



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Safety-Aware Aggregation for Federated Fine-Tuning of Large Language Models

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE AND ENGINEERING - INGEGNERIA INFORMATICA

Author: **Leonardo Lei**

Student ID: 252566

Advisor: Prof. Cinzia Cappiello

Co-advisors: Mattia Sabella

Academic Year: 2025-26

Abstract

The rapid democratization of Large Language Models (LLMs) has amplified a dual-use dilemma: the same capabilities that let these models advance legitimate science also let them generate unsafe content in domains such as biosecurity, chemistry, and cybersecurity. At the same time, the demand for privacy-preserving training has driven the adoption of Federated Learning (FL), in which a central server aggregates client updates but never sees the raw local data. This privacy guarantee disables the dominant safety paradigm: server-side inspection and filtering of training data becomes impossible, so a client fine-tuning on hazardous data can silently poison the global model. Our key observation is that, although the server cannot select data, it can still act on a standardized, trusted data risk signal computed locally and shared without ever exposing raw samples. The goal of this work is therefore to relocate safety enforcement from the data space into the aggregation space, using this risk signal to properly weight the incoming updates. Building on a LoRA-based Parameter-Efficient Fine-Tuning (PEFT) approach, we evaluate a series of aggregation strategies, including a novel technique, Risk-Aware Projection for Federated Learning (RAP-FL). Across extreme and realistic data-heterogeneity scenarios, we quantify the trade-off between suppressing hazardous knowledge (WMDP) and preserving general-domain utility (MMLU), and show that RAP-FL achieves the strongest safety-utility balance.

Keywords: Federated Learning, AI Safety, Large Language Models, PEFT, Trustworthy AI.

Abstract in lingua italiana

La rapida democratizzazione dei Large Language Models (LLM) ha amplificato un dilemma dual-use: le stesse capacità che permettono a questi modelli di favorire la ricerca scientifica legittima permettono loro anche di generare contenuti non sicuri in ambiti come la biosicurezza, la chimica e la cybersecurity. Allo stesso tempo, la richiesta di un addestramento che preservi la privacy ha spinto l'adozione del Federated Learning (FL), in cui un server centrale aggrega gli aggiornamenti dei client senza mai osservare i dati locali grezzi. Questa garanzia di privacy disabilita il paradigma di sicurezza dominante: l'ispezione e il filtraggio dei dati di addestramento lato server diventano impossibili, per cui un client che esegue il fine-tuning su dati pericolosi può avvelenare silenziosamente il modello globale. La nostra osservazione chiave è che, sebbene il server non possa selezionare i dati, può comunque agire sulla base di un segnale di rischio standardizzato e affidabile, calcolato localmente e condiviso senza mai esporre i campioni grezzi. L'obiettivo di questo lavoro è quindi spostare l'applicazione della sicurezza dallo spazio dei dati allo spazio dell'aggregazione, utilizzando questo segnale di rischio per pesare opportunamente gli aggiornamenti in arrivo. Sfruttando un approccio di Parameter-Efficient Fine-Tuning (PEFT) basato su LoRA, valutiamo uno spettro di strategie di aggregazione, tra cui una nuova tecnica: Risk-Aware Projection for Federated Learning (RAP-FL). In scenari di eterogeneità dei dati sia estremi che realistici, quantifichiamo il trade-off tra la soppressione della conoscenza pericolosa (WMDP) e la preservazione dell'utilità generale (MMLU), mostrando che RAP-FL raggiunge il miglior equilibrio di sicurezza-utilità.

Parole chiave: Apprendimento Federato, Sicurezza dell'Intelligenza Artificiale, Modelli Linguistici di Grandi Dimensioni, PEFT, Intelligenza Artificiale Affidabile.

Contents

Abstract	i
Abstract in lingua italiana	iii
Contents	v
1 Introduction	1
1.1 Thesis Structure	1
2 State of the Art	3
2.1 Foundations of Pre-trained Language Models	3
2.1.1 Encoder-Only Models: Text Representation and Classification . . .	4
2.1.2 Decoder-Only Models: Autoregressive Generation	5
2.1.3 The Open-Weight Paradigm and Safety Implications	6
2.2 Parameter-Efficient Fine-Tuning (PEFT)	7
2.2.1 Low-Rank Adaptation (LoRA)	7
2.2.2 Model Quantization and QLoRA	8
2.3 AI Safety in Large Language Models	9
2.3.1 The "Dual-Use" Dilemma	10
2.3.2 Existing Mitigation Strategies	11
2.3.3 Proactive Safety: The Deep Ignorance Framework	12
2.4 Federated Learning (FL)	13
2.4.1 Federated Averaging (FedAvg)	14
2.4.2 Federated PEFT (FedPEFT) and FedLoRA	15
2.5 Research Gap: The Decentralized Safety Challenge	16
2.6 Research Question	17
3 Methodology	19
3.1 Module 1: Centralized Baseline Verification	20

3.2	Module 2: Federated Transition	21
3.2.1	Method 0: Standard Federated Averaging (Baseline)	24
3.2.2	Method 1: Federated Averaging with Safety Filtering	25
3.3	Module 3: Safety-Aware Aggregation Test	26
3.3.1	Method 2: Dynamic Weighted Aggregation	27
3.3.2	Method 3: Federated Task Arithmetic	28
3.3.3	Method 4: Risk-Aware Projection for Federated Learning (RAP-FL)	33
3.4	Phase 4: Evaluation	38
3.4.1	Metrics	40
4	Experimental Setup	43
4.1	Technology Used	43
4.1.1	Programming Language and Deep Learning Frameworks	43
4.1.2	LLM and Fine-Tuning Libraries	43
4.1.3	Data Processing and Evaluation Tools	44
4.1.4	Pre-Trained Models	44
4.2	Dataset and Benchmark Selection	45
4.2.1	Training Datasets	45
4.2.2	Benchmark Datasets	48
4.3	Centralized Verification Baseline Configurations	48
4.3.1	Architectural and Optimization Hyperparameters	49
4.3.2	Centralized Experimental Paradigms	51
4.4	Federated Learning Simulation Setup	52
4.4.1	Network Architecture and Client Partitioning	52
4.4.2	Local Optimization and Global Synchronization Loops	53
4.4.3	Federated Experimental Scenarios	53
4.4.4	Algorithmic Implementations of Defensive Strategies	55
5	Results and Discussion	65
5.1	Centralized Baselines Verification (RQ1)	65
5.1.1	Analysis of Knowledge Absorption and Variance Loss	66
5.2	Federated Replication of Baseline Paradigms (RQ2)	68
5.2.1	Analysis of Decentralized Knowledge Acquisition	69
5.2.2	Conclusion on Replicability	71
5.3	Comparative Evaluation of Safety-Aware Strategies (RQ3)	71
5.4	General Knowledge and Catastrophic Forgetting	75
6	Conclusion	79

6.1	Summary of Contributions and Key Findings	79
6.2	Limitations	80
6.3	Future Work	81
 Bibliography		 83
 A Appendix A		 87
A.1	Detailed Experimental Results per Seed	87
A.2	Loss and Confidence Metrics for Federated Solutions	91
 List of Figures		 95
 List of Tables		 99
 List of Symbols		 101
 Acknowledgements		 103

1 | Introduction

The rapid evolution of **Large Language Models (LLMs)** has brought unprecedented capabilities in natural language understanding, reasoning, and generation. However, as these massive models continue to grow in parameter count, techniques designed to reduce computational and memory overhead, such as **Parameter-Efficient Fine-Tuning (PEFT)** have become very popular.

Concurrently, these technological advancements introduce a "dual-use" risks. The vast corpora used during training embed not only benign, foundational knowledge but also hazardous information that could assist malicious actors in domains such as bioterrorism or chemterrorism.

Simultaneously, the growing demand for privacy-preserving AI has driven the adoption of **Federated Learning (FL)**. In this paradigm, training is pushed to decentralized edge nodes where raw data cannot be centrally inspected, filtered, or sanitized. In a federated environment, the central server is effectively blind to the local data distributions. Consequently, if a participating client fine-tunes the model on hazardous data, whether maliciously or inadvertently, the resulting local weight updates can easily inject dangerous capabilities into the global model, compromising the safety of the entire network. Ensuring proactive safety in FL, therefore, remains an open and critical architectural challenge.

To this end, this thesis seeks to address this critical intersection between decentralized training and **AI safety**. We introduce and evaluate comprehensive solutions for Safe Federated Fine-Tuning, aiming to transpose proactive safety mechanisms into a privacy-preserving environment. Rather than relying on centralized data inspection, we explore how to equip the federated pipeline with risk-aware aggregation strategies at the server level, as well as novel training strategies that can be integrated within a federated setup.

1.1. Thesis Structure

The structure of this thesis follows a logical progression designed to build from foundational concepts to complex experimental evaluations. After this introduction, the subse-

quent chapters are organized as follows:

- **Chapter 2: State of the Art.** A comprehensive survey of the background and existing literature.
- **Chapter 3: Methodology.** A detailed explanation of the rationale behind our research pipeline, structured into four modules. The chapter details the mathematical and operational logic behind all the methods introduced and the evaluation.
- **Chapter 4: Experimental Setup.** A thorough description of how we implemented the experiments and solutions outlined in the methodology. This chapter covers the selection of technologies, models and dataset used, as well as the configuration setup adopted to make our experiments.
- **Chapter 5: Results and Discussion.** A comprehensive presentation of the experimental outcomes. This chapter analyzes the quantitative results across different federated scenarios, deeply examining the trade-off between the model's general utility and its *safety-robustness* against the revival of malicious capabilities.
- **Chapter 6: Conclusion and Future Work.** A concise overview of our research, highlighting the key findings and its limits, summarizing our main contributions, and outlining potential directions for future investigation in safe federated environments.

2 | State of the Art

This chapter provides the theoretical foundation and reviews the existing literature essential to contextualize the challenges of safe decentralized training. We begin by introducing Large Language Models (LLMs) and examining their inherent "dual-use" risks. Subsequently, we explore state-of-the-art centralized mitigation strategies. Finally, we introduce Parameter-Efficient Fine-Tuning (PEFT) and Federated Learning (FL), highlighting the critical research gap that emerges when attempting to bridge decentralized training with proactive safety.

2.1. Foundations of Pre-trained Language Models

The rapid advancement of artificial intelligence has been largely driven by the development of Large Language Models (LLMs) [34]. Built upon the foundational Transformer architecture [29] (illustrated in Figure 2.1), these models leverage the self-attention mechanism to process context in parallel, overcoming the sequential limitations of previous recurrent neural networks.

The original Transformer architecture is composed of two distinct functional blocks, an **Encoder-Decoder** structure. The **Encoder** maps an input sequence into a rich, contextualized numerical representation, where each token's embedding is informed by the entire surrounding sequence. The **Decoder**, conversely, generates an output sequence one token at a time, using both its own previously generated tokens and the representation produced by the encoder. Figure 2.1 (left) illustrates this Encoder-Decoder structure.

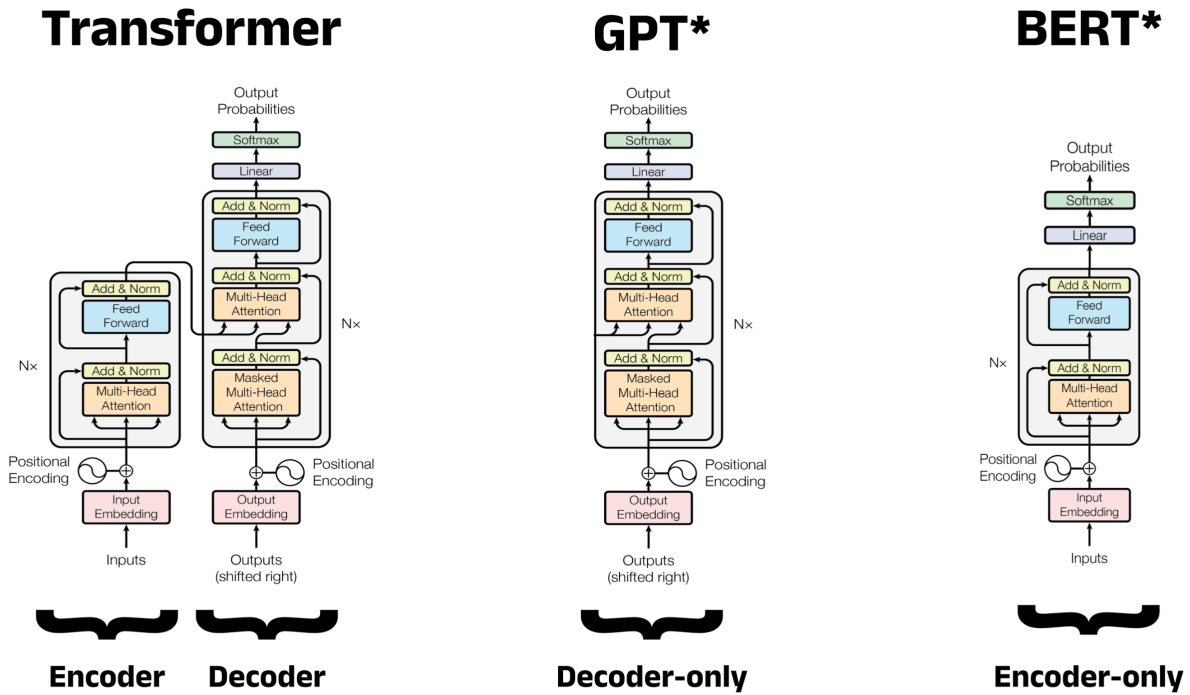
While the original Transformer was designed for machine translation using this dual-block configuration, the modern NLP landscape has predominantly specialized into two distinct architectural branches, each retaining only one of these two blocks and serving different computational objectives within our research:

- **Encoder-Only Models (e.g., BERT, ModernBERT):**

These architectures retain only the encoder component of the original Transformer. [8]

- **Decoder-Only Models (e.g., GPT family, LLaMA, Mistral):**

Conversely, these models rely exclusively on the decoder stack. They represent the standard architecture behind the most famous **generative** and conversational AI systems widely known today, such as ChatGPT. [27] [4]



*Illustrative example, exact model architecture may vary slightly

Figure 2.1: Overview of the architectures of Transformer models, BERT models and GPT models compared to each other. Figure adapted from [24].

2.1.1. Encoder-Only Models: Text Representation and Classification

Models belonging to the BERT (Bidirectional Encoder Representations from Transformers) [8] family utilize only the encoder block of the original architecture. They employ a bidirectional self-attention mechanism, allowing each token to attend to both preceding and succeeding tokens simultaneously. These models are typically pre-trained using a **Masked Language Modeling (MLM)** objective, where the model learns to predict missing words within a sentence based on the full surrounding context.

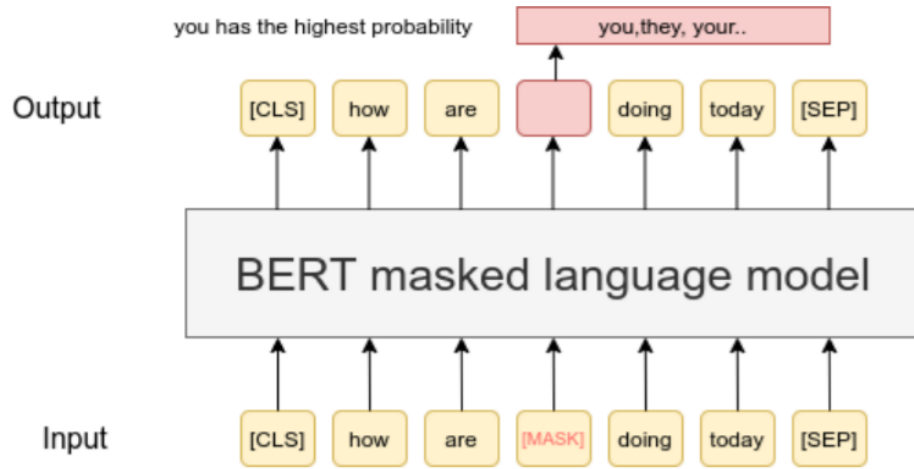


Figure 2.2: Illustration of BERT’s Masked Language Modeling (MLM) pre-training task. The diagram demonstrates how an input sentence is processed: a specific token is masked (e.g., [MASK]), and the model analyzes the full, bidirectional surrounding context to predict the original token with the highest probability. Image sourced from the official SentenceTransformers documentation [22].

Due to their deep contextual understanding, encoder-only models are state-of-the-art for discriminative tasks such as sentiment analysis, named entity recognition, and text classification. In the context of this thesis, an advanced encoder-only model (**ModernBERT**) is employed as a highly accurate classifier to evaluate and label the training datasets, distinguishing between benign utility data and hazardous knowledge.

2.1.2. Decoder-Only Models: Autoregressive Generation

Conversely, modern generative LLMs (such as the GPT family, LLaMA, and Pythia) are predominantly based on deep, decoder-only structures. These models employ causal (or unidirectional) masked self-attention, meaning a token can only attend to prior tokens in the sequence. They are optimized for autoregressive generation via a next-token prediction objective.

Because they excel at open-ended reasoning and text generation, decoder-only models are the primary subjects of current AI safety research. In our framework, a decoder-only LLM serves as the base generative model upon which we apply our Federated Learning and unlearning techniques.

The Generative LLM Lifecycle

For the scope of this research, it is crucial to understand the bipartite training lifecycle of these generative Large Language Models. Their development is fundamentally divided into two main stages:

- **Pre-training:** In this initial phase, the model processes massive and largely unfiltered corpora of text using the aforementioned next-token prediction objective. Through this process, the model acquires statistical representations of language, semantic patterns, and broad world knowledge. *Crucially, it is during this phase that the model concurrently absorbs hazardous and dual-use information inherently present in raw internet data.*
- **Fine-tuning / Alignment:** After pre-training, the model undergoes **fine-tuning**, a refinement stage in which its pre-trained weights are further updated through training on a smaller, task-specific dataset, allowing it to adapt to more specialized objectives [10]. When this adaptation process is specifically aimed at guiding the model's behavior towards human preferences and safety principles, it is referred to as **alignment**. In generative LLMs, alignment is commonly achieved through **Supervised Fine-Tuning (SFT)** on instruction-response pairs, often followed by more advanced techniques such as Reinforcement Learning from Human Feedback (RLHF) [19], teaching the model to follow instructions and refuse harmful or unsafe requests.

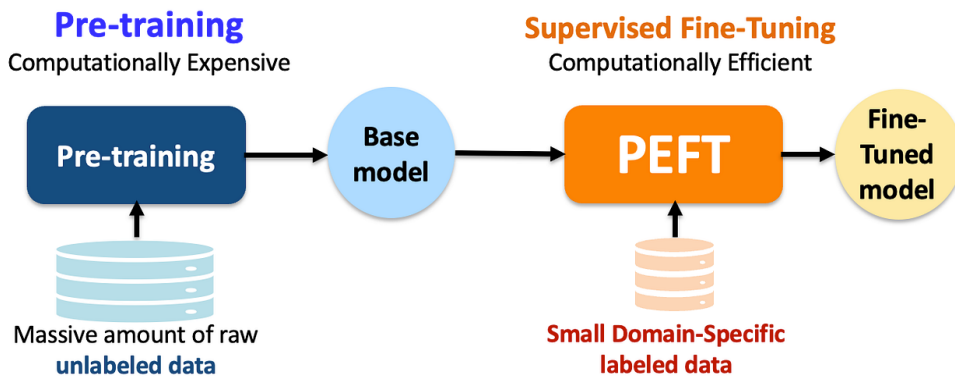


Figure 2.3: LLM lifecycle phases. Source: adapted from [18]

2.1.3. The Open-Weight Paradigm and Safety Implications

Recently, the AI landscape has experienced a paradigm shift with the release of powerful *open-weight* models (e.g., LLaMA, Pythia). Unlike proprietary models accessible only

via closed APIs, open-weight models allow researchers and edge users to download the complete parameter set and execute local adaptations. While this drastically democratizes access to advanced capabilities and enables decentralized architectures like Federated Learning, it inadvertently shifts the control of model safety from the original developers directly into the hands of the end-users. This open paradigm necessitates novel, robust unlearning and risk-aware projection techniques to ensure models remain safe even when fine-tuned in decentralized environments.

2.2. Parameter-Efficient Fine-Tuning (PEFT)

Adapting Large Language Models to specific downstream tasks or attempting to edit their capabilities through standard full-parameter fine-tuning is computationally prohibitive. During standard backpropagation, the system must store not only the model’s weights but also the optimizer states, gradients, and forward activations. For modern models with billions of parameters, these massive computational and memory demands strictly confine training to high-end, centralized GPU clusters, making such procedures entirely unfeasible for resource-constrained edge devices.

To address this critical bottleneck, **Parameter-Efficient Fine-Tuning (PEFT)** methods have become the standard for downstream model adaptation. PEFT strategies aim to match the performance of full fine-tuning while updating only a minimal fraction of the total parameters [33]. They achieve this by strictly freezing the vast majority of the pre-trained base model’s weights, introducing and optimizing only a small set of targeted parameters. This structural approach serves a dual purpose: first, it drastically reduces the computational and memory overhead, since the system no longer needs to calculate gradients or maintain optimizer states for billions of parameters, and second, it inherently protects the model against catastrophic forgetting, ensuring that the foundational capabilities acquired during pre-training remain intact during the adaptation phase.

2.2.1. Low-Rank Adaptation (LoRA)

Among PEFT methodologies, **Low-Rank Adaptation (LoRA)** [11] has emerged as the most widely adopted technique. LoRA hypothesizes that the updates to the weights during neural network adaptation reside on a low intrinsic dimension.

Rather than modifying the pre-trained model weights directly, LoRA freezes the original parameter matrices and injects trainable rank-decomposition matrices into targeted layers of the Transformer architecture (typically the attention mechanisms). Mathematically, for

a pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$, the update is constrained by representing it with two smaller matrices $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$, where the rank $r \ll \min(d, k)$. The forward pass is modified as:

$$h = W_0x + \Delta Wx = W_0x + BAx \quad (2.1)$$

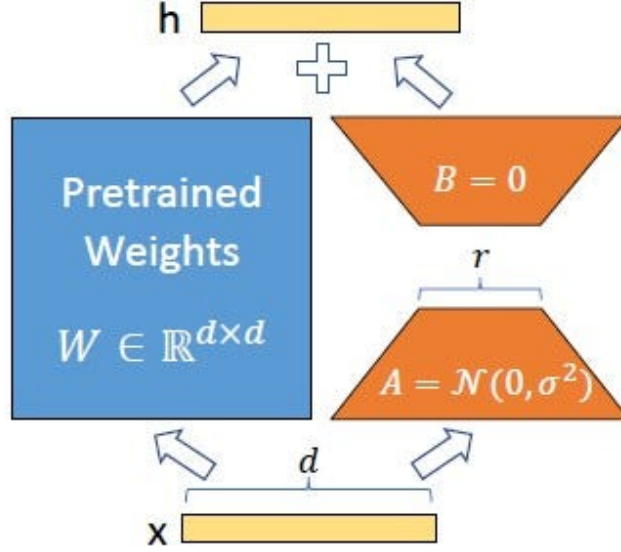


Figure 2.4: Representation of LoRA adapter matrices A and B. The base model parameters are frozen. Adapted from [11]

This approach drastically reduces the number of trainable parameters, often by a factor of 10,000, and significantly lowers the memory footprint. Crucially for safety interventions, LoRA isolates the newly acquired knowledge (or the suppressed behaviors) into modular, lightweight adapters while keeping the foundational model perfectly intact.

2.2.2. Model Quantization and QLoRA

While LoRA significantly reduces the memory required for optimizer states and gradients, the sheer size of the frozen base model (W_0) still poses a prohibitive VRAM barrier for decentralized nodes. To bridge this gap, PEFT is frequently combined with **model quantization**, a technique that reduces the numerical precision of the model's weights (e.g., from 16-bit floating-point to 8-bit or 4-bit integer representations) [13].

A breakthrough in this domain is *QLoRA* (Quantized LoRA) [7]. QLoRA enables the fine-tuning of massive models on consumer-grade hardware without degrading performance in

a significant way. It achieves this by quantizing the frozen base model to a highly efficient 4-bit NormalFloat (NF4) data type, while the small, injected LoRA adapters (A and B) are trained in standard 16-bit precision. During the forward and backward passes, the 4-bit weights are temporarily dequantized to compute the gradients for the adapters.

By synergizing the parameter efficiency of LoRA with the extreme memory compression of QLoRA, it becomes technologically viable to push the fine-tuning process to edge nodes. These innovations are the exact enablers that make decentralized paradigms, such as Federated Learning, more accessible for modern LLMs. Quantitatively, this approach reduces the average memory footprint of fine-tuning by up to 75% compared to standard 16-bit procedures, allowing models with tens of billions of parameters to be effectively adapted on single consumer GPUs with as little as 24GB of VRAM [7].

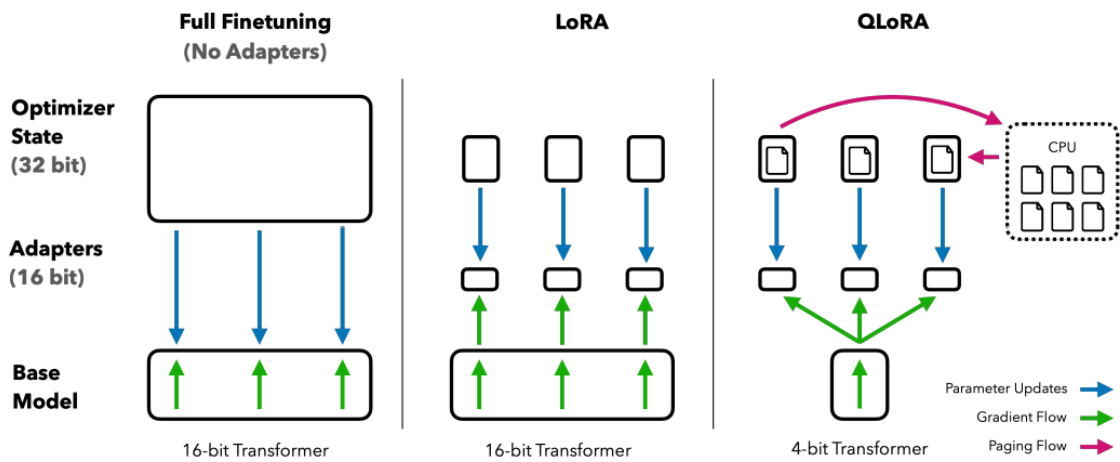


Figure 2.5: Comparison between Full Parameter Training, LoRA and QLoRA, adapted from [7].

