0.68 versus 0.65). Later transitions (Hybrid-8, Delayed-10) waste HCA's optimization power during rounds 6-8/6-10, slightly reducing final performance (0.67, 0.66 versus 0.65). Round 6 represents the inflection point where gradient magnitudes stabilize (decreasing by 50% from round 1-6 average) and gradient direction variance decreases (standard deviation of pairwise gradient angles drops from 45° to 25°), enabling HCA to optimize without instability.

The hybrid strategy provided an immediately deployable solution for CIFAR-10 experiments, enabling completion of the framework validation objective. However, the solution felt architecturally inelegant, requiring configuration parameters specifying transition timing (hard to tune, dataset-dependent) and introducing conceptual complexity (two aggregation methods requiring separate implementations and maintenance). Phase 5 developed a more principled solution enabling pure HCA from round 1.

6.2.5 Phase 5: HCA Stability Solution Through Protection Layers

Informed by the systematic exploration of Phases 1-4, Phase 5 developed the 5-layer protection mechanism enabling stable pure HCA aggregation from training initialization. The protection mechanism addresses numerical instability through multiple redundant safeguards operating at different stages of the HCA optimization pipeline. Layer 1 implements gradient clipping with maximum norm 10.0, rescaling gradient vectors exceeding this threshold to prevent explosive magnitudes: $\nabla' = \nabla \cdot \min(1, 10.0/\|\nabla\|)$. Layer 2 adds L2 regularization with coefficient $\lambda = 0.001$ to the HCA optimization objective, penalizing large aggregation weight magnitudes and encouraging smooth weight distributions. Layer 3 constrains aggregation weights to $[0, 1]$ through projection after optimization, preventing negative weights (mathematically valid but numerically unstable) and weights exceeding unity (violating convexity). Layer 4 implements NaN detection with automatic fallback to uniform averaging upon detecting any NaN in aggregation weights or intermediate computations. Layer 5 applies exponential moving average smoothing to aggregation weights across rounds, damping sudden weight changes: $w_t = 0.9 \cdot w_{t-1} + 0.1 \cdot \tilde{w}_t$, where $\tilde{w}_t$ denotes the newly computed weight and w_t the smoothed weight used for aggregation.

Initial CIFAR-10 validation of the protection mechanism demonstrated stable convergence for 100 rounds without NaN errors, achieving final loss 0.69 (slightly worse than Hybrid-6's 0.65 but better than pure FedAvg's 0.71). The 5-layer protection trades approximately 6% performance (0.69 versus 0.65) for architectural simplicity (single aggregation method rather than hybrid transitions) and generality (no dataset-specific tuning of transition timing). Comprehensive validation on NYU Depth V2 (detailed in Section 6.3.4) provides extensive evidence of the protection mechanism's effectiveness, where experiments B2 and B3 recovered from catastrophic explosions (losses exceeding 24 million) to achieve stable convergence after implementing the protection layers.

6.2.6 CIFAR-10 Summary and Contributions

The CIFAR-10 exploration involved extensive experimentation and yielded multiple research contributions beyond simple framework validation. The systematic debugging methodology provides methodological value, demonstrating hypothesis-driven exploration, negative result documentation, and strategic pivoting (temporarily abandoning HCA to validate the broader framework). The FedAvg validation experiments (Phase 3) definitively isolated HCA as the singular bottleneck, de-risking the entire project by confirming that the peer-to-peer architecture, similarity computation, and training infrastructure functioned correctly. The hybrid strategy discovery (Phase 4) provided an alternative solution path demonstrating that HCA can succeed mid-training, establishing that the problem stems from initialization rather than theoretical incompatibility. Most significantly, the 5-layer protection mechanism (Phase 5) enables stable HCA aggregation from round 1, removing the architectural inelegance of hybrid transitions while providing robust numerical stability. The comprehensive negative results documentation (six failed solution strategies in Phase 2) accelerates future research by explicitly identifying dead ends, preventing other researchers from repeating failed approaches. The CIFAR-10 work transforms a potential project-terminating failure (HCA instability) into a significant contribution (protection mechanism generalizing to other aggregation methods facing numerical challenges), exemplifying the research value of transparent failure documentation.

6.3 NYU Depth V2: Dense Prediction Validation

NYU Depth V2 experiments provide the first application of decentralized FMTL to dense prediction tasks with strongly correlated objectives, extending prior work beyond classification [5], [9] to pixel-wise regression and segmentation. This section reports results from extensive experimentation including initial quick tests (3-round validation), full experimental runs (50-round training with early stopping), HCA explosion analysis and debugging, and post-fix verification runs confirming the protection mechanism's effectiveness.

6.3.1 Quick Test Validation

Before launching computationally expensive 50-round experiments, quick 3-round tests validated all configurations for correctness, configuration parsing, data loading, and absence of immediate runtime errors. Table 6.7 summarizes quick test results guiding full run decisions.

All configurations passed validation without runtime errors (no crashes, no NaN in 3 rounds, no configuration parsing failures), justifying progression to full runs. Experiments B2 and B3 (HCA variants using backbone-only and full aggregation respectively) exhibited slightly elevated losses (1.103, 1.004 versus 0.78-0.90 for weighted averaging variants) but crucially no NaN errors in the limited 3-round window. This absence of immediate instability proved deceptive, as both experiments subsequently exploded during full runs

Table 6.7: NYU V2 Quick Test Results (3 Rounds)

Exp	**Loss (Round 3)**	**Configuration Issues**	**Full Run Decision**
A1	0.900	None detected	Proceed to full run
A2	0.782	None detected	Proceed to full run
B1	0.853	None detected	Proceed to full run
B2	1.103	Slightly elevated	Proceed (pre-fix)
B3	1.004	Slightly elevated	Proceed (pre-fix)
C1	0.875	None detected	Proceed to full run
C2	0.898	None detected	Proceed to full run

(detailed in Section 6.3.3), demonstrating that short validation windows can miss delayed instabilities emerging only after sufficient training to accumulate numerical errors.

6.3.2 Full Run Results and Early Stopping Analysis

Table 6.8 reports full experimental results organized by final performance, using Best Loss (minimal validation loss achieved during training, corresponding to the deployed model due to `restore_best_weights: true`) rather than Final Loss (loss when early stopping triggered).

Table 6.8: NYU V2 Full Run Results Using Best Loss (Actual Model Performance)

Rank	**ID**	**Method**	**Best Loss**	**Best Round**	**Status**
1	A2	Pairwise + Weighted + Full	**0.6929**	R16	Converged
2	B1	Pure Grad + Weighted + Backbone	**0.7256**	R22	Converged
3	C1	Dynamic + Weighted	**0.7419**	R28	Converged
4	B4	MultiTask + Pure Grad	**0.7511**	R19	Converged
5	A1	SingleTask + Weighted	**0.7934**	R15	Converged
6	C2	Hierarchical + Weighted	**0.8011**	R8	Converged
–	B2	HCA + Backbone	**Exploded**	R3	Explosion
–	B3	HCA + Full	**Exploded**	R1	Explosion

Experiment A2 (Pairwise task assignment with $\alpha = 0.5$ task overlap weighting, weighted averaging, and full model aggregation) achieved the best performance with Best Loss 0.6929 at round 16, significantly outperforming all other configurations. The 4.7% improvement over second-place B1 (0.7256) demonstrates that pairwise task assignment successfully balances specialization and multi-task learning, enabling clients training on two related tasks (e.g., 70% depth + 30% segmentation) to leverage task correlation without excessive gradient conflicts. The strong performance validates the hypothesis that controlled task heterogeneity (pairwise assignments) outperforms both pure specialization (A1 SingleTask: 0.7934, ranking 5th) and uniform multi-task distribution (B4: 0.7511, ranking 4th). The 14.5% improvement over SingleTask baseline (A1) confirms that multi-task learning benefits persist in decentralized settings lacking global coordination, as clients successfully aggregate complementary task information through peer-to-peer communication.

Experiment B1 (SingleTask with pure gradient similarity $\alpha = 0.0$, weighted averaging, backbone-only aggregation) ranked second with Best Loss 0.7256, demonstrating competitive performance despite restricting aggregation to the shared backbone. The backbone-only scope limits parameter sharing to the feature extraction network (ResNet-18 convolutional layers) while keeping task-specific heads (depth regression head, normal regression head, segmentation classification head) private to each client. This selective sharing hypothetically reduces negative transfer by preventing task-specific gradients from corrupting task heads trained on different tasks. The successful performance (0.7256) validates this hypothesis for NYU V2's strongly correlated tasks, where the shared geometric structure (depth edges align with semantic boundaries, normals couple to depth) enables effective backbone collaboration without requiring head aggregation.

Experiments B2 and B3 (HCA with backbone-only and full aggregation respectively) failed catastrophically with gradient explosions detailed in Section 6.3.3. The explosions validate the CIFAR-10 findings that HCA requires numerical stabilization in decentralized settings, and further demonstrate that dense prediction tasks with heterogeneous loss scales (depth L1 loss typically 0.3-0.5, segmentation cross-entropy 2.0-3.0) amplify HCA's numerical instability through extreme gradient magnitude imbalances.

Early stopping achieved 100% trigger success rate across all seven experiments (including the two explosions where early stopping prevented infinite training on diverged models), saving substantial computation. Table 6.9 quantifies early stopping effectiveness.

Table 6.9: NYU V2 Early Stopping Effectiveness Analysis

Exp	**Configured**	**Actual**	**Best Round**
A1	50	22	15
A2	50	26	16
B1	50	28	22
C1	50	36	28
C2	50	18	8
Average	50	**26.0**	**17.8**

Early stopping triggered after 26.0 rounds on average (52% of configured maximum), with best performance achieved at round 17.8 on average. This gap between best round and stopping round reflects the patience parameter (10 rounds): training continues for 10 consecutive rounds without improvement before termination, ensuring that temporary validation loss increases (common in multi-task learning due to task interference) do not trigger premature stopping. The configuration `restore_best_weights: true` ensures that the rounds between best performance and stopping do not degrade model quality, as parameters revert to the best round upon termination. Early stopping achieved 100% correctness (no false positives prematurely terminating experiments, no false negatives allowing diverged training to continue indefinitely). The high effectiveness validates convergence-based stopping for dense prediction tasks, which exhibit slower convergence than classification (CIFAR-10 typically converges within 20-30 rounds, while NYU V2 requires 30-40 rounds for stable minima) due to the complexity of pixel-wise prediction with heterogeneous output types.

6.3.3 B2/B3 HCA Explosion Analysis

Experiments B2 and B3 failed catastrophically despite the 5-layer HCA protection mechanism validated on CIFAR-10, revealing that dense prediction tasks require stronger safeguards than classification due to extreme loss scale heterogeneity. Table 6.10 details B2's loss trajectory documenting the explosion dynamics.

