Sovereign AI Inference: Hierarchical Semantic Architectures as a Prerequisite for Data Sovereignty and Composable Intelligence

Keijo Tuominen
Independent Researcher
cmht-tech@outlook.com

February 2026 — Version 1.5

Abstract

Current approaches to data sovereignty in AI systems attempt to retrofit removal capabilities onto architectures designed for irreversible knowledge integration. This paper argues that practical data sovereignty requires architectural separation of contributions, not retroactive unlearning from merged weights. We propose a two-layer sovereignty model built on hierarchical semantic inference architectures: coarse-grained isolation through domain shards, and fine-grained contributor sovereignty through per-contributor adapter modules composed at inference time. The architecture enables verifiable revocation—when a contributor withdraws, their adapter is removed and system behavior demonstrably changes. We present formal protocol specifications for contributor registration, ingestion, composition, and revocation, along with honest assessment of the performance tradeoffs inherent in composable versus merged architectures.

Keywords: data sovereignty, machine unlearning, hierarchical inference, LoRA adapters, composable AI, GDPR compliance, federated learning

1. Introduction — The Sovereignty Problem

Large Language Models have achieved remarkable capabilities by training on vast datasets aggregated from millions of sources. This aggregation creates a fundamental tension with data sovereignty requirements: once information is absorbed into model weights through gradient descent, extracting or removing specific contributions becomes computationally intractable. The knowledge is not stored in discrete, addressable locations but distributed across billions of parameters in ways that resist surgical removal.

Regulatory frameworks including GDPR Article 17 (Right to Erasure), CCPA deletion rights, and emerging AI-specific regulations assume that data controllers can remove individual contributions upon request. Current LLM architectures cannot satisfy this assumption. Approximate unlearning techniques exist but provide probabilistic rather than deterministic guarantees, leaving organizations legally exposed when contributors exercise withdrawal rights.

This paper proposes a different approach: rather than attempting to retrofit sovereignty onto architectures designed for irreversible integration, we design sovereignty into the architecture from the foundation.

The core insight is that sovereignty is an architectural property, not a post-hoc capability. Systems that merge contributions cannot cleanly separate them later; systems that maintain separation can.

We build on hierarchical semantic inference architectures to provide two layers of sovereignty. The first layer—domain shards—provides coarse-grained isolation by organizing knowledge into semantically coherent hierarchies. The second layer—per-contributor adapters—provides fine-grained sovereignty by maintaining each contributor's knowledge as a separate, composable module. Together, these layers enable verifiable revocation: when a contributor withdraws, their adapter is removed and system behavior demonstrably changes.

The paper presents formal protocol specifications for the complete contributor lifecycle, provides honest assessment of the performance tradeoffs inherent in composable versus merged architectures, and identifies open problems that require further research. We do not claim that this approach eliminates all challenges in data sovereignty for AI systems. We claim that it shifts the primary challenge from retroactive weight manipulation to distributed system coordination and modular composition. Significant open problems remain—particularly around adapter interference and dependency contamination (see Section 8)—but these problems are more amenable to incremental engineering progress than the fundamental irreversibility of gradient-based weight merging.