2.3. AI Safety in Large Language Models

As Pre-trained Language Models (PLMs) become increasingly capable and deeply integrated into societal infrastructure, ensuring their secure and responsible deployment has emerged as a critical field of research known as AI Safety. Broadly defined, AI Safety encompasses a wide spectrum of challenges aimed at ensuring that artificial intelligence systems operate reliably and in alignment with human values. This includes mitigating unintended model behaviors, such as hallucinations, biases, and the generation of toxic content, as well as preventing the deliberate misuse of AI capabilities by malicious actors.

Historically, the primary objective of safety research has been to make models "helpful,

honest, and harmless" through alignment techniques like RLHF and SFT. However, with the advent of open-weight models and the extraordinary reasoning capabilities of modern architectures, the focus of AI Safety has necessarily expanded. Researchers must now address profound security risks, transitioning from preventing offensive text generation to safeguarding against high-harm, real-world exploitation. The urgency of these emerging threats has recently been formalized by international policy frameworks, such as the U.S. Executive Order on Safe, Secure, and Trustworthy AI [3], which mandate rigorous safety evaluations for advanced foundation models.

2.3.1. The "Dual-Use" Dilemma

At the heart of these real-world security risks lies a severe "dual-use" dilemma, which has been massively amplified by the democratization of Large Language Models. The core of the issue is that the exact same extensive knowledge required to assist researchers in chemistry, biology, or computer science can be readily exploited by malicious actors to lower the barrier for developing cyber-weapons, biological threats [15] or illegal drugs [28].

To systematically quantify these threat levels, recent literature has introduced targeted evaluation benchmarks, such as the *Weapons of Mass Destruction Proxy* (WMDP) benchmark [15]. Unlike general-purpose benchmarks, WMDP is designed to measure the retention of hazardous knowledge specifically in biosecurity, cybersecurity, and chemical security. The benchmark consists of thousands of multiple-choice questions (MCQs) curated by subject-matter experts to serve as a proxy for weaponizable information. A key methodological feature of WMDP is the attempt to "disentangle" benign scientific knowledge from dangerous capabilities, allowing researchers to evaluate whether a model can remain useful for legitimate research while being "sanitized" of high-risk information.

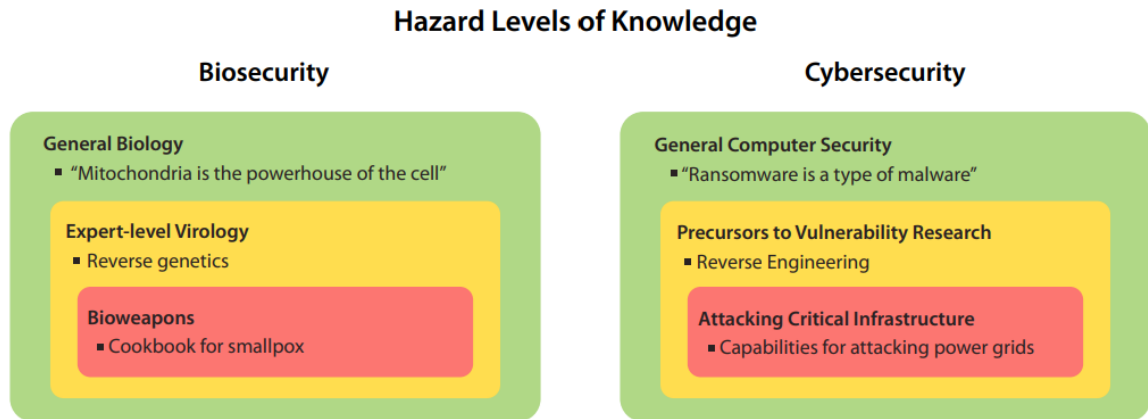


Figure 2.6: Overview of hazardous knowledge domains categorized by the WMDP benchmark. Figure adapted from [15].

2.3.2. Existing Mitigation Strategies

Alongside standard alignment techniques, the research community has developed various advanced interventions aimed at fundamentally suppressing, editing, or removing hazardous knowledge from Large Language Models. The two main paradigms that are the most common are:

- Data Selection and Filtering:** A foundational and preventative approach involves the rigorous curation of training corpora to exclude hazardous or dual-use content [31]. This is a **pre-training** solution, by selectively filtering out sensitive information from the pre-training or fine-tuning datasets, developers aim to prevent the model from ever acquiring the underlying capabilities necessary to generate malicious outputs. Implementations of this strategy range from heuristic keyword blocklists to sophisticated, multi-stage machine learning classification pipelines designed to identify and remove proxy threat knowledge at scale. While this data-centric strategy establishes a strong, tamper-resistant baseline of "deep ignorance", curating web-scale datasets is notoriously difficult and computationally expensive. Furthermore, models relying solely on this mitigation may still remain vulnerable if adversaries intentionally reintroduce hazardous data through subsequent fine-tuning stages. [21]
- Machine Unlearning:** We move on a **post training solution**. This paradigm aims to selectively "erase" specific concepts or behaviors from a model's parameters after the initial training. A prominent optimization-based technique in this domain is *Representation Misdirection for Unlearning* (RMU) [32] [6], which perturbs the

model’s internal activations when exposed to hazardous data to steer representations toward meaningless outputs. Alternatively, other unlearning methods operate strictly in the weight space, such as a particular use of the *Task Arithmetic* [12]. By calculating a "task vector", defined as the element-wise difference between fine-tuned and pre-trained weights, associated with hazardous knowledge, practitioners can subtract or negate this vector from the global model to actively inhibit malicious capabilities. This modular, algebraically-driven unlearning is particularly relevant for decentralized applications.

- **Circuit Breaking:** Staying on **post training** solutions we have Circuit Breaking. Rather than permanently erasing knowledge from the weights, this approach utilizes representation engineering to control the model’s outputs dynamically. As proposed by Zou et al. [35], "circuit breakers" identify and interrupt the neural pathways responsible for generating harmful outputs. By mapping harmful generative processes into incoherent latent spaces, this technique effectively short-circuits the model’s ability to produce malicious content when triggered by adversarial prompts.

2.3.3. Proactive Safety: The Deep Ignorance Framework

A recent work on AI safety of **data filtering** solutions is the *Deep Ignorance* framework [17]. While the aforementioned strategies intervene on models that have already acquired hazardous knowledge, introduces a highly effective proactive paradigm. This data-centric approach operates on the principle that the most robust defense against capability acquisition is to prevent the model from learning the dangerous information in the first place.

The core idea of Deep Ignorance is to build highly tamper-resistant safeguards by meticulously filtering dual-use topics directly from the pre-training and fine-tuning corpora. To achieve this scalably across massive datasets, the methodology employs an efficient multi-stage filtering pipeline:

1. **Initial Blocklist Filtering:** The entire dataset is rapidly scanned against a curated list of domain-specific hazardous keywords (e.g., biosecurity terms). Documents containing no blocked terms are immediately retained for training, bypassing further inspection and vastly reducing the overall computational burden.
2. **Classifier Escalation:** Documents flagged by the blocklist (e.g., containing two or more suspicious keywords) are escalated to a highly accurate semantic classifier, such as a fine-tuned ModernBERT model [30]. This step is crucial to distinguish true threats from benign mentions of the same vocabulary.

3. **Data Sanitization:** The classifier analyzes the semantic content and assigns a threat probability score to each escalated document. Text chunks exceeding a strict safety threshold are permanently discarded, while those scoring below the threshold (false positives) are safely re-integrated into the training pool.
4. **Safe Optimization:** Finally, the language model is pre-trained or continually fine-tuned exclusively on the remaining sanitized corpus.

By ensuring that the training set is "ignorant" of specific threats, this centralized framework guarantees that malicious capabilities are never encoded into the model's parameters, making them highly resistant to adversarial fine-tuning and revival. Because of its structural robustness, adapting the principles of this centralized data-filtering pipeline into a privacy-preserving federated environment forms the foundational baseline of our research. This can come at a cost of the overall knowledge of the models, but the research result demonstrate a minimal loss in such benchmarks.

2.4. Federated Learning (FL)

As the demand for data privacy and regulatory compliance intensifies, traditional centralized machine learning, where raw data is fully contained in a single datacenter, faces significant structural and legal barriers. Federated Learning (FL) [16] has been proposed as a decentralized alternative that mitigates these concerns by shifting the paradigm: rather than bringing the data to the model, FL brings the model to the data.

In a federated architecture, raw data never leaves the edge devices or local data silos. Instead, participating clients locally compute weight updates using their private datasets and transmit only these mathematical updates to a central server. Consequently, this decentralized paradigm is uniquely suited for highly sensitive domains like hospital data, personal devices or corporate servers. For example it enables collaborative medical research across independent hospitals without sharing patient records, facilitates privacy-preserving AI applications on personal mobile devices, and allows secure intelligence sharing among isolated corporate entities.

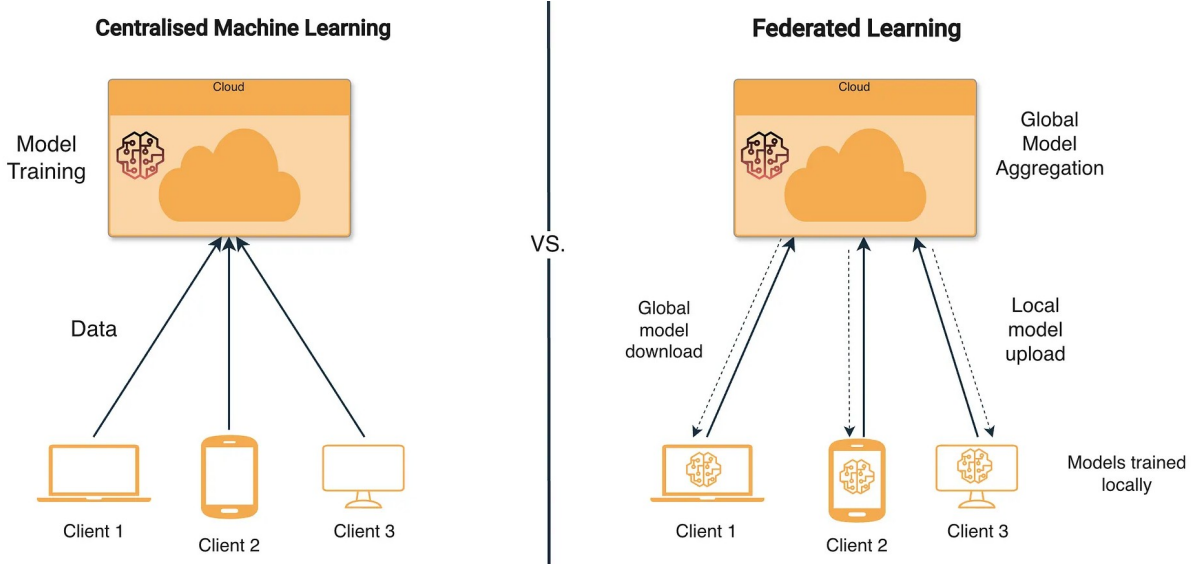


Figure 2.7: Comparison between a centralized training setup (left) and a federated training setup (right). Figure adapted from [20].

2.4.1. Federated Averaging (FedAvg)

The foundational algorithm driving this decentralized orchestration is **Federated Averaging (FedAvg)** [16] [26]. FedAvg is an aggregation algorithm, the most commonly used in Federated Learning. The training process proceeds over discrete communication rounds. At the beginning of round t , the central server broadcasts the current global model parameters to a set of K selected clients. Each client k performs local training on its private dataset containing n_k samples to optimize the parameters.

After local optimization, clients send their updated parameters back to the server. The server then constructs the new global model for round $t + 1$ by computing a weighted average of the received updates, strictly proportional to the volume of data processed by each client. Mathematically, the aggregation is defined as:

$$\Phi_{t+1} = \sum_{k=1}^K \frac{n_k}{N} \Phi_{t+1}^k$$

where $N = \sum_{k=1}^K n_k$ is the total number of training samples across all participating clients, and Φ represents the trainable parameters.

2.4.2. Federated PEFT (FedPEFT) and FedLoRA

While standard FL was initially designed for smaller architectures, applying it directly to Large Language Models introduces a severe communication bottleneck. Transmitting billions of parameters back and forth across a network at every communication round is bandwidth-prohibitive.

To resolve this limitation, recent literature has formalized the intersection of decentralized training and efficient adaptation through frameworks broadly categorized as **Federated PEFT (FedPEFT)** [14] [25].

These approaches integrate techniques like the very common LoRA into the federated pipeline. In a typical *FedLoRA* setup, the massive pre-trained base model remains completely frozen and is shared across all clients. During the federated rounds, clients locally optimize only the low-rank adapter matrices ($\Delta\Phi$) and transmit these lightweight payloads back to the server.

Consequently, the central server utilizes standard aggregation algorithms like FedAvg to average solely the adapter updates. By exchanging merely a fraction of the total parameters (often less than 1%), using FedPEFT techniques it can successfully aligns the communication overhead with the constraints of edge networks, making the collaborative fine-tuning of frontier LLMs practically viable.

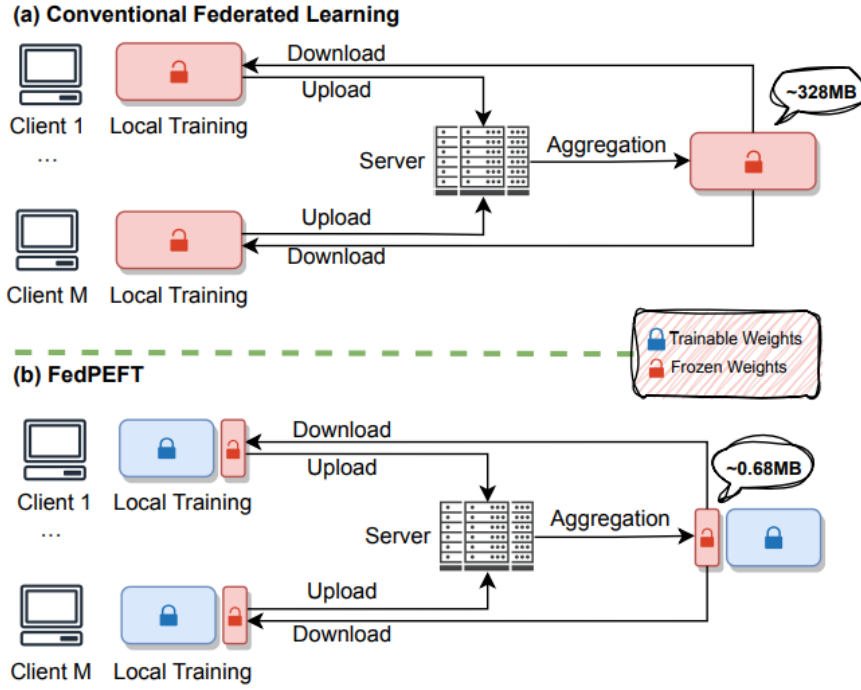


Figure 2.8: Comparison between a classical federated learning setup and fedPEFT setting . Figure adapted from [25].

2.5. Research Gap: The Decentralized Safety Challenge

While Federated Learning successfully resolves data privacy concerns, it introduces a severe vulnerability regarding proactive model safety. By design, the central orchestrator in a federated network is strictly prohibited from inspecting local data distributions. Consequently, highly effective centralized safeguards, such as the explicit data-filtering pipeline of the *Deep Ignorance* framework [17], cannot be directly applied by the server. However, in principle, an individual client could still apply such filtering unilaterally on its own local data.

This architectural constraint creates the *"blind server problem"*, representing a gap in current AI safety literature. In a standard FedPEFT setup, the server simply aggregates incoming LoRA adapters without any semantic awareness of the knowledge they encode. To the central orchestrator, a weight update enhancing general biological reasoning is mathematically indistinguishable from an update injecting hazardous virology instructions.

Consequently, if a participating client fine-tunes the model on an unfiltered local dataset containing dual-use content, as could plausibly occur even among non-malicious collaborators, such as a legitimate biomedical lab handling sensitive virology documents, the server will blindly average this contaminated weight update into the global model. This process bypasses the model’s initial alignment, irrevocably injecting hazardous capabilities into the global parameters.

Overcoming this limitation requires a fundamental shift in the AI safety paradigm. Since centralized data inspection is impossible in FL, the current state-of-the-art lacks a mechanism to evaluate the semantic threat level of a decentralized update *before* it poisons the global model. Addressing this critical vulnerability requires moving towards risk-aware global aggregation strategies that operate directly within the PEFT weight space. Bridging this gap by integrating data-filtering logic, task arithmetic, novel weight-space techniques and federated orchestration is the primary objective of the methodology proposed in the following chapter

2.6. Research Question

Building upon the established research gaps, this research will try to answer the following question: *How can we perform Parameter-Efficient Fine-Tuning (PEFT) in a federated environment while effectively mitigating safety risks and preventing the emergence of harmful capabilities?*

Specifically, this work focuses on:

- **RQ1 (PEFT Replicability):** Can the data-filtering interventions of Deep Ignorance be effectively replicated when fine-tuning is strictly bounded to a Parameter-Efficient (PEFT) regime?
- **RQ2 (Federated Replicability):** Can the training methodology and safety properties established in the initial phase of the Deep Ignorance study be effectively replicated within a Federated Learning framework?
- **RQ3 (Safety-Robustness beyond Data Selection):** Are there federated aggregation and weight-space manipulation techniques that offer higher *safety-robustness* against the revival of hazardous capabilities compared to simple data selection or reduction strategies?
- **RQ4 (Performance Trade-offs):** What is the impact of these decentralized safety-aware strategies on the balance between risk mitigation (safety) and the gen-

eral reasoning capabilities (utility) of the model?

3 | Methodology

In this chapter, we provide a detailed overview of the research methodology adopted to develop a safe federated fine-tuning framework. Our approach follows a structured, multi-stage process: it begins with the adaptation of established centralized safety techniques to a resource-efficient setting, progresses into transposing these techniques to a decentralized architecture, and concludes with the formulation of novel safety-aware aggregation strategies.



Figure 3.1: Methodology pipeline steps.

The methodology is organized into four main phases pipeline (illustrated in Figure 3.1), where each module is a self-contained experiment whose outcome *gates* the transition to the next one. The gating outcome of each module is assessed along two complementary axes, *Risk* (or safety), which captures the extent to which a model retains or exhibits dual-use hazardous knowledge, and *Utility* captures the extent to which general-purpose model capability is preserved. A favorable outcome at any gate requires Risk to decrease while Utility remains stable.

The four modules are:

1. **Centralized Baseline Verification:** Adapting the safety principles to a resource-efficient setting (QLoRA) to check that data-filtering safety survives a parameter-efficient regime.
2. **Federated Transition:** Re-establishing the same behaviour under a decentralized paradigm to evaluate vulnerabilities and fix the reference baselines.

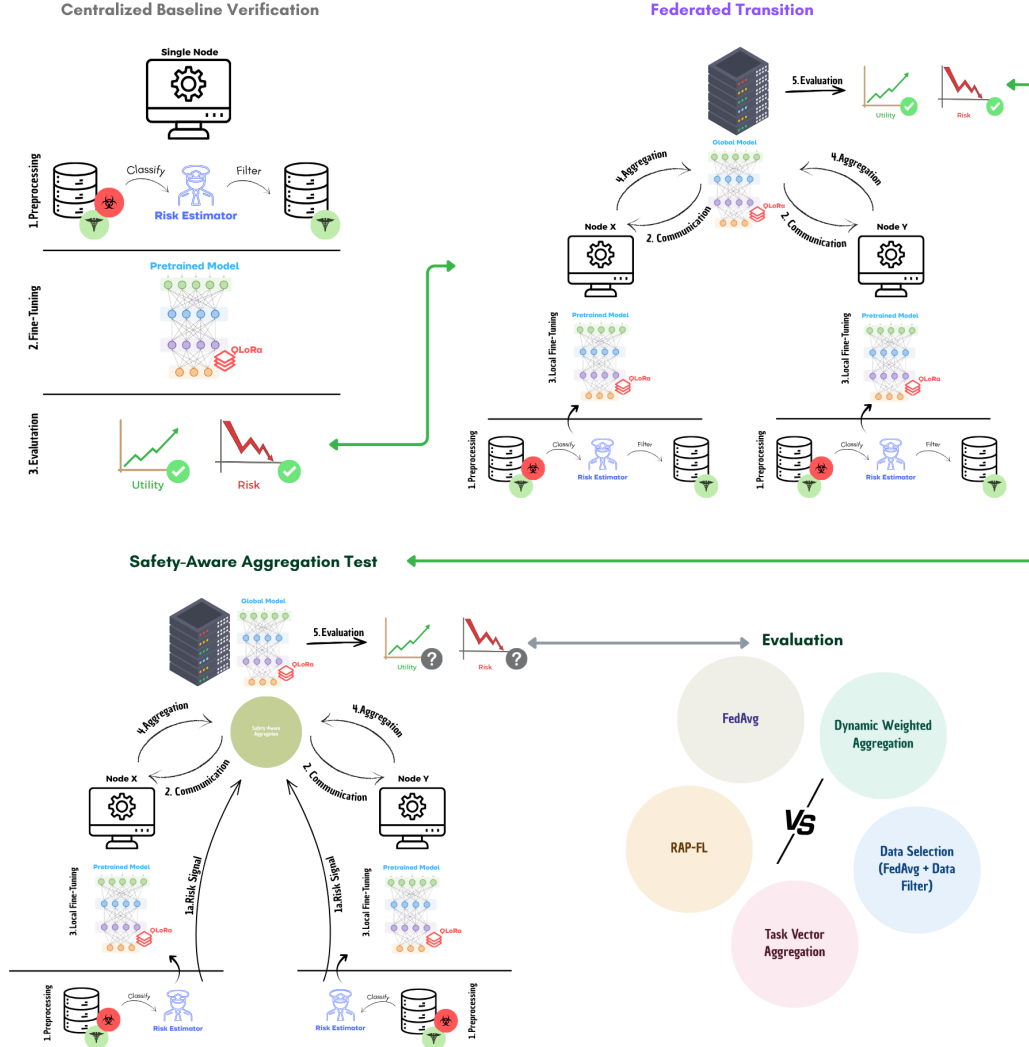


Figure 3.2: Methodology Overview

3. **Safety-Aware Aggregation Test:** The core contribution of this thesis, where we implement and run FL configurations with novel safety-aware aggregation, that move beyond simple data selection.
4. **Evaluation:** A final module that compares all applied strategies head-to-head on the crucial safety-utility trade-off.

3.1. Module 1: Centralized Baseline Verification

The first step of our methodology derives from the *Deep Ignorance* study on a *single node* and tries to answer to our first research question (RQ1). However, rather than full-

parameter training, we utilize a low-rank, 4-bit-quantized adapter (QLoRA). Since the original safeguard was validated with full-parameter training, this computational bottleneck could plausibly alter the dynamics that make data filtering effective; we therefore verify the mechanism in a centralized setting before incurring the cost of federation. Only when this pattern holds, Risk decreasing while Utility is preserved (the green-check outcome in Fig. 3.2), the safeguard is considered transposable and the pipeline moves to the Federated setting.

To formalize this approach within our PEFT context, we build upon the mechanics of LoRA, defined in 2.2.1, let θ represent the entire parameter space of our model. We can decompose it as:

$$\theta = \theta_0 \oplus \phi \quad (3.1)$$

where θ_0 represents the frozen pre-trained weights and ϕ represents the trainable PEFT parameters. The forward pass is the one shown in 2.1. This architectural choice was primarily driven by the need to significantly reduce computational overhead (memory footprint) due to our limited resources.

As shown in our pipeline, this module runs three steps:

1. **Preprocessing:** A *Risk Estimator*, a classifier, scores every document.
2. **Fine-Tuning:** The frozen pre-trained model is adapted on controlled dataset compositions through QLoRA.
3. **Evaluation:** The adapted model is measured on Utility and Risk benchmarks.

The purpose is to confirm two preconditions: that exposure to hazardous data measurably raises hazardous capabilities, and that excluding it suppresses that capability at an acceptable utility cost. Only when the required pattern holds, risk decreasing while utility is preserved at a good level (the green-check outcome in the pipeline), the safeguard is considered transposable, and the framework moves to the Federated setting.

3.2. Module 2: Federated Transition

The second module addresses the second research question (RQ2) and applies the safety baseline approach under a decentralized architecture. In this FL setup, raw training data remains strictly localized on individual client nodes to preserve data privacy.

Building upon the global PEFT formulation established in Module 1 ($\theta = \theta_0 \oplus \phi$), the frozen pre-trained base model θ_0 is distributed and shared across all clients. Only the

lightweight LoRA adapters ϕ are trained and exchanged. Crucially, transmitting only these low-rank parameters drastically reduces the communication overhead, making the adaptation of Large Language Models viable over edge networks.