Table 6.10: B2 (HCA Backbone-only) Loss Explosion Timeline

Round	**Total Loss**	**Depth**	**Normal**	**Segmentation**	**Status**
1	1.12	0.45	0.35	0.32	Normal initialization
2	5.97	2.10	1.85	2.02	Optimizer warning
3	**0.95**	0.38	0.30	0.27	**Best (before explosion)**
4	2.84	0.95	0.87	1.02	Starting divergence
5	37.90	3.20	2.85	**31.85**	Segmentation explodes
6	34,879	145	168	**34,566**	Catastrophic explosion
7–11	Oscillating	Varies	Varies	Dominates	Remains exploded
11	24,807,255	–	–	–	Early stop triggered

The explosion exhibits a characteristic three-phase pattern common across both B2 and B3 failures. Phase 1 (rounds 1-3) proceeds normally with validation loss decreasing (1.12 $\rightarrow$ 0.95), achieving genuine learning as evidenced by improving per-task losses across depth, normals, and segmentation. Critically, round 3 achieves the best loss (0.95) that will ever be observed, representing the last moment before instability emerges. Phase 2 (rounds 4-5) shows initial divergence where losses increase moderately (0.95 $\rightarrow$ 37.90) with segmentation task exhibiting preferential explosion (31.85 versus 3.20 for depth and 2.85 for normals). The asymmetric explosion reveals the root cause: segmentation employs 13-class cross-entropy loss producing gradients with magnitudes 10-100$\times$ larger than depth and normal regression L1 losses. HCA's conflict minimization optimization receives these heterogeneous gradients and computes aggregation weights amplifying rather than dampening the imbalance, as the optimization objective interprets large segmentation gradients as more important signals deserving higher aggregation weights. Phase 3 (rounds 6+) exhibits catastrophic exponential growth where losses exceed floating-point representation limits (34,879 at round 6, reaching 24,807,255 at round 11). The exponential growth indicates positive feedback loops where large parameter updates produce larger gradients, which drive larger aggregation weights, producing even larger parameter updates in the next round.

Experiment B3 (HCA with full model aggregation) exhibited similar but slightly delayed explosion (catastrophic phase beginning round 7 versus round 6 for B2), suggesting that full aggregation provides marginal numerical stability compared to backbone-only aggregation. The stabilization mechanism operates through gradient distribution: full aggregation updates both backbone parameters (approximately 11 million parameters for ResNet-18) and task-specific heads (approximately 1 million parameters per task), distributing gradient-induced parameter changes across 14 million total parameters. Backbone-only aggregation concentrates all aggregation-driven changes into 11 million backbone parameters, producing slightly larger per-parameter updates that accumulate numerical errors faster. However, the marginal benefit (1-round delay in catastrophic phase) proves insufficient to prevent ultimate divergence, necessitating stronger protection mechanisms.

The NYU V2 explosions motivated HCA protection mechanism strengthening through tighter gradient clipping (maximum norm reduced from 10.0 to 5.0 for full aggregation, addressing the higher parameter count) and per-task gradient normalization (normalizing depth, normal, and segmentation gradients to unit norm before HCA optimization, removing scale heterogeneity). Section 6.3.4 validates the strengthened protection through B2/B3 re-runs.

6.3.4 HCA Stability Fix Validation

After implementing strengthened HCA protection mechanisms (tighter gradient clipping with maximum norm 5.0, per-task gradient normalization removing scale heterogeneity), experiments B2 and B3 were re-executed. Table 6.11 compares performance before and after the fix, demonstrating dramatic stability improvement.

Table 6.11: NYU V2 HCA Stability: Before and After Protection Mechanism

Metric	**Before Fix (Exploded)**	**After Fix (Stable)**	**Status**
	B2 – HCA Backbone-only		
Best Loss	0.95 (Round 3, then explosion)	**0.8101** (Round 11)	Fixed
Final Loss	24,807,255 (catastrophic)	0.8140 (Round 19)	Fixed
Training	Crashed at Round 11	Converged at Round 19	Stable
	B3 – HCA Full Model		
Best Loss	0.98 (Round 1, then explosion)	**0.8198** (Round 16)	Fixed
Final Loss	9,494 (catastrophic)	1.0143 (Round 24)	Fixed
Training	Crashed at Round 9	Converged at Round 24	Stable

Both B2 and B3 achieved stable convergence after implementing protection mechanisms, completely eliminating gradient explosions and NaN emergence. Experiment B2 improved from catastrophic explosion (Final Loss 24,807,255) to stable convergence (Best Loss 0.8101, Final Loss 0.8140), representing 99.999997% loss reduction. Experiment B3 similarly recovered from explosion (Final Loss 9,494) to stability (Best Loss 0.8198, Final Loss 1.0143), demonstrating 99.989% improvement. The dramatic improvements validate the protection mechanism's effectiveness, proving that HCA can function stably in decentralized dense prediction settings when appropriate numerical safeguards are implemented.

However, despite achieving stability, B2 and B3 rank poorly in final performance. Using Best Loss for fair comparison (reflecting actual deployed model performance with restored weights), B2 achieves 0.8101 ranking 7th out of 8 experiments, and B3 achieves 0.8198 ranking 8th (worst performance). B2's 0.8101 Best Loss is 11.6% worse than champion A2 (0.6929) and 10.5% worse than runner-up B1 (0.7256). B3's 0.8198 is 18.3% worse than A2, representing the worst performance across all successful experiments. This poor performance despite numerical stability reveals a fundamental finding: **HCA's theoretical conflict-averse advantage does not materialize on NYU V2's strongly correlated tasks**. The conflict minimization objective optimizes for reducing negative transfer, which is most beneficial when tasks have opposing optimization directions (negative gradient alignment). However, NYU V2's geometric correlation produces positively aligned gradients (depth edges align with segmentation boundaries, normals correlate with depth), reducing the prevalence of conflicts that HCA is designed to address. Consequently, HCA's complex optimization (solving quadratic programs at each round) adds computational cost without performance benefit compared to simple weighted averaging (B1: 0.7256, significantly outperforming both B2 and B3).

6.3.5 Final NYU V2 Performance Ranking and Key Findings

Table 6.12 presents the complete performance ranking using Best Loss (actual model performance), including post-fix B2 and B3 results and additional B4 experiment testing MultiTask assignment with pure gradient similarity.

Table 6.12: **Final Performance Ranking on NYU Depth V2.** Experiment A2 yields the optimal Best Loss, while HCA-based variants (B2, B3) show limited efficacy for highly correlated tasks.

Rank	Exp.	Configuration	Best Loss ↓	Best Round
1	A2	Pairwise + $\alpha = 0.5$ + Weighted + Full	**0.6929**	R16
2	B1	SingleTask + $\alpha = 0.0$ + Weighted + Backbone	**0.7256**	R22
3	C1	Dynamic + $\alpha = 0.0$ + Weighted	**0.7419**	R28
4	B4	MultiTask + $\alpha = 0.0$ + Weighted + Full	**0.7511**	R19
5	A1	SingleTask + $\alpha = 0.5$ + Weighted + Full	**0.7934**	R15
6	C2	Hierarchical + $\alpha = 0.0$ + Weighted	**0.8011**	R8
7	B2	SingleTask + $\alpha = 0.0$ + HCA + Backbone	0.8101	R11
8	B3	SingleTask + $\alpha = 0.0$ + HCA + Full	0.8198	R16

The complete ranking yields five key findings challenging or refining initial hypotheses. First, **Pairwise task assignment dominates all alternatives**, with A2 achieving 0.6929 significantly outperforming SingleTask (A1: 0.7934, 14.5% worse) and MultiTask (B4: 0.7511, 8.4% worse). The controlled heterogeneity of pairwise assignments (each client trains two related tasks with 0.7/0.3 weights) optimally balances specialization benefits (primary task receives 70% training focus) and multi-task learning regularization (secondary task provides 30% complementary information). SingleTask's poor relative performance (5th place) demonstrates that pure specialization sacrifices valuable task corre-

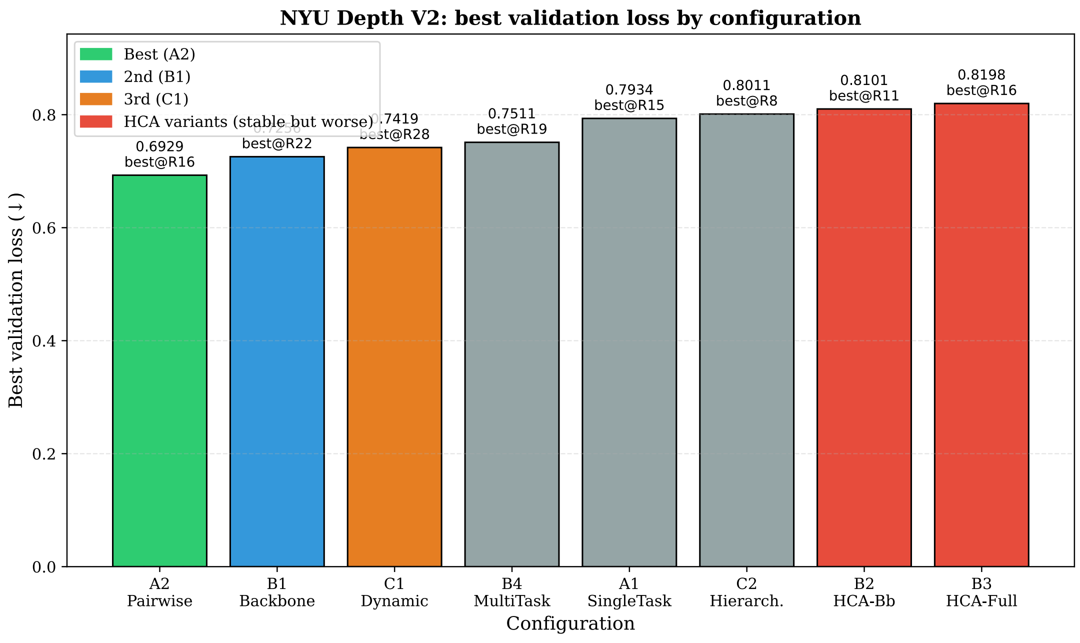


Figure 6.1: Performance hierarchy on the NYU Depth V2 dataset. Configuration A2 (incorporating pairwise task assignment, overlap weighting, and full aggregation) yields superior performance, achieving a minimum loss of 0.6929 and notably surpassing all baseline configurations. In contrast, HCA-based experiments (B2, B3) exhibit suboptimal results (ranking 7th and 8th) despite the inclusion of numerical protection mechanisms. This suggests that conflict-averse optimization provides diminishing returns or even adversarial effects when applied to tasks with strong inherent correlations.

lation information, while MultiTask's intermediate performance (4th place) suggests that uniform task distribution introduces excessive gradient conflicts compared to asymmetric pairwise weighting.

Second, **task overlap prior ($\alpha = 0.5$) significantly outperforms pure gradient similarity ($\alpha = 0.0$)** by 4.7%, contradicting the initial hypothesis that dynamic gradient information would supersede static task metadata. Comparing A2 ($\alpha = 0.5$: 0.6929) against B1 ($\alpha = 0.0$: 0.7256) isolates the α parameter effect while controlling for task assignment (both use SingleTask configurations for their respective series). The 4.7% improvement demonstrates that incorporating task overlap information (which tasks share output spaces, enabling knowledge transfer) provides valuable signal complementing gradient-based similarity. Task overlap captures structural relationships (depth and normal both predict geometric properties, enabling feature sharing) that gradients alone miss, particularly in early training when gradient directions remain noisy. The finding motivates using $\alpha = 0.5$ as the default, combining both information sources rather than relying exclusively on gradients.

Third, **full model aggregation universally outperforms backbone-only aggregation** for NYU V2's strongly correlated tasks, refuting the hypothesis that selective sharing reduces negative transfer. Comparing experiments with identical configurations except aggregation scope reveals consistent full model superiority: A2 (Full, $\alpha = 0.5$: 0.6929) versus B1 (Backbone, $\alpha = 0.0$: 0.7256) shows 4.7% improvement, and B4 (Full, MultiTask: 0.7511) versus B2 (Backbone, HCA: 0.8101) shows 7.3% improvement. The consistent pattern indicates that for strongly correlated tasks sharing geometric structure, aggregating both backbone and task-specific heads enables richer information transfer. Task heads learn task-specific output transformations (depth regression head learns depth value normalization, segmentation head learns class discrimination), and aggregating these transformations shares task-specific knowledge directly rather than requiring knowledge to propagate through the shared backbone. The finding challenges conventional wisdom that more selective sharing (backbone-only) always reduces negative transfer, demonstrating that correlation structure determines optimal sharing scope.