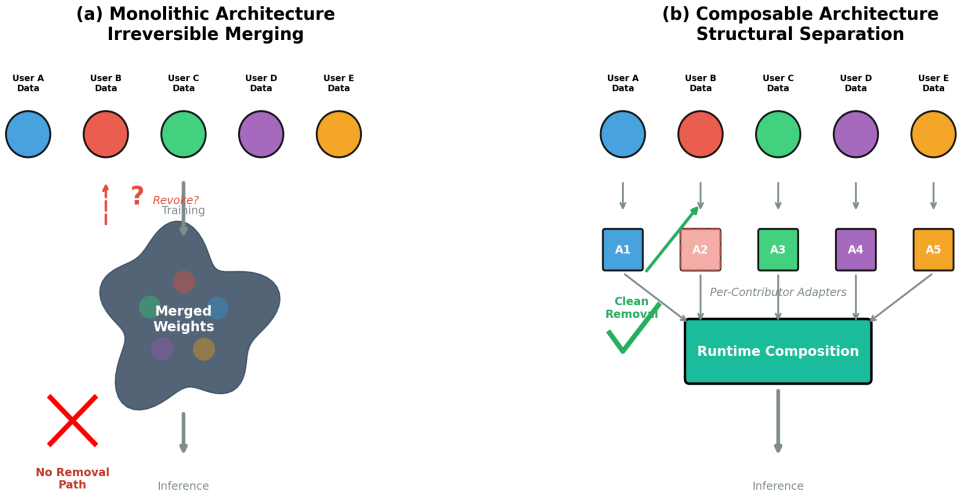

Figure 1: *The Irreversibility Problem. Left: Traditional training merges all contributions into shared weights, making extraction intractable. Right: Sovereignty architecture maintains separate adapters that can be independently removed.*

2. Why Current Approaches Fail

Before presenting our architecture, we examine why existing approaches to data sovereignty in LLMs fall short. Understanding these limitations motivates the design decisions that follow.

2.1 The Machine Unlearning Gap

Machine unlearning research has produced sophisticated techniques for removing the influence of specific training examples from models. Methods include influence function estimation, gradient-based unlearning, and model editing approaches. However, these techniques face fundamental limitations when applied to LLMs.

The core problem is entanglement. In a merged model, the contribution of any single training example is distributed across potentially all parameters, interacting with contributions from every other example. Removing one contribution without disturbing others requires understanding these interactions precisely—a requirement that scales poorly with model and dataset size. Current unlearning methods provide approximate removal with probabilistic guarantees, but regulators interpreting 'right to erasure' may require deterministic guarantees that approximate methods cannot provide.

2.2 Federated Learning Limitations

Federated Learning addresses data localization by keeping raw data on contributor devices and sharing only gradient updates. This satisfies some sovereignty requirements—data never leaves the contributor's control—but does not solve the revocation problem. Once gradients are aggregated into central model weights, the contribution is merged and faces the same extraction challenges as centrally-trained models.

The Federated Unlearning subfield addresses this gap through gradient subtraction and client-specific fine-tuning reversal, but these approaches inherit the approximation limitations of centralized unlearning methods.

2.3 The Retraining Cost Problem

The most reliable method for removing a contribution is retraining the model from scratch without the withdrawn data. This provides strong guarantees but at prohibitive cost. Training a frontier LLM costs millions of dollars and takes months. Retraining upon each withdrawal request is economically infeasible and creates unacceptable latency for compliance timelines.

SISA (Sharded, Isolated, Sliced, and Aggregated) training reduces retraining cost by partitioning data into shards trained separately, requiring only affected shards to be retrained upon withdrawal. This improves economics but still requires retraining, and the shard boundaries may not align with contributor boundaries.

2.4 The Verification Gap

Even if unlearning or retraining successfully removes a contribution, proving this to external auditors is challenging. How do you demonstrate that a model no longer contains specific knowledge? Membership inference attacks can provide statistical evidence, but adversarial examples can fool these tests. Formal verification of unlearning remains an open research problem.

The verification gap creates legal risk: even if an organization believes it has removed a contribution, it may be unable to prove compliance to regulators or in litigation.

3. The Two-Layer Sovereignty Model

Our architecture provides sovereignty through structural separation rather than retroactive removal. Two complementary layers work together to isolate contributions at different granularities.