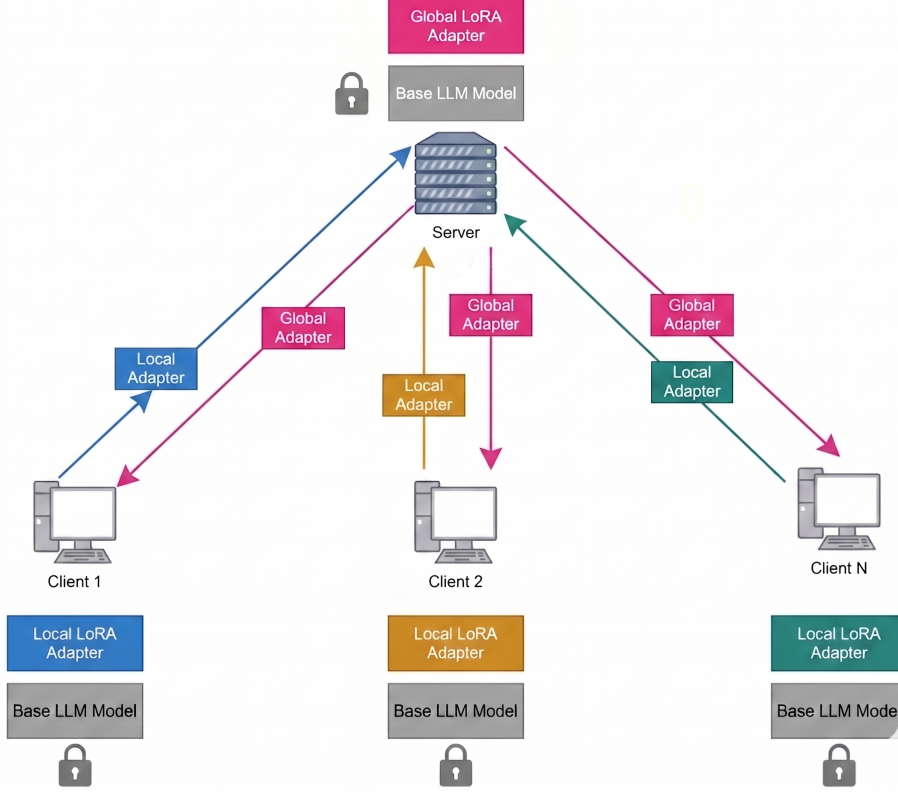


Figure 3.3: Illustration of Federated PEFT LoRA. Each nodes shares the same frozen base LLM model, only the adapters are exchanged.

Let $\mathcal{K} = \{1, \dots, K\}$ be the set of participating clients. The federated learning process proceeds over discrete communication rounds $t = 0, 1, \dots, T - 1$.

Each client performs the federated counterpart of the centralized pipeline. More in detail the process is divided into a one-time setup phase and an iterative training loop

Part 1: Local Initialization (Executed Once)

1. **Local Data Assessment:** Independently, each client $k \in \mathcal{K}$ utilizes an identical, locally deployed Risk Estimator to evaluate its private dataset $\mathcal{D}_k$. The classifier assigns a continuous safety and threat score, $P_{threat}(x)$, to each individual sample x .
2. **Data Pre-processing and Scoring:** Based on the estimator's output, a method-specific preprocessing step is applied. For our data-selection baseline (Method 1), a filter is utilized to completely discard malicious samples from $\mathcal{D}_k$.

Part 2: Federated Training (Iterative Communication Rounds)

The federated learning process proceeds over discrete communication rounds $t = 0, 1, \dots, T-1$.

3. **Global Synchronization:** At the beginning of each round t , the central server broadcasts the current global model parameters (the LoRA weights, denoted as Φ_t) to all participating clients in $\mathcal{K}$.
4. **Local Fine-Tuning:** Each client k initializes its local adapter with the global weights ($\Phi_{k,t} \leftarrow \Phi_t$) and performs Parameter-Efficient Fine-Tuning on its prepared dataset $\mathcal{D}_k$. By minimizing a language modeling loss $\mathcal{L}_k$, the client computes an updated local adapter $\Phi_{k,t+1}$.
5. **Upload and Global Aggregation:** Clients calculate their explicitly updated parameter deltas ($\Delta\Phi_{k,t} = \Phi_{k,t+1} - \Phi_t$) and transmit these lightweight payloads back to the central server. The server then aggregates these updates to form the new global model ($\Phi_{t+1} = \Phi_t + \Delta\Phi_{global}$). This step represents the core of our methodological variations. For our reference baselines, the server applies standard **Federated Averaging (FedAvg)**, which computes a data-weighted average of the client updates:

$$\Delta\Phi_{global} = \sum_{k=1}^K \frac{n_k}{N} \Delta\Phi_{k,t} \quad (3.2)$$

where n_k is the number of training samples processed by client k , and $N = \sum n_k$ is the total number of samples across the active network.

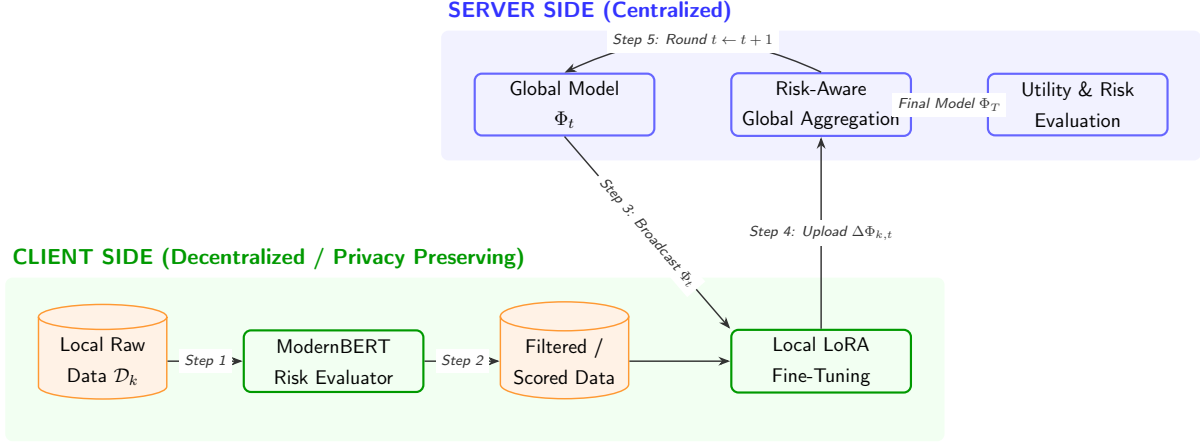


Figure 3.4: Overview of the **general** Safe Federated Fine-Tuning pipeline. The process separates the one-time local data assessment (Steps 1-2) from the iterative federated learning loop (Steps 3-5). The central server aggregates weight updates without ever accessing the raw local data.

The primary objective of this phase is to answer our second research question (RQ2) regarding federated replicability. To achieve this, we transposed the paradigms established in Phase 1 into a two-client federated architecture using standard **Federated Averaging (FedAvg)**.

A concluding **(5) Evaluation** verifies whether the safety pattern survives once training is decentralized. A positive (or suboptimal) outcome here justifies the necessity of the advanced safety-aware mechanisms developed in the next module.

This module also fixes the two references that propagate to the final comparison: *Method 0 (Standard FedAvg)* with no intervention, which establishes the ground baseline of comparison for decentralized vulnerability, and *Method 1 (FedAvg with Safety Filtering)*, which replicates the *Deep Ignorance* approach under the federated constraint: each client applies the same data-level safety classifier locally to its own dataset prior to fine-tuning, after which standard FedAvg aggregation proceeds unmodified.

Mathematical details are presented in the following subsections:

3.2.1. Method 0: Standard Federated Averaging (Baseline)

As our absolute reference point, we utilize the standard **Federated Averaging (FedAvg)** algorithm. To formalize this approach within our Parameter-Efficient Fine-Tuning (PEFT) context, we first define the mechanics of Low-Rank Adaptation (LoRA) that stands for all methods.

During the local fine-tuning phase, the original pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$ remains frozen. The model is updated by learning a low-rank adapter ΔW , parameterized by two smaller matrices $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$, where the rank $r \ll \min(d, k)$. The forward pass is thus modified as:

$$h = W_0x + \Delta Wx = W_0x + BAx \quad (3.3)$$

In our federated framework, the term $\Delta\Phi_k$ represents the concatenated vector of all LoRA parameters (the matrices A and B) learned by client k .

In this baseline scenario (Method 0), the central server performs a standard weighted average (FedAvg algorithm) of these local LoRA adapters based solely on the number N of data samples processed by each client:

$$\Delta\Phi_{global} = \sum_{k=1}^K \frac{n_k}{N} \Delta\Phi_k \quad (3.4)$$

where n_k is the number of training samples on client k , N is the total number of samples across all K clients, and $\Delta\Phi_{global}$ is the resulting global adapter update.

In this scenario, the local safety classifier is entirely bypassed, as no risk-aware filtering or specialized aggregation strategies are implemented. Rather than being a proposed solution, this approach serves as an experimental baseline. By subjecting the standard FedAvg protocol to our experiment provides us the base ground behavioral trend against which we can evaluate the effectiveness and the trade-offs of our proposed safety-aware strategies.

3.2.2. Method 1: Federated Averaging with Safety Filtering

As our baseline for data-level safety and second baseline, this strategy transposes the strict data-filtering mechanism of the original *Deep Ignorance* study into our federated pipeline. Unlike the following weight-manipulation techniques, Method 1 operates entirely during the *Local Data Assessment* and *Filtering* phases (Steps 1 and 2 of our pipeline) before any training occurs.

For each client k with a local dataset D_k , the safety Risk Estimator evaluates every text sample $x_i \in D_k$, outputting a threat probability $P_{threat}(x_i)$. We define a "hard filter" mechanism governed by a predefined safety threshold τ . The client constructs a sanitized data subset, $D_{safe,k}$, by completely discarding any sample whose threat probability exceeds

this threshold:

$$D_{safe,k} = \{x_i \in D_k \mid P_{threat}(x_i) \leq \tau\}$$

Following this filtering step, the training pipeline proceeds exactly as defined in Method 0. Each client performs standard local fine-tuning exclusively on its sanitized subset $D_{safe,k}$, and the central server aggregates the resulting LoRA adapters using standard FedAvg. In this scenario the complementary subset of data to the $D_{safe,k}$, instead labeled as $D_{threat,k}$ is just discarded without use.

3.3. Module 3: Safety-Aware Aggregation Test

This module represents the core contribution of this thesis. It directly addresses our third research question (**RQ3**) by introducing techniques that move beyond simple data selection, laying the necessary architectural groundwork to subsequently evaluate the utility-safety trade-offs (**RQ4**) in the next module. Rather than proposing a standalone solution, this phase introduces 3 safety-aware strategies designed to actively mitigate risk at both the client and server levels.

The core concept relies on integrating local data evaluation with risk-aware global aggregation. Instead of silently dropping data at the client level (as in the Method 1 baseline), our advanced methods fundamentally bypass hard data filtering. Instead, each client emits a *Risk Signal*. This signal is sent (or utilized locally) alongside the adapter update, ensuring that the global training process incorporates vital safety constraints without ever exposing the raw data itself to the central server.

The scoring function is standardized across the network, but its honest execution represents an explicit trust assumption in our framework: the local Risk Estimator is taken to be a certified, non-forgable entity, since a malicious client could otherwise arbitrarily report benign scores for hazardous data to bypass defenses.

The server (or the client’s local graph) consumes this signal inside a *Safety-Aware Aggregation Mechanism*. To systematically compare these different solutions (Methods 2, 3, and 4) against our baselines and establish a common methodological ground, we re-define a similar five-step operational pipeline to the one previously defined in module 2:

Part 1: Local Initialization (Executed Once)

1. **Local Data Assessment:** Unchanged from Module 2 (Step 1): each client independently scores its private dataset with the same Risk Estimator.

2. **Data Pre-processing and Scoring:** For our proposed advanced solutions (Methods 2, 3, and 4), the filtering of the baseline Method 1 is removed; instead, the continuous threat probabilities scored by the Risk Estimator are retained to explicitly guide the subsequent data partitioning, local optimization or risk-aware aggregation depending on the method.

Part 2: Federated Training (Iterative Communication Rounds)

3. **Global Synchronization:** Unchanged from Module 2 (Step 3).
4. **Local Fine-Tuning Round:** Clients perform Parameter-Efficient Fine-Tuning (LoRA) on their prepared local datasets. Depending on the specific strategy being tested, this training phase can optimize a single, standard LoRA adapter or compute distinct, risk-weighted gradient updates for the "Safe" and "Threat" data partitions.
5. **Global Aggregation:** Clients transmit their locally computed LoRA updates back to the server. The server then aggregates these adapters into a new global model. This step represents the core of our methodological variations: Method 2 retains the structural form of a weighted sum but replaces the data-proportional weight n_k/N with a risk-aware scalar w_k , deliberately omitting renormalization; Methods 3 and 4 instead go beyond simple weighted averaging, operating directly on the adapters as vectors in weight space.

The following subsections detail the mathematical formulation and operational logic specific to each of these three methods: *Method 2 (Dynamic Weighted Aggregation)*, *Method 3 (Federated Task Arithmetic)*, and *Method 4 (Risk-Aware Projection for Federated Learning, RAP-FL)*.

3.3.1. Method 2: Dynamic Weighted Aggregation

Moving beyond the standard data-volume weighting of FedAvg and the binary exclusion of the baseline Method 1, Method 2 introduces a **risk-aware, sample-level weighting** system. Instead of assigning a static, data-proportional weight to the entire client as in FedAvg, this algorithm dynamically evaluates the exact composition of the mini-batches processed during the local training round. Structurally, Method 2 retains the weighted-sum form of FedAvg's aggregation rule, but replaces the data-proportional weight n_k/N (in eq. 3.4) with this risk-aware scalar w_k .

To achieve this while strictly preserving data privacy (as the server remains blind to

local distributions), the weight computation is performed entirely on the client side. We establish an asymmetric reliability baseline: each sample classified as "Safe" (S) by the local risk estimator classifier contributes a high reliability fixed score of 0.8, whereas a "Threat" (T) sample contributes a heavily penalized score of 0.2.

At the conclusion of the local training round, each client k calculates its raw aggregation weight w_k by averaging the reliability scores of the samples it actually processed. Let $N_{S,k}$ and $N_{T,k}$ represent the number of safe and threat samples processed during the *current round*, respectively, and $N_{tot,k}$ represent the total number of processed samples *in that same round*. The local weight is defined as:

$$w_k = \frac{0.8 \cdot N_{S,k} + 0.2 \cdot N_{T,k}}{N_{tot,k}} \quad (3.5)$$

Once computed, the clients transmit their respective scalar weights w_k alongside their LoRA adapters $\Delta\Phi_k$ to the central server.

Crucially, upon receiving these updates, the central server performs the aggregation *without* applying any mathematical renormalization (i.e., without dividing by the sum of all local weights). The global model update is computed strictly as:

$$\Delta\Phi_{global} = \sum_{k=1}^K w_k \Delta\Phi_k \quad (3.6)$$

By deliberately omitting the renormalization step, the absolute magnitude of the global update becomes strictly dependent on the overall safety profile of the network. Mathematically, if the participating clients process a high volume of threat data, the sum of w_k will naturally decrease, proportionally shrinking the global update. Consequently, each client's update is directly scaled according to its underlying data type distribution: clients with more hazardous data exert proportionally less influence.

We designed this architectural choice to investigate whether this implicit dampening effect could act as a safety-driven decay mechanism. The empirical validity of this hypothesis, alongside any unintended consequences on training stability or model utility, is a subject of our critical evaluation in Chapter 5.

3.3.2. Method 3: Federated Task Arithmetic

The third proposed method is based on the *Task Arithmetic* paradigm, a vector arithmetic approach that interprets model weight updates as specific directions (task vectors) within

the parameter space.

Formal Definition of Task Vector

As formalized by Ilharco et al. [12], a **task vector** specifies a direction in the weight space of a pre-trained model such that movement in that direction improves performance on a target task. Let $\theta_{pre} \in \mathbb{R}^d$ be the weights of a pre-trained model, and $\theta_{ft} \in \mathbb{R}^d$ be the corresponding weights of the same model after fine-tuning on a specific task. The task vector $\tau \in \mathbb{R}^d$ is formally defined as the element-wise difference between the fine-tuned and pre-trained weights:

$$\tau = \theta_{ft} - \theta_{pre} \quad (3.7)$$

The core premise of this paradigm is that these task vectors can be steered and combined through standard arithmetic operations, such as addition and negation, to edit the model's behavior. For instance, while adding task vectors can improve performance on multiple tasks simultaneously, negating a task vector (i.e. $-\tau$) effectively decreases the model's performance on the target task, inducing a targeted "forgetting" mechanism.

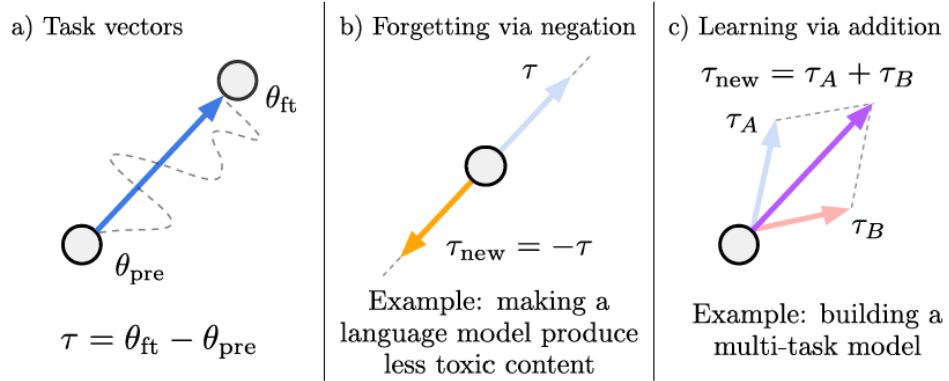


Figure 3.5: Visual representation of Task Arithmetic. Figure reproduced from Ilharco et al. [12].

Translating Task Vectors to Federated Learning

While Task Vectors and Task Arithmetic have proven highly effective for model editing and safety alignment in centralized environments, deploying them within a Federated Learning (FL) architecture presents unique challenges and opportunities. To integrate this paradigm into our federated safety framework, we decouple the process into two distinct phases: Local Task Vector Generation (Client) and Global Task Arithmetic Aggregation

(Server).

A great focus is put on the aggregation phase to isolate and combine client-specific learning components, allowing the server to selectively "negate" representations associated with hazardous information through destructive interference.

In a federated scenario, the methodology is structured into the following operational phases:

1. **Local Data Split:** After each client k has evaluated its local dataset D_k using the internal safety classifier, the data is strictly partitioned into two disjoint pools based on a predefined safety threshold τ . This yields a safe subset $D_{safe,k} = \{x_i \in D_k \mid P_{threat}(x_i) \leq \tau\}$ and a threat subset $D_{threat,k} = \{x_i \in D_k \mid P_{threat}(x_i) > \tau\}$.
2. **Dual Local Fine-Tuning:** The client executes two separate training processes on the frozen base model with distinct objectives. The safe data pool $D_{safe,k}$ is utilized to train a standard benign adapter update, denoted as $\Delta\Phi_{safe,k}$. Conversely, the threat data pool $D_{threat,k}$ is used to train a separate adapter, effectively isolating the malicious knowledge into a specific threat task vector, $\Delta\Phi_{threat,k}$.
3. **Aggregation and Negation:** The central server first aggregates the client-specific updates into two distinct global task vectors using standard FedAvg. Let $|D_{safe}|$ and $|D_{threat}|$ represent the total number of safe and threat samples across all K clients, respectively:

$$\Delta\Phi_{safe_global} = \sum_{k=1}^K \frac{|D_{safe,k}|}{|D_{safe}|} \Delta\Phi_{safe,k} \quad (3.8)$$

$$\Delta\Phi_{threat_global} = \sum_{k=1}^K \frac{|D_{threat,k}|}{|D_{threat}|} \Delta\Phi_{threat,k} \quad (3.9)$$

Subsequently, the server performs a vector subtraction to derive the final aggregated model update. The global threat vector is scaled by a steering factor α and subtracted from the safe global update to actively inhibit hazardous capabilities:

$$\Delta\Phi_{final} = \Delta\Phi_{safe_global} - \alpha\Delta\Phi_{threat_global} \quad (3.10)$$

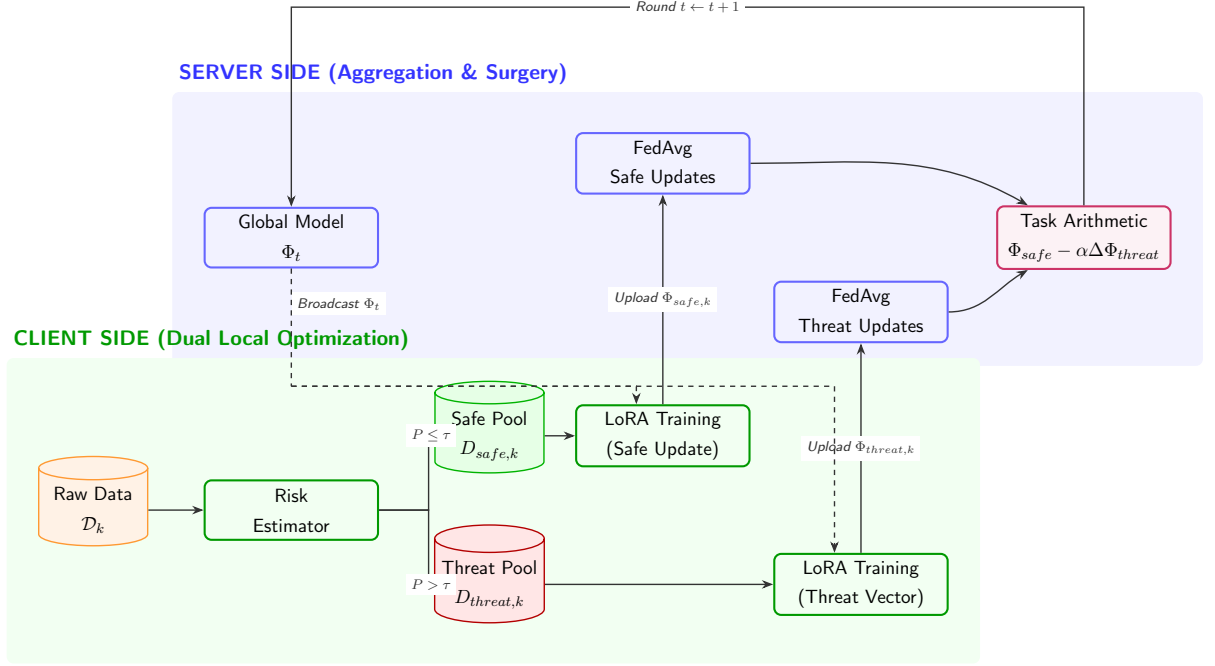


Figure 3.6: Architecture of Method 3 (Federated Task Arithmetic). The client splits its data and computes two distinct parameter updates. The server aggregates these into separate global representations and performs surgical vector negation to actively suppress the hazardous capabilities.

Mathematical Adaptation for LoRA and Weight-Space Surgery

In a standard full-parameter fine-tuning scenario, the task arithmetic operation $\Delta\Phi_{safe} - \alpha\Delta\Phi_{threat}$ would be mathematically straightforward, as the updates represent direct gradients applied to the model weights. However, since our architecture exclusively trains LoRA adapters while keeping the base model frozen, a specific adaptation is required to prevent algebraically induced catastrophic forgetting and model collapse.

In our federated setup, at the beginning of each round t , the server broadcasts the previous global adapter, which we denote as Φ_t . After the local training phase, the clients do not return the isolated gradient $\Delta\Phi$, but rather the absolute updated weights of their adapters. Assuming the server aggregates these into two global representations, we have:

- **Global Safe Adapter:** $\Phi_{safe_global} = \Phi_t + \Delta\Phi_{safe_global}$
- **Global Threat Adapter:** $\Phi_{threat_global} = \Phi_t + \Delta\Phi_{threat_global}$

A naive subtraction of these absolute weights (e.g., $\Phi_{naive} = \Phi_{safe_global} - \alpha\Phi_{threat_global}$) would inadvertently subtract the historical knowledge embedded in Φ_t . By expanding the

terms of this naive approach, the algebraic flaw becomes evident:

$$\Phi_{naive} = (\Phi_t + \Delta\Phi_{safe_global}) - \alpha(\Phi_t + \Delta\Phi_{threat_global}) \quad (3.11)$$

$$\Phi_{naive} = (1 - \alpha)\Phi_t + \Delta\Phi_{safe_global} - \alpha\Delta\Phi_{threat_global} \quad (3.12)$$

As demonstrated, the coefficient representing the prior historical knowledge (Φ_t) is artificially reduced to $(1 - \alpha)$. For instance, applying a steering factor of $\alpha = 0.5$ yields:

$$\Phi_{naive} = 0.5\Phi_t + \Delta\Phi_{safe_global} - 0.5\Delta\Phi_{threat_global} \quad (3.13)$$

This mathematically explains the algebraically induced catastrophic forgetting: systematically halving (or otherwise diminishing) the foundational parameter set Φ_t at every communication round leads to a rapid, exponential degradation of the model's general utility.

To preserve the foundational knowledge intact, our implementation surgically extracts the isolated *Threat Task Vector* by subtracting the starting global parameters from the received threat adapter. The exact aggregation formula implemented on the server is defined as:

$$\Phi_{t+1} = \Phi_{safe_global} - \alpha(\Phi_{threat_global} - \Phi_t) \quad (3.14)$$

The mathematical validity of this surgical extraction is demonstrated by expanding the terms:

$$\Phi_{t+1} = (\Phi_t + \Delta\Phi_{safe_global}) - \alpha((\Phi_t + \Delta\Phi_{threat_global}) - \Phi_t) \quad (3.15)$$

$$\Phi_{t+1} = \Phi_t + \Delta\Phi_{safe_global} - \alpha\Delta\Phi_{threat_global} \quad (3.16)$$

This operation ensures that the global model retains 100% of its prior knowledge (maintaining a coefficient of 1 for Φ_t), while applying the unlearning mitigation exclusively to the malicious directional components acquired during the current round.

Algorithm 3.1 Federated Task Arithmetic Aggregation

- 1: **Server receives:** $\{\Phi_{safe,k}, \Phi_{threat,k}\}$ from clients
 - 2: $\Phi_{safe_global} \leftarrow \sum \text{FedAvg}(\Phi_{safe,k})$
 - 3: $\Phi_{threat_global} \leftarrow \sum \text{FedAvg}(\Phi_{threat,k})$
 - 4: $\Delta\Phi_{threat_round} \leftarrow \Phi_{threat_global} - \Phi_t$ ▷ Task Arithmetic
 - 5: $\Phi_{t+1} \leftarrow \Phi_{safe_global} - \alpha\Delta\Phi_{threat_round}$
 - 6: **return** Φ_{t+1}
-

3.3.3. Method 4: Risk-Aware Projection for Federated Learning (RAP-FL)

The final and most sophisticated strategy evaluated in our framework is **Risk-Aware Projection for Federated Learning (RAP-FL)**. While the previous non-baseline methods (Methods 2 and 3) rely heavily on specialized server-side aggregation strategies and vector arithmetic, Method 4 introduces an exclusively client-side solution centered on a just the labeling and a peculiar training procedure. In this approach, the core of the safety intervention consist disentangling utility and risk directly during the local optimization phase through orthogonal projections within the parameter space. RAP-FL shifts the complexity of safety enforcement to the client nodes, enabling the server to subsequently to just perform a regular aggregation.