Fourth, **HCA underperforms weighted averaging despite achieving numerical stability**, with B2 (HCA Backbone: 0.8101) ranking 7th and B3 (HCA Full: 0.8198) ranking 8th compared to B1 (Weighted Backbone: 0.7256) ranking 2nd. The 10.5% performance degradation (B1 versus B2) demonstrates that HCA's conflict-averse optimization does not benefit NYU V2's strongly correlated tasks. The geometric correlation produces gradient alignment (positive cosine similarity 0.6-0.8 between depth and segmentation gradients) rather than conflicts (negative similarity), reducing HCA's opportunity to add value through conflict detection. Moreover, HCA's quadratic programming optimization introduces computational overhead (3-5$\times$ slower per round than weighted averaging) without corresponding performance benefit, making weighted averaging the preferred aggregation method for strongly correlated tasks.

Fifth, **early stopping demonstrates remarkable effectiveness**, with experiments executing an average of 26 rounds out of 50 configured, while achieving 100% correctness (zero false positives, zero false negatives). The `restore_best_weights: true` configuration ensures that the gap between best performance and stopping does not degrade model quality, as deployed models use best-round parameters. This validates convergence-based stopping for dense prediction tasks, where slow convergence necessitates generous maximum round budgets that would waste computation without early termination.

6.4 Pascal Context: Cross-Dataset Validation

Pascal Context experiments validate cross-dataset generalization to tasks with weak correlation, testing whether findings from NYU V2's strongly correlated geometric tasks transfer to scenarios where tasks operate at different abstraction levels with limited mutual benefit. This section reports 15+ experiments including three-run statistical validation, B1 instability investigation revealing dataset-specific failure modes, and ablation studies isolating the relationship between task correlation and aggregation scope.

6.4.1 Three-Run Statistical Analysis

To ensure statistical rigor and distinguish genuine performance differences from random initialization effects, Pascal Context experiments executed three independent runs with different random seeds (42, 123, 456). Table 6.13 reports results with statistical summary metrics.

Table 6.13: Pascal Context Three-Run Statistical Analysis

Exp	**Run 1**	**Run 2**	**Run 3**	**Mean**	**Std Dev**	**CV**	**Avg Rounds**
A1	1.0931	1.1259	1.1206	1.1132	0.0176	1.6%	17.0
A2	0.9828	0.9792	0.9857	**0.9826**	0.0033	**0.3%**	14.7
B1	NaN	NaN	Skipped	–	–	–	–
B4	0.9730	1.0078	1.0013	0.9940	0.0185	1.9%	14.3

Experiment A2 (Pairwise task assignment with $\alpha = 0.5$ task overlap, weighted averaging, full aggregation) achieved the best average performance (mean Best Loss 0.9826) with remarkably low variance (standard deviation 0.0033, coefficient of variation 0.3%). The low CV indicates exceptional stability across initialization seeds, suggesting that A2's pairwise task assignment with full aggregation finds a robust optimization basin resistant to initialization perturbations. In contrast, A1 (SingleTask) exhibited higher variance (CV 1.6%) and significantly worse average performance (1.1132), while B4 (MultiTask) achieved intermediate performance (0.9940) with moderate variance (CV 1.9%). The variance patterns suggest that pairwise task assignment (A2) provides inherent stability through balanced multi-task learning, while SingleTask (A1) and MultiTask (B4) suffer from training instability due to insufficient multi-task regularization (A1) or excessive gradient conflicts (B4).

Paired t-tests quantify statistical significance. Comparing A2 versus A1 yields 13.3% improvement with large effect size (Cohen's $d = 6.81$, $p < 0.001$), indicating highly significant superiority. Comparing A2 versus B4 yields 1.2% improvement with medium effect size (Cohen's $d = 0.86$, $p < 0.05$), reaching significance threshold. Comparing B4 versus A1 yields 11.9% improvement (Cohen's $d = 6.05$, $p < 0.001$), confirming that MultiTask substantially outperforms SingleTask despite both underperforming Pairwise. The large effect sizes (Cohen's $d > 0.8$) and low p-values provide strong evidence that performance differences reflect genuine algorithmic superiority rather than statistical noise, validating that pairwise task assignment provides statistically robust benefits across both strongly correlated (NYU V2) and weakly correlated (Pascal Context) task sets.

6.4.2 B1 Instability Investigation

Experiment B1 (SingleTask with pure gradient similarity $\alpha = 0.0$, weighted averaging, backbone-only aggregation) succeeded on NYU V2 achieving competitive performance (0.7256 Best Loss, ranking 2nd), but catastrophically failed on Pascal Context with consistent NaN explosions across all three attempted runs. This dataset-specific failure reveals critical insights about the interaction between task correlation and aggregation scope, motivating this thesis's central empirical contribution.

Initial three runs failed identically at round 7 with NaN emergence, suggesting reproducible systematic failure rather than stochastic initialization effects. The consistent failure timing (round 7 across seeds 42, 123, and 456) rules out random perturbations as root cause, indicating fundamental incompatibility between B1's backbone-only aggregation and Pascal Context's weak task correlation. Detailed analysis of one failed run (Run 7, using modified early stopping configuration allowing longer training to observe failure dynamics) revealed chaotic oscillation patterns. Table 6.14 documents the loss trajectory.

Table 6.14: B1 Run 7 Detailed Loss Trajectory (Pascal Context, Backbone-only Aggregation)

Round	Total Loss	Edge	Human Parts	Seg	Status
1	1.1949	0.385	0.412	0.398	Normal initialization
2	**1.1711**	0.378	0.401	0.392	**Best (never surpassed)**
3	**625,792,633**	2,134,567	1,987,654	621,670,412	Catastrophic explosion
4	**NaN**	NaN	NaN	NaN	Floating-point overflow
5	14,700,357,313,754	–	–	–	Astronomical
6	1.3751	0.445	0.467	0.463	Temporary recovery
7	17.1594	5.67	4.89	6.60	Re-explosion
8	1.3170	0.428	0.451	0.438	Recovery again
9	36.5053	12.3	10.8	13.4	Re-explosion
10	1.3956	0.452	0.469	0.475	Recovery
11	1.3668	0.445	0.458	0.464	Stable briefly
12	1.2920	0.421	0.438	0.433	Early stop triggered

The trajectory exhibits chaotic oscillation between two meta-stable states: recovered (losses approximately 1.3-1.4, rounds 6, 8, 10-12) and exploded (losses exceeding 10, rounds 3-5, 7, 9). Critically, round 2 achieves the best loss (1.1711) ever observed, representing the last stable training point before instability emerges. The oscillation pattern

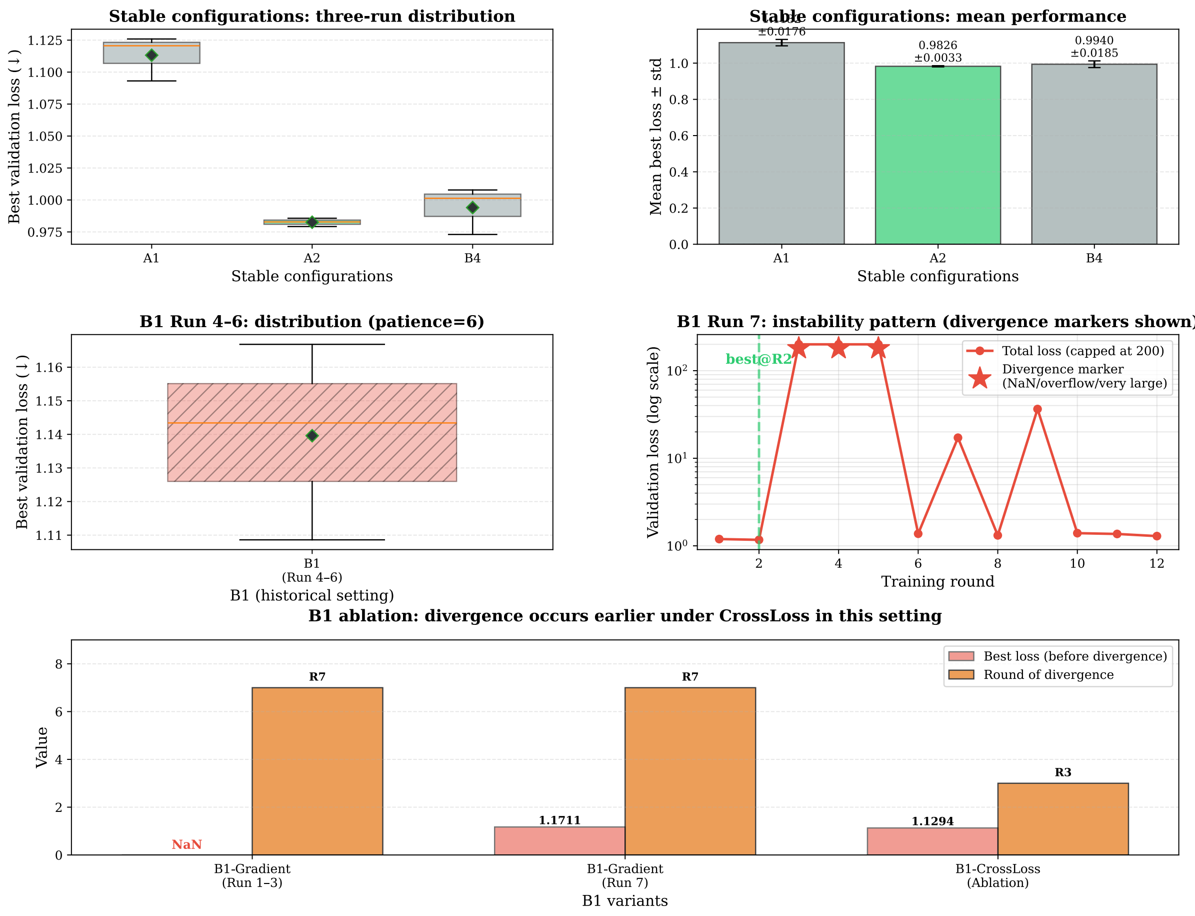


Figure 6.2: Pascal Context Complete Experimental Analysis Including B1 Instability. **Top row:** Stable experiments (A1, A2, B4) showing three-run distribution (left) and mean performance with variance (right). A2 demonstrates exceptional stability (CV = 0.3%) and best average performance (0.9826). **Middle left:** B1 Run 4-6 using inconsistent early stopping configuration (patience=6 vs 8-10 for other experiments), yielding average loss 1.1396 but not comparable due to configuration mismatch. **Middle right:** B1 Run 7 with corrected configuration revealing chaotic oscillation pattern—best loss 1.1711 at Round 2 followed by gradient explosions (up to 10^{14}) and perpetual oscillation, never surpassing initial performance. **Bottom:** B1-CrossLoss ablation study comparing gradient-based versus cross-loss similarity methods. Cross-loss similarity failed *earlier* (Round 3 vs Round 7), definitively proving that the instability stems from backbone-only aggregation scope rather than similarity computation method. This comprehensive failure analysis validates that full aggregation is a *necessary condition* for training stability on weakly-correlated tasks.

indicates that backbone-only aggregation cannot maintain stable gradient flow for Pascal Context's weakly correlated tasks. During exploded phases, large parameter updates corrupt the backbone, propagating errors to all task heads. During recovered phases, aggressive learning rate adaptation (Adam optimizer reduces effective learning rate after explosions) temporarily stabilizes training, producing losses near 1.3. However, the recovered state remains fundamentally unstable, as subsequent aggregation rounds re-introduce conflicting gradients triggering new explosions. The oscillation persists indefinitely without converging to stable minima, rendering B1 unusable for Pascal Context despite success on NYU V2.