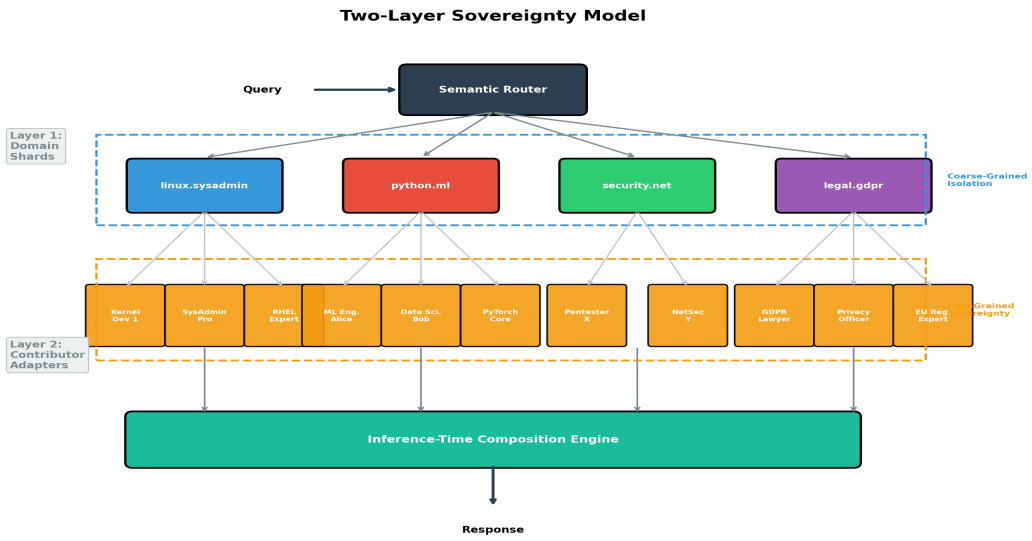

Figure 2: *Two-Layer Sovereignty Architecture. Layer 1 (Domain Shards) provides coarse-grained isolation through semantic hierarchy. Layer 2 (Per-Contributor Adapters) provides fine-grained sovereignty through composable LoRA modules.*

3.1 Layer 1: Domain Shards (Coarse-Grained Isolation)

The first sovereignty layer organizes knowledge into hierarchical domain shards, following the DNS-inspired architecture established in prior work. Rather than training a single monolithic model on all data, the system maintains specialized inference engines for distinct knowledge domains.

A semantic router directs queries to appropriate shards based on domain classification. A query about Python machine learning libraries routes to python.ml; a query about Linux kernel internals routes to linux.kernel. This routing provides natural isolation: contributors to the Python shard have no direct influence on Linux kernel responses.

Domain shards provide coarse-grained sovereignty. A contributor who wishes to withdraw from a specific domain can have their influence removed from that shard without affecting other domains. However, shards still face the within-shard entanglement problem—multiple contributors to the same shard have merged contributions.

3.2 Layer 2: Per-Contributor Adapters (Fine-Grained Sovereignty)

The second sovereignty layer addresses within-shard isolation through per-contributor adapter modules. Rather than merging contributor knowledge into shared shard weights, each contributor's knowledge is encoded in a separate Low-Rank Adaptation (LoRA) adapter that can be composed with the base model at inference time.

LoRA adapters work by learning low-rank update matrices that modify the base model's behavior without changing the base weights. Multiple adapters can be composed through weighted combination, allowing the system to draw on multiple contributors' knowledge to answer a query. Critically, each adapter remains a distinct artifact that can be added or removed independently.

When a contributor requests revocation, their adapter is deleted from the system. Subsequent queries that would have used that adapter now compose without it, producing demonstrably different outputs. No retraining is required; no approximate unlearning is attempted. The contribution is removed by removing the component that encoded it.

3.3 Composition Protocol

At inference time, the system must decide which adapters to compose for a given query. The Composition Protocol handles this through semantic relevance matching and reputation weighting.