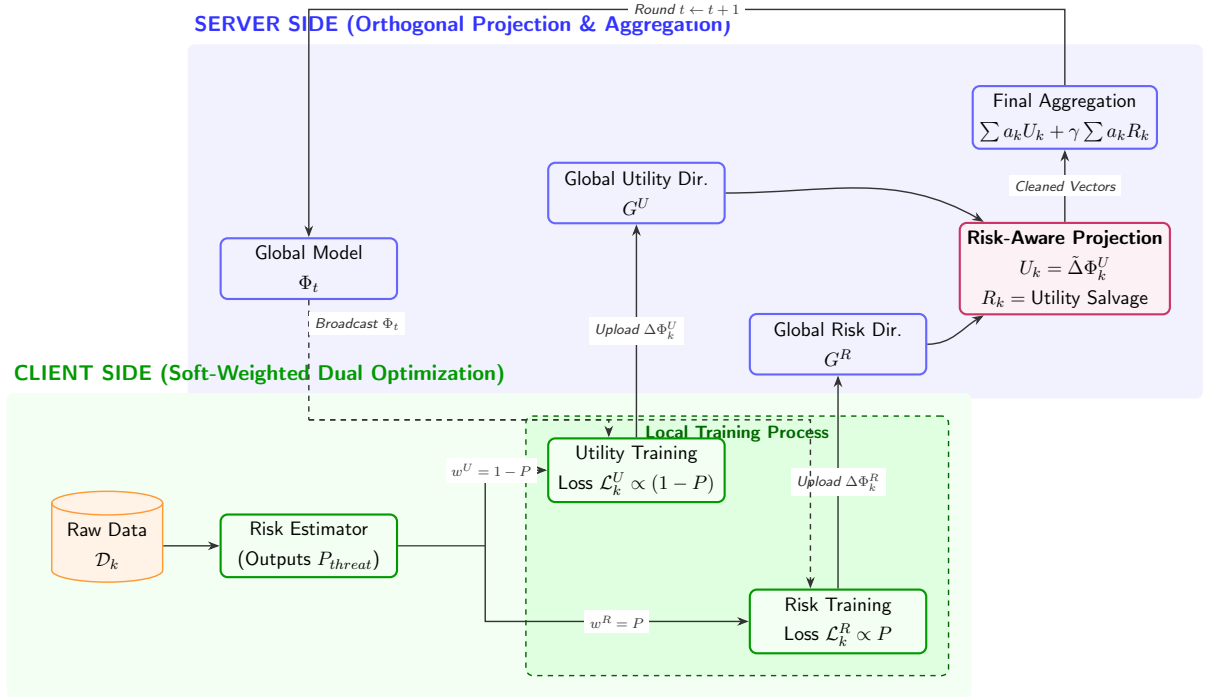


Figure 3.7: Architecture of RAP-FL...

The methodology is structured into the following operational phases:

1. Classifier-Guided Score Assignment (Client-Side)

Unlike previous methods where the classifier score $P_{threat}(x)$ serves only as label set by a threshold τ , RAP-FL doesn't directly label data into *Safe* or *Threat* but keep the probability output of the classifier $P_{threat}(x)$ as score. For every sample $x \in \mathcal{D}_k$, the client

uses the local safety classifier’s threat probability, $P_{threat}(x)$, to assign two distinct soft weights:

- **Risk Weight:** $w_k^R(x) = P_{threat}(x)$
- **Utility Weight:** $w_k^U(x) = 1 - P_{threat}(x)$

Those scores have a key role in the subsequent customized training.

2. Client-Side Dual Updates

In this method we go back to one singular training, the client performs local optimization to produce two distinct adapter updates from each sample and step. To achieve this, the standard language modeling loss $l(x; \Phi)$ (e.g., cross-entropy) for each sample x is explicitly scaled by the assigned soft weights. This defines two separate, risk-aware objective functions for the client k :

$$\mathcal{L}_k^U(\Phi) = \mathbb{E}_{x \sim D_k} [w_k^U(x) \cdot l(x; \Phi)] \quad (3.17)$$

$$\mathcal{L}_k^R(\Phi) = \mathbb{E}_{x \sim D_k} [w_k^R(x) \cdot l(x; \Phi)] \quad (3.18)$$

By minimizing these dual objectives starting from the current global parameters Φ_t , the client optimizes for safe capabilities and captures threat representations, obtaining two new sets of weights: Φ_k^U and Φ_k^R . The final updates transmitted to the server are the computed parameter deltas:

$$\Delta\Phi_k^U = \Phi_k^U - \Phi_t \quad \text{and} \quad \Delta\Phi_k^R = \Phi_k^R - \Phi_t \quad (3.19)$$

3. Global Direction Estimation

Once the clients transmit their dual updates, the server estimates the global trajectories for both safety and threat. Using a base aggregation weight a_k (e.g., proportional to the local token count), the server computes the Global Utility Direction (G^U) and the Global Risk Direction (G^R):

$$G^U = \sum_{k=1}^K a_k \Delta\Phi_k^U \quad \text{and} \quad G^R = \sum_{k=1}^K a_k \Delta\Phi_k^R \quad (3.20)$$

These aggregated vectors serve a critical geometric purpose for the subsequent phases of

the RAP-FL algorithm. G^R acts as the mathematical definition of the "threat axis" within the weight space, establishing the exact sub-space that must be neutralized. Conversely, G^U represents the global consensus on benign knowledge acquisition, acting as a structural anchor to prevent catastrophic forgetting and to evaluate which components of the threat updates might still contain salvageable general utility

4. Risk-Aware Projection and Utility Salvage

This phase represents the core of the RAP-FL algorithm. The geometric intuition behind this approach is based on the assumption that benign and hazardous capabilities may become entangled within the high-dimensional parameter space. The primary intention of this phase is to decouple these two concepts. To pursue this separation, the server utilizes orthogonal projections, aiming to neutralize any update vector component that aligns with the malicious parameter sub-space.

The server utilizes the Frobenius inner product $\langle A, B \rangle_F$ to measure the degree of alignment, or similarity, between two vectors. By leveraging this metric, we define a projection operator onto the global risk direction G^R . Geometrically, this operator evaluates how much of any given update V runs parallel to the threat axis, effectively extracting a "shadow" vector that isolates its hazardous contribution. To prevent mathematical instability, the projection is conditioned on the magnitude of the risk vector:

$$Proj_{G^R}(V) = \begin{cases} \frac{\langle V, G^R \rangle_F}{\|G^R\|_F^2} G^R & \text{if } \langle G^R, G^R \rangle_F > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.21)$$

To construct a risk-suppressed update, the server aims to remove any component of the local utility update that aligns with the global risk direction. This is achieved through a direct vector subtraction: the server calculates the projection of the safe update onto the threat axis and explicitly subtracts this extracted hazardous component from the original update. This algebraic operation forces the resulting utility update to become orthogonal to the risk direction, ensuring that the optimization of benign capabilities does not inadvertently drag the model toward malicious parameter sub-spaces:

$$\tilde{\Delta}\Phi_k^U = \Delta\Phi_k^U - \lambda Proj_{G^R}(\Delta\Phi_k^U) \quad (3.22)$$

where $\lambda \geq 0$ dictates the risk-suppression strength. For the final aggregation, we denote this cleaned utility update as $U_k = \tilde{\Delta}\Phi_k^U$.

Furthermore, RAP-FL attempts to extract ("salvage") benign knowledge structurally embedded within the threat updates. It projects out the risk components from $\Delta\Phi_k^R$ and retains only the residual vectors that exhibit a positive inner product with the Global Utility Direction G^U :

$$\tilde{\Delta}\Phi_k^R = \Delta\Phi_k^R - \mu Proj_{G^R}(\Delta\Phi_k^R) \quad (3.23)$$

$$R_k = \tilde{\Delta}\Phi_k^R \cdot \mathbf{1}[\langle \tilde{\Delta}\Phi_k^R, G^U \rangle_F > 0] \quad (3.24)$$

where $\mu \geq 0$ is the risk-suppression strength for threat data, and R_k represents the salvaged safe components.

5. Final Aggregation

Since our methodology deliberately bypasses any optional secondary filtering to maximize the retention of benign knowledge, this risk-suppressed update is directly assigned as the final utility contribution for the client:

$$U_k = \tilde{\Delta}\Phi_k^U \quad (3.25)$$

The final global PEFT update is formed by combining the risk-suppressed utility updates (U_k) and the salvaged safe components from the risk updates (R_k), scaled by a salvage coefficient γ :

$$\Delta\Phi_{final} = \sum_{k=1}^K a_k U_k + \gamma \sum_{k=1}^K a_k R_k \quad (3.26)$$

After computing this aggregated update, the central server updates the global PEFT parameters for the next communication round ($t + 1$) by adding the final delta to the current model weights (Φ_t):

$$\Phi_{t+1} = \Phi_t + \Delta\Phi_{final} \quad (3.27)$$

This projection-based mechanism aims to neutralizing threat vectors at the algebraic level while maximizing the retention of general linguistic capabilities.

Here following the sequential overview of the entire process, the complete procedural flow of our implementation is summarized in Algorithm 3.2.

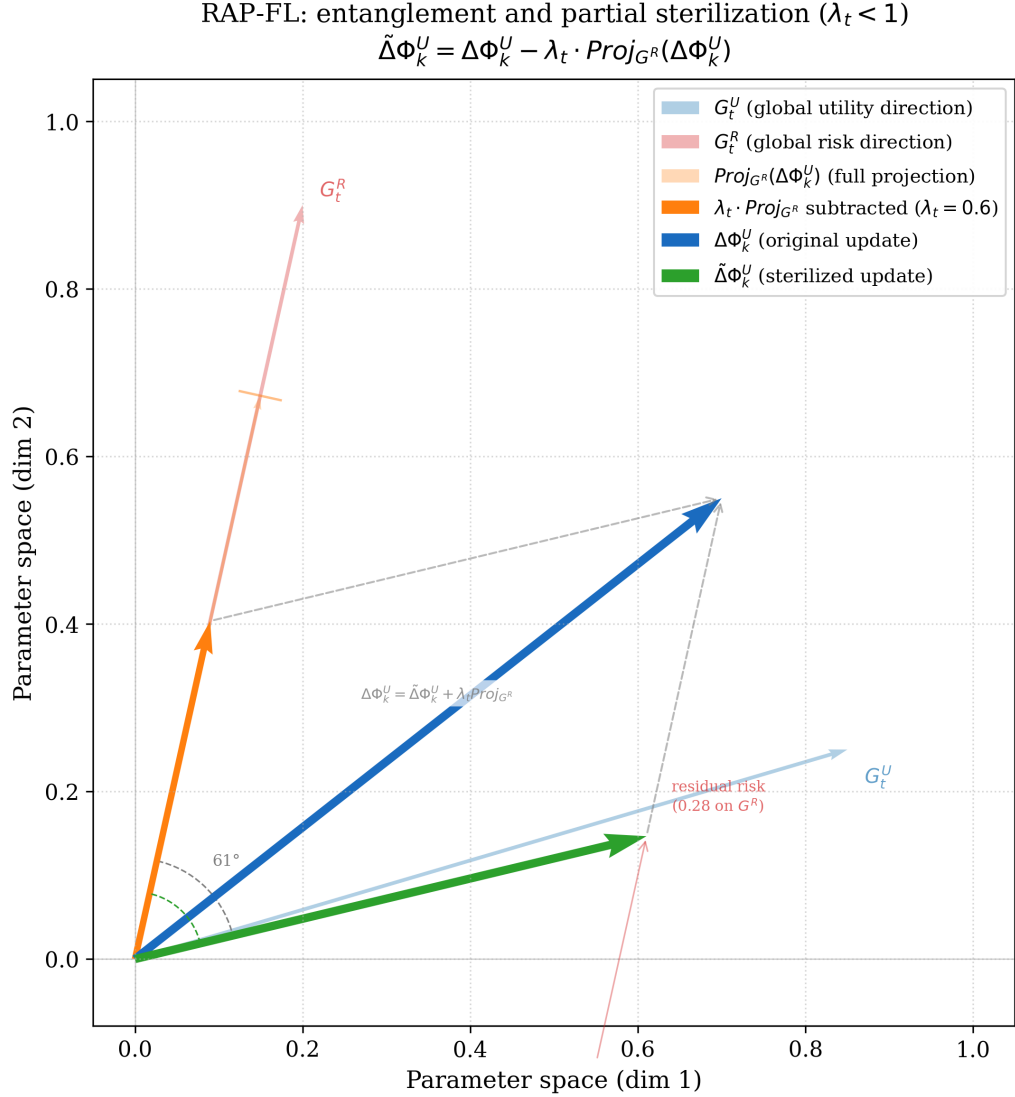


Figure 3.8: Geometric representation of partial update sterilization in RAP-FL. The plot illustrates the projection mechanics within a simplified 2D parameter space. The estimated global utility (G_t^U) and risk (G_t^R) directions form an acute angle, reflecting the *entanglement* between safe capabilities and vulnerabilities in the model's weights. The original local update $\Delta\Phi_k^U$ (in blue) is decomposed by computing its orthogonal projection along the risk axis (transparent orange vector). By subtracting only a fraction of this component, modulated by the suppression factor $\lambda_t = 0.6$ (*illustrative value*) (solid orange vector), the sterilized update $\tilde{\Delta}\Phi_k^U$ (in green) is obtained. As highlighted by the decomposition parallelogram, this operation "bends" the vector away from the threat. Since $\lambda_t < 1$, the system accepts a slight residual risk (red arrow) in order to preserve a strong alignment with the global utility direction.

Algorithm 3.2 Risk-Aware Projection Federated Learning (RAP-FL)

```

1: Initialize global PEFT parameters  $\Phi_0$ 
2: for  $t = 0$  to  $T - 1$  do
3:   Server samples clients  $\mathcal{S}_t$  and broadcasts  $\Phi_t$ 
4:   for all  $k \in \mathcal{S}_t$  in parallel do
5:     Client computes soft weights  $w_k^U(x) = 1 - P_{threat}(x)$  and  $w_k^R(x) = P_{threat}(x)$ 
6:     Client trains PEFT to obtain utility update  $\Delta\Phi_{k,t}^U$  minimizing  $\mathcal{L}_k^U$ 
7:     Client trains PEFT to obtain risk update  $\Delta\Phi_{k,t}^R$  minimizing  $\mathcal{L}_k^R$ 
8:     Client sends  $\Delta\Phi_{k,t}^U$  and  $\Delta\Phi_{k,t}^R$  to the server
9:   end for
10:  Server computes global directions:  $G^U = \sum_k a_{k,t} \Delta\Phi_{k,t}^U$  and  $G^R = \sum_k a_{k,t} \Delta\Phi_{k,t}^R$ 
11:  for all  $k \in \mathcal{S}_t$  do
12:     $\tilde{\Delta}\Phi_{k,t}^U = \Delta\Phi_{k,t}^U - \lambda_t Proj_{G^R}(\Delta\Phi_{k,t}^U)$ 
13:     $U_{k,t} = \tilde{\Delta}\Phi_{k,t}^U$  ▷ Utility gate deliberately bypassed
14:     $\tilde{\Delta}\Phi_{k,t}^R = \Delta\Phi_{k,t}^R - \mu_t Proj_{G^U}(\Delta\Phi_{k,t}^R)$ 
15:     $R_{k,t} = \tilde{\Delta}\Phi_{k,t}^R \cdot \mathbf{1}[\langle \tilde{\Delta}\Phi_{k,t}^R, G^U \rangle_F > 0]$  ▷ Utility salvage gate
16:  end for
17:   $\Delta\Phi_{final} = \sum_k a_{k,t} U_{k,t} + \gamma_t \sum_k a_{k,t} R_{k,t}$ 
18:   $\Phi_{t+1} = \Phi_t + \Delta\Phi_{final}$ 
19: end for
20: return  $\Phi_T$ 

```

3.4. Phase 4: Evaluation

The final module evaluates the safety-aware methods by comparing every strategy head-to-head on the safety-utility trade-off: the two references fixed in the Federated Transition, standard *FedAvg* and *Data Selection* (FedAvg + data filtering), against the three safety-aware solutions developed, *Dynamic Weighted Aggregation*, *Task Vector Aggregation*, and *RAP-FL*.

The primary objective of the evaluation phase is to quantify the effectiveness of the proposed federated risk-mitigation methodology, taking Specifically, the assessment aims to verify the dual-objective of the system: successfully suppressing the model’s hazardous knowledge while simultaneously preserving its general-purpose capabilities and linguistic utility.

To achieve a standardized and reproducible assessment, we evaluate each method by the knowledge that remains *inside* the model, not by the text it produces. The reason is

that a genuinely safe model should *lose* hazardous facts, not merely learn to avoid stating them; we therefore probe its **latent knowledge** [5], the factual content encoded in the weights, through a normalized Multiple-Choice Question Answering (MCQA) framework, heavily inspired by the Massive Multitask Language Understanding (MMLU) benchmark methodology [9].

For a given question Q and a set of possible choices $\{C_1, C_2, \dots, C_k\}$, the evaluation pipeline concatenates each potential answer to the original prompt, forming a full sequence $S_j = Q \oplus C_j$ for each choice $j \in \{1, \dots, k\}$. Let $S_j = (t_1, t_2, \dots, t_N)$ be the tokenized representation of the concatenated sequence, where N is the total number of tokens. The probability of the sequence, according to the autoregressive language model, is given by the product of the conditional probabilities of each token:

$$P(S_j) = \prod_{i=1}^N P(t_i | t_1, \dots, t_{i-1}) \quad (3.28)$$

To prevent numerical underflow and improve computational stability, we compute the log-probability of the sequence. Specifically, we focus only on the log-probability of the tokens constituting the completion C_j , conditioned on the prompt Q . Let M be the number of tokens in the prompt Q , such that the tokens of C_j are indexed from $M + 1$ to N . The log-probability of the completion C_j given the prompt Q is calculated as follows:

$$\log P(C_j | Q) = \sum_{i=M+1}^N \log P(t_i | t_1, \dots, t_{i-1}) \quad (3.29)$$

Because choices can vary significantly in length, simply taking the sum of log-probabilities inherently biases the model toward shorter answers. To counteract this, we normalize the log-probability by the length of the completion (number of tokens $L = N - M$):

$$\text{Score}(C_j) = \frac{1}{L} \sum_{i=M+1}^N \log P(t_i | t_1, \dots, t_{i-1}) \quad (3.30)$$

The model's deterministic prediction, $\hat{y}$, is selected as the candidate choice that maximizes this normalized log-probability score

$$\hat{y} = \arg \max_{j \in \{1, \dots, k\}} \text{Score}(C_j) \quad (3.31)$$

In simpler terms the answer with highest overall probability is chosen. This approach provides a mathematically robust measure of the model’s internal belief distribution, mitigating the confounding effects of length bias and tokenization artifacts.

The evaluation process is strictly partitioned into two complementary dimensions:

- **Threat Evaluation:** Designed to assess the success of the risk-mitigation strategy. We utilize domain-specific proxy benchmarks (such as the WMDP dataset) that focus on sensitive biosecurity and chemical hazards. A successful unlearning or data-filtering process should severely degrade the model’s performance on these specific tasks, ideally driving it down to random chance levels.
- **Control Evaluation (Utility):** Designed to verify that the core competencies of the LLM remain intact. We employ standard evaluation datasets representing benign, general-purpose knowledge (e.g., general biology, computer science, or standard MMLU categories). The objective is to maintain a performance level that is as close to the original baseline model as possible, demonstrating that the safety interventions do not cause catastrophic forgetting of safe knowledge.

3.4.1. Metrics

The primary metric employed to evaluate both dimensions is **Accuracy**, defined as the proportion of correctly predicted answers out of the total number of evaluated questions:

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N_{\text{total}}} \quad (3.32)$$

where N_{correct} represents the number of instances where the model’s prediction $\hat{y}$ matches the ground truth, and N_{total} is the total number of questions in the evaluation set. In the context of our framework, success is indicated by a sharp, targeted decrease in Threat Accuracy paired with a stable, high Control Accuracy.

To gain deeper insights into the model’s internal certainty and the severity of its errors, we introduce two additional continuous metrics: **Confidence Margin** and **Prediction Loss**. Rather than relying on unnormalized log-probabilities, we first convert the raw sequence scores, denoted as S_j , into a valid probability distribution over the available choices using a Softmax normalization:

$$P(C_j) = \frac{e^{S_j}}{\sum_{i=1}^k e^{S_i}} \quad (3.33)$$

Let $P(C^{(1)})$ and $P(C^{(2)})$ represent the probabilities of the model's first (Top-1) and second (Top-2) most likely predictions, respectively, and let $P(C^{(target)})$ be the probability assigned to the ground-truth correct answer.

- **Confidence Margin:** This metric quantifies how "convinced" the model is of its best prediction by measuring the probability distance between its top two choices:

$$\text{Confidence Margin} = P(C^{(1)}) - P(C^{(2)}) \quad (3.34)$$

Analyzing shifts in this margin provides critical secondary evidence of unlearning. A marked decrease in the Confidence Margin on threat tasks indicates that the model's internal representations regarding hazardous concepts have been disrupted, introducing significant uncertainty even if the raw accuracy remains temporarily above random chance.

- **Prediction Loss:** This metric evaluates the magnitude of the model's error. If the model correctly predicts the target answer ($C^{(1)} = C^{(target)}$), the loss is strictly evaluated as 0. Otherwise, it is calculated as the probability gap between the incorrect Top-1 prediction and the actual Target:

$$\text{Loss} = \begin{cases} 0 & \text{if } C^{(1)} = C^{(target)} \\ P(C^{(1)}) - P(C^{(target)}) & \text{otherwise} \end{cases} \quad (3.35)$$

A higher loss on threat tasks signifies a highly successful unlearning intervention, demonstrating that the model not only selects the wrong answer but actively assigns a heavily degraded probability to the hazardous truth.

Finally, for both the Confidence Margin and the Prediction Loss, we aggregate the results over the entire evaluation dataset by calculating the **Mean** values.

4 | Experimental Setup

In this chapter, we present a detailed overview of the experimental setup and pipeline developed to collect the data required to address our research questions.

Datasets, code and details for reproducibility can be retrieved in this repo <https://github.com/LeoLe12/MASTER-THESIS-Federated-Learning-and-Safety>.

4.1. Technology Used

To conduct the experiments, train the models, and evaluate the risk-mitigation strategies, we relied on a robust software stack based on open-source libraries and frameworks. The experimental pipeline integrates standard machine learning tools with specialized libraries for LLMs and Federated Learning.

4.1.1. Programming Language and Deep Learning Frameworks

- **Python:** The entire experimental pipeline, including data preparation, model training, and evaluation, was developed using Python 3 (specifically version 3.11).
- **PyTorch:** We utilized PyTorch as the primary deep learning framework. It served as the foundational backend for all neural network operations, tensor manipulations, and gradient calculations, enabling hardware acceleration via CUDA. PyTorch was also used to handle the manual aggregation of parameters in our custom Federated Learning scripts.

4.1.2. LLM and Fine-Tuning Libraries

- **Hugging Face Transformers:** We extensively used the `transformers` library to load, instantiate, and manage both causal and sequence-classification language models. This library provided the `Trainer` API for supervised fine-tuning and the necessary tokenizers (`AutoTokenizer`) to process text inputs.
- **Hugging Face PEFT:** To make the training computationally feasible and to sim-

ulate edge-device adaptation, we used the Parameter-Efficient Fine-Tuning (`peft`) library. Specifically, we employed Low-Rank Adaptation (LoRA) through the `LoraConfig` and `get_peft_model` modules to train only a small subset of the model’s parameters.

- **Hugging Face Datasets:** The `datasets` library was used to efficiently convert pandas DataFrames and load remote datasets, preparing them for the PyTorch Trainer.
- **BitsAndBytes:** To further reduce VRAM consumption during generative model fine-tuning, we integrated the `bitsandbytes` library. We applied 4-bit quantization (using the NF4 quantization type) through the `BitsAndBytesConfig` module.

4.1.3. Data Processing and Evaluation Tools

- **Pandas and NumPy:** For dataset manipulation, loading CSV/Parquet files, and handling data structures, we utilized `pandas`. `NumPy` was employed for array manipulations, extracting argmax values, and calculating mathematical metrics.
- **Custom Benchmark and FL Scripts:** We developed custom Python modules to support our specific methodology. This includes a custom sequence log-probability evaluator for Multiple-Choice Question Answering tasks, and a custom Federated Averaging (`FedAvg`) implementation to aggregate LoRA weights across simulated clients.

4.1.4. Pre-Trained Models

To ensure methodological consistency and build directly upon established baselines in alignment safety, our experimental pipeline leverages two specialized architectures adapted from the *Deep Ignorance* framework [17]. Rather than training foundational layers from scratch, this choice allows our research to systematically evaluate decentralized unlearning and projection methodologies on verified, stable validation checkpoints.

- **Classifier Model (Encoder-Only):** For data labeling, semantic inspection, and simulated client-side threat detection, we integrated `ModernBERT-large`, specifically its downstream `fine-tuned variant` to classify whether a corpora of text can contain bio-hazardous knowledge or not. This modern encoder-only architecture features an extended context window of up to 8,192 tokens, making it optimal for digesting long scientific documents and text sequences. Within our framework, this model serves as the automated discriminator to evaluate user requests and soft-partition

training payloads into utility and risk dimensions.

- **Generative Target Model (Decoder-Only):** As the primary target for our safety interventions, decentralized updates, and alignment evaluations, we utilized [deep-ignorance-pretraining-stage-unfiltered](#). Sourced directly from the EleutherAI research suite, this 6.9B parameter decoder-only model represents a critical intermediate checkpoint trained on 500B unfiltered tokens. Because it fundamentally contains dual-use proxy knowledge before the introduction of post-training safeguards or dataset annealing, it provides the ideal, decontaminated baseline to test parameter-efficient fine-tuning (PEFT/LoRA) and risk-aware global aggregations.