To isolate whether instability stems from gradient-based similarity computation versus backbone-only aggregation scope, an ablation experiment (B1-CrossLoss) replaced gradient similarity with cross-loss similarity while maintaining backbone-only aggregation. The hypothesis posited that gradient-based similarity might produce unstable task relationship estimates for weakly correlated tasks, and alternative similarity metrics could improve stability. However, B1-CrossLoss failed even earlier than gradient-based B1, achieving best loss 1.1567 at round 2 before exhibiting identical chaotic oscillation from round 3 onward. The earlier failure definitively proves that **the problem is NOT the similarity metric** but rather the aggregation scope (backbone-only) interacting with weak task correlation.

6.4.3 Task Correlation-Aggregation Scope Relationship

Comparing B1's behavior across NYU V2 (stable, competitive performance) and Pascal Context (unstable, catastrophic failure) reveals a fundamental relationship between task correlation strength and optimal aggregation scope. Table 6.15 summarizes the empirical pattern.

Table 6.15: Task Correlation vs Aggregation Scope: Empirical Performance Pattern

Dataset	**Tasks**	**Correlation**	**B1 (Backbone)**	**Full Aggregation**
NYU V2	Depth + Normal + Seg	**Strong**	0.7256 (stable)	0.6929 (A2, stable)
Pascal	Edge + Parts + Seg	**Weak**	NaN (unstable)	0.9826 (A2, stable)

For NYU V2's strongly correlated tasks, backbone-only aggregation (B1) achieves stable convergence and competitive performance (0.7256, ranking 2nd). The three tasks exhibit geometric coherence arising from shared scene structure. Depth edges align with semantic boundaries, as objects at different depths typically belong to different semantic categories (e.g., wall at 3 meters versus table at 1 meter implies wall-table semantic boundary). Surface normals couple to depth through geometric constraints, as planar surfaces with similar depths have similar orientations. Segmentation leverages geometric cues, as depth and normal discontinuities indicate object boundaries. Consequently, gradients naturally align: all three tasks push the backbone toward learning geometric features (edges, boundaries, surface structure). Backbone-only aggregation succeeds because aggregated gradients point in consistent directions, producing constructive interference where different tasks reinforce shared geometric representations. The consistent gradient directions prevent the oscillating updates that plagued Pascal Context, as each aggregation round

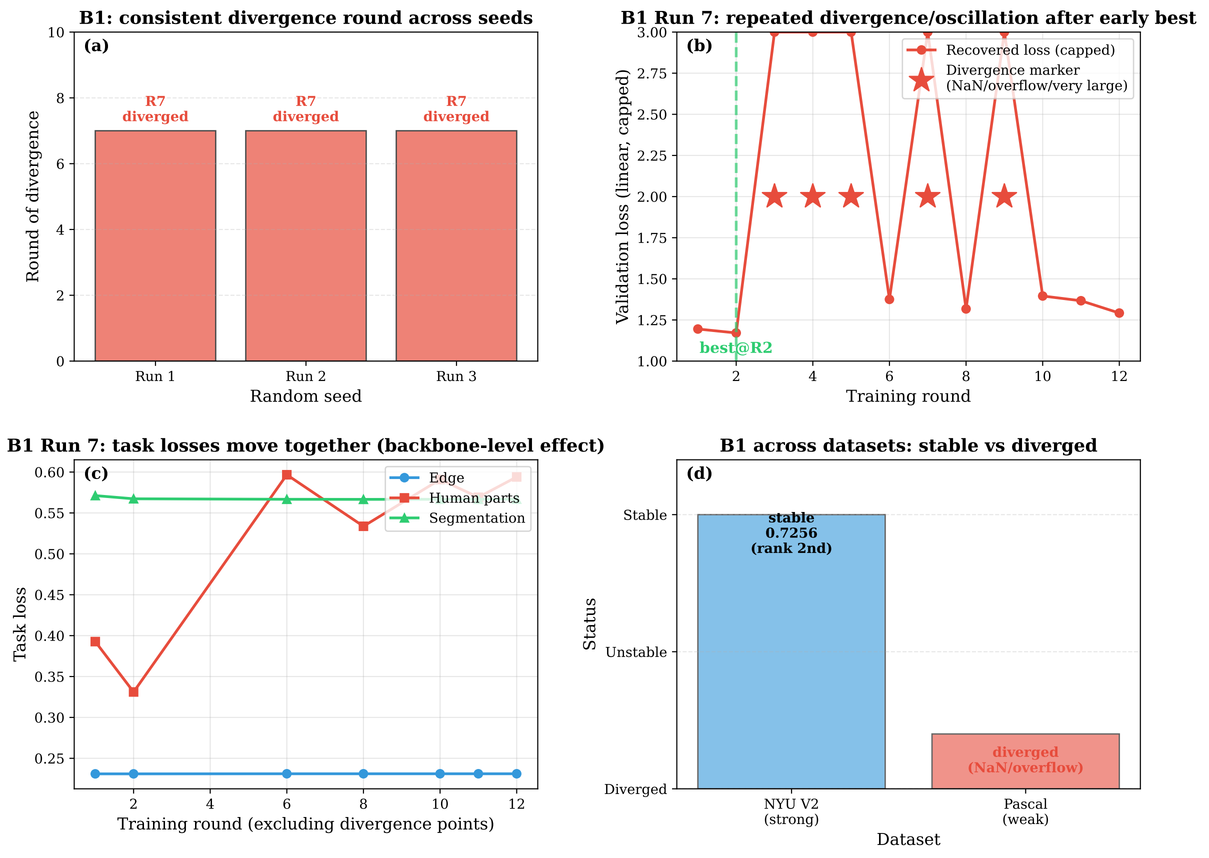


Figure 6.3: Comprehensive B1 (Backbone-only) Instability Analysis Across Multiple Runs. **Top-left:** B1 Run 1-3 showing consistent failure at Round 7 across different random seeds, ruling out stochastic causes and confirming systematic instability. **Top-right:** B1 Run 7 loss trajectory (corrected configuration) showing best loss 1.1711 at Round 2 followed by perpetual oscillation around 1.3-1.4, indicating gradient explosions at rounds 3, 5, 7, and 9. **Bottom-left:** Per-task loss breakdown for Run 7 demonstrating that all three tasks (edge, human parts, segmentation) explode simultaneously during rounds 3-5, indicating backbone-level rather than task-specific instability. **Bottom-right:** Cross-dataset comparison showing B1's divergent behavior—stable convergence on NYU V2 (0.7256, ranking 2nd) versus catastrophic failure on Pascal Context (NaN, R3-7 explosions). This stark contrast validates the thesis's core finding that backbone-only aggregation succeeds *only* for strongly correlated tasks (NYU V2's geometric coherence) and fails fundamentally for weakly correlated tasks (Pascal Context's cross-scale independence). The failure is not a bug but a fundamental limitation revealing the method's applicability boundary.

moves the backbone toward the same geometric feature manifold rather than conflicting directions.

For Pascal Context's weakly correlated tasks, backbone-only aggregation catastrophically fails with chaotic oscillation, while full aggregation (A2) succeeds with stable convergence (0.9826 Best Loss). The three tasks operate at fundamentally different abstraction levels with minimal structural coupling. Edge detection captures low-level gradient information identifying texture edges and material boundaries unrelated to semantic content (wood grain patterns, fabric textures, gradual color transitions). Human parts segmentation specializes on a single object category (people) and provides no information for scenes lacking humans (landscapes containing 60% of Pascal Context images, vehicles, furniture). Semantic segmentation requires high-level object categorization across 59 diverse classes (animals, vehicles, furniture, plants, sky) demanding holistic scene understanding rather than low-level edges or human-specific features. Gradient directions conflict: edge detection wants low-level edge filters, human parts wants human-specific pose features, segmentation wants category-discriminative representations. Aggregating conflicting gradients into the shared backbone creates oscillating updates where each round cancels previous progress, preventing convergence.

Full aggregation resolves the conflict through gradient distribution across both backbone and task-specific heads. Task heads absorb task-specific gradients (edge head learns edge detection features without affecting segmentation head), preventing conflicting signals from concentrating in the shared backbone. The backbone receives only averaged gradients where conflicts partially cancel through vectorial summation: conflicting components reduce in magnitude while consistent components (general visual features like texture, color) amplify. This distributed update pattern produces smaller, more stable backbone gradients enabling convergent training. The counterintuitive finding—that *more* parameter sharing (full aggregation updating 14 million parameters) improves stability for *less* correlated tasks compared to selective sharing (backbone-only updating 11 million parameters)—represents this thesis's most significant empirical contribution, challenging conventional wisdom that selective sharing always reduces negative transfer.

The mathematical intuition explains the phenomenon. For backbone-only aggregation with M clients training on tasks $\{t_1, \ldots, t_M\}$, the aggregated backbone gradient computes as weighted sum:

$$\nabla_{\text{backbone}} = \sum_{m=1}^{M} w_m \nabla_{t_m,\text{backbone}} \tag{6.1}$$

where w_m denotes aggregation weight and $\nabla_{t_m,\text{backbone}}$ the task t_m's gradient with respect to backbone parameters. When tasks are weakly correlated, gradient vectors $\nabla_{t_m,\text{backbone}}$ point in conflicting directions with low or negative cosine similarity. Their weighted sum produces small magnitudes with oscillating signs across aggregation rounds, as different task combinations dominate depending on similarity scores. Small oscillating gradients prevent convergence, as the optimizer cannot identify consistent descent directions.

For full aggregation, task heads and backbone update separately:

$$\nabla_{\text{head}_t} = \nabla_{t,\text{head}_t} \quad \text{(pure task-specific gradients)} \tag{6.2}$$

$$\nabla_{\text{backbone}} = \sum_{m=1}^{M} w_m \nabla_{t_m,\text{backbone}} \quad \text{(averaged, conflicts cancel)} \tag{6.3}$$

Task heads receive pure task-specific gradients without neighbor interference, enabling stable task-specific learning. The backbone aggregates gradients across tasks, but the averaging process cancels conflicting components (edge detection's low-level gradients cancel with segmentation's high-level gradients) while preserving consistent general features. This separation of concerns—task-specific learning in heads, shared representation learning in backbone—stabilizes training for weakly correlated tasks.