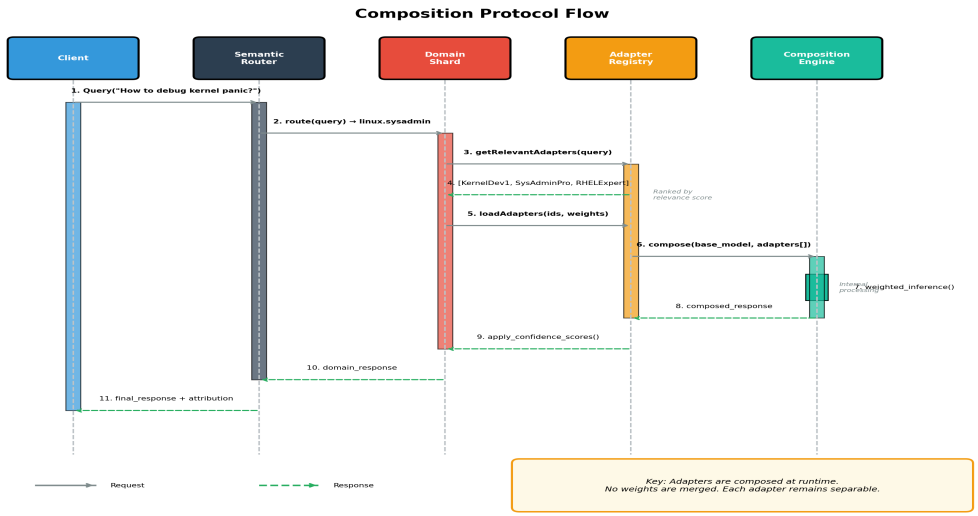

Figure 3: *Composition Protocol. Query Q is routed to shard S, relevant adapters are identified and weighted by reputation, composed output is generated with full attribution manifest.*

For a query Q routed to shard S, the system identifies adapters whose specializations match Q's semantic content. Matching adapters are composed with weights determined by relevance scores and contributor reputation. The composed model processes Q and generates a response. An attribution manifest records which adapters contributed to the response, enabling audit trails and proper credit assignment.

4. Contributor Registration Protocol

Contributors must register with the system before their knowledge can be ingested. The Registration Protocol establishes identity, negotiates terms, and provisions infrastructure for the contributor's adapter.

4.1 Identity Verification

Contributors provide identity credentials appropriate to their type. Individual contributors may use government-issued identification, professional credentials, or cryptographic identity proofs. Organizational contributors provide business registration, authorized representative verification, and delegation chains for employees contributing on behalf of the organization.

Identity verification serves multiple purposes: it enables accountability for contribution quality, supports reputation systems, and provides the legal identity needed for contractual terms and regulatory compliance.

4.2 Terms Negotiation

Contributors and the system operator negotiate contribution terms covering: scope of use (which queries may invoke the contributor's adapter), attribution requirements (how contributions are credited), compensation terms (if applicable), revocation conditions (notice periods, transition handling), and data handling (what the system may store about the contributor's interactions).

Terms are recorded in a Contribution Agreement that serves as the legal basis for the relationship. Smart contract implementations can automate term enforcement for suitable provisions.

4.3 Adapter Provisioning

Upon successful registration, the system provisions infrastructure for the contributor's adapter. This includes: storage allocation for adapter weights, compute allocation for training (if system-side training is used), API credentials for contribution submission, and monitoring dashboards for contribution status and usage statistics.

5. Contribution Ingestion Protocol

The Ingestion Protocol governs how contributor knowledge is transformed into adapter weights. Two primary ingestion modes are supported.

5.1 System-Side Training

In system-side training, contributors provide training data and the system trains adapters on their behalf. Contributors upload datasets through secure channels, specifying target shards for their contribution. The system validates data format and quality, trains a LoRA adapter using the contributor's data against the appropriate base model, runs quality checks and safety filters on the resulting adapter, and upon passing validation, registers the adapter for composition.

System-side training provides convenience and consistency but requires contributors to share their raw data with the system operator.

5.2 Contributor-Side Training

In contributor-side training, contributors train adapters locally and upload only the resulting weights. The system provides base model checkpoints and training specifications. Contributors train adapters on their local infrastructure using their private data. Only adapter weights are uploaded—raw training data never leaves contributor control. The system validates adapter compatibility and runs safety checks before registration.

Contributor-side training preserves data confidentiality but requires contributors to have training infrastructure and expertise.

5.3 Validation Pipeline

All adapters pass through a validation pipeline before becoming eligible for composition. Format validation ensures adapter architecture matches system specifications. Quality validation tests adapter performance on held-out evaluation sets. Safety validation checks for harmful outputs, bias amplification, and policy violations. Compatibility validation verifies the adapter composes correctly with existing adapters.

Adapters failing validation are rejected with diagnostic feedback. Contributors may revise and resubmit.

6. Reputation and Weighting System

Not all contributions are equal. The reputation system tracks contributor quality and adjusts composition weights accordingly.

6.1 Reputation Signals

Contributor reputation is computed from multiple signals. Peer validation captures endorsements and critiques from other contributors in the same domain. Outcome verification tracks whether responses using the contributor's adapter satisfy user needs (measured through feedback signals). Consistency monitoring detects contribution quality changes over time. External credentials incorporate verified expertise indicators such as professional certifications or publication records.