4.2. Dataset and Benchmark Selection

To ensure strict experimental consistency and a rigorous comparative analysis, the datasets selected for this study are directly inherited from the foundational *Deep Ignorance* research. These corpora are systematically categorized into two distinct functional groups within our federated pipeline: **Training Datasets**, which are localized on client nodes to drive the fine-tuning process, and **Benchmark Datasets**, which are held out externally to evaluate the global model’s performance post-aggregation.

4.2.1. Training Datasets

In the context of Safe Federated Fine-Tuning, the choice of data strictly dictates the validity of both the training dynamics and the final evaluation. Because Parameter-Efficient Fine-Tuning naturally operates on smaller amounts of data, the quality and representativeness of the local datasets heavily dictate the overall success.

Training datasets are selected to simulate dual-use scientific domains. We utilize two specialized text corpora created and shared by the [CAMEL AI](#) community, an open-source community focused on autonomous and communicative agents.

- [Camel Chemistry](#): A dense dataset covering chemical formulations, procedural descriptions, and molecular theory. It encompasses both benign foundational chemistry and advanced synthetic pathways that present significant dual-use risks.
- [Camel Biology](#): A corpus focused on molecular biology, genetics, and cellular systems, embedding sophisticated biological concepts that could potentially be exploited in hazardous applications.

This choice was motivated by the need to preserve comparability with the original Deep

Ignorance framework while remaining aligned with the constraints of PEFT-based training. Since both Camel Chemistry and Camel Biology were used in the original study, they provide a consistent experimental foundation. At the same time, their moderate size make them well suited for efficient domain adaptation in federated PEFT settings.

Training Dataset building

The Camel Datasets were the source of our Training Dataset, here we explain how we built the set, the process we went through and its statistics. Each dataset originally consists of 20,000 instructional samples which are reduced just slightly upon dropping the duplicates.

The raw data was subjected to a specific formatting pipeline to optimize it for causal language modeling. Each sample was structured by concatenating the `topic`, `sub_topic`, and conversational messages into a unified text block. This structural formatting helps the model distinguish between conceptual categories during the fine-tuning phase.

Once formatted, every sample was evaluated by our local Risk Estimator, the fine-tuned ModernBERT-large model. The classifier processed each text block, outputting a softmax probability distribution. Specifically, for each sample, we extracted:

- **Probability Class 0 (P_{safe}):** The likelihood that the text contains benign, safe knowledge.
- **Probability Class 1 (P_{threat}):** The likelihood that the text encodes hazardous or dual-use information.

These soft labels were added to the data set and serve as the foundational weights (w^U and w^R).

Based on these confidence scores, we established a strict safety threshold of $\tau = 0.8$. Samples scoring $P_{threat} \geq 0.8$ were classified as "Threat", while those scoring below were classified as "Safe". This categorization partitioned the available raw data into distinct semantic pools, ready for the construction of our specific training distributions.

During this phase, we observed a natural disparity in the distribution of hazardous knowledge across domains. While the biology corpus contained an abundance of high-confidence threat samples (over 9,900), the chemistry corpus yielded a maximum of 3,391 high-confidence threat samples. To maximize data utilization while preventing domain distribution from becoming a confounding variable across our experiments, we fixed the chemistry subset size to 3,390 samples and the biology subset size to 9,924 samples for all experimental setups.

Construction of the Experimental Paradigms

We constructed three distinct training sets of identical overall size (13,314 samples) and identical domain composition (approximately 74.5% Biology and 25.5% Chemistry). The three datasets, which perfectly mirror the centralized experimental paradigms established in Chapter 3, were assembled as follows:

- **Threat Dataset:** Composed entirely (100%) of samples extracted from the high-confidence threat pools. This serves as our reference for maximum vulnerability.
- **Safe Dataset:** Composed entirely (100%) of samples extracted from the safe pools. This serves as our reference for maximum safety-robustness.
- **Mixed Dataset:** Constructed to simulate a standard, unfiltered environment. It contains an exact, unweighted 50/50 split between safe and threat samples (6,657 samples each). Importantly, this 50/50 split is maintained proportionally within the domains themselves (e.g., 4,962 Bio Safe and 4,962 Bio Threat, alongside 1,695 Chem Safe and 1,695 Chem Threat).

Dataset	Total Size	Bio / Chem Split	Safe Samples	Threat Samples
Threat	13,314	9,924 / 3,390	0 (0%)	13,314 (100%)
Safe	13,314	9,924 / 3,390	13,314 (100%)	0 (0%)
Mixed	13,314	9,924 / 3,390	6,657 (50%)	6,657 (50%)

Table 4.1: Summary of the training datasets composition built for the experiments. The domain split is consistently maintained across all paradigms to isolate the specific effects of the safety interventions.

Furthermore, to accurately gauge the computational requirements and linguistic density of our specific training distributions, we performed a token-level analysis on the constructed datasets. Using the tokenizer associated with our target base model, we computed the total and average sequence lengths for each experimental paradigm.

Dataset	Total Samples	Total Tokens	Average Tokens / Sample
Safe Dataset	13,314	7,339,611	551.27
Mixed Dataset	13,314	7,764,230	583.16
Threat Dataset	13,314	8,216,727	617.15

Table 4.2: Token-level statistics for the three constructed experimental datasets. Calculations are based on the tokenizer of the target generative model.

4.2.2. Benchmark Datasets

To quantify the impact of our safety-aware strategies without compromising privacy or inspecting local data, we leverage two standardized evaluation benchmarks that perfectly align with the training domains:

- **Weapon Matter Declassification Project (WMDP)** [15]: This benchmark serves as our primary security and safety metric. It is explicitly designed to measure hazardous, weaponizable knowledge across biosecurity, chemical safety, and cybersecurity. A lower score on WMDP indicates a successful suppression of dangerous capabilities.
- **Massive Multitask Language Understanding (MMLU)** [9]: Specifically focusing on subjects related to biology and chemistry (such as MMLU-Bio). This benchmark serves as our primary utility metric, allowing us to monitor whether our aggressive safety interventions cause catastrophic forgetting of benign, legitimate scientific education.

By testing the aggregated global model across both the WMDP and MMLU frameworks, we can systematically plot the safety-utility trade-off introduced by each defensive strategy.

4.3. Centralized Verification Baseline Configurations

Before transposing the continuous training process into a distributed, federated setup, we aim to adapt the work to a LoRA setup. These baselines serve a dual purpose: first, they validate the capability of Parameter-Efficient Fine-Tuning (PEFT) to adapt the target architecture under computational constraints; second, they provide a quantitative reference point to measure the exact trade-off between safety mitigation and utility retention before introducing the architectural complexities of federated orchestration.

To ensure a perfectly controlled environment, all centralized experiments share an identical hardware abstraction, architectural quantization, and optimization pipeline. This shared infrastructure isolates the data composition as the sole independent variable across the baseline runs.

4.3.1. Architectural and Optimization Hyperparameters

As established, all evaluations target our previously defined 6.9B parameter generative model. Given the immense memory footprint required for full-parameter backpropagation at this scale, we implement a unified **QLoRA (Quantized Low-Rank Adaptation)** regime across all configuration scripts.

- Model Quantization:** To fit the model within consumer-grade VRAM constraints, the base model parameters are completely frozen and compressed into a 4-bit NormalFloat (NF4) data type via the `bitsandbytes` library. Double quantization is systematically enabled (`bnb_4bit_use_double_quant=True`) to minimize placeholder overhead by quantizing the quantization constants themselves. Crucially, to ensure numerical stability and prevent mathematical underflow during backpropagation, the compute data type for activation gradients is set to standard 16-bit Floating Point (`float16`), forcing the system to cast parameters to 16-bit precision exclusively during the forward and backward execution passes.
- LoRA Configuration:** Trainable low-rank adapter matrices are injected into all linear layers of the Transformer architecture. This comprehensive targeting encompasses both the attention projection matrices (`q_proj`, `k_proj`, `v_proj`, `o_proj`) and the multi-layer perceptron (MLP) blocks (`gate_proj`, `up_proj`, `down_proj`). This highly restrictive parameter sub-space isolates the newly acquired configurations into exactly 33,554,432 trainable parameters, bounding the optimization process to just 0.4870% of the global network (6,889,807,872 total parameters). The exact PEFT hyperparameters applied across all runs are detailed in Table 4.3.

Hyperparameter	Value
Intrinsic Rank (r)	16
Scaling Factor (α)	32
Dropout Probability	0.05
Target Modules	<code>all-linear</code>
Trainable Parameters	33,554,432
Total Parameters	6,889,807,872
Trainable Parameter Ratio	0.4870%

Table 4.3: LoRA and PEFT hyperparameters applied to the target generative model.

- **Training Configurations:** The optimization process is orchestrated utilizing the Hugging Face `Trainer` API. To ensure stable convergence while strictly adhering to memory constraints, we employ 16-bit mixed precision (`fp16`) coupled with gradient checkpointing. The optimization is driven by the `paged_adamw_8bit` optimizer with a base learning rate of 5×10^{-5} , a weight decay of 0.03, and gradient clipping set to a maximum norm of 0.5.

The computational workload is distributed using a per-device batch size of 4 and 8 gradient accumulation steps, resulting in an effective global batch size of 32. A `cosine` learning rate scheduler is deployed with an initial warmup ratio of 0.03. While the theoretical maximum number of training epochs is set to 10, we implement an aggressive, high-frequency evaluation strategy: the model is evaluated and checkpointed every 47 steps. This step-based strategy, paired with custom stopping callbacks, ensures that the system automatically retrieves the highest-performing weights based on the minimum validation loss (`eval_loss`) at the end of the run, effectively preventing catastrophic overfitting. The complete set of hyperparameters is summarized in Table 4.4.

Hyperparameter	Value
Optimizer	<code>paged_adamw_8bit</code>
Base Learning Rate	5×10^{-5}
LR Scheduler	<code>cosine</code> (Warmup ratio: 0.03)
Weight Decay	0.03
Max Gradient Norm	0.5
Max Epochs	10
Per-Device Batch Size	4
Gradient Accumulation Steps	8
Effective Global Batch Size	32
Precision	<code>fp16</code> (Mixed Precision)
Evaluation & Save Strategy	Every 47 steps
Best Model Selection Metric	<code>eval_loss</code>
Gradient Checkpointing	True

Table 4.4: Training and optimization hyperparameters applied across all centralized baseline experiments.

4.3.2. Centralized Experimental Paradigms

By maintaining the exact technical hyperparameters detailed above, we executed three distinct centralized training paradigms, differentiated exclusively by the risk-profile of the processed datasets:

- **Centralized Safe Baseline:** This run optimizes the model exclusively on the *Safe Dataset* (13,314 samples, where $P_{threat} < 0.8$). It serves as our upper-bound control for safety alignment, designed to monitor whether continuous fine-tuning on purely benign text causes a degradation of general reasoning capabilities (*catastrophic forgetting*) or safely expands scientific domain competence.
- **Centralized Threat Baseline:** This run exposes the model exclusively to the *Threat Dataset* (13,314 samples, where $P_{threat} \geq 0.8$). Acting as our maximum vulnerability anchor, this baseline maps the worst-case scenario, demonstrating how rapidly the target LLM absorbs weaponizable and hazardous concepts when unconstrained by any safety-aware data selection or algorithmic defense.
- **Centralized Mixed Baseline:** This paradigm leverages the *Mixed Dataset* (13,314 samples containing an unweighted 50/50 safe-to-threat ratio). It simulates a stan-

dard, unfiltered continuous pre-training process where benign utility and hazardous dual-use knowledge coexist. This setup acts as the primary control baseline to evaluate the natural evolution and interference of both capabilities within a single, centralized weight space.

4.4. Federated Learning Simulation Setup

This section details the distributed infrastructure developed to evaluate safe federated fine-tuning.

Transitioning from a centralized environment to a Federated Learning (FL) paradigm introduces critical orchestration challenges, such as weight synchronization, local-to-global optimization loops, and the inherent constraint of a "blind server" unable to inspect client data payloads.

This section outlines the shared network architecture, the data partitioning methodologies, and the specific configurations applied to both baseline protocols and advanced weight-space defensive solutions.

4.4.1. Network Architecture and Client Partitioning

To evaluate the distributed learning dynamics, the simulation models a collaborative network consisting of a central orchestrator (server) and $K = 2$ distinct participating clients. This choice of $K = 2$ represents the minimal viable federated architecture. It was dictated by the severe computational overhead and VRAM limitations associated with simulating concurrent Large Language Model optimization processes on a centralized hardware setup.

Each client performs local training on the identical generative base architecture employed throughout the centralized experiments.

- **Data Splitting:** the original global datasets (13,314 samples each) are explicitly partitioned prior to distribution. A structured **Global Validation Set** consisting of 1,282 samples is held out externally at the server level. The remaining 12,032 training samples are then perfectly bifurcated into two disjoint, equal halves. Each half is assigned locally to one of the two clients as **Local Training Set**, ensuring that both nodes possess an identical local training size of 6,016 samples..

4.4.2. Local Optimization and Global Synchronization Loops

The temporal progression of our federated framework is governed by discrete, sequential communication rounds. At each round $t \in \{0, 1, \dots, T - 1\}$, the synchronization loop operates through the following unified parameters:

- **Global Orchestration:** The training is scheduled over a fixed number of global communication rounds T . In our experiments, for methods M0, M1, M2, and M3, T is set to 20. For M4, having a different step synchronization, T is set to 940 instead. At the beginning of each round, the central server broadcasts the current global adapter configuration Φ_t to all active clients.
- **Local Fine-Tuning and Step Synchronization:** Upon receiving the global weights, each client initializes its local adapter parameters ($\Phi_{k,t} \leftarrow \Phi_t$) and executes local optimization. Rather than synchronizing at the boundary of a full epoch, the orchestrator enforces a high-frequency synchronization interval fixed at exactly 47 **local steps**. Given that each client processes a training subset of 6,016 samples with an effective batch size of 32 (a per-device batch size of 4 combined with 8 gradient accumulation steps), a complete local epoch consists of exactly 188 optimization steps ($6,016/32 = 188$). Therefore, the decision to aggregate every 47 steps mathematically corresponds to exactly one-quarter ($1/4$) of a local epoch. This fine-grained synchronization aims to avoid excessive local client drift and provides the server with highly reactive parameter trajectories. M4 differs again as mentioned previously and aggregates every 1 step. Detailed explanation are presented in 4.4.4 Combining with the number of rounds this yields to a total of **940 local steps** per client for every method.
- **Weight-Space Payload Upload:** At the conclusion of the local training phase, clients calculate their parameter updates (deltas) or compile their complete state dictionaries, transmitting these lightweight weight-space payloads back to the central server for aggregation.

4.4.3. Federated Experimental Scenarios

To rigorously evaluate the robustness of our defensive aggregation strategies across varying network conditions, we defined three distinct experimental scenarios. These setups manipulate the proportion of Safe and Threat data assigned to the two participating clients, allowing us to simulate different degrees of network vulnerability and data heterogeneity. Given that each client’s local training set is fixed at 6,016 samples, the scenarios

are structured as follows:

- **Scenario 1: Extreme Asymmetry (Limit Case):** In this setup, the data distribution is entirely polarized regarding safety. Client 1 is trained exclusively on Safe data (100%), while Client 2 is trained exclusively on Threat data (100%). This scenario represents a boundary condition where data types are perfectly homogeneous within individual clients but entirely heterogeneous across the network. It simulates a clean, controlled environment where a purely benign node is mathematically contrasted against an entirely compromised or "rogue" node.
- **Scenario 2: Symmetric Entanglement:** Both Client 1 and Client 2 are assigned an exact 50/50 split of Safe and Threat samples. This scenario represents a standard baseline for inadvertently exposed networks, where hazardous and benign knowledge are uniformly entangled across all participating edge nodes. It allows us to observe how the global model behaves when all clients contribute equally to both the utility and the risk objectives.
- **Scenario 3: Realistic Heterogeneity:** This setup introduces a more realistic, non-uniform distribution of risk across the network. Client 1 processes a predominantly benign dataset (60% Safe, 40% Threat), whereas Client 2 operates on a heavily compromised dataset (30% Safe, 70% Threat). This asymmetry mimics real-world federated deployments (e.g., distinct research labs or hospital networks) where different nodes have varying degrees of exposure to dual-use or hazardous knowledge, testing the aggregation algorithm's resilience against skewed data distributions.

To provide a clear and immediate overview of the data partitioning across these scenarios, the exact sample distributions and their corresponding percentages are detailed in Table 4.5.

Scenario	Node	Safe Samples	Threat Samples	Total
1. Extreme Asymmetry	Client 1	6,016 (100%)	0 (0%)	6,016
	Client 2	0 (0%)	6,016 (100%)	6,016
2. Symmetric Entanglement	Client 1	3,008 (50%)	3,008 (50%)	6,016
	Client 2	3,008 (50%)	3,008 (50%)	6,016
3. Realistic Heterogeneity	Client 1	3,609 ($\sim 60\%$)	2,407 ($\sim 40\%$)	6,016
	Client 2	1,804 ($\sim 30\%$)	4,212 ($\sim 70\%$)	6,016

Table 4.5: Distribution of Safe and Threat samples across the two federated clients for each experimental scenario. The total local dataset size remains strictly fixed at 6,016 samples per client to ensure uniform volume-based aggregation weights.

4.4.4. Algorithmic Implementations of Defensive Strategies

Within this synchronized loop and across the defined experimental scenarios, the server-side aggregation represents the primary variable of our research. Depending on the active experimental setup, the system transitions from standard averaging to sophisticated vector manipulations. The operational settings and implementation details for each evaluated method are structured as follows:

Method 0: Standard Federated Averaging (FedAvg Baseline)

As theoretically established in our Methodology (Chapter 3), Method 0 represents the absolute experimental baseline, executing standard **Federated Averaging (FedAvg)** without any active risk mitigation or safety-filtering logic. From an implementation perspective, this orchestrates a "blind" aggregation where the server treats all incoming adapter updates identically, regardless of their underlying threat exposure.

In our simulated environment, the server aggregates the lightweight LoRA payloads (the rank-decomposition matrices A and B) transmitted by the clients. Given that the local dataset sizes are uniformly fixed at exactly 6,016 samples for both clients across Scenarios 1 and 2, the standard token-weighted average algebraically simplifies to an unweighted arithmetic mean. The core aggregation logic is implemented on the server by loading the `safetensors` state dictionaries and computing the element-wise midpoint for every specific key in the parameter space:

$$w_{avg}[key] = \frac{w_1[key] + w_2[key]}{2.0}$$

Implementation Caveat regarding LoRA Aggregation It is important to highlight an inherent mathematical approximation introduced by direct parameter-space aggregation in federated LoRA fine-tuning. In standard full-parameter federated learning, averaging gradients or model parameters yields an exact weighted aggregation in parameter space. However, in our PEFT setting, each client optimizes low-rank adapter matrices such that $\Delta W = BA$.

Consequently, the element-wise average of the decomposed matrices does not generally correspond to the average of the reconstructed updates in the original parameter space. Specifically:

$$\frac{B_1 A_1 + B_2 A_2}{2} \neq \left(\frac{B_1 + B_2}{2} \right) \left(\frac{A_1 + A_2}{2} \right)$$

Recent studies, such as FedEx-LoRA [23], propose exact aggregation strategies based on reconstruction and singular value decomposition (SVD) re-projection of the averaged updates. While mathematically more accurate, these approaches introduce significant computational overhead at the server side.

Therefore, consistent with recent FedPEFT literature [25], our implementation adopts direct element-wise averaging of the LoRA tensors A and B . Although this introduces an intrinsic structural approximation in the aggregation process, it remains highly communication and memory efficient for edge-device federated settings and has been empirically shown to preserve stable convergence and effective parameter interpolation during continuous fine-tuning. [2]

Method 1: Federated Averaging with Local Safety Filtering

As conceptually introduced in Chapter 3, Method 1 serves as our data-level safety baseline. This strategy transposes the strict "data-dropping" paradigm of the centralized *Deep Ignorance* framework directly to the federated edge nodes. Prior to initiating the local optimization loop, each client autonomously inspects its local dataset using the ModernBERT risk evaluator and enforces a strict exclusion policy.

In terms of software implementation, this translates to a hard boolean filter applied to the local dataframes, permanently discarding any sample whose threat probability exceeds the threshold ($P_{threat} \geq 0.8$, corresponding to `final_label == 1`). The clients subsequently

perform local LoRA fine-tuning exclusively on the surviving benign samples (`final_label == 0`). Finally, the central server aggregates the resulting weight updates using standard proportional Federated Averaging ($\Delta\Phi_{global} = \sum \frac{n_k}{N} \Delta\Phi_k$).

This pre-training filtration drastically alters the data distribution and learning dynamics of the simulated network. As established in Section 4.4.1, the network begins with a perfectly balanced data allocation, where both clients hold exactly 6,016 samples (yielding an equal 50/50 aggregation weight). However, because hazardous knowledge is heterogeneously distributed across the scenarios, dropping the threat data dynamically breaks this initial volume balance. This asymmetric data reduction forces the FedAvg algorithm to shift the aggregation weights based on the surviving benign samples, resulting in unique computational consequences for each experimental setup:

- **Impact on Scenario 1 (Extreme Asymmetry):** Client 1, possessing only Safe data, retains all 6,016 samples. Conversely, Client 2, possessing only Threat data, drops 100% of its dataset, leaving it with 0 actionable samples. Consequently, Client 2 is entirely excluded from the training round. The aggregation weight of Client 1 becomes 1.0 (6,016/6,016), effectively collapsing the federated architecture into a single-node centralized training loop.
- **Impact on Scenario 2 (Symmetric Entanglement):** Both clients perfectly drop half of their datasets, retaining 3,008 samples each. While the network remains perfectly balanced (each client contributing exactly 50% to the global FedAvg), the overall volume of data processed by the network is halved, potentially impacting the acquisition rate of benign scientific utility.
- **Impact on Scenario 3 (Realistic Heterogeneity):** This scenario demonstrates the most complex dynamic. Client 1 drops 40% of its data (retaining 3,609 samples), while Client 2 drops 70% (retaining 1,804 samples). The total surviving data across the network drops to $N = 5,413$. As a result, the standard FedAvg algorithm autonomously shifts the aggregation weights: the update from Client 1 now dominates the global model with an influence of approximately 66.7%, while Client 2's contribution shrinks to 33.3%.

Table 4.6 provides a comprehensive quantitative summary of the post-filtration dataset sizes and the resulting global aggregation weights applied by the server during Method 1.

Scenario	Node	Pre-Filter Size	Post-Filter Size	Aggregation Weight ($\frac{n_k}{N}$)
1. Extreme Asymmetry	Client 1	6,016	6,016	100.0%
	Client 2	6,016	0	0.0% (Excluded)
2. Symmetric	Client 1	6,016	3,008	50.0%
	Client 2	6,016	3,008	50.0%
3. Realistic	Client 1	6,016	3,609	66.7%
	Client 2	6,016	1,804	33.3%

Table 4.6: Impact of local safety filtering (Method 1) on client dataset sizes and subsequent FedAvg aggregation weights. Discarding threat samples dynamically alters the influence each client has on the global model update.

Method 2: Dynamic Weighted Aggregation

Unlike Method 1, which actively discards malicious data, Method 2 executes a standard local federated training loop across the entirety of the local datasets. In all experimental scenarios, both clients process their full allocation of 6,016 samples, preserving the original volume balance of the network. As theoretically formulated in Chapter 3, the safety intervention in this method is purely algebraic and operates at a batch-level granularity: clients compute a dynamic confidence scalar ($w_{k,t}$) for each communication round based strictly on the internal composition of the mini-batches processed during the preceding 47 local optimization steps.

During these 47 steps (amounting to 1,504 processed samples per round), the client evaluates the exact Safe/Threat ratio sampled by the dataloader, applying an asymmetric scoring system (0.8 for Safe, 0.2 for Threat). Because the dataloader shuffles the training set, the transmitted weight is stochastic and fluctuates per round, gravitating towards an expected mathematical value ($\mathbb{E}[w_k]$) dictated by the global local distribution.

Translating this batch-level formulation across our three defined scenarios yields the following dynamic behaviors:

- **Scenario 1 (Extreme Asymmetry):** Because the data distribution is entirely homogeneous within the individual clients, every sampled mini-batch is entirely pure. Consequently, the dynamic equation collapses into fixed bounds. At every round, Client 1 transmits a static weight of $w_1 = 0.8$, while the rogue Client 2 transmits $w_2 = 0.2$.
- **Scenario 2 (Symmetric Entanglement):** With an exact 50/50 internal split on both nodes, the mini-batch composition fluctuates per round. The transmitted

weights $w_{1,t}$ and $w_{2,t}$ dynamically oscillate based on the sampling, but mathematically converge to an expected value of $\mathbb{E}[w] = 0.5$.

- **Scenario 3 (Realistic Heterogeneity):** Here, the batch-level volatility fully exercises its dynamic capability. The weights shift at every synchronization step: Client 1 (predominantly Safe) oscillates around a robust expected weight of $\mathbb{E}[w_1] \approx 0.56$, whereas Client 2 (predominantly Threat) is dynamically suppressed, fluctuating around $\mathbb{E}[w_2] \approx 0.38$.

Upon receiving these per-round scalars alongside the LoRA adapters, the central server executes the direct linear combination: $w_{avg}[key] = w_{1,t} \cdot w_1[key] + w_{2,t} \cdot w_2[key]$. As designed, the server intentionally skips the mathematical renormalization step (i.e., it does not divide by $w_{1,t} + w_{2,t}$).

Method 3: Federated Task Arithmetic (Vector Negation)

As formally defined in Chapter 3, Method 3 shifts the defensive mechanism from the data dimension to the parameter space. This approach leverages Task Arithmetic to actively isolate and surgically suppress hazardous representations through vector negation. The implementation of this strategy requires a structural modification to both the local client optimization and the global server aggregation loops.

Client-Side Dual Optimization and Asymmetric Hyperparameters: Prior to training, each client evaluates its local dataset using the safety classifier. Instead of discarding the threat data, the client strictly bifurcates its dataset into a *Safe Pool* ($P_{threat} < 0.8$) and a *Threat Pool* ($P_{threat} \geq 0.8$). The volumetric distribution of these dual training pools across the experimental scenarios strictly follows the data proportions previously detailed in Section 4.4. In boundary conditions (such as Scenario 1), the dual optimization dynamically skips empty pools to conserve computational resources.

