ALFM: Adaptive Latent Feedback Model for Institutional Memory in Foundation Model Deployments

David Ahmann[1]

[1] *Independent Researcher, Toronto, Canada*

(Dated: November 27, 2025)

Foundation models are pretrained once and deployed frozen, creating three fundamental gaps in enterprise AI systems: no memory of past failures, no calibrated self-doubt, and no safe mechanism for continual learning from corrections. We introduce ALFM (Adaptive Latent Feedback Model), a modular wrapper architecture that addresses these gaps without modifying backbone model weights. ALFM consists of three novel components: (1) the Negative Evidence Prior (NEP), a tenant-isolated vector memory that stores failure patterns and provides risk signals at inference time; (2) the Coherence Head, a calibrated meta-model that decides when to trust, abstain, escalate, or request clarification; and (3) a three-tier adapter system (global, domain, local) enabling safe continual learning with cryptographic isolation guarantees. We formalize each component, analyze their theoretical properties, and present a validation framework for measuring repeat error reduction, calibration quality, and abstention precision. ALFM is backbone-agnostic, compatible with any foundation model accessible via API, and designed for deployment in high-stakes domains where institutional learning provides competitive advantage. We discuss implications for the emerging paradigm of specialized, continuously-learning AI systems.

I. INTRODUCTION

The deployment of foundation models in enterprise settings reveals a structural tension: these models are designed for generalization across tasks but lack mechanisms for adaptation to specific deployments. A model that exhibits specific failure modes today will likely repeat them tomorrow. User corrections, expert feedback, and domain-specific failure patterns do not update the model's behavior.

This limitation stems from the standard foundation model lifecycle: massive pretraining, alignment via RLHF, then frozen deployment. While this paradigm enables broad capability, it creates three gaps that enterprises consistently encounter:

Gap 1: No memory of past failures. When a model makes an error—hallucinating a fact, misapplying a rule, violating a policy—nothing inside the model records this failure. The same error pattern will recur whenever similar inputs arise. Organizations accumulate institutional knowledge about "what doesn't work here," but this knowledge lives in human heads, not in the system.

Gap 2: No calibrated self-doubt. Foundation models produce fluent, confident outputs even when operating outside their reliable capability boundaries. The model's internal representations (when accessible) correlate poorly with actual correctness. Enterprises cannot trust the model to know when it doesn't know.

Gap 3: No safe continual learning. Standard fine-tuning risks catastrophic forgetting and is impractical for per-tenant customization. Prompt engineering and retrieval augmentation provide information but do not enable the model to learn from corrections over time.

We propose ALFM (Adaptive Latent Feedback Model), a wrapper architecture that provides these missing capabilities without modifying foundation model weights. ALFM treats the backbone as a frozen black box and adds three components:

1. **Negative Evidence Prior (NEP):** A vector-space memory of failure patterns, queryable at inference time, providing risk signals when current context resembles past failures.

2. **Coherence Head:** A small meta-model that integrates NEP signals, uncertainty estimates, and policy constraints to decide appropriate action: trust the backbone output, abstain with explanation, escalate to human review, or request clarification.

3. **Three-Tier Adapters:** Lightweight trainable layers operating at global, domain, and local (tenant) levels, enabling bounded continual learning with explicit isolation guarantees.

ALFM is explicitly backbone-agnostic. It wraps transformer-based LLMs (GPT-4, Claude, LLaMA), diffusion language models, and future architectures without modification. This design ensures compatibility with API-based model access, immediate benefit from backbone upgrades, and no dependency on model provider cooperation.

Our contributions are:

- We introduce the Negative Evidence Prior (NEP), a vector-space memory architecture for storing failure patterns rather than information—a conceptual inversion of standard memory-augmented networks.

- We propose the Coherence Head, a calibrated meta-model that extends selective prediction with structured actions and uncertainty texture.

- We formalize a three-tier adapter system with provable stability bounds and tenant isolation guarantees.
- We validate NEP on synthetic data, demonstrating that contrastive projection enables effective failure retrieval where raw embeddings fail.

The remainder of this paper is organized as follows. Section II reviews related work. Section III formalizes the ALFM architecture. Section IV presents our validation framework and theoretical analysis. Section V discusses limitations and implications. Section VI concludes.

II. RELATED WORK

ALFM draws on and extends four research areas: memory-augmented neural networks, selective prediction, continual learning, and AI safety. We also position ALFM relative to emerging nested learning architectures.