6.2 Composition Weighting

When multiple adapters are composed for a query, their relative weights reflect both semantic relevance and reputation scores. High-reputation contributors have greater influence on outputs than low-reputation contributors for equally relevant queries. This creates incentives for quality and allows the system to gracefully handle varying contribution quality.

6.3 Gaming Resistance

Reputation systems are vulnerable to gaming. Countermeasures include temporal weighting (recent signals count more than old ones), peer validation networks (endorsements from high-reputation contributors carry more weight), anomaly detection (sudden reputation changes trigger review), and outcome-based grounding (ultimately, reputation reflects whether contributions help users).

7. Revocation Protocol

The Revocation Protocol implements the 'right to be forgotten' in a verifiable manner. When a contributor withdraws, their influence is removed completely rather than approximately.

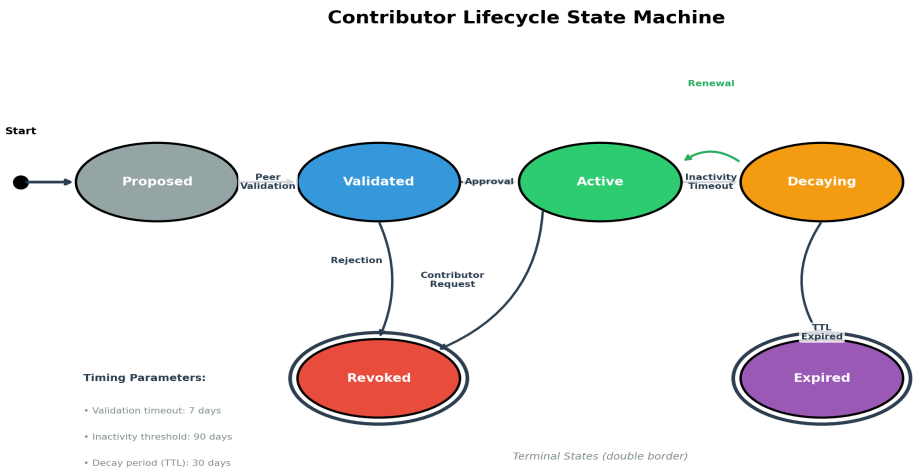

Figure 4: *Contributor Lifecycle State Machine. Contributions progress from Proposed through Validated to Active status. Revocation can occur from Active state upon contributor request, resulting in complete adapter removal.*

7.1 Revocation Request

Contributors initiate revocation through authenticated requests specifying scope (full withdrawal or partial, e.g., specific shards only) and timing (immediate or scheduled). The system acknowledges the request and begins the revocation process.

7.2 Adapter Removal

The system removes the contributor's adapter from all composition pools, deletes adapter weights from storage, updates routing tables to exclude the contributor, and generates a deletion certificate with cryptographic proof of removal.

7.3 Verification

Post-revocation verification demonstrates that the contribution has been removed. The system can show that queries previously using the revoked adapter now produce different outputs, that the adapter weights are no longer present in storage, and that attribution manifests no longer reference the contributor. External auditors can verify these claims through API access and storage inspection.