The client then executes two independent Parameter-Efficient Fine-Tuning runs on the frozen base model to optimize a benign adapter ($\Phi_{safe,k}$) and a separate, isolated adapter explicitly capturing the malicious trajectory ($\Phi_{threat,k}$). To maintain strict temporal alignment with the baseline synchronization interval (47 local steps per client), the computational budget is dynamically split between the two sub-tasks (alternating 23 and 24 steps per round).

Crucially, as demonstrated in [12] task vector extraction are obtained with parameters setting lower than regular fine-tuning, so we implemented a highly specialized **asymmetric hyperparameter configuration** for the dual training loops:

- **Safe Branch:** Optimized using the standard configuration established in the centralized baselines, featuring a Learning Rate of 5×10^{-5} and a Weight Decay of 0.03. This ensures stable and reliable utility acquisition.
- **Threat Branch:** Configured with a strictly penalized Learning Rate of 1×10^{-5} and an aggressive Weight Decay of 0.1. This asymmetry is a critical design choice: the significantly lower learning rate prevents gradient explosion and catastrophic distortion of the base parameters when mapping malicious behaviors. Simultaneously, the high weight decay forces parameter sparsity (disentanglement), ensuring that the resulting threat adapter isolates only the most critical malicious parameter directions for cleaner downstream surgery.

Server-Side Vector Surgery: Once the dual adapters are uploaded, the central orchestrator processes them through a two-tier aggregation protocol:

- **Tier 1 (Independent Averaging):** The server performs standard proportional FedAvg independently on the two branches, yielding a Global Safe Adapter (Φ_{safe_global}) and a Global Threat Adapter (Φ_{threat_global}).
- **Tier 2 (Surgical Task Vector Aggregation):** To prevent algebraically induced catastrophic forgetting, the server extracts the isolated threat task vector for the current round. As implemented in our pipeline, this is achieved by subtracting the prior global state (Φ_{prev}) from the newly aggregated threat adapter: $\Delta\Phi_{threat} = \Phi_{threat_global} - \Phi_{prev}$. (During the initial round, where no prior state exists, the system relies directly on the threat adapter). Finally, the server applies the Task Arithmetic negation by subtracting this isolated threat vector from the global safe adapter.

Based on empirical evaluation, the steering coefficient was fixed with base $\alpha = 0.5$ across all experimental runs. Such value is kept as is in scenario 1 and 2, but in scenario 3 the exact algebraic formulation implemented in the aggregation script is defined as:

$$\alpha_{dynamic} = \alpha_{base} \cdot \left(\frac{\sum_{k=1}^K N_{threat,k}}{\sum_{k=1}^K N_{safe,k}} \right)$$

To prevent model collapse or catastrophic parameter distortion in extremely skewed rounds, a safety clipping limit is enforced at the code level ($\alpha_{dynamic} \leq 1.2$). This dynamic approach allows the server to adapt its defensive magnitude directly to the actual threat exposure of the federated network:

- **Scenarios 1 and 2:** Due to the perfectly balanced global data volume (6,016

Safe vs. 6,016 Threat samples across the network), the risk ratio is exactly 1.0. Consequently, the formula mathematically collapses to the baseline $\alpha = 0.5$.

- **Scenario 3 (Realistic Heterogeneity):** The network processes a total of 5,413 Safe samples and 6,619 Threat samples. The elevated risk ratio (~ 1.22) automatically scales the steering coefficient up to $\alpha \approx 0.61$, providing a proportionally stronger destructive interference against the dominant malicious update.

The final algebraic formulation implemented in the aggregation script is thus defined as:

$$\Phi_{new} = \Phi_{safe_global} - \alpha_{dynamic} \cdot (\Phi_{threat_global} - \Phi_{prev})$$

This surgical, risk-aware intervention induces destructive interference specifically along the malicious parameter directions. By dynamically adjusting the negation intensity, the system effectively neutralizes the threat while safely preserving the foundational knowledge anchored by the benign updates.

Method 4: Risk-Aware Projection for Federated Learning (RAP-FL)

The experimental implementation of the **RAP-FL** algorithm strictly follows the theoretical framework detailed in Chapter 3. While the fundamental geometric projections are handled at the server level, the most significant experimental challenge involved adapting the client-side infrastructure to support simultaneous dual-objective optimization efficiently.

Client-Side Single-Pass Dual Optimization: Unlike Method 3, which requires executing two completely separate fine-tuning runs (doubling the computational overhead), RAP-FL implements an elegant *single-pass* strategy. Every client k relies on the local risk classifier $C_k(x) \in [0, 1]$ to evaluate the threat level of each sample x . For every sample within a given mini-batch B , two complementary continuous weights are dynamically defined:

$$\begin{aligned} w_k^U(x) &= 1 - C_k(x) \\ w_k^R(x) &= C_k(x) \end{aligned}$$

Let $\ell(x; \theta_0, \Phi)$ represent the standard language modeling loss (e.g., autoregressive negative log-likelihood) evaluated on the trainable LoRA parameters Φ while keeping the base model θ_0 frozen. Instead of optimizing the standard average loss, the client must

accumulate two distinct gradients simultaneously for the same mini-batch:

$$g_k^U = \sum_{x \in B} w_k^U(x) \nabla_{\Phi} \ell(x; \theta_0, \Phi)$$

$$g_k^R = \sum_{x \in B} w_k^R(x) \nabla_{\Phi} \ell(x; \theta_0, \Phi)$$

PyTorch Implementation (Linearity of the Derivative): From a software engineering perspective, computing these dual gradients efficiently poses a memory and execution bottleneck. However, by exploiting the mathematical linearity of the derivative, where $\sum [w_k(x) \nabla \ell(x)] = \nabla \sum [w_k(x) \ell(x)]$, we optimized the pipeline natively within PyTorch’s computational graph.

The unreduced per-sample loss is explicitly multiplied by w^U and w^R to create two distinct scalar objectives ($\mathcal{L}^U$ and $\mathcal{L}^R$). The client then invokes two separate backward passes: the first retains the computational graph (`lossU.backward(retain_graph=True)`), and the second completes the cycle (`lossR.backward()`). The local optimizers process these dual gradients to yield two distinct sets of updated weights ($\Phi_{k,t}^U$ and $\Phi_{k,t}^R$). Finally, the explicit parameter deltas transmitted to the server are derived as:

$$\Delta_{k,t}^U = \Phi_{k,t}^U - \Phi_t$$

$$\Delta_{k,t}^R = \Phi_{k,t}^R - \Phi_t$$

Step-by-Step Federated Synchronization: This customized single-pass architecture necessitates a critical divergence in communication frequency: transitioning from a 47-step local window to a **per-step aggregation paradigm**. This shift is not a mere performance optimization, but a strict mathematical requirement of the algorithm.

Because both utility and risk gradients are calculated from a single shared forward pass evaluated on the frozen anchor Φ_t , accumulating updates locally for multiple steps would cause the Φ^U and Φ^R buffers to mathematically diverge. If local divergence occurred, a single forward pass could no longer accurately represent the state of both parameter trajectories simultaneously. Synchronizing with the central server at every single optimization step ensures that the dual updates are immediately geometrically filtered and aggregated. This instantaneously produces a newly unified global model Φ_{t+1} , providing a mathematically consistent and shared starting point for the subsequent forward pass.

Server-Side Configuration and Hyperparameters: Upon receiving the updates, the central server executes the orthogonal sterilizations and utility gating. The Frobenius inner products ($\langle A, B \rangle_F$) required for the projections are efficiently computed utilizing Py-

Torch’s element-wise tensor multiplication and summation (`torch.sum(A * B)`). Based on our empirical tuning across the simulated network, the variables governing the projection magnitude were configured as follows:

- **Utility Sterilization Penalty (λ_t):** Set to **0.3**.
- **Risk Sterilization Penalty (μ_t):** Set to **0.8**.
- **Utility Salvage Coefficient (γ_t):** Set to **0.2**.

Those parameters could be tuned through hyper-parameter tuning, but such techniques requires an amount of resources we unfortunately lack.

Structural Uniformity Across Experimental Scenarios: A defining characteristic of RAP-FL, which sharply contrasts with the previous baselines, is its complete structural uniformity across all experimental scenarios. Because the method relies entirely on continuous soft-weighting rather than hard data filtering (Method 1) or dataset bifurcation (Method 3), the operational setting remains perfectly identical regardless of the underlying data distribution.

In Scenarios 1, 2, and 3, every participating client consistently processes its full allocation of $N = 6,016$ samples. The computational pipeline, which consists of the single-pass forward execution and the dual-backward gradient accumulation, never varies in size, memory footprint, or synchronization timing. The algorithm dynamically adapts to extreme asymmetries not by altering the training architecture, but purely through the mathematical decay of the specific loss components ($\mathcal{L}^U$ and $\mathcal{L}^R$) dictated by the classifier’s scores. Consequently, RAP-FL proves to be completely *scenario-agnostic* at the system level, shifting the burden of threat isolation entirely to the server’s geometric projections.

Open-Source Implementation To support the experimental setup described above, we developed a dedicated open-source Python library that implements the full RAP-FL pipeline. The library is built on top of **Flower** framework [1], which handles the federated client-server coordination and simulation environment. On the client side, it wraps the Hugging Face **Trainer** class to introduce the single-pass dual-backward logic, while on the server side it extends Flower’s **Strategy** interface to perform the orthogonal projection and aggregation steps. The underlying tensor operations rely on **PyTorch**, and parameter-efficient fine-tuning is managed through the Hugging Face **PEFT** library for LoRA adapter handling. The framework is designed to be plug-and-play, allowing practitioners to integrate RAP-FL into existing federated pipelines with minimal configuration. The source code is publicly available at <https://github.com/LeoLe12/Federated-Risk-Aware-Projection>.

5 | Results and Discussion

This chapter presents the experimental results of our proposed methodology run through the experiments in Chapter 4 and provides a discussion of the findings. The primary objective of this evaluation is to systematically address the research questions formulated in Chapter 1 by quantifying the effectiveness of various risk-mitigation strategies in a Parameter-Efficient Federated Learning environment. Specifically we discuss the trade-off we achieve in each method.

To achieve a structured and objective analysis, the evaluation is divided into several progressive stages: First, we analyze the **centralized baselines** (Section 4.3) to establish the theoretical replicability of experiment in a PEFT setting. Next we explore the same results on the **federated transposition** of the centralized baselines defined earlier. Lastly we conclude with the **Federated Solutions** 4.4 to demonstrate the inherent vulnerability of the standard Federated Averaging (FedAvg) algorithm, effectively answering our second research question (**RQ2**). Subsequently, we present a detailed comparative analysis of the **Safety-Aware Strategies**, evaluating Methods 1 through 4 across three distinct data heterogeneity scenarios. This addresses **RQ3** by highlighting the strengths and critical limitations of data filtering, dynamic weighting, task arithmetic, and risk-aware projections.

Throughout the chapter, the model’s acquisition of malicious knowledge is quantified using the WMDP benchmark, while its benign scientific competence is measured via the MMLU-Bio benchmark. Additional general knowledge benchmarks (MMLU-NoBio) are employed to assess the occurrence of catastrophic forgetting.

5.1. Centralized Baselines Verification (RQ1)

These baseline configurations isolate data composition as the sole independent variable, mapping the direct impact of fine-tuning paradigms on capability shifting.

Figure 5.1 illustrates the trajectories of the models across five training epochs (1880 steps) on the WMDP benchmark (our primary proxy for hazardous knowledge acquisition) and

the MMLU-Bio benchmark (our metric for general domain utility).

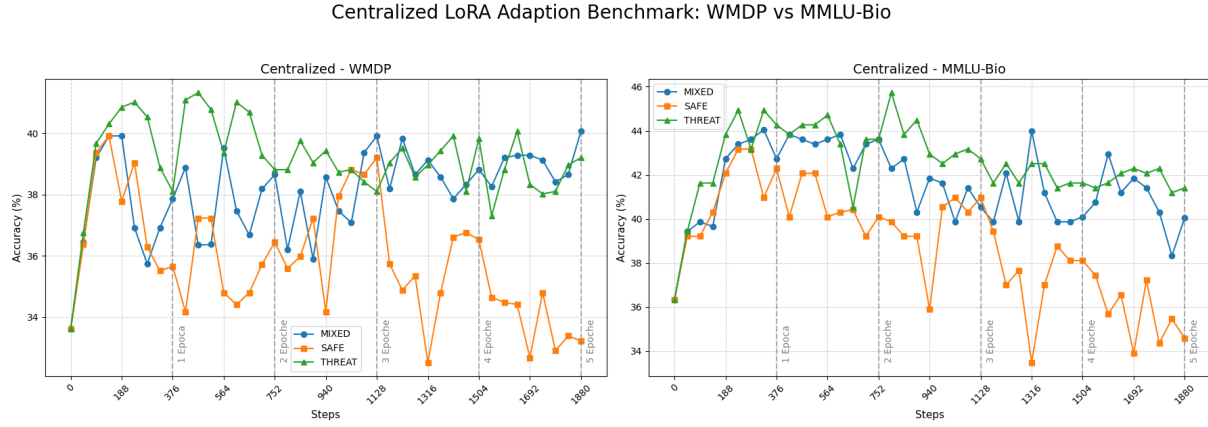


Figure 5.1: Centralized PEFT baselines evaluation. **Left:** WMDP accuracy demonstrates rapid hazardous capability acquisition for Threat and Mixed datasets, while the Safe-Only baseline remains successfully suppressed. **Right:** MMLU-Bio accuracy highlights the eventual utility degradation of the Safe-Only baseline due to variance loss, whereas Threat-Only improves general scientific utility.

5.1.1. Analysis of Knowledge Absorption and Variance Loss

The results validate the parameter-efficient adaptation capability of QLoRA while highlighting profound safety risks and learning dynamics. The unfiltered base model starts with a native WMDP accuracy of 33.62% and an MMLU-Bio accuracy of 36.34%.

Continuous fine-tuning on the **Threat-Only** dataset causes a rapid, clear escalation in malicious capabilities, driving WMDP accuracy to a peak of 41.32% early in the training process (at step 470). Crucially, this exposure to advanced, albeit dangerous, scientific texts also significantly improves general biological utility, increasing MMLU-Bio accuracy to a peak of 44.83% due to the sheer information enrichment.

The **Mixed Baseline** uncovers an alarming alignment dynamic: when benign and malicious inputs are ingested simultaneously in equal proportions, the acquisition of hazardous capabilities follows a dangerously similar trajectory to the pure Threat baseline, climbing to a WMDP accuracy of 40.06% while mirroring the Threat baseline’s utility gains.

Conversely, the **Safe-Only** baseline confirms the core premise of data selection for safety: it successfully drops malicious context, keeping WMDP accuracy suppressed and ending slightly below its pre-training levels (33.23% at step 1880). However, the analysis of its MMLU-Bio trajectory reveals a critical limitation. While it maintains strong utility during the first three epochs (hovering around 40-42% accuracy), extended training results in

a noticeable, sharp drop, reaching 34.58% accuracy in the final epoch. We attribute this degradation to the data filtering process itself. By actively removing complex, dual-use scientific content, the dataset loses critical variance. Although this constrained data pool is sufficient for safe short-term adaptation, the limited variety makes the model highly susceptible to overfitting during extended fine-tuning, ultimately reducing its overall reasoning ability and causing variance loss.

Loss and Confidence While raw accuracy tracks the discrete prediction success, analyzing the continuous probability metrics such as Confidence and Loss margin provides more insights and details of the results.

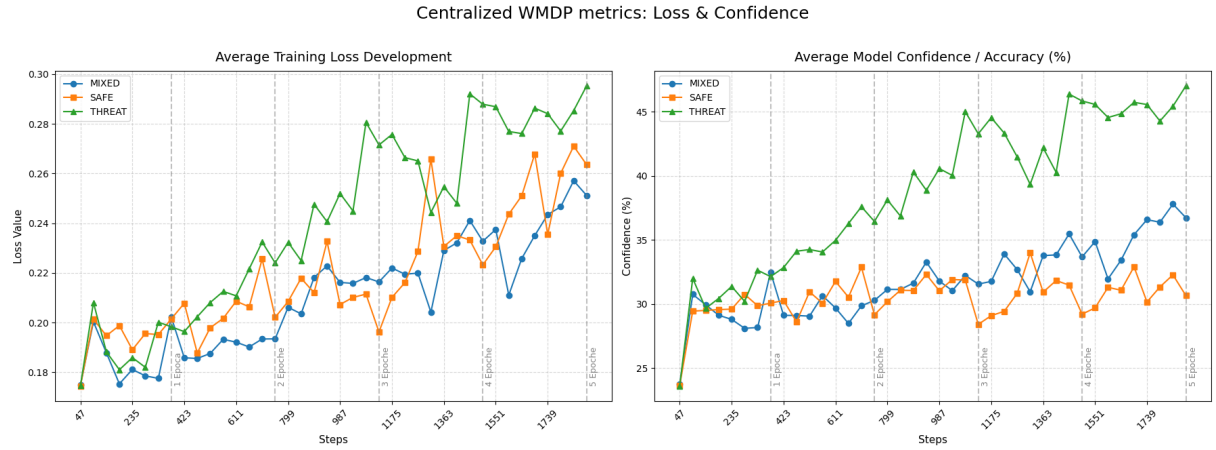


Figure 5.2: Prediction Loss and Confidence Margin of the centralized baselines evaluated on the WMDP benchmark.

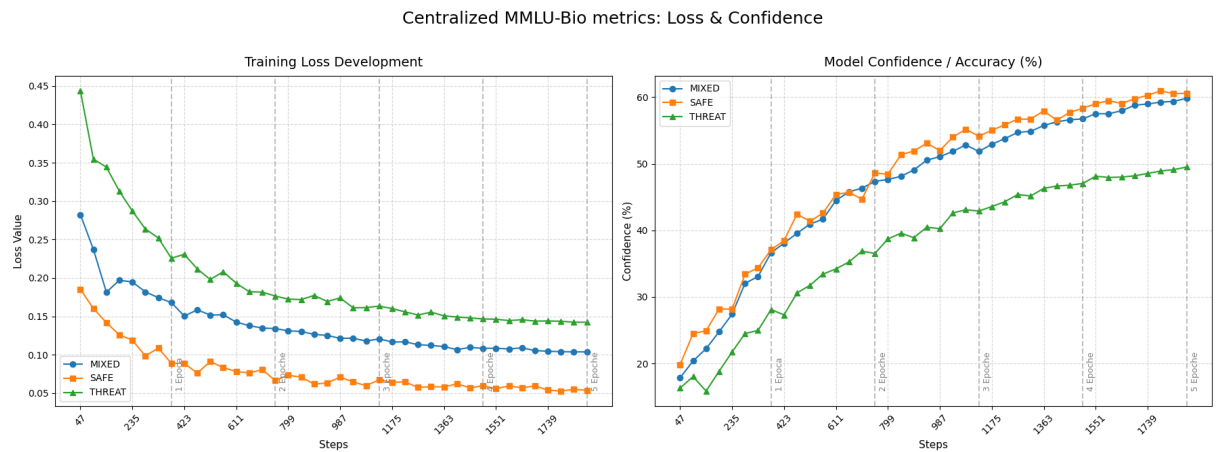


Figure 5.3: Prediction Loss and Confidence Margin of the centralized baselines evaluated on the MMLU-Bio benchmark.

The data reveals distinct behavioral signatures based on the training composition:

- **Hazardous Knowledge (WMDP):** The continuous metrics reveal a stark divergence in how the models internalize threats. The **Threat-Only** baseline rapidly absorbs hazardous concepts, doubling its average confidence margin from 23.6% to 47.0%. Coupled with a high prediction loss (~ 0.29), this indicates that the compromised model makes highly confident errors. Conversely, the **Safe-Only** baseline successfully prevents this capability acquisition. Its confidence on hazardous tasks is effectively suppressed (peaking at just 30.6%), meaning the model remains uncertain and effectively "guesses" when faced with biosecurity questions, thereby validating the proactive data filtration strategy.
- **Benign Utility Calibration (MMLU-Bio):** On safe scientific tasks, the **Safe-Only** model becomes exceptionally well-calibrated. As it fine-tunes, its prediction loss drops to 0.05 while its confidence soars to 60.5%. In contrast, although the **Threat-Only** model also improves its MMLU-Bio accuracy, it does so with significantly less certainty (confidence peaking at 49.5%, with a higher loss of 0.14).

In conclusion, these continuous metrics confirm that data filtering not only prevents the overt generation of hazardous text but fundamentally alters the latent space: a safe model becomes highly confident in benign reasoning while remaining structurally uncertain of malicious concepts.

5.2. Federated Replication of Baseline Paradigms (RQ2)

We address our first research question (**RQ2**): *Can the capability acquisition and safety dynamics observed in centralized training be replicated within a decentralized, Federated Learning framework?*

To evaluate this, we transposed the three foundational paradigms (Safe-Only, Threat-Only, and Mixed) into a two-client federated architecture using standard Federated Averaging (FedAvg). In this strictly controlled setup, the global datasets from the centralized experiments were perfectly bifurcated. Both clients received an exact 50/50 split of the respective data pool, and synchronized their LoRA adapters with the central server every 47 local steps for a total of 20 communication rounds (spanning 5 local epochs).

Figure 5.4 illustrates the global model's trajectories across the communication rounds for both the WMDP and MMLU-Bio benchmarks.

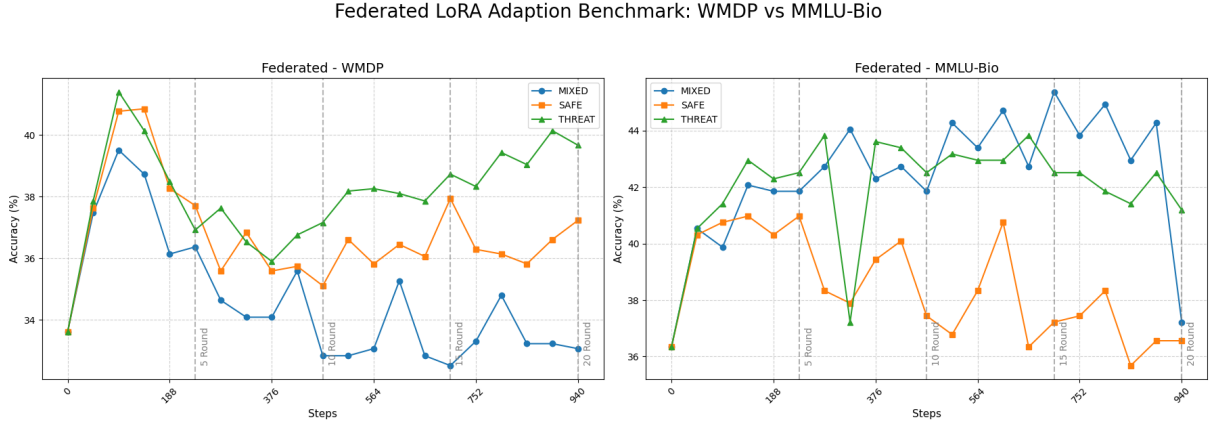


Figure 5.4: Federated replication of baseline paradigms using standard FedAvg across 20 communication rounds. **Left:** WMDP accuracy shows the rapid integration of threat capabilities. **Right:** MMLU-Bio accuracy confirms the eventual variance-loss degradation in the Safe-Only network, mirroring centralized dynamics.

5.2.1. Analysis of Decentralized Knowledge Acquisition

The empirical results confirm that FedAvg effectively acts as a reliable as reliable baseline aggregation strategy, mirroring the behavioral trends established in the centralized bounds, albeit with the expected introduction of some aggregation noise.

Rather than introducing new learning dynamics, the homogeneous federated networks directly replicate the phenomena observed in the centralized baselines:

- **Threat Replication:** The **Threat-Only** distribution rapidly injects malicious capabilities into the global model, with WMDP accuracy spiking to 41.40% by Round 2 and sustaining a high accuracy (39.67%), perfectly matching the centralized poisoning effect.
- **Variance Loss Replication:** The **Safe-Only** paradigm successfully suppresses the threat but replicates the exact same variance-loss degradation. After an initial utility peak (MMLU-Bio at 40.97%), the lack of complex dual-use data causes the general scientific utility to collapse back to 36.56% by the final round.
- **Aggregation Noise:** The only notable divergence from the centralized baseline is the inherent volatility of the federated synchronization loop. In the **Mixed** paradigm, the model experiences sharp transient spikes in threat acquisition (up to 39.51%) at the synchronization boundaries before the competing safe and threat gradients average out to settle around 33.07%.

Federated Confidence and Loss. Analysis of the continuous probability metrics in the federated environment confirms and trends observed in the centralized baselines.

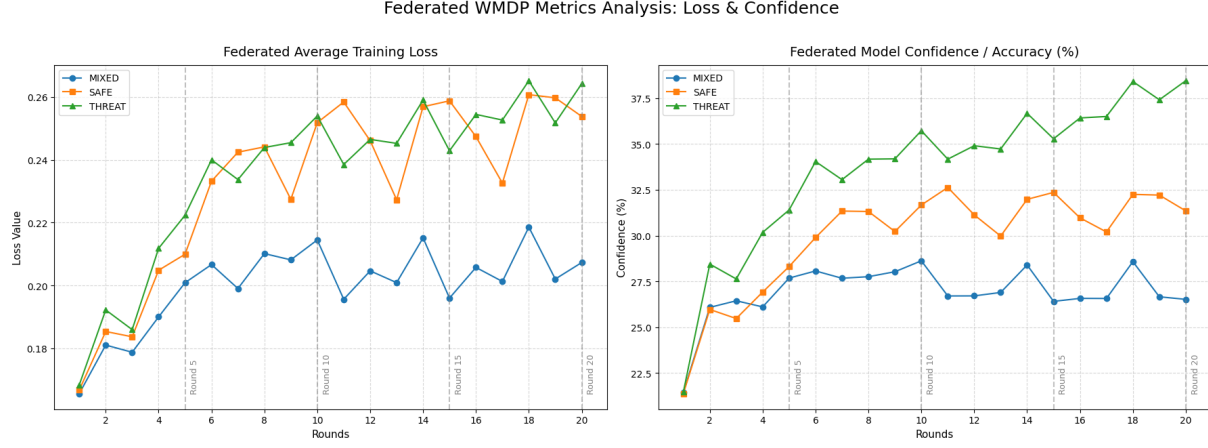


Figure 5.5: Prediction Loss and Confidence Margin of the federated replication evaluated on the WMDP benchmark.