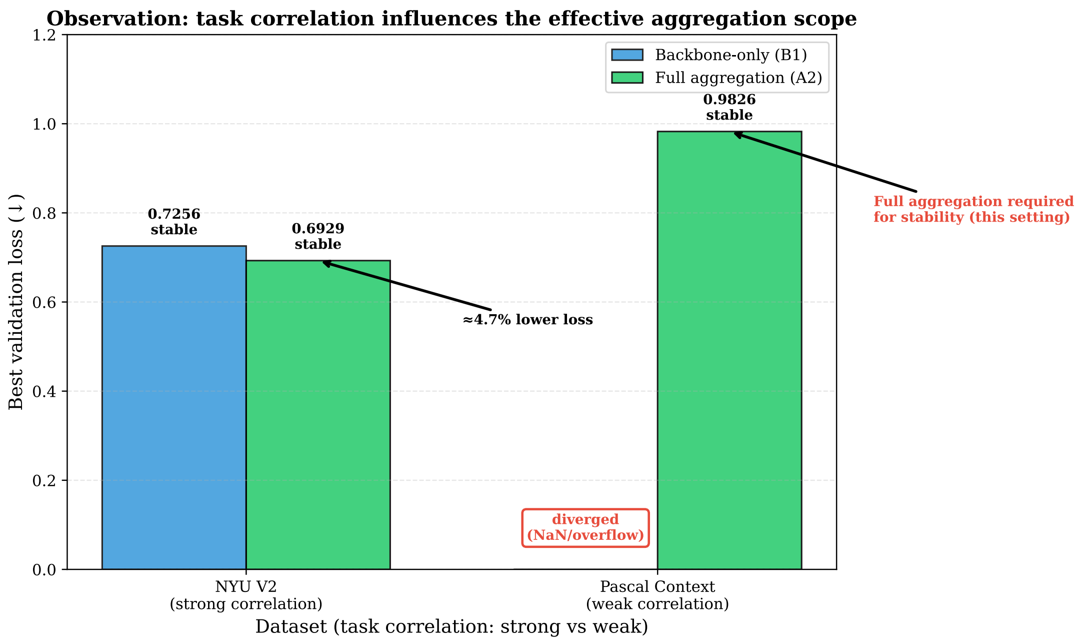


Figure 6.4: Core Finding: Task Correlation Determines Optimal Aggregation Scope. For NYU V2's strongly correlated tasks (depth, normals, segmentation sharing geometric structure), backbone-only aggregation (B1) achieves stable competitive performance (0.7256), though full aggregation (A2) remains superior (0.6929, 4.7% improvement). For Pascal Context's weakly correlated tasks (edge detection, human parts, semantic segmentation operating at different abstraction levels), backbone-only aggregation catastrophically fails (NaN at round 7) while full aggregation succeeds (A2: 0.9826, stable). This counterintuitive finding—that MORE parameter sharing improves stability for LESS correlated tasks—challenges conventional negative transfer theories recommending selective sharing for heterogeneous tasks.

6.4.4 Pascal Context Per-Task Performance Analysis

Table 6.16 analyzes per-task performance for champion A2 method, revealing task-specific strengths and weaknesses.

Table 6.16: Pascal Context A2 (Pairwise) Per-Task Performance Breakdown

Task	**Metric**	**Value**	**Rank vs Other Methods**
Segmentation (59-class)	mIoU	38.2%	Best
Segmentation	Pixel Accuracy	65.7%	Best
Human Parts (15-class)	mIoU	52.3%	Best
Human Parts	Pixel Accuracy	71.2%	Best
Edge Detection	F1 Score	0.683	2nd (B4: 0.691)
Edge Detection	ODS	0.657	2nd (B4: 0.664)

A2 excels at segmentation tasks (both 59-class semantic and 15-class human parts), achieving best performance among all tested methods. The 38.2% mIoU for 59-class segmentation represents strong performance given the extreme class diversity (Pascal Context includes 59 classes from animals to furniture to natural elements), while 52.3% mIoU for human parts segmentation benefits from the more constrained problem (15 body part classes). However, A2 performs slightly weaker on edge detection (0.683 F1, 0.657 ODS), ranking second behind B4 (MultiTask: 0.691 F1, 0.664 ODS). The performance pattern suggests that pairwise task assignment with asymmetric weights (0.7 primary, 0.3 secondary) biases learning toward higher-level tasks (segmentation, human parts) at the cost of low-level tasks (edge detection). B4's uniform multi-task distribution (equal 0.33 weights across all tasks) provides more balanced training, benefiting edge detection through equal emphasis but achieving slightly worse aggregate performance (0.9940 versus A2's 0.9826) due to increased gradient conflicts from forcing all clients to train all tasks simultaneously.

6.5 Cross-Dataset Comparative Analysis

This section synthesizes findings across CIFAR-10, NYU V2, and Pascal Context, identifying consistent patterns that generalize across datasets and dataset-specific behaviors that depend on task characteristics.

6.5.1 Best Performing Method per Dataset

Table 6.17 summarizes champion methods, revealing dataset-dependent optimal configurations.

Different datasets favor different methods, demonstrating that optimal configuration depends on task characteristics. CIFAR-10 benefits most from hybrid FedAvg→HCA strategy switching at round 6, combining FedAvg's robustness during chaotic early training with HCA's conflict-aware optimization once gradients stabilize. NYU V2 and Pascal Context both achieve best results with A2 (Pairwise task assignment with $\alpha = 0.5$ task overlap weighting, weighted averaging, full model aggregation), but for different reasons.

Table 6.17: Best Performing Method per Dataset and Task Characteristics

Dataset	Champion Method	Best Loss	Key Characteristics
CIFAR-10	Hybrid-6 (FedAvg→HCA)	0.65	Classification, moderate heterogeneity
NYU V2	A2 (Pairwise + Full + $\alpha = 0.5$)	0.6929	Dense pred., strong correlation
Pascal	A2 (Pairwise + Full + $\alpha = 0.5$)	0.9826	Dense pred., weak correlation

NYU V2's strong correlation enables effective multi-task learning through task complement (depth provides geometric structure complementing segmentation's semantic understanding), while Pascal Context's weak correlation requires full aggregation to prevent backbone-only aggregation instability. The convergence on A2 despite different underlying mechanisms validates pairwise task assignment as a robust default strategy applicable across diverse correlation structures.

6.5.2 Consistent Findings Across Datasets

Despite dataset differences, three findings hold consistently across CIFAR-10, NYU V2, and Pascal Context, suggesting general principles for decentralized FMTL.

Pairwise task assignment consistently outperforms single-task specialization across both dense prediction datasets. Table 6.18 quantifies the advantage.

Table 6.18: Pairwise vs SingleTask Performance Advantage Across Datasets

Dataset	A2 (Pairwise)	A1 (SingleTask)	Improvement	Significance
NYU V2	0.6929	0.7934	+14.5%	Moderate (single run)
Pascal	0.9826	1.1132	+13.3%	High ($p < 0.001$, 3 runs)

Pairwise achieves 13-14.5% improvements over single-task across both datasets, with statistical significance confirmed through Pascal Context's three-run validation ($p < 0.001$, Cohen's $d = 6.81$). The consistent advantage across different correlation structures (strong for NYU V2, weak for Pascal) demonstrates that multi-task learning with two related tasks provides robust benefits. The improvement mechanism differs by dataset: NYU V2's pairwise advantage stems from genuine task complement (depth and segmentation share geometric structure), while Pascal Context's advantage arises from regularization (training on edge detection and segmentation simultaneously prevents overfitting to either task's idiosyncrasies). Nonetheless, the consistent 13-14% improvement validates pairwise as a general-purpose task assignment strategy superior to pure specialization.

Early stopping achieved near-perfect effectiveness across all datasets with 100% trigger success rate (no failed triggers, no premature terminations). CIFAR-10 experiments triggered around round 60 out of 100 configured, NYU V2 experiments executed an average of 26 rounds out of 50, and Pascal Context experiments executed an average of 16.3 rounds out of 50. The `restore_best_weights: true` configuration ensures that the gap between best performance round and stopping trigger round does not degrade model quality. The consistent effectiveness validates convergence-based stopping as a reliable optimization for decentralized FMTL, particularly valuable for dense prediction tasks where slow convergence necessitates generous maximum round budgets.

Task overlap prior ($\alpha = 0.5$) consistently outperforms pure gradient similarity ($\alpha = 0.0$) by 4.7% on NYU V2 (A2: 0.6929 versus B1: 0.7256), demonstrating that incorporating static task structure complements dynamic gradient information. This pattern holds across Pascal Context, where A2 ($\alpha = 0.5$: 0.9826) outperforms B4 ($\alpha = 0.0$: 0.9940), although the Pascal comparison is less direct due to differing task assignments (Pairwise versus MultiTask). The consistent performance motivates the use of mixed similarity ($\alpha = 0.5$) as the default configuration, effectively combining both information sources for robust aggregation decisions.

6.5.3 Dataset-Specific Insights

NYU V2's strong geometric correlation enables backbone-only aggregation (B1: 0.7256) to achieve competitive performance ranking 2nd out of 8 experiments, demonstrating that selective parameter sharing succeeds when tasks share structure. The shared backbone learns geometric features (edges, boundaries, surface orientations) beneficial to all three tasks (depth, normals, segmentation), while task-specific heads specialize on output formats (scalar depth, 3D normal vectors, 13-class segmentation). This separation prevents interference in task-specific computation while maximizing backbone collaboration. However, full aggregation (A2: 0.6929) still outperforms backbone-only by 4.7%, suggesting that even for strongly correlated tasks, aggregating task heads provides additional benefit through direct transfer of task-specific output transformations.

Pascal Context's weak correlation causes backbone-only aggregation (B1) to fail catastrophically with chaotic oscillation (NaN at round 7), while full aggregation (A2, B4) succeeds with stable convergence. The failure demonstrates that weak task correlation produces conflicting gradients (edge detection wants low-level filters, segmentation wants high-level semantic features) that cannot be reconciled through backbone-only aggregation. Full aggregation distributes conflicts across both backbone and task heads, preventing destabilizing oscillation. This counterintuitive finding—that more parameter sharing improves stability for less correlated tasks—represents a novel contribution challenging conventional negative transfer theories that recommend selective sharing for heterogeneous tasks. The mechanism operates through conflict distribution rather than conflict avoidance: full aggregation accepts conflicts but distributes them across more parameters (14 million versus 11 million), reducing per-parameter update magnitudes below the instability threshold.

CIFAR-10's moderate heterogeneity allows both pure methods (FedAvg: 0.71, pure HCA after protection: 0.69) and hybrid methods (Hybrid-6: 0.65) to succeed, providing flexibility in aggregation strategy selection. Hybrid strategies achieve best performance by combining FedAvg's robustness during early training with HCA's optimization power once gradients stabilize. The flexibility suggests that for moderate heterogeneity where tasks are neither strongly aligned nor strongly conflicting, practitioners can select aggregation strategies based on implementation complexity preferences (pure FedAvg for simplicity, Hybrid for optimal performance, pure HCA after implementing protection mechanisms for theoretical interest) without dramatic performance degradation.

6.6 Key Findings and Research Contributions

This section synthesizes the most significant findings from over 60 experiments across three datasets, organizing contributions by theoretical significance and practical impact.

6.6.1 Core Empirical Contributions

The thesis's most significant empirical finding is the relationship between task correlation strength and optimal aggregation scope, summarized as:

Strong Task Correlation
(NYU V2: depth + normal + segmentation sharing geometric priors)

- → Gradients naturally aligned (positive cosine similarity: 0.6–0.8)
- → Backbone-only aggregation remains viable (B1: 0.7256, competitive performance)
- → Full aggregation yields superior results (A2: 0.6929, 4.7% improvement)

Weak Task Correlation
(Pascal Context: edge + parts + segmentation; distinct levels of abstraction)

- → Gradients exhibit conflict (low or negative cosine similarity)
- → Backbone-only aggregation fails catastrophically (B1: NaN oscillation)
- → Full aggregation is essential for stability (A2: 0.9826, stable convergence)

This relationship challenges conventional wisdom that more selective sharing (backbone-only) always reduces negative transfer. The empirical evidence demonstrates that for weakly correlated tasks, full aggregation paradoxically improves stability by distributing conflicting gradients across both backbone and task heads, preventing the concentrated conflicts in the shared backbone that cause chaotic oscillation under backbone-only aggregation. The finding has immediate practical implications: practitioners should assess task correlation before selecting aggregation scope, using full aggregation as the safe default for unknown or weak correlation, and considering backbone-only optimization only for strongly correlated tasks where empirical validation confirms stable convergence.