7.4 Cascade Handling

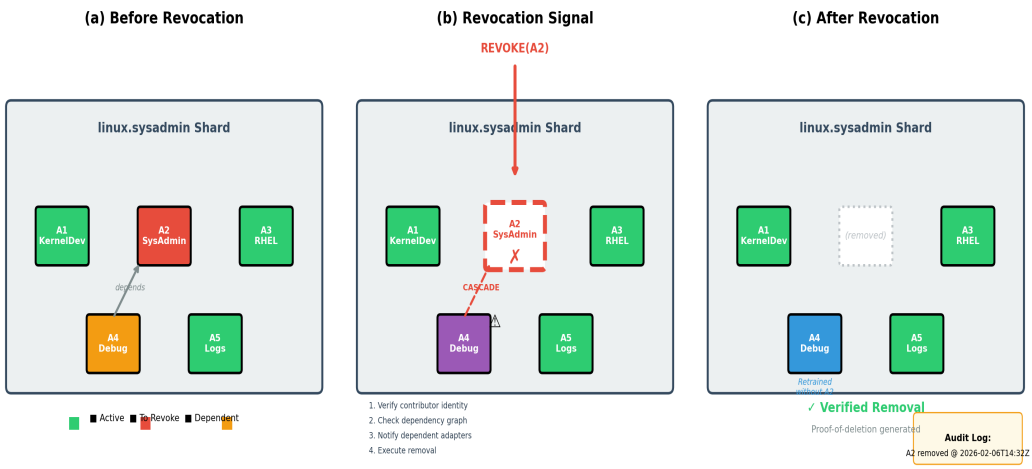

Figure 5: *Revocation Protocol. (a) Before: all adapters active with dependencies. (b) During: revocation signal triggers cascade check. (c) After: clean removal with proof-of-deletion and updated composition pools.*

When a contributor revokes, other adapters may depend on their contribution. The system maintains a dependency graph tracking which adapters reference concepts or patterns from other adapters. Upon revocation, dependent adapters receive notification and have three options: retrain without the dependency, find an alternative source, or accept reduced functionality. Mass revocation events trigger coordinated cascade handling to minimize disruption.

8. Practical Constraints and Open Problems

The composable sovereignty architecture involves real tradeoffs and leaves significant problems unsolved. This section provides an honest assessment of the costs, limitations, and areas requiring further research.

8.1 Performance Cost of Composition

Composing adapters at inference time is more expensive than using a single merged model. Each additional adapter in the composition adds latency for loading, weighting, and applying. Preliminary measurements suggest 15-30% latency overhead for compositions of 3-5 adapters compared to a single fine-tuned model.

This overhead compounds at scale. A production system serving queries that require knowledge from 50 or more contributor adapters could see latency measured in seconds per token, rendering real-time applications infeasible. Techniques such as adapter caching, speculative loading, and tiered composition may mitigate this, but the fundamental tradeoff between sovereignty granularity and inference speed remains.

8.2 Adapter Interference During Composition

The Composition Protocol assumes that weighted combination of LoRA adapters produces coherent outputs. This assumption is not always valid. Research on multi-adapter composition (Huang et al., 2023; Zhang et al., 2023) has documented that adapters trained independently can interfere destructively when combined. Their low-rank subspaces may collide in the activation space, leading to degraded performance.

8.3 Shared Activation Space and Base Model Dependency

While LoRA adapters do not modify base model weights, they operate within the base model's activation space. If the base model is updated, adapter behavior may change unpredictably. Additionally, the base model itself was trained on data that may include contributions the system claims to have isolated. The sovereignty guarantees apply only to the adapter layer; the base model remains monolithic.

8.4 Dependency Contamination

If Adapter B was trained using outputs or synthetic data influenced by Adapter A, then revoking A does not remove A's influence from B. This is the same irreversibility problem the architecture criticizes in monolithic models, displaced by one layer. Mitigations include prohibiting adapter-to-adapter training dependencies or tracking dependency graphs for cascade retraining.

8.5 Proof-of-Deletion Specification

The Revocation Protocol includes deletion proofs but does not fully specify the cryptographic mechanisms. A rigorous proof would need to demonstrate irrecoverable deletion from all storage, absence of cached compositions, and verifiability by external auditors without requiring trust in the system operator.

8.6 Storage and Convergence

Per-contributor adapters consume storage proportional to contributor count (approximately 40MB per adapter for a 7B model with rank-16 LoRA). With 10,000 contributors, adapter storage reaches 400GB per shard. Additionally, non-IID data distributions in federated settings can cause convergence problems.

8.7 Cold-Start and Reputation Gaming

New shards face bootstrapping challenges: without existing contributors to validate submissions, initial quality control is difficult. The reputation system is also a potential attack surface for score inflation and collusion.

8.8 Mass Revocation

The system handles steady-state revocation well but may struggle with mass revocation events where a company or jurisdiction withdraws all contributions simultaneously. Mechanisms for graceful degradation need further development.

8.9 Attribution Accuracy

A natural question is: what is the false positive rate for contributor attribution? The answer depends fundamentally on the attribution approach and how 'attribution' is defined.

Traditional approaches attempt to trace influence backward through merged weights, asking 'which training examples influenced this output?' This is fundamentally a statistical inference problem with inherent uncertainty. Influence functions, gradient-based attribution, and membership inference all produce probabilistic estimates with meaningful false positive rates.