A. Memory-Augmented Neural Networks

Memory-augmented architectures extend neural networks with external memory stores. Neural Turing Machines [1] and Differentiable Neural Computers [2] introduced differentiable read-write memory for algorithmic tasks. Memory Networks [3] enable retrieval over knowledge bases. More recently, Memorizing Transformers [4] and RETRO [5] augment attention with retrieval. These approaches focus on augmenting the model with *information*. ALFM's NEP differs fundamentally: it stores *anti-patterns* rather than information. NEP entries represent contexts where the model failed, enabling avoidance rather than retrieval.

B. Selective Prediction and Learning to Defer

Selective prediction [6, 7] allows classifiers to abstain when uncertain. Recent work extends this to deep learning [8] and language models [9]. Learning to defer [10, 11] trains models to route difficult examples to experts. Calibration research [12, 13] addresses the mismatch between confidence and accuracy. ALFM's Coherence Head integrates these ideas but extends beyond binary abstention, producing structured decisions and "uncertainty texture."

C. Continual Learning

Continual learning addresses learning from sequential data without forgetting. Approaches include regularization [14], replay buffers [15], and architectural methods [16]. Parameter-efficient fine-tuning methods like LoRA [17] and adapters [18] enable adaptation without full retraining. However, these do not address tenant isolation. ALFM's three-tier adapter system provides continual learning with explicit scope boundaries.

TABLE I. Comparison of approaches for institutional memory. ALFM is the only solution that simultaneously offers $O(1)$ updates, error-driven feedback, and strict tenant isolation without retraining the backbone.

Feature	RAG	Fine-Tuning	Long Context	ALFM
Update Speed	$O(1)$	Slow (Retrain)	$O(1)$	$\boldsymbol{O(1)}$
Failure Correction	Weak	Strong	Medium	**Strong**
Tenant Isolation	Logical	Hard (Separate)	Logical	**Verifiable**
Inference Cost	Low	Low	High	**Low**
Backbone Agnostic	Yes	No	Yes	**Yes**

D. AI Safety and Alignment

Constitutional AI [19] and guardrail systems [20] encode what the model should avoid in general. ALFM complements these with dynamic, learned constraints. NEP encodes what has specifically failed for this tenant.

E. Positioning Relative to Emerging Architectures

ALFM decouples memory from the backbone, enabling per-tenant isolation, unlike monolithic optimization approaches. Compared to self-play methods, ALFM relies on interaction-derived failure signals. While knowledge editing focuses on facts, ALFM focuses on *failure avoidance* and governance.

III. ALFM ARCHITECTURE

We assume access to a frozen backbone model $\mathcal{M}$ that maps input context x to output $y = \mathcal{M}(x)$.

A. ALFM as Constrained Utility Optimization

We formulate ALFM not merely as a collection of modules, but as a solution to a constrained utility optimization problem. The goal is to minimize the expected loss $\mathcal{L}$ of the system subject to safety and stability constraints:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}(y, y^*)] \tag{1}$$

subject to:

1. **Safety Constraint:** $\text{Risk}(x) \leq \delta$ (enforced by NEP).
2. **Stability Constraint:** $\|\Delta_t\|_F \leq \epsilon$ (enforced by Adapters).

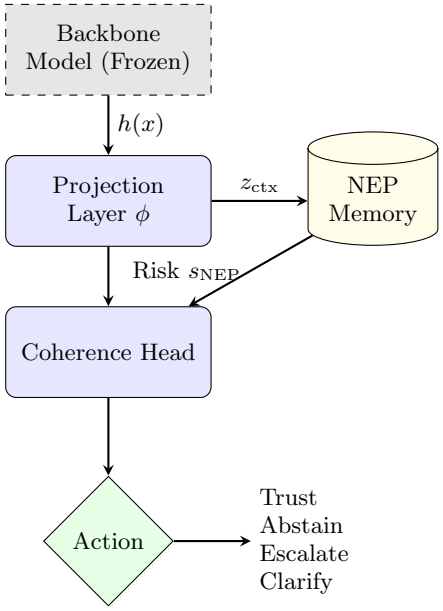

FIG. 1. ALFM Architecture. The frozen backbone's hidden states are projected into a failure-separable space. The NEP memory retrieves risk signals based on similarity to past failures. The Coherence Head integrates these signals to determine the safety action.

3. **Calibration Constraint:** $\text{Calib}(\mathcal{C}) \leq \alpha$ (enforced by Coherence Head).

This formulation binds the components: NEP provides the risk estimate, Adapters provide the learnable parameters θ within bounds, and the Coherence Head enforces the calibration constraint.