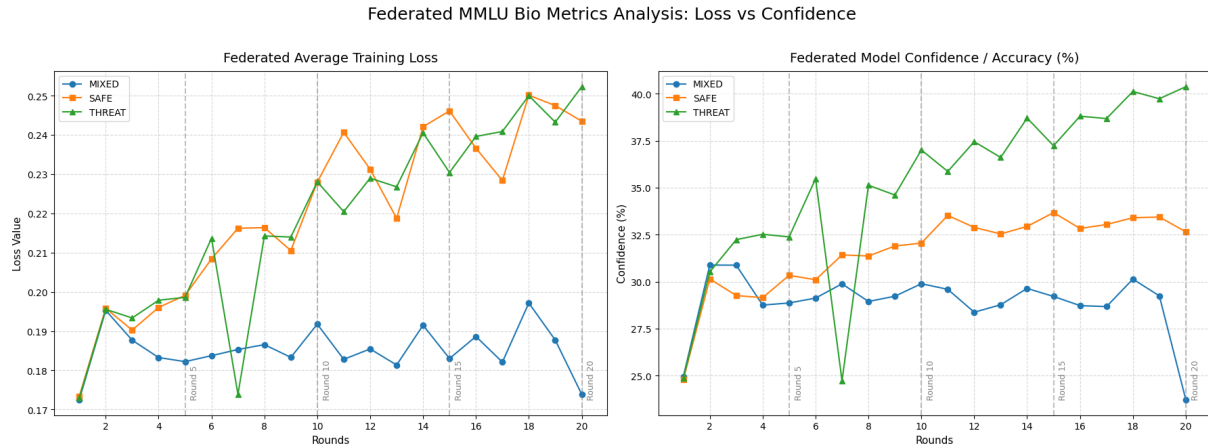


Figure 5.6: Prediction Loss and Confidence Margin of the federated replication evaluated on the MMLU-Bio benchmark.

On the hazardous **WMDP** benchmark, the **Threat-Only** network reliably internalizes malicious concepts, with its confidence margin escalating from 21.4% to 38.4%. Conversely, the **Safe-Only** network successfully suppresses threat certainty, peaking at just 32.3% and confirming that decentralized data filtration keeps the model probabilistically uncertain about hazardous knowledge.

However, the evaluation of benign utility (**MMLU-Bio**) exposes the severe, compounded cost of federated data starvation. Unlike the centralized Safe baseline, which achieved excellent calibration on safe biology, the decentralized **Safe-Only** network struggles to

gain internal certainty, with its confidence margin stagnating around 32.6%. Because the already-filtered, low-variance dataset is further partitioned across isolated edge nodes, the global model suffers an exacerbated variance loss. In contrast, the **Threat-Only** network, which retains the complex (albeit dangerous) scientific texts, achieves higher confidence in general biology (40.3%).

This divergence perfectly illustrates the core limitation of edge-side data filtering: permanently discarding data in a federated network can starve the model of linguistic complexity, degrading benign utility and motivating the need for volume-preserving interventions beyond data selection.

5.2.2. Conclusion on Replicability

These findings successfully answer **RQ2**. The training methodology and safety properties established in centralized environments *can* be effectively replicated within a Federated Learning framework using PEFT. Standard FedAvg efficiently aggregates both general utility and hazardous capabilities.

However, this successful replication of the "Threat" baseline exposes the fundamental vulnerability of decentralized architectures: if FedAvg perfectly incorporates knowledge from homogeneous threat nodes, it will blindly incorporate malicious updates in heterogeneous scenarios where the server cannot inspect the data. This sets the need for advanced risk-aware aggregation strategies evaluated in the following sections.

5.3. Comparative Evaluation of Safety-Aware Strategies (RQ3)

To address **RQ3**, we performed a systematic evaluation of our proposed defensive interventions compared to the established baselines. We observe how each strategy reacts to the structural variations of the network topology across the three defined scenarios (S1, S2, and S3, as detailed in Table 4.5) to address our **RQ4**.

To ensure statistical reliability and mitigate the optimization variance of LLMs, the empirical results presented and discussed in this section represent the average performance across two distinct experimental seeds. Figure 5.7 illustrates the trajectories of the global models across all five training epochs.

(Detailed per-seed learning trajectories and individual scenario breakdowns are provided for completeness in Appendix A.1).

Benchmark Results: Average (Seed 42 & Seed 73) - WMDP vs MMLU-Bio (All Scenarios)



Figure 5.7: WMDP (Threat) and MMLU-Bio (Utility) accuracy across the three federated scenarios (S1-S3) over five training epochs. The results represent the average of two experimental seeds. Each row corresponds to one scenario; left plots track threat suppression, right plots track utility preservation.

Standard Federated Averaging (Method 0). The empirical evidence demonstrates a clear structural vulnerability: the standard FedAvg algorithm systematically fails to maintain proactive safety. In all scenarios, the "blind" aggregation of malicious adapter updates causes a rapid escalation in threat capabilities. The WMDP accuracy consistently escalates from the pretrained baseline of 33.6% to peak around $\sim 40.3\%$ on average across the network configurations (settling at 39.5% terminally in S3). While this unmitigated aggregation preserves excellent general utility (MMLU-Bio $> 42\%$), the global model completely fails its primary safety objective, confirming that decentralized training without architectural interventions acts as a direct vector for malicious poisoning.

Method 1: Local Data Filtering. Applying the *Deep Ignorance* filtering logic directly at the edge nodes reveals significant, structural limitations in decentralized environments. In the Extreme Asymmetry scenario (S1), Method 1 performs acceptably for safety (WMDP $\approx 34.0\%$) because the entirely compromised client can be fully excluded. However, this comes at the severe cost of *data starvation*: the drastic data reduction triggers variance-loss overfitting, plunging MMLU-Bio accuracy down to 37.8%, well below the baseline utility. Crucially, in the realistic heterogeneous scenario (S3), Method 1 fails on both axes simultaneously. The WMDP score remains dangerously elevated at 37.7%, while MMLU-Bio degrades further to 38.0%. This unexpected failure stems from the mechanics of federated aggregation: dropping data heterogeneously across clients dynamically skews the volume-based aggregation weights ($\frac{n_k}{N}$), granting disproportionate influence to certain updates and failing to effectively neutralize the threat. Consequently, in our experiments we cannot conclude that simple data selection provides meaningful advantage over undefended FedAvg in realistic deployments in our experiments, but we are limited to a 2 client network.

Method 2: Dynamic Weighted Aggregation. This strategy exhibits extreme volatility and severe topology dependence. In S1, where safe and hazardous data are cleanly partitioned across clients (pure local batches), the dynamic scalars perfectly penalize the rogue node, effectively suppressing the threat (WMDP $\approx 33.6\%$) while preserving outstanding general utility (MMLU-Bio $\approx 40.5\%$). However, this defense mechanism completely collapses in realistic environments. Once safe and threat samples are entangled within the same local datasets (S2 and S3), the per-batch risk scores mathematically average out. The dampening effect vanishes entirely: in S3, WMDP escalates to 41.7%, performing even worse than the standard baseline. This renders Method 2 structurally unsafe for any real-world federated deployment.

Method 3: Federated Task Vector Aggregation. The vector negation approach represents the opposite failure mode: it is geometrically powerful but fundamentally non-surgical. In S1, the explicitly isolated threat vector is so mathematically potent that its aggressive negation drives WMDP safely below the baseline (32.8%). Yet, this blunt subtraction induces severe *catastrophic forgetting*, collapsing the model’s general scientific reasoning (MMLU-Bio plunges to 35.0%). Furthermore, in the more complex Realistic scenario (S3), the mathematical extraction of the task vector becomes inherently noisy due to the variance in heterogeneous local data distributions. As a result, the defense loses its effectiveness, failing to suppress the threat (WMDP climbs back to 38.6%) while maintaining a somewhat arbitrary utility gain (MMLU-Bio $\approx 41.3\%$).

Method 4: Risk-Aware Projection (RAP-FL). RAP-FL emerges as the strictly superior and uniquely robust algorithmic solution, achieving an optimal and consistent safety-utility balance across all three scenarios. Its defining strength is not merely an absolute best-in-class score on a single isolated metric, but rather a **remarkably stable operating point**. The terminal WMDP accuracy for RAP-FL varies only between 34.2% and 34.5% across S1, S2, and S3, a remarkably tight spread of less than 0.3 percentage points. No other evaluated method comes close to this level of scenario-agnostic consistency.

This stability is a direct mathematical consequence of the per-step geometric projection. At each local aggregation step, hazardous directions are surgically identified and projected out of the adapter space, while the residual benign knowledge embedded within dual-use updates is actively salvaged. By geometrically decoupling benign concepts from malicious parameter sub-spaces at the server level, RAP-FL effectively neutralizes threats without relying on the fragile data-partitioning assumptions that cause Method 2 to collapse, and without inducing the catastrophic forgetting that afflicts Methods 1 and 3.

Table 5.1, Table 5.2, Table 5.3 provide a concise summary of the terminal-epoch performance for all evaluated strategies for each scenario tested, highlighting the strategy that achieves the best safety-utility trade-off.

The plots of the Loss (eq. 3.35) and Confidence Margin (eq. 3.34) are provided in the appendix A.2.

Average terminal-epoch performance (S1 – Extreme Asymmetric)

Method	WMDP (Threat) ↓	MMLU-Bio (Utility) ↑
Pretrained Baseline	33.6%	36.3%
Method 0: FedAvg	40.3%	42.1%
Method 1: Data Selection	34.0%	37.8%
Method 2: Dyn. Weighting	33.6%	40.5%
Method 3: Task Vector	32.8%	35.0%
Method 4: RAP-FL	34.5%	38.7%

Table 5.1: Comparison of terminal-epoch accuracy in the extreme asymmetric scenario (S1), averaged across two experimental seeds. In this scenario Method 2 (*Dynamic Weight*) achieves the best results.

Average terminal-epoch performance (S2 – Mixed Homogeneous)

Method	WMDP (Threat) ↓	MMLU-Bio (Utility) ↑
Pretrained Baseline	33.6%	36.3%
Method 0: FedAvg	35.6%	41.1%
Method 1: Data Selection	35.5%	35.5%
Method 2: Dyn. Weighting	37.1%	40.3%
Method 3: Task Vector	37.1%	42.3%
Method 4: RAP-FL	34.4%	39.8%

Table 5.2: Comparison of terminal-epoch accuracy in the mixed homogeneous scenario (S2), averaged across two experimental seeds. RAP-FL achieves the best threat suppression while simultaneously preserving benign utility.

Average terminal-epoch performance (S3 – Mixed Heterogenous)

Method	WMDP (Threat) ↓	MMLU-Bio (Utility) ↑
Pretrained Baseline	33.6%	36.3%
Method 0: FedAvg	39.5%	44.0%
Method 1: Data Selection	37.7%	38.0%
Method 2: Dyn. Weighting	41.7%	43.9%
Method 3: Task Vector	38.6%	41.3%
Method 4: RAP-FL	34.2%	39.1%

Table 5.3: Comparison of terminal-epoch accuracy in the heterogeneous scenario (S3), the most *realistic* scenario, averaged across two experimental seeds. RAP-FL achieves the best threat suppression while simultaneously preserving benign utility.

5.4. General Knowledge and Catastrophic Forgetting

To complete our answer to **RQ4** regarding the global safety-utility trade-off, we evaluate whether the proposed safety-aware methods affect broader reasoning capabilities beyond our targeted scientific domains (Biology and Chemistry). To this end, we evaluated the final global models on the remaining, non-biological categories of the Massive Multitask Language Understanding benchmark (**MMLU-NoBio**) [9].

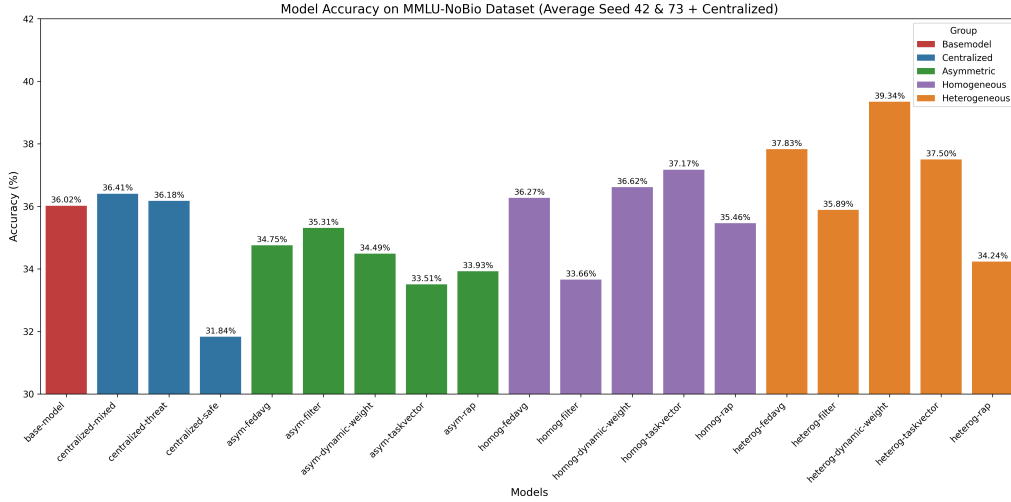


Figure 5.8: Evaluation on General Knowledge (MMLU-NoBio) across defensive strategies run on 2 seeds, compared to the base model and centralized experiments. Singular seed results are provided for completeness in Appendix A.1.

Figure 5.8 illustrates that the algorithmic choices of defensive strategies have a profound and varied impact on the model’s foundational knowledge. As a reference, standard FedAvg (**M0**), which operates without any safety defense, maintains general knowledge intact, yielding an average accuracy delta of +0.26 percentage points (pp) compared to the pre-trained base model (36.02%).

When defenses are introduced, their behavior diverges significantly depending on the client distribution. Strict dataset exclusion (**M1**, filter) induces noticeable catastrophic forgetting in the homogeneous setting (dropping to 33.66%, a delta of -2.36 pp), while showing negligible effects in the other configurations. On the other hand, surgical task vector negation (**M3**) exhibits highly scenario-dependent behavior: it causes a significant performance drop under the asymmetric setting (33.51%, -2.51 pp), but actually recovers and slightly surpasses the baseline in homogeneous (+1.15 pp) and heterogeneous (+1.48 pp) environments, resulting in an average delta of +0.04 pp. This suggests that vector subtraction does not induce widespread forgetting, but its impact is highly sensitive to distribution shifts. The negation effect is more evident on the domain benchmarks discussed in the previous section, which is a good sign for this method.

In comparison, softer weight-adjusting interventions (**M2**, dynamic-weight) preserve utility well, even exhibiting the highest peaks in heterogeneous scenarios (39.34%, +3.32 pp). However, as discussed in Section 5.3, this high general knowledge comes at the cost of failing the primary safety objective.

Method 4 (RAP-FL), which operates via orthogonal projections, consistently ranks among the two lowest federated methods in terms of general knowledge across all scenarios, exhibiting the worst average delta overall (-1.48 pp). Moreover, while RAP-FL appears to be the most stable method in terms of average accuracy range across scenarios (1.53 pp), this stability is slightly less pronounced when looking at individual runs (e.g., in seed 73, the dataset filtering method **M1** exhibits a narrower range of 1.00 pp compared to RAP’s 1.69 pp).

However, RAP-FL’s utility degradation must be contextualized. while it sacrifices some general knowledge compared to other federated methods, it assures safety with previous benchmark analysis.

6 | Conclusion

This thesis addressed the 'dual-use' dilemma exacerbated by the rapid democratization of Large Language Models (LLMs). While these architectures offer immense potential for scientific advancement, their capacity to encode and generate hazardous knowledge poses severe real-world security risks. As the adoption of Federated Learning (FL) accelerates to meet privacy demands, ensuring proactive AI safety in decentralized environments has emerged as a critical necessity. To bridge this gap, our research systematically investigated LLM safety risks within an FL pipeline, focusing specifically on the inherent vulnerability of the blind server problem, the architectural constraint where the central server aggregates local parameter updates but is strictly prohibited from inspecting the underlying private data.

6.1. Summary of Contributions and Key Findings

To address this gap, we designed a highly structured, four-module methodological pipeline. We first successfully adapted the centralized, data-centric *Deep Ignorance* [17] safeguard into a highly resource-efficient Parameter-Efficient Fine-Tuning (QLoRA) regime. Following this, we relocated the safety enforcement mechanism from the raw data space into the weight-aggregation space. Driven by a per-sample *Risk Signal* computed by a local classifier, we proposed and evaluated three advanced risk-aware aggregation mechanisms against standard Federated Averaging (FedAvg) and edge-side data filtering.

Our extensive empirical evaluation provided clear answers to the Research Questions formulated at the beginning of this work:

- **RQ1 & RQ2 (PEFT Viability and Federated Replicability):** We demonstrated that proactive data-filtering effectively mitigates hazardous capabilities even when bounded to a low-rank adapter space. Furthermore, standard FedAvg can replicate these dynamics perfectly in homogeneous networks, with an acceptable level of aggregation noise, that can also possibly attributed to our approximation in the aggregation. However, this perfectly highlights the vulnerability of the system:

in heterogeneous scenarios, standard FedAvg acts as a blind vector injector, allowing a single compromised node to rapidly poison the global model and escalate threat capabilities (WMDP accuracy) above 41%.

- **RQ3 (Safety-Robustness Beyond Data Selection):** We investigated whether safety in federated fine-tuning can be enforced directly at the aggregation stage, without relying on server-side data selection. To this end, we introduced and compared three safety-aware aggregation mechanisms. **Dynamic Weighted Aggregation** incorporates risk information by scaling each client contribution according to the estimated proportion of safe and hazardous samples, reducing the influence of risky updates while preserving participation from all clients. **Task Vector Aggregation** explicitly separates safe and hazardous knowledge into distinct adaptation vectors and applies task arithmetic to suppress the threat-related component during global model construction. Finally, we propose **Risk-Aware Projection for Federated Learning (RAP-FL)**, a geometrical approach that treats safety as a directional property in parameter space. Rather than weighing or removing entire updates, RAP-FL identifies hazardous directions and projects them out while preserving orthogonal utility-preserving components.
- **RQ4 (Performance Trade-offs):** Our empirical analysis reveals that safety-aware aggregation inevitably introduces a trade-off between hazardous knowledge suppression and utility preservation, but the magnitude of this trade-off strongly depends on the aggregation strategy. Dynamic Weighted Aggregation provides a lightweight mechanism for reducing the influence of risky clients, yet its effectiveness diminishes when benign and hazardous knowledge coexist within the same update. Task Vector Aggregation offers a more explicit form of hazard removal by isolating and negating threat-related adaptations. Its performance is highly sensitive to the structure of the learned task vectors, which heavily depends on the underlying data distribution. In contrast, RAP-FL achieves the most favorable balance between safety and utility. RAP-FL therefore emerges as the most effective and consistent solution across scenario for navigating the safety-utility trade-off, maintaining strong performance on benign benchmarks while keeping hazardous capabilities substantially constrained.

6.2. Limitations

While this research establishes a strong foundational framework for safe federated fine-tuning, several limitations must be acknowledged:

- **The Trust Assumption:** As defined in the Research Gap 2.5, this work aims to

prevent inadvertent contamination of local unfiltered data rather than intentional and malicious actors. Our work does not permanently *solve* the blind server problem but rather reframes it under a narrower trust assumption. Because the client computes the continuous risk signal locally, our mathematical guarantees hold only if the Risk Estimator is assumed to be an "honest" entity. A truly adversarial client could intentionally forge these scores, reporting benign probabilities for highly hazardous payloads to bypass the server's geometric projections.

- **Computational Scale and Topology:** Due to the VRAM and computational constraints associated with executing multiple 7-billion-parameter LLMs simultaneously, our federated simulation was strictly bounded to a minimal, two-client architecture. While sufficient to validate the algorithmic math and boundary conditions, real-world FL networks operate with hundreds of highly asynchronous and Non-IID nodes. Furthermore, resource constraints limited the implementation of computationally intensive aggregation mechanisms, as highlighted in Section 4.4.4, our use of element-wise averaging introduces a mathematical approximation, an aspect that could be refined in future works with more accurate aggregation strategies.
- **Hyperparameter Tuning:** The advanced methodologies, particularly RAP-FL and Task Arithmetic, rely on critical scaling coefficients (e.g., projection strengths λ , μ , salvage rate γ , and steering factor α). In our experiments, these were fixed based on empirical heuristics. The lack of an exhaustive hyperparameter search grid means the results, while strong, may not represent the absolute theoretical maximum of the algorithms.

6.3. Future Work

Building upon these findings and limitations, several promising avenues for future research emerge in the intersection of decentralized learning and AI safety:

- **Verifiable Risk Execution:** The most critical next step is eliminating the trust assumption. Future work should investigate the integration of Cryptographic Proofs or Trusted Execution Environments (TEEs) at the edge. If the Risk Estimator (e.g., the ModernBERT classifier) is encapsulated within a hardware-secured enclave, the central server could cryptographically verify that the transmitted Risk Signal was genuinely computed by the certified model, mathematically preventing adversarial score forgery.
- **Large-Scale Simulations:** To validate the scalability of RAP-FL, subsequent re-

search should deploy the algorithm across a massively distributed network ($K \geq 100$). This would allow for the evaluation of the projection mechanism under extreme data heterogeneity, high client dropout rates, and varying degrees of local computational budgets; though such scale would demand considerably greater computational resources than those available in this study.

- **Integration with Secure Aggregation (SecAgg):** Modern privacy regulations often require that the central server cannot inspect even the individual parameter updates of the clients. Integrating RAP-FL’s vector projections with cryptographic Secure Aggregation protocols, where the server performs algebraic manipulations on encrypted payloads or shares, represents a highly challenging but essential direction for fully secure, privacy-preserving AI frameworks.

In conclusion, this thesis demonstrates that the decentralization of AI training does not necessitate the abandonment of proactive safety. By shifting the defensive paradigm from the data space to the parameter space, mechanisms like RAP-FL provide a robust, scenario-agnostic blueprint for scaling the collaborative capabilities of open-weight LLMs while systematically mitigating their most severe dual-use risks.

Bibliography

- [1] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.
- [2] J. Bian, L. Wang, L. Zhang, and J. Xu. Lora-fair: Federated lora fine-tuning with aggregation and initialization refinement, 2025. URL <https://arxiv.org/abs/2411.14961>.
- [3] J. R. Biden. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence. 2023.
- [4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [5] C. Burns, H. Ye, D. Klein, and J. Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.
- [6] H.-T. Dang, T. Pham, H. Thanh-Tung, and N. Inoue. On effects of steering latent representation for large language model unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23733–23742, 2025.
- [7] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. Qlora: Efficient fine-tuning of quantized llms. *Advances in neural information processing systems*, 36: 10088–10115, 2023.
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL <https://arxiv.org/abs/1810.04805>.
- [9] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- [10] J. Howard and S. Ruder. Universal language model fine-tuning for text classification.

- In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.
- [11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.
 - [12] G. Ilharco, M. T. Ribeiro, M. Wortsman, S. Gururangan, L. Schmidt, H. Hajishirzi, and A. Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.
 - [13] J. Lang, Z. Guo, and S. Huang. A comprehensive study on quantization techniques for large language models. In *2024 4th International conference on artificial intelligence, robotics, and communication (ICAIRC)*, pages 224–231. IEEE, 2024.
 - [14] R. Lee, M. Kim, F. Rezk, R. Li, S. I. Venieris, and T. Hospedales. Fedp²eft: Federated learning to personalize peft for multilingual llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 22805–22813, 2026.
 - [15] N. Li, A. Pan, A. Gopal, S. Yue, D. Berrios, A. Gatti, J. D. Li, A.-K. Dombrowski, S. Goel, L. Phan, et al. The wmdp benchmark: Measuring and reducing malicious use with unlearning. *arXiv preprint arXiv:2403.03218*, 2024.
 - [16] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. Pmlr, 2017.
 - [17] K. O’Brien, S. Casper, Q. Anthony, T. Korbak, R. Kirk, X. Davies, I. Mishra, G. Irving, Y. Gal, and S. Biderman. Deep ignorance: Filtering pretraining data builds tamper-resistant safeguards into open-weight llms. *arXiv preprint arXiv:2508.06601*, 2025.
 - [18] L. Oliveira. Parameter-efficient fine-tuning of large language models. <https://medium.com/@lmpo/parameter-efficient-fine-tuning-of-large-language-models-4ed51860e1da>, 2023. Accessed: 2026-04-15.
 - [19] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
 - [20] P. Pandey. Federated learning (part 1): The basics of training models where the data lives. <https://pandeyparul.medium.com/>

- [federated-learning-part-1-the-basics-of-training-models-where-the-data-lives-a](#)
2020. Accessed: 2026-05-14.
- [21] X. Qi, Y. Zeng, T. Xie, P.-Y. Chen, R. Jia, P. Mittal, and P. Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to!, 2023. URL <https://arxiv.org/abs/2310.03693>.
- [22] N. Reimers and I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics, Nov. 2019. doi: 10.18653/v1/D19-1410. URL <https://aclanthology.org/D19-1410>.
- [23] R. Singhal, K. Ponkshe, and P. Vepakomma. Fedex-lora: Exact aggregation for federated and efficient fine-tuning of foundation models. *arXiv preprint arXiv:2410.09432*, 2024.
- [24] B. Smith. A complete guide to BERT with code, May 2024. URL <https://towardsdatascience.com/a-complete-guide-to-bert-with-code-9f87602e4a11/>.
- [25] G. Sun, U. Khalid, M. Mendieta, P. Wang, and C. Chen. Exploring parameter-efficient fine-tuning to enable foundation models in federated learning. In *2024 IEEE International Conference on Big Data (BigData)*, pages 8015–8024. IEEE, 2024.
- [26] T. Sun, D. Li, and B. Wang. Decentralized federated averaging, 2021. URL <https://arxiv.org/abs/2104.11375>.
- [27] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [28] F. Urbina, F. Lentzos, C. Invernizzi, and S. Ekins. Dual use of artificial-intelligence-powered drug discovery. *Nature Machine Intelligence*, 4(3):189–191, 2022.
- [29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [30] B. Warner, A. Chaffin, B. Clavié, O. Weller, O. Hallström, S. Taghadiouni, A. Gallagher, R. Biswas, F. Ladhak, T. Aarsen, N. Cooper, G. Adams, J. Howard, and I. Poli. Smarter, better, faster, longer: A modern bidirectional encoder for

- fast, memory efficient, and long context finetuning and inference, 2024. URL <https://arxiv.org/abs/2412.13663>.
- [31] J. Welbl, A. Glaese, J. Uesato, S. Dathathri, J. Mellor, L. A. Hendricks, K. Anderson, P. Kohli, B. Coppin, and P.-S. Huang. Challenges in detoxifying language models. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2447–2469, Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.210. URL <https://aclanthology.org/2021.findings-emnlp.210/>.
- [32] J. Xu, Z. Wu, C. Wang, and X. Jia. Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 8(3):2150–2168, 2024.
- [33] L. Xu, H. Xie, S.-Z. J. Qin, X. Tao, and F. L. Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023. URL <https://arxiv.org/abs/2312.12148>.
- [34] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2):1–124, 2023.
- [35] A. Zou, L. Phan, J. Wang, D. Duenas, M. Lin, M. Andriushchenko, R. Wang, Z. Kolter, M. Fredrikson, and D. Hendrycks. Improving alignment and robustness with circuit breakers. *Advances in Neural Information Processing Systems*, 37:83345–83373, 2024.