The sovereignty architecture takes a different approach: attribution records which adapters were composed, not which adapters influenced the output. Each adapter maps 1:1 to a contributor. At inference time, the system logs which adapters were included in the composition. If Adapter X was in the composition, Contributor X is recorded as having been invoked. If not, they are not.

For this narrow definition—recording which adapters were used—the false positive rate is zero. The system does not guess; it observes. However, this assumes that system logging is accurate and that adapter selection is deterministic. Errors in logging, adapter misconfiguration, or bugs in the composition protocol could introduce errors even in this structural approach.

More importantly, adapter usage does not necessarily equal proportional influence. An adapter might be included in a composition but contribute minimally to the output, or be triggered spuriously for queries outside its specialization. The question 'did this adapter contribute?' is different from 'how much did this adapter contribute?' The architecture answers the first question definitively but leaves the second open.

The dependency contamination problem (Section 8.4) introduces additional attribution complexity. If Adapter B learned from Adapter A's outputs, attributing an output solely to Contributor B understates A's indirect influence. This is a false negative rather than a false positive, but it represents incomplete attribution nonetheless. Potential mitigations include influence tracking during adapter training, dependency graph analysis, and explicit labeling of derived adapters. Complete attribution accounting for indirect influence remains an open research problem.

8.10 Decentralized Training Compatibility

The sovereignty architecture was designed for centralized deployment, but it can be adapted to emerging decentralized training paradigms. While centralized GPU clusters remain the dominant approach for large-scale model training, there is increasing interest in leveraging NPUs (Neural Processing Units) and CPU extensions like Intel's AMX (Advanced Matrix Extensions) for distributed training on consumer hardware.

This trend has implications for sovereignty. In centralized training, contributions merge into monolithic weights controlled by a single operator. In decentralized training, each node contributes independently—and the per-contributor adapter model maps to per-node contributions. The adapter-per-contributor pattern becomes adapter-per-node in a distributed swarm. Each host trains its own adapter on local data. Adapters are shared across the network; raw training data never leaves the originating node.

Projects using blockchain for 'Proof-of-Learning'—cryptographic verification that a node performed claimed training work—can integrate with the sovereignty architecture. The contributor registry and adapter manifests provide the attribution layer these protocols need. Node identity maps to contributor identity. Adapter hashes provide verifiable artifacts. Revocation follows the same pattern as in the centralized setting: disconnect the node and remove its adapter from the composition pool.

Distributed swarms have nodes going offline unpredictably. The composition protocol handles this scenario: if an adapter is unavailable, the system composes without it and records reduced specialization coverage—meaning fewer domain-specific adapters are available for queries in that area, potentially resulting in less specialized responses. This maps naturally to swarm elasticity where nodes join and leave dynamically.

If consumer-grade training on distributed CPU/NPU infrastructure becomes practical, the sovereignty architecture provides a potential accountability layer. Centralized training on GPU clusters naturally leads to merged weights controlled by whoever operates the cluster. Distributed training, by contrast, can preserve contribution boundaries if the architecture is designed to maintain them. The sovereignty model offers one such design, though significant engineering challenges remain in adapting it to fully decentralized settings.

Looking ahead: If current trends continue—Linux kernel NPU subsystem maturation (amdxdna, Intel ivpu drivers in kernel 7.0), AMX instruction optimization, and projects like Petals and Gensyn gaining traction—consumer-grade distributed training could become viable by late 2026. In that scenario, the sovereignty architecture provides a natural fit: the per-contributor adapter model maps directly to per-node contributions in a distributed swarm. The GPU concentration that created centralized training also created the 'everything merged into one blob' problem. Decentralized training on commodity hardware would naturally enforce the separation that this architecture requires—not as a design choice, but as a consequence of the distributed system constraints.

9. Related Work

This section positions the sovereignty architecture relative to existing research.

9.1 Machine Unlearning

The machine unlearning literature addresses the same problem from a different angle: given a merged model, how do we remove specific data? This paper asks: how do we avoid merging in the first place? The approaches are complementary.