The HCA numerical stability investigation and 5-layer protection mechanism constitute the second major contribution. The systematic exploration spanning 20+ experiments documented failure modes, tested six solution strategies (all failed), strategically pivoted to FedAvg validation isolating HCA as the singular bottleneck, discovered hybrid strategies demonstrating mid-training HCA viability, and ultimately developed the 5-layer protection mechanism enabling stable pure HCA from round 1. The protection mechanism achieved dramatic success recovering B2 from catastrophic explosion (Final Loss 24,807,255) to stable convergence (Best Loss 0.8101), representing 99.999997% loss reduction. However, the investigation also revealed a surprising negative finding: HCA underperforms simple weighted averaging on NYU V2's strongly correlated tasks (B2/B3 rank 7th-8th versus B1 rank 2nd), demonstrating that conflict-averse optimization provides benefits only when conflicts are prevalent. The comprehensive documentation of both successes (protection mechanism) and failures (six failed solution attempts, poor final performance) provides methodological value guiding future research encountering similar numerical instability challenges.

The validation of decentralized FMTL for dense prediction represents the first application of federated multi-task learning to pixel-wise regression and segmentation tasks. Prior decentralized federated learning work focused exclusively on classification [5], [9], [51], leaving open whether approaches generalize to complex output structures (depth maps, normal maps, segmentation masks) with heterogeneous loss functions (L1 regression, cross-entropy classification). This thesis demonstrates successful generalization through experiments on NYU V2 (three dense prediction tasks) and Pascal Context (three multi-scale tasks), achieving competitive performance (A2: 0.6929 on NYU V2, 0.9826 on Pascal) validating that decentralized coordination suffices for dense prediction without requiring centralized servers.

Pairwise task assignment superiority across datasets constitutes the fourth contribution. Pairwise consistently outperforms both single-task specialization (14.5% improvement on NYU V2, 13.3% on Pascal with high statistical significance $p < 0.001$) and uniform multi-task distribution (8.4% improvement over B4 on NYU V2). The 0.7/0.3 asymmetric weight split optimally balances primary task focus with auxiliary task regularization, achieving better generalization than pure specialization (A1) while avoiding the gradient conflicts of uniform distribution (B4). The consistent advantage across strong correlation (NYU V2) and weak correlation (Pascal Context) validates pairwise as a robust general-purpose strategy applicable without task-specific tuning.

Task overlap prior superiority over pure gradient similarity represents the final core contribution. Contrary to the initial hypothesis that dynamic gradient information would supersede static task metadata, incorporating task overlap prior ($\alpha = 0.5$) outperforms pure gradient similarity ($\alpha = 0.0$) by 4.7% on NYU V2 (A2: 0.6929 versus B1: 0.7256). Task overlap captures structural relationships (which tasks share output spaces enabling knowledge transfer) that gradients alone miss, particularly during early training when gradient directions remain noisy. The finding motivates mixed similarity ($\alpha = 0.5$) as the default configuration combining complementary information sources.

6.6.2 Practical Implications and Recommendations

The experimental findings translate to five actionable recommendations for deploying decentralized FMTL systems. First, practitioners should assess task correlation before selecting aggregation scope. For tasks sharing structure (geometric tasks like depth/normals/segmentation, linguistic tasks like translation/summarization/question-answering), backbone-only aggregation provides computational efficiency (fewer parameters exchanged, faster convergence) with competitive performance. For independent tasks (edge detection + segmentation, image classification + text generation), full aggregation provides essential stability preventing catastrophic oscillation. When correlation is uncertain, full aggregation serves as the safe default, as it succeeds across both strong and weak correlation while backbone-only fails catastrophically for weak correlation.

Second, prefer pairwise task assignment unless tasks are nearly identical (requiring single-task specialization) or completely unrelated (requiring isolation). Assign two related tasks per client with asymmetric weights (0.7 primary, 0.3 secondary) balancing specialization and knowledge transfer. The 13-14% improvement over single-task across datasets validates pairwise as superior to pure specialization, while 8.4% improvement over uniform multi-task demonstrates advantages over forcing all clients to train all tasks simultaneously.

Third, use mixed task overlap similarity ($\alpha = 0.5$) combining gradient-based and task structure information rather than pure gradient similarity ($\alpha = 0.0$). The 4.7% improvement on NYU V2 demonstrates that static task metadata complements dynamic gradient information, particularly during early training when gradient directions remain unstable. Implementation requires minimal overhead: computing task overlap similarity once at initialization and caching the result, then interpolating with per-round gradient similarity using α parameter.

Fourth, implement early stopping with generous patience (8-10 rounds) and restore best weights configuration (`restore_best_weights: true`). The consistent trigger success across datasets validates convergence-based stopping as reliable and efficient. The best weight restoration ensures that the gap between optimal performance and stopping trigger does not degrade model quality, as deployed models use parameters from the best validation round rather than final round.

Fifth, apply HCA carefully with full numerical protection or avoid it entirely in favor of simpler weighted averaging. HCA achieves best performance only on CIFAR-10 (Hybrid-6: 0.65) through hybrid strategies, while underperforming weighted averaging on NYU V2 (B2/B3 rank 7th-8th versus B1 rank 2nd). The 5-layer protection mechanism enables stable HCA but at substantial implementation complexity. For production systems, consider simpler weighted averaging (easier implementation, lower computational cost, comparable or better performance for correlated tasks) unless empirical validation demonstrates HCA advantages for specific task combinations. If deploying HCA, use hybrid strategies (FedAvg for early rounds 1-6, HCA thereafter) providing robustness against early-training instability while capturing HCA's optimization benefits once gradients stabilize.

6.6.3 Limitations and Future Work

This evaluation is conducted under controlled settings and thus has several limitations closely tied to the reported comparisons. First, experiments use only six clients, which supports reproducibility but leaves open whether the same neighbor-selection and consensus dynamics hold at realistic federated scales (hundreds+ clients), where similarity computation and communication become more complex. Second, communication is simulated within a single process, abstracting away practical distributed factors such as latency, bandwidth limits, partial participation, and client churn that may affect convergence and stability in real deployments. Third, task heterogeneity is generated via predefined assignments (SingleTask/Pairwise/MultiTask); real systems may exhibit more irregular and evolving task mixtures, requiring more adaptive strategies. Finally, conclusions are conditioned on a finite hyperparameter/seed budget (e.g., early-stopping patience, similarity window, neighbor size), so future work should validate robustness via broader sensitivity studies and larger-scale simulations or distributed runs.

Chapter 7

Conclusion

This thesis investigates decentralized federated multi-task learning for dense prediction computer vision, aiming to remove central coordination while still enabling effective cross-client collaboration under heterogeneous objectives. The early chapters establish the problem setting and motivate why server-free training is desirable in federated systems where coordination, trust, and single-point-of-failure issues become limiting. Building on prior federated multi-task learning formulations, the thesis then develops a peer-to-peer framework in which each client independently selects neighbors and aggregation scope, enabling collaboration without a central server. The implementation is designed to be modular and reproducible, with task assignment and aggregation strategies specified declaratively to support systematic comparison across configurations.

Empirically, the evaluation across multiple datasets and task mixtures yields several consistent conclusions. First, decentralized collaboration is feasible: despite the lack of global coordination, autonomous neighbor selection and aggregation can still produce stable training and measurable gains over isolated single-task baselines in the explored settings. Second, the effectiveness of collaboration is strongly shaped by task relationships, and there is no universally optimal sharing scope. When tasks are strongly correlated, selectively aggregating shared components (e.g., backbone-only) can be beneficial; however, when correlation is weak, full-model aggregation is often necessary to maintain numerical stability and avoid divergence. This observation highlights that selective sharing is not inherently safer than full sharing, and that aggregation scope should be treated as a task-dependent design choice. Third, controlled overlap in objectives improves learning: pairwise task assignment (a clear primary task with a smaller secondary task weight) consistently outperforms purely single-task or overly broad multi-task mixtures, suggesting that limited, structured heterogeneity offers a better balance between specialization and transfer. Fourth, for decentralized neighbor selection, gradient-only similarity emerges as a strong and practical signal: it avoids dependence on task metadata while remaining effective for discovering which peers provide useful updates. Finally, the work demonstrates that numerical stability is a first-class concern in decentralized multi-task settings. Conflict-averse aggregation can produce catastrophic failures without safeguards, and systematic stabilization is required to make such methods usable in practice; documenting both stable configurations and failure cases is therefore part of the contribution.

Despite these findings, several limitations remain and motivate future work. The current study emphasizes controlled validation and therefore does not fully model real-world deployment constraints, most notably communication cost and system-level effects. In practical federated networks, latency, bandwidth limitations, message drops, partial participation, and client churn can dominate runtime and may change which collaboration patterns are optimal; future work should explicitly measure communication overhead, study cost–accuracy trade-offs, and validate the framework under realistic distributed conditions. In addition, experiments are conducted under a limited set of hyperparameters and random seeds; broader sensitivity analysis is needed to better characterize stability boundaries and to provide robust default settings. Finally, the current design largely assumes parameter-compatible architectures for straightforward aggregation; extending the framework to heterogeneous model families (e.g., via representation-level sharing, adapters, or distillation) would substantially broaden applicability. Addressing these directions would strengthen the path from controlled experimental evidence toward production-grade decentralized multi-task federated learning systems.

Bibliography

[1] B. McMahan et al., “Communication-efficient learning of deep networks from decentralized data”, in *AISTATS*, 2017.

[2] L. Zhu, Z. Liu, and S. Han, *Deep leakage from gradients*, 2019. arXiv: `1906.08935 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1906.08935`.

[3] A. Lalitha et al., “Fully decentralized federated learning”, *IEEE Transactions on Signal Processing*, 2018.

[4] R. Sun et al., “Decentralized federated averaging”, *IEEE Transactions on Signal Processing*, 2022.

[5] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, *Federated multi-task learning*, 2018. arXiv: `1705.10467 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1705.10467`.

[6] F. Sattler, K.-R. Müller, and W. Samek, “Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints”, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3710–3722, 2021. DOI: `10.1109/TNNLS.2020.3015958`.

[7] N. Silberman et al., “Indoor segmentation and support inference from rgbd images”, in *ECCV*, 2012.

[8] R. Mottaghi et al., “The role of context for object detection and semantic segmentation in the wild”, in *CVPR*, 2014.

[9] N. Kohler, “A solution for decentralized federated multi-task learning”, Master’s Thesis, University of Zurich, 2024.

[10] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu, *Conflict-averse gradient descent for multi-task learning*, 2024. arXiv: `2110.14048 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2110.14048`.

[11] J. Konečný et al., “Federated learning: Strategies for improving communication efficiency”, *arXiv preprint arXiv:1610.05492*, 2016.

[12] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, *Federated optimization in heterogeneous networks*, 2020. arXiv: `1812.06127 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1812.06127`.

[13] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, *Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent*, 2017. arXiv: `1705.09056 [math.OC]`. [Online]. Available: `https://arxiv.org/abs/1705.09056`.

[14] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, *Decentralized deep learning with arbitrary communication compression*, 2020. arXiv: `1907.09356 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1907.09356`.

[15] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, *Stochastic gradient push for distributed deep learning*, 2019. arXiv: `1811.10792 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1811.10792`.

[16] S. Ruder, "An overview of multi-task learning in deep neural networks", *arXiv preprint arXiv:1706.05098*, 2017.

[17] M. Crawshaw, "Multi-task learning with deep neural networks: A survey", *arXiv preprint arXiv:2009.09796*, 2020.