A | Appendix A

A.1. Detailed Experimental Results per Seed

This appendix provides the disaggregated empirical results for the comparative evaluation discussed in Chapter 5. Due to the inherent stochasticity of neural network optimization, particularly within Parameter-Efficient Fine-Tuning regimes, the primary evaluation was conducted across multiple initializations. The figures below are organized into two groups:

1. **Domain Threat-Controll Evaluation (WMDP and MMLU-Bio):** Figures A.1 (seed 42) and A.2 (seed 73) present the raw, per-seed learning trajectories for both the WMDP (Threat) and MMLU-Bio (Utility) benchmarks across the three federated scenarios.
2. **General Knowledge Evaluation (MMLU-NoBio):** To further validate that our safety interventions do not induce widespread catastrophic forgetting, Figures A.3 (seed 42) and A.4 (seed 73) illustrate the model’s accuracy on general-purpose MMLU categories.

Benchmark Results seed 42: WMDP vs MMLU-Bio (All Scenarios)

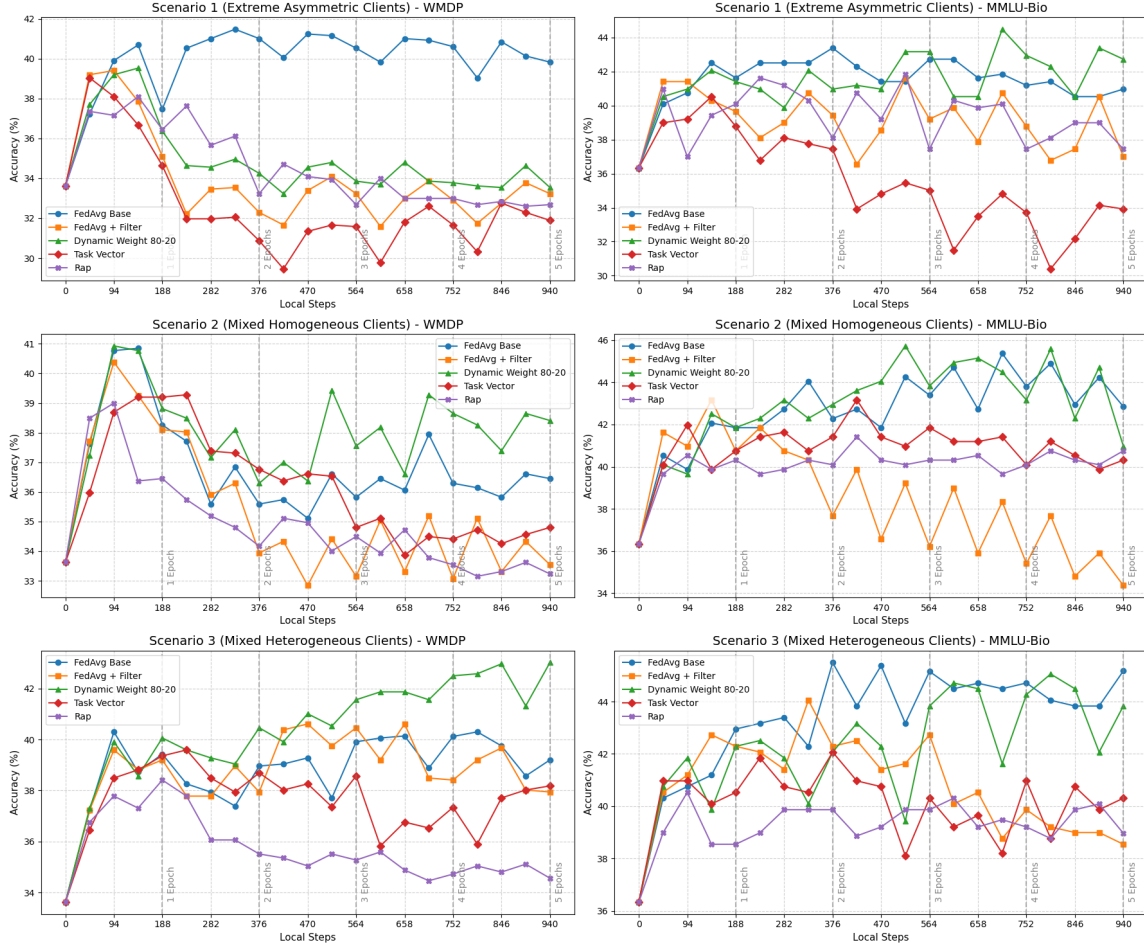


Figure A.1: WMDP and MMLU-Bio benchmark accuracy across the three federated scenarios (S1-S3) over five training epochs. Results obtained using experimental data seed 42.

Benchmark Results Seed 73: WMDP vs MMLU-Bio (All Scenarios)



Figure A.2: WMDP and MMLU-Bio benchmark accuracy across the three federated scenarios (S1–S3) over five training epochs. Results obtained using experimental data seed 73.

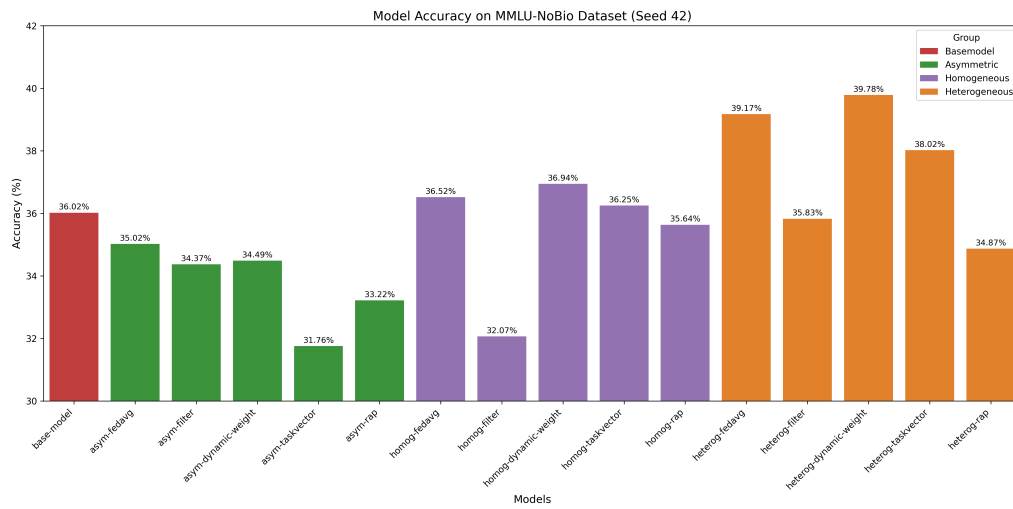


Figure A.3: MMLU-Nobio categories to evaluate general knowledge. Results obtained using experimental data **seed 42**.

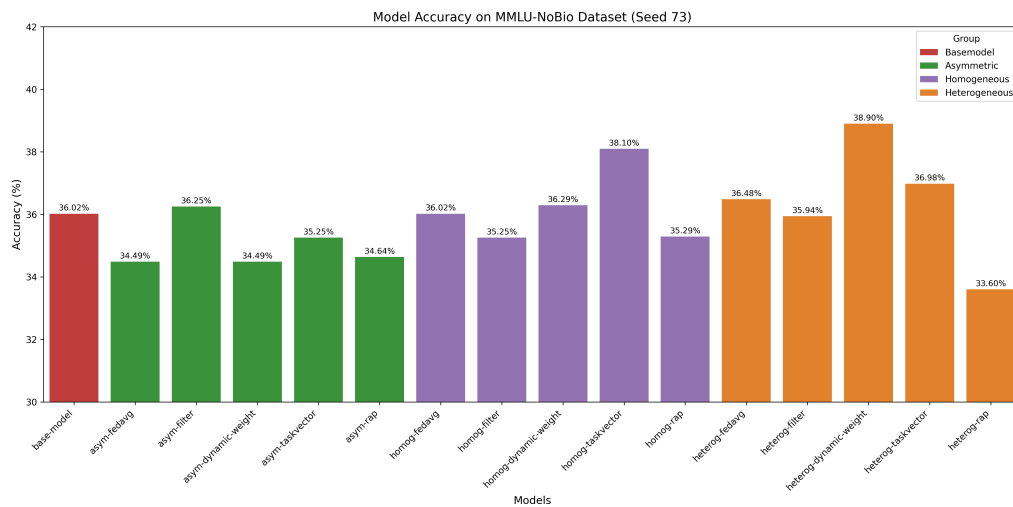


Figure A.4: MMLU-Nobio categories to evaluate general knowledge. Results obtained using experimental data **seed 73**.

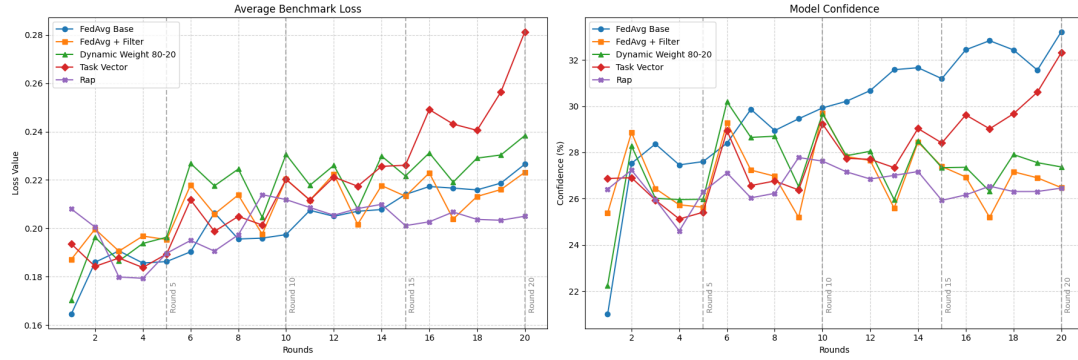
A.2. Loss and Confidence Metrics for Federated Solutions

To offer additional insights on the results in 5.3, this section presents the accompanying metrics computed across the evaluated configurations and methods tested.

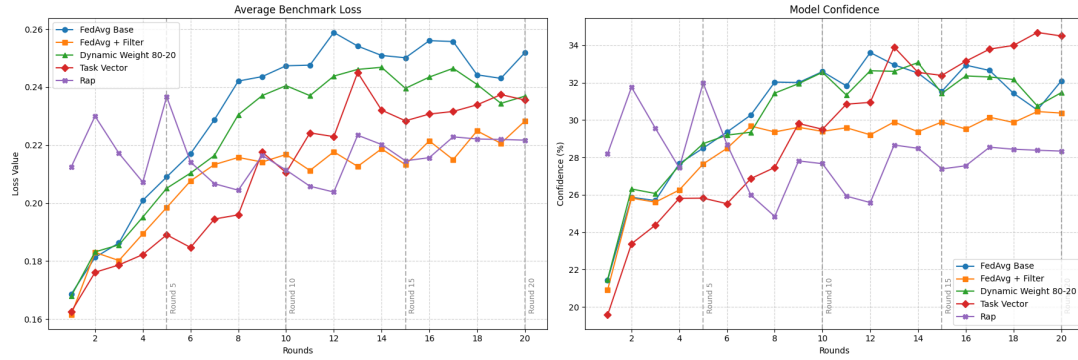
The figures below display the aggregated loss and confidence margin trajectories over the training epochs:

1. **Safety Alignment Dynamics (WMDP):** Figure A.5 details the trajectories of the 2 metrics on the results obtained on threat benchmark WMDP.
2. **Utility Maintenance Dynamics (MMLU-Bio):** Figure A.6 illustrates the trajectories of the 2 metrics on the results obtained on domain utility benchmark, MMLU-Bio.

Federated WMDP Metrics Analysis: Loss vs Confidence (Scenario 1 (Extreme Asymmetric Clients))



Federated WMDP Metrics Analysis: Loss vs Confidence (Scenario 2 (Mixed Homogeneous Clients))



Federated WMDP Metrics Analysis: Loss vs Confidence (Scenario 3 (Mixed Heterogeneous Clients))

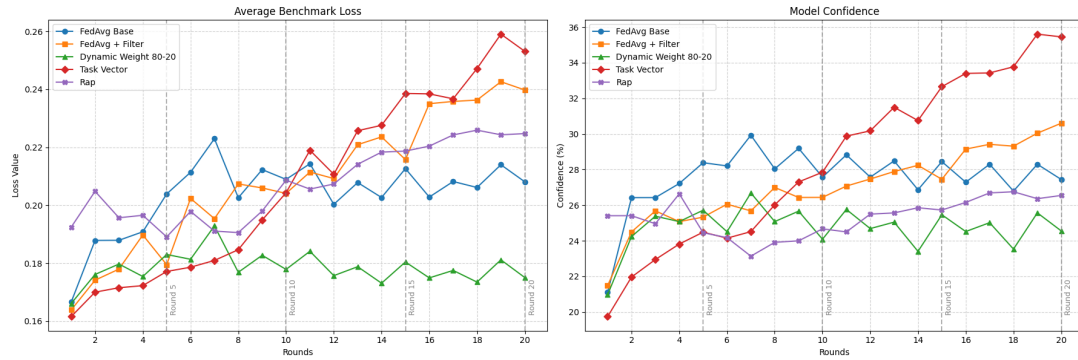
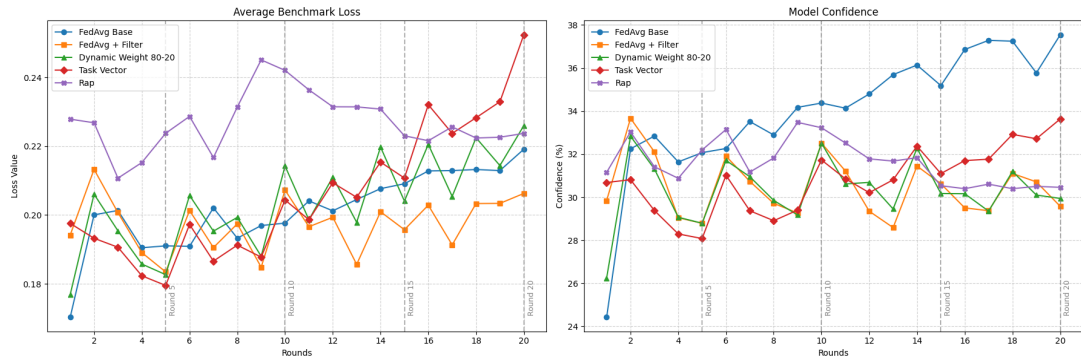
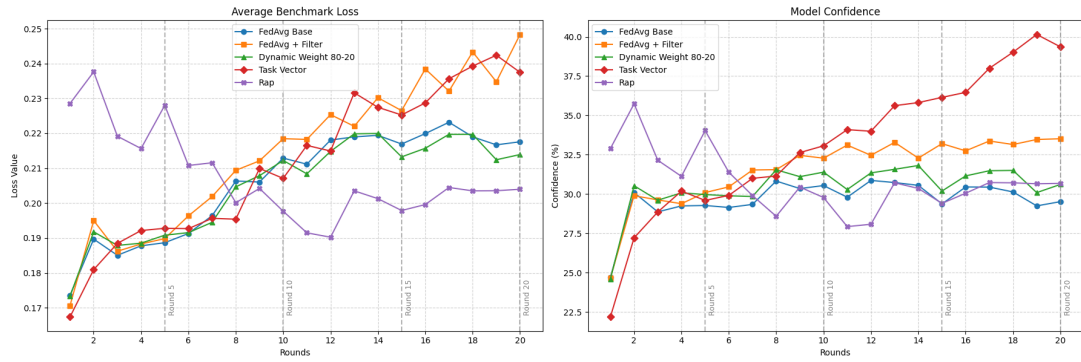


Figure A.5: Loss and prediction confidence trajectories evaluated on the WMDP benchmark.

Federated MMLU-Bio Metrics Analysis: Loss vs Confidence (Scenario 1 (Extreme Asymmetric Clients))



Federated MMLU-Bio Metrics Analysis: Loss vs Confidence (Scenario 2 (Mixed Homogeneous Clients))



Federated MMLU-Bio Metrics Analysis: Loss vs Confidence (Scenario 3 (Mixed Heterogeneous Clients))

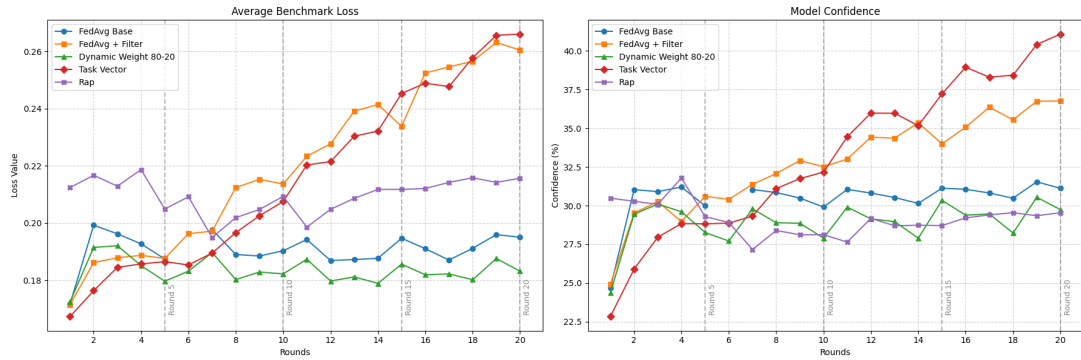


Figure A.6: Loss and prediction confidence trajectories evaluated on the MMLU-Bio dataset.

List of Figures

2.1	Overview of the architectures of Transformer models, BERT models and GPT models compared to each other. Figure adapted from [24].	4
2.2	Illustration of BERT's Masked Language Modeling (MLM) pre-training task. The diagram demonstrates how an input sentence is processed: a specific token is masked (e.g., [MASK]), and the model analyzes the full, bidirectional surrounding context to predict the original token with the highest probability. Image sourced from the official SentenceTransformers documentation [22].	5
2.3	LLM lifecycle phases. Source: adapted from [18]	6
2.4	Representation of LoRA adapter matrices A and B. The base model parameters are frozen. Adapted from [11]	8
2.5	Comparison between Full Parameter Training, LoRA and QLora, adapted from [7].	9
2.6	Overview of hazardous knowledge domains categorized by the WMDP benchmark. Figure adapted from [15].	11
2.7	Comparison between a centralized training setup (left) and a federated training setup (right). Figure adapted from [20].	14
2.8	Comparison between a classical federated learning setup and fedPEFT setting . Figure adapted from [25].	16
3.1	Methodology pipeline steps.	19
3.2	Methodology Overview	20
3.3	Illustration of Federated PEFT LoRA. Each nodes shares the same frozen base LLM model, only the adapters are exchanged.	22
3.4	Overview of the general Safe Federated Fine-Tuning pipeline. The process separates the one-time local data assessment (Steps 1-2) from the iterative federated learning loop (Steps 3-5). The central server aggregates weight updates without ever accessing the raw local data.	24
3.5	Visual representation of Task Arithmetic. Figure reproduced from Ilharco et al. [12].	29

3.6	Architecture of Method 3 (Federated Task Arithmetic). The client splits its data and computes two distinct parameter updates. The server aggregates these into separate global representations and performs surgical vector negation to actively suppress the hazardous capabilities.	31
3.7	Architecture of RAP-FL...	33
3.8	Geometric representation of partial update sterilization in RAP-FL. The plot illustrates the projection mechanics within a simplified 2D parameter space. The estimated global utility (G_t^U) and risk (G_t^R) directions form an acute angle, reflecting the <i>entanglement</i> between safe capabilities and vulnerabilities in the model's weights. The original local update $\Delta\Phi_k^U$ (in blue) is decomposed by computing its orthogonal projection along the risk axis (transparent orange vector). By subtracting only a fraction of this component, modulated by the suppression factor $\lambda_t = 0.6$ (<i>illustrative value</i>) (solid orange vector), the sterilized update $\tilde{\Delta}\Phi_k^U$ (in green) is obtained. As highlighted by the decomposition parallelogram, this operation "bends" the vector away from the threat. Since $\lambda_t < 1$, the system accepts a slight residual risk (red arrow) in order to preserve a strong alignment with the global utility direction.	37
5.1	Centralized PEFT baselines evaluation. Left: WMDP accuracy demonstrates rapid hazardous capability acquisition for Threat and Mixed datasets, while the Safe-Only baseline remains successfully suppressed. Right: MMLU-Bio accuracy highlights the eventual utility degradation of the Safe-Only baseline due to variance loss, whereas Threat-Only improves general scientific utility.	66
5.2	Prediction Loss and Confidence Margin of the centralized baselines evaluated on the WMDP benchmark.	67
5.3	Prediction Loss and Confidence Margin of the centralized baselines evaluated on the MMLU-Bio benchmark.	67
5.4	Federated replication of baseline paradigms using standard FedAvg across 20 communication rounds. Left: WMDP accuracy shows the rapid integration of threat capabilities. Right: MMLU-Bio accuracy confirms the eventual variance-loss degradation in the Safe-Only network, mirroring centralized dynamics.	69
5.5	Prediction Loss and Confidence Margin of the federated replication evaluated on the WMDP benchmark.	70

5.6	Prediction Loss and Confidence Margin of the federated replication evaluated on the MMLU-Bio benchmark.	70
5.7	WMDP (Threat) and MMLU-Bio (Utility) accuracy across the three federated scenarios (S1–S3) over five training epochs. The results represent the average of two experimental seeds. Each row corresponds to one scenario; left plots track threat suppression, right plots track utility preservation. . .	72
5.8	Evaluation on General Knowledge (MMLU-NoBio) across defensive strategies run on 2 seeds, compared to the base model and centralized experiments. Singular seed results are provided for completeness in Appendix A.1.	76
A.1	WMDP and MMLU-Bio benchmark accuracy across the three federated scenarios (S1–S3) over five training epochs. Results obtained using experimental data seed 42	88
A.2	WMDP and MMLU-Bio benchmark accuracy across the three federated scenarios (S1–S3) over five training epochs. Results obtained using experimental data seed 73	89
A.3	MMLU-NoBio categories to evaluate general knowledge. Results obtained using experimental data seed 42	90
A.4	MMLU-NoBio categories to evaluate general knowledge. Results obtained using experimental data seed 73	90
A.5	Loss and prediction confidence trajectories evaluated on the WMDP benchmark.	92
A.6	Loss and prediction confidence trajectories evaluated on the MMLU-Bio dataset.	93

List of Tables

4.1	Summary of the training datasets composition built for the experiments. The domain split is consistently maintained across all paradigms to isolate the specific effects of the safety interventions.	47
4.2	Token-level statistics for the three constructed experimental datasets. Calculations are based on the tokenizer of the target generative model.	48
4.3	LoRA and PEFT hyperparameters applied to the target generative model.	50
4.4	Training and optimization hyperparameters applied across all centralized baseline experiments.	51
4.5	Distribution of Safe and Threat samples across the two federated clients for each experimental scenario. The total local dataset size remains strictly fixed at 6,016 samples per client to ensure uniform volume-based aggregation weights.	55
4.6	Impact of local safety filtering (Method 1) on client dataset sizes and subsequent FedAvg aggregation weights. Discarding threat samples dynamically alters the influence each client has on the global model update.	58
5.1	Comparison of terminal-epoch accuracy in the extreme asymmetric scenario (S1), averaged across two experimental seeds. In this scenario Method 2 (<i>Dynamic Weight</i>) achieves the best results.	74
5.2	Comparison of terminal-epoch accuracy in the mixed homogeneous scenario (S2), averaged across two experimental seeds. RAP-FL achieves the best threat suppression while simultaneously preserving benign utility.	75
5.3	Comparison of terminal-epoch accuracy in the heterogeneous scenario (S3), the most <i>realistic</i> scenario, averaged across two experimental seeds. RAP-FL achieves the best threat suppression while simultaneously preserving benign utility.	75

List of Symbols

Symbol	Description
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
FedAvg	Federated Averaging
FL	Federated Learning
GPU	Graphics Processing Unit
IID	Independent and Identically Distributed
LLM	Large Language Model
LoRA	Low-Rank Adaptation
MCQA	Multiple-Choice Question Answering
MLM	Masked Language Modeling
MMLU	Massive Multitask Language Understanding
NLP	Natural Language Processing
PEFT	Parameter-Efficient Fine-Tuning
QLoRA	Quantized Low-Rank Adaptation
RAP-FL	Risk-Aware Projection for Federated Learning
RLHF	Reinforcement Learning from Human Feedback
RMU	Representation Misdirection for Unlearning
SFT	Supervised Fine-Tuning
SVD	Singular Value Decomposition
VRAM	Video Random Access Memory
WMDP	Weapons of Mass Destruction Proxy

Acknowledgements

I would like to sincerely thank Professor Cinzia Cappiello, my thesis advisor, for giving me this opportunity and for her guidance throughout this work.

Special thanks also to Mattia Sabella, my co-advisor, for his constant support and feedback throughout the entire duration of this thesis. He also played a key role in the design of the RAP-FL method.