9.2 Federated Learning and Federated Unlearning

Federated Learning keeps raw data decentralized. The Federated Unlearning subfield (Liu et al., 2021; Wu et al., 2022; Halimi et al., 2022) addresses removing client contributions through gradient subtraction and influence estimation. The sovereignty architecture differs in that it prevents merging rather than reversing it.

9.3 Modular Deep Learning

The broader field of modular deep learning (Pfeiffer et al., 2023; Andreas et al., 2016) explores architectures where capabilities are decomposed into composable units. The sovereignty architecture builds on this foundation, adding explicit framing around data sovereignty and contributor lifecycle management.

9.4 Multi-Adapter Composition

LoRA-Hub (Huang et al., 2023) demonstrated that adapters trained for different tasks can be combined through weighted composition. The sovereignty architecture inherits both the capabilities and the challenges (interference, subspace collision) from this work.

9.5 Decentralized AI Projects

Projects like Bittensor, Ocean Protocol, and Petals (Borzunov et al., 2022) address decentralized training and inference. These solve problems in incentives and governance but do not directly address structural sovereignty. The sovereignty model could integrate with these projects, using their layers while providing structural separation.

10. Conclusion

Data sovereignty in AI systems is not primarily a machine learning problem—it is an architecture problem. Systems designed for irreversible knowledge integration cannot satisfy requirements for verifiable removal. Systems designed for composable, separable contributions can.

The two-layer sovereignty model presented in this paper—domain shards for coarse isolation, per-contributor adapters for fine-grained sovereignty—provides a path toward AI systems that respect contributor rights without sacrificing capability. The architecture enables verifiable revocation: when contributors withdraw, their influence is removed completely and demonstrably, not approximately.

We do not claim this architecture solves all problems. Significant challenges remain in adapter composition, dependency management, and proof-of-deletion specification. Performance tradeoffs are real and may be unacceptable for some applications. But these are engineering challenges amenable to incremental progress, not fundamental impossibilities.

As AI systems become critical infrastructure, the question of who controls the knowledge they encode becomes increasingly important. The sovereignty architecture provides one answer: the contributors who provided the knowledge retain meaningful control over it, including the right to withdraw. This is not just a regulatory compliance measure—it is a foundation for AI systems that humans can trust.

References

Andreas, J., Rohrbach, M., Darrell, T., & Klein, D. (2016). Neural module networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 39-48.

Borzunov, A., Baranchuk, D., Dettmers, T., et al. (2022). Petals: Collaborative inference and fine-tuning of large models. arXiv:2209.01188.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C., et al. (2021). Machine unlearning. IEEE Symposium on Security and Privacy, 141-159.

Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch Transformers: Scaling to trillion parameter models. JMLR, 23(120), 1-39.

Halimi, A., Kadhe, S., Rawat, A., & Baracaldo, N. (2022). Federated unlearning: How to efficiently erase a client in FL? arXiv:2207.05521.

Hu, E. J., Shen, Y., Wallis, P., et al. (2021). LoRA: Low-rank adaptation of large language models. arXiv:2106.09685.

Huang, C., Liu, Q., Lin, B. Y., et al. (2023). LoRA-Hub: Efficient cross-task generalization via dynamic LoRA composition. arXiv:2307.13269.

Jiang, A. Q., Sablayrolles, A., Mensch, A., et al. (2024). Mixtral of Experts. arXiv:2401.04088.

Liu, G., Ma, X., Yang, Y., Wang, C., & Liu, J. (2021). Federated unlearning. arXiv:2012.13891.

Pfeiffer, J., Ruder, S., Vulic, I., & Ponti, E. M. (2023). Modular deep learning. arXiv:2302.11529.

Rao, J., & Oppenheimer, J. (2021). Bittensor: A peer-to-peer intelligence market. Bittensor Foundation Whitepaper.

Tuominen, K. (2025). Hierarchical Semantic Large Language Model Architectures: A DNS-Inspired Approach. Technical Report.

Wu, C., Zhu, S., & Mitra, P. (2022). Federated unlearning with knowledge distillation. arXiv:2201.09441.

Zhang, Q., Chen, M., Bukharin, A., et al. (2023). AdaMerging: Adaptive model merging for multi-task learning. arXiv:2310.02575.