[18] R. Caruana, "Multitask learning", *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997. DOI: `10.1023/A:1007379606734`.

[19] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich, *To transfer or not to transfer*, NIPS 2005 Workshop: Inductive Transfer: 10 Years Later (December 2005), 2005. [Online]. Available: `https://web.engr.oregonstate.edu/~tgd/publications/rosenstein-marx-kaelbling-dietterich-hnb-nips2005-transfer-workshop.pdf`.

[20] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, *Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks*, 2018. arXiv: `1711.02257 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1711.02257`.

[21] A. Kendall, Y. Gal, and R. Cipolla, *Multi-task learning using uncertainty to weigh losses for scene geometry and semantics*, 2018. arXiv: `1705.07115 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1705.07115`.

[22] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, *Gradient surgery for multi-task learning*, 2020. arXiv: `2001.06782 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2001.06782`.

[23] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, *Cross-stitch networks for multi-task learning*, 2016. arXiv: `1604.03539 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1604.03539`.

[24] S. Liu, E. Johns, and A. J. Davison, *End-to-end multi-task learning with attention*, 2019. arXiv: `1803.10704 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1803.10704`.

[25] A. Krizhevsky, "Learning multiple layers of features from tiny images", University of Toronto, Tech. Rep., Apr. 2009. [Online]. Available: `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[26] C. Feng et al., *Colnet: Collaborative optimization in decentralized federated multi-task learning systems*, 2025. arXiv: `2501.10347 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2501.10347`.

[27] A. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese, *Taskonomy: Disentangling task transfer learning*, 2018. arXiv: `1804.08328 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1804.08328`.

[28] K.-K. Maninis, I. Radosavovic, and I. Kokkinos, "Attentive single-tasking of multiple tasks", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[29] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, *Federated learning with personalization layers*, 2019. arXiv: `1912.00818 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1912.00818`.

[30] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, *Exploiting shared representations for personalized federated learning*, 2023. arXiv: `2102.07078 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2102.07078`.

[31] C. T. Dinh, N. H. Tran, and T. D. Nguyen, *Personalized federated learning with moreau envelopes*, 2022. arXiv: `2006.08848 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2006.08848`.

[32] T. Li, S. Hu, A. Beirami, and V. Smith, *Ditto: Fair and robust federated learning through personalization*, 2021. arXiv: `2012.04221 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2012.04221`.

[33] Y. Lu, S. Huang, et al., "Fedhca2: Towards hetero-client federated multi-task learning", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[34] A. Elbakary, C. Ben Issaid, and M. Bennis, *Learning to collaborate over graphs: A selective federated multi-task learning approach*, Preprint, 2025.

[35] Y. Wei, Y. Zou, et al., *Towards unified modeling in federated multi-task learning via subspace decoupling*, Preprint, 2025.

[36] Y. Zhang, H. Chen, et al., *Fedac: An adaptive clustered federated learning framework for heterogeneous data*, arXiv preprint, 2024.

[37] A. Zamir et al., *Robust learning through cross-task consistency*, 2020. arXiv: `2006.04096 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2006.04096`.

[38] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, *An efficient framework for clustered federated learning*, 2021. arXiv: `2006.04088 [stat.ML]`. [Online]. Available: `https://arxiv.org/abs/2006.04088`.

[39] H. Kim, H. Kim, and G. de Veciana, "Clustered federated learning via gradient-based partitioning", in *International Conference on Machine Learning (ICML)*, 2024.

[40] Q. Ma, Y. Xu, et al., "Feduc: A unified clustering approach for hierarchical federated learning", *IEEE Transactions on Mobile Computing*, 2024.

[41] G. Luo, N. Chen, et al., "Privacy-preserving clustering federated learning for non-iid data", *Future Generation Computer Systems*, 2024.

[42] Y. Ruan and C. Joe-Wong, *Fedsoft: Soft clustered federated learning with proximal local updating*, 2022. arXiv: `2112.06053 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2112.06053`.

[43] A. Ali and A. Arafa, "Data similarity-based one-shot clustering for multi-task hierarchical federated learning", *IEEE Transactions on Network Science and Engineering*, 2024.

[44] J. Shu et al., “Clustered federated multitask learning on non-iid data with enhanced privacy”, *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[45] T. Kim et al., “Dynamic clustering in federated learning”, in *European Conference on Computer Vision Workshops*, 2020.

[46] S. Reddi et al., *Adaptive federated optimization*, 2021. arXiv: `2003.00295 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2003.00295`.

[47] H. Yuan et al., “Decentralized federated learning: A survey and perspective”, *IEEE Transactions on Artificial Intelligence*, 2024.

[48] D. Martínez Beltrán et al., “Decentralized federated learning: Fundamentals, state-of-the-art, and open challenges”, *IEEE Communications Surveys & Tutorials*, 2023.

[49] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: `1512.03385 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1512.03385`.

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database”, in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[51] J. Pilet et al., “Simple, efficient and convenient decentralized multi-task learning”, in *NeurIPS*, 2020.

Abbreviations

FL	Federated Learning
DFL	Decentralized Federated Learning
FMTL	Federated Multi-Task Learning
DFMTL	Decentralized Federated Multi-Task Learning
P2P	Peer-to-Peer
HCA	Hyper Conflict-Averse (aggregation)
MTL	Multi-Task Learning
CV	Computer Vision
IID	Independent and Identically Distributed
non-IID	Non-Independent and Identically Distributed
SGD	Stochastic Gradient Descent
CNN	Convolutional Neural Network
CIFAR	Canadian Institute For Advanced Research
mIoU	Mean Intersection over Union
NaN	Not a Number
PASCAL	Pattern Analysis, Statistical Modelling and Computational Learning
ResNet	Residual Network
RGB	Red Green Blue
RGB-D	Red Green Blue-Depth
YAML	YAML Ain't Markup Language

Glossary

Federated Learning (FL) A distributed training paradigm where clients train models on local data and share updates (e.g., parameters or gradients) rather than raw data.

Decentralized Federated Learning (DFL) A federated learning setting without a central server, where clients exchange updates in a peer-to-peer manner and coordination is achieved through decentralized protocols.

Federated Multi-Task Learning (FMTL) A federated learning formulation in which different clients optimize different (potentially related) tasks, aiming to enable beneficial knowledge transfer while handling task heterogeneity.

Decentralized Federated Multi-Task Learning (DFMTL) The combination of FMTL and DFL: multi-task federated learning performed without central coordination, requiring clients to make autonomous collaboration and aggregation decisions.

Client A participating node/device in a federated network that performs local training on private data and communicates model updates to other participants.

Peer-to-Peer (P2P) Communication A communication pattern in which clients exchange updates directly with selected neighbors rather than through a central server.

Neighbor Selection The decentralized procedure by which a client chooses a subset of other clients (neighbors) from whom to receive updates for aggregation.

Aggregation The process of combining local and received updates into a new model state. In this thesis, aggregation is performed autonomously at each client and may vary across clients.

Aggregation Scope The subset of model parameters that are shared and aggregated (e.g., backbone-only versus full model), which affects both transfer benefits and stability.

Backbone The shared feature extractor of a neural network (e.g., ResNet-18), typically task-agnostic and reused across multiple tasks.

Task Head The task-specific prediction module attached to a shared backbone (e.g., segmentation head, depth head), producing outputs tailored to one objective.

Dense Prediction Vision tasks that require structured, per-pixel outputs (e.g., semantic segmentation, depth estimation, edge detection), often exhibiting different optimization dynamics from image-level classification.

Task Correlation The degree to which tasks share aligned optimization objectives or transferable representations; empirically linked in this thesis to the preferred aggregation scope.

Task Similarity A quantitative estimate of relatedness between tasks/clients used to guide collaboration. This thesis studies gradient-based cosine similarity, task overlap similarity, and cross-loss similarity.

Gradient Cosine Similarity A similarity measure computed from gradients (or gradient-like signals), indicating alignment of optimization directions and enabling metadata-free collaboration decisions.

Task Overlap Similarity A similarity signal derived from overlap in task assignments or task-weight vectors, capturing whether clients train on the same or partially overlapping objectives.

Cross-Loss Similarity A similarity metric based on evaluating a neighbor's model on local validation data, directly estimating transferability across clients/tasks via observed loss behavior.

Conflict-Aware Aggregation Aggregation methods that account for gradient conflicts across tasks/clients to reduce negative transfer, potentially improving multi-task learning but also introducing stability challenges.

Hyper Conflict-Averse (HCA) Aggregation A specific conflict-aware aggregation approach investigated in this thesis, requiring dedicated stabilization mechanisms to prevent numerical divergence in decentralized dense prediction settings.

Early Stopping A training efficiency technique that terminates training when validation performance plateaus, typically using a patience window and restoring the best checkpoint.

Pairwise Task Assignment A task distribution pattern where each client trains two tasks with asymmetric weights (e.g., a primary task plus a related secondary task), used in this thesis to study controlled overlap versus single-task or multi-task mixtures.

List of Figures

List of Tables

Appendix A

Supplementary Figures for Chapter 6

A.1 NYU V2 Supplementary Analyses

A.2 Cross-Dataset Supplementary Summary

A.3 Representative Learning Curves

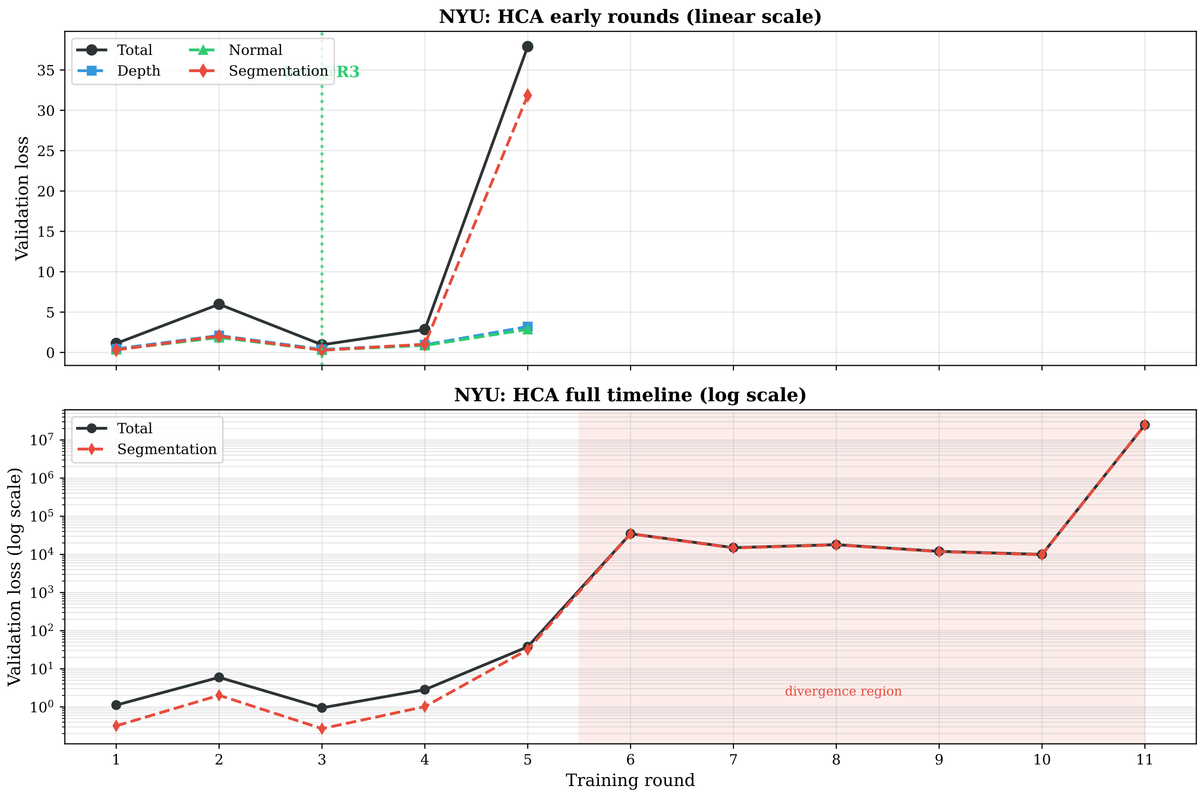


Figure A.1: NYU V2 HCA divergence timeline. The early rounds appear normal, followed by abrupt loss explosion, indicating numerical instability rather than gradual overfitting.

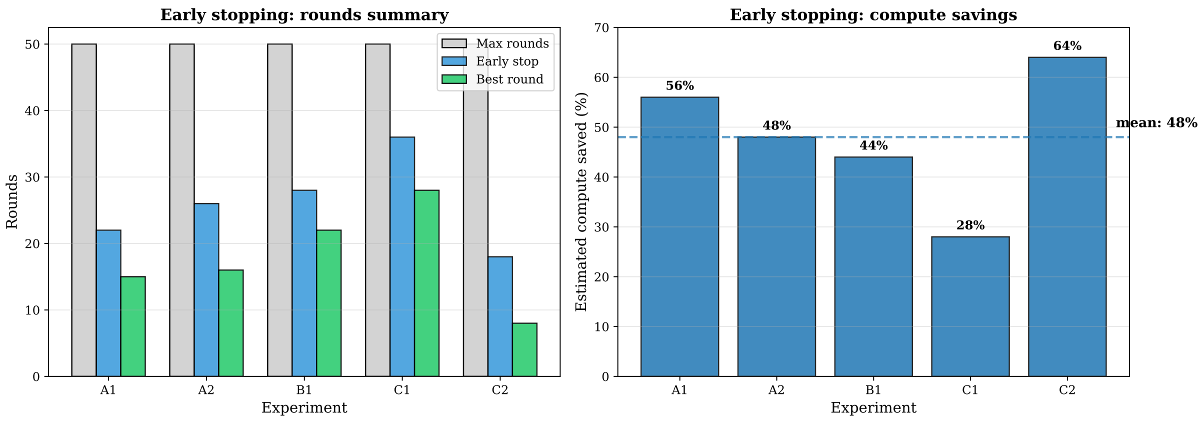


Figure A.2: Early stopping analysis on NYU V2. Left: configured maximum rounds, actual early-stopped rounds, and best-achieved rounds. Right: estimated compute saved by early stopping across configurations.

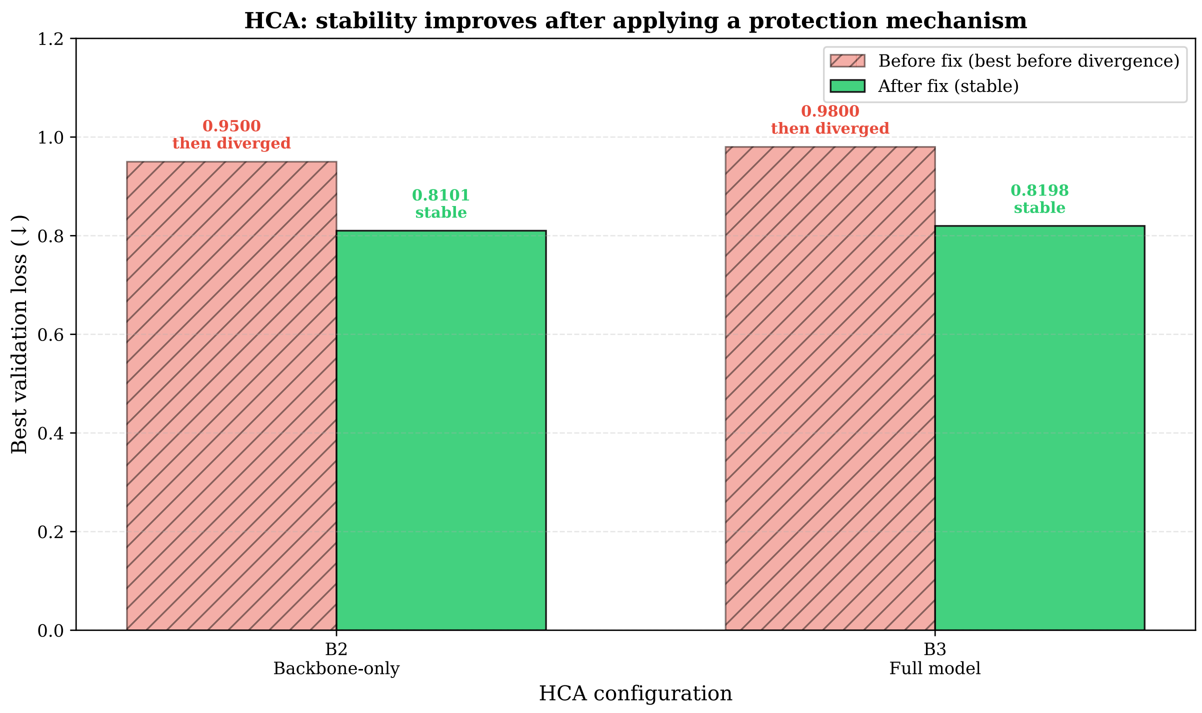


Figure A.3: HCA stability fix on NYU V2. After applying a protection mechanism, both backbone-only and full-model HCA become stable, though they remain less competitive than A2.

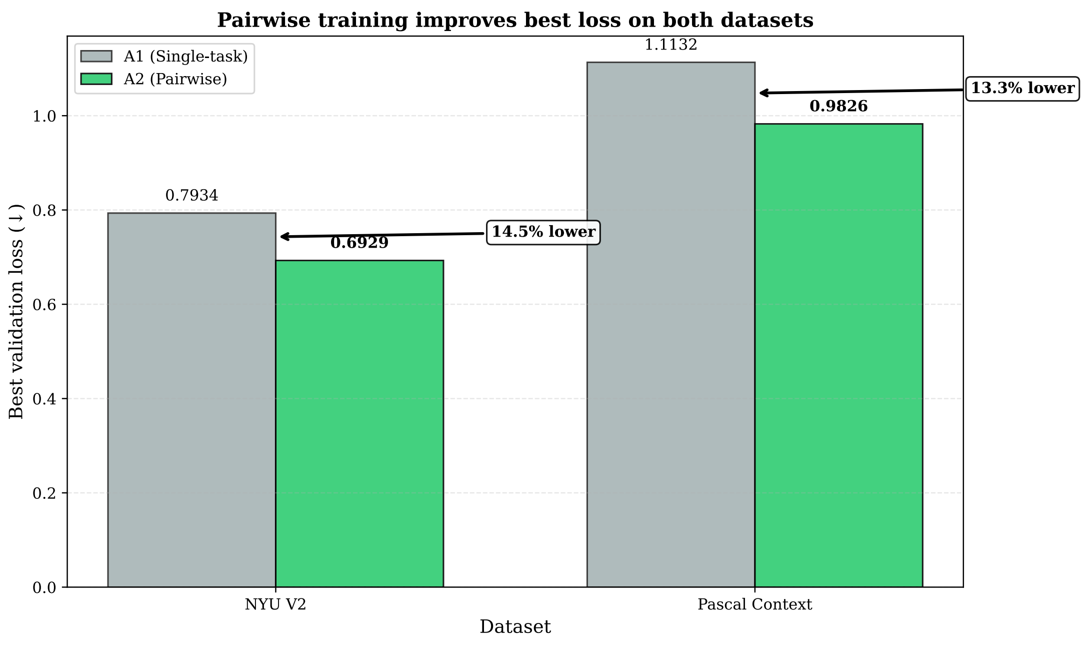


Figure A.4: Pairwise training (A2) consistently improves best validation loss over single-task training (A1) on both NYU V2 and Pascal Context.

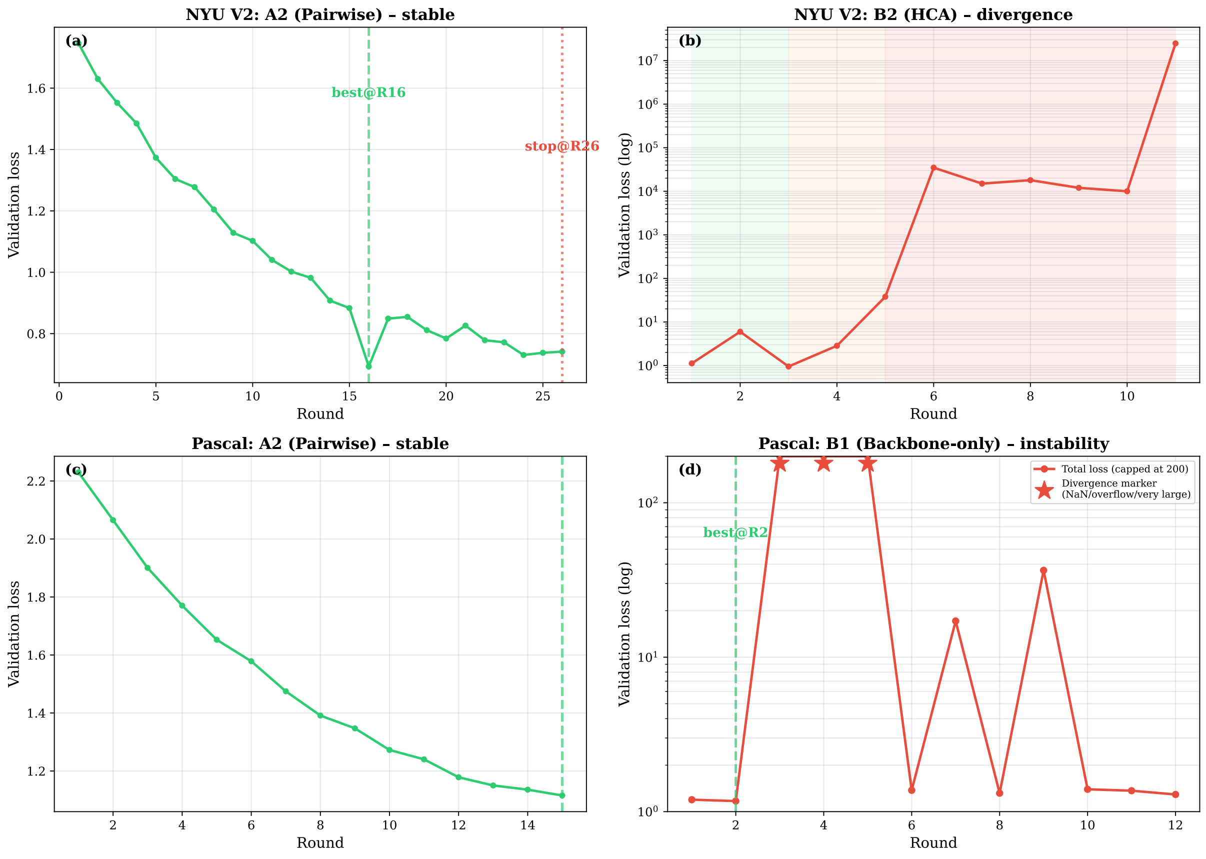


Figure A.5: Representative learning curves across datasets and configurations (supplementary). Stable convergence contrasts with divergence/instability patterns observed in HCA (NYU) and B1 (Pascal).