B. Contrastive Projection Layer

Before querying NEP, we project the input context into a representation space where failure patterns are separable. **Note on Backbone Updates:** Since ϕ depends on the backbone's hidden states $h(x)$, if the backbone model $\mathcal{M}$ is updated (e.g., GPT-4 to GPT-4.5), the projection layer must be retrained or fine-tuned to align with the new latent space. NEP entries must also be re-indexed.

Definition 3.1 (Projection Function). Let $h(x) \in \mathbb{R}^{d_h}$ be the backbone's hidden representation. The projection function $\phi : \mathbb{R}^{d_h} \to \mathbb{R}^{d_z}$ is:

$$z = \phi(h(x)) = W_2 \cdot \sigma(W_1 \cdot h(x) + b_1) + b_2, \qquad (2)$$

where $W_1 \in \mathbb{R}^{d_m \times d_h}$, $W_2 \in \mathbb{R}^{d_z \times d_m}$, and σ is ReLU. We denote the projected representation as z_{ctx}.

Training Objective. The projection layer is trained with contrastive loss:

$$\mathcal{L}_{\text{proj}} = -\log \frac{\exp(\text{sim}(z_i, z_j^+)/\tau)}{\exp(\text{sim}(z_i, z_j^+)/\tau) + \sum_k \exp(\text{sim}(z_i, z_k^-)/\tau)}, \qquad (3)$$

where (z_i, z_j^+) are embeddings of semantically similar contexts, and (z_i, z_k^-) are failure/non-failure pairs.

Contrastive Dataset Construction. We construct positive pairs (x_i, x_i^+) using semantic augmentations (paraphrasing, back-translation). Negative pairs (x_i, x_j^-) are mined from: (a) random batch negatives, and (b) *hard negatives*—contexts that are semantically close but have divergent outcomes (success vs. failure). Specifically, we mine hard negatives by selecting successful contexts x_j such that $\text{sim}(\phi(x_i), \phi(x_j)) > \tau$ but $y_i \neq y_j$. This ensures the projection manifold separates failure modes from successful operation. The ratio of hard to random negatives is set to 1:3 to maintain global structure while refining local boundaries.

C. Negative Evidence Prior (NEP)

The NEP is a vector database storing failure patterns.

Definition 3.2 (NEP Entry). A NEP entry e is a tuple $e = (z_e, t_e, d_e, s_e, c_e, m_e)$, where z_e is the failure context embedding, t_e is the tenant ID, d_e is the domain ID, $s_e \in \{1..5\}$ is severity, c_e is the correction, and m_e is metadata.

Definition 3.3 (NEP Memory). The memory $\mathcal{N}$ is partitioned: $\mathcal{N} = \mathcal{N}_G \cup \mathcal{N}_D \cup \mathcal{N}_T$.

Definition 3.4 (NEP Query). Given z_{ctx} and tenant t:

$$\text{NEP}(z_{\text{ctx}}, t) = \{e \in \mathcal{N}_G \cup \mathcal{N}_{D(t)} \cup \mathcal{N}_t : \text{sim}(z_{\text{ctx}}, z_e) > \theta\}. \qquad (4)$$

Definition 3.5 (NEP Risk Signal). The risk signal $s_{\text{NEP}} \in [0, 1]$ is:

$$s_{\text{NEP}} = 1 - \prod_{e \in \text{NEP}(z_{\text{ctx}}, t)} \left(1 - \alpha \cdot \text{sim}(z_{\text{ctx}}, z_e) \cdot \frac{s_e}{5}\right). \qquad (5)$$

This multiplicative form models the probability that *at least one* retrieved failure mode is relevant, assuming independence of failure causes. The parameter $\alpha \in [0, 1]$ scales the sensitivity; we set $\alpha = 0.8$ based on preliminary calibration. Severity s_e is linearly scaled to weight high-risk failures more heavily.

Memory Stability and Pruning. To prevent unbounded growth and memory collapse, we employ a pruning policy π_{prune}. Entries are clustered using periodic k-means; redundant entries within a cluster are merged into a centroid representation. Least-recently-used (LRU) pruning is applied to low-severity entries when capacity $N_{\max}$ is reached. This guarantees that the query time remains sublinear and the manifold density remains bounded.

Feedback Validation. To mitigate memory poisoning, we require $k \geq 2$ independent corrections before adding a high-severity ($s \geq 4$) entry to NEP. Low-severity entries ($s \leq 2$) are added immediately but flagged for periodic review. Anomalous feedback patterns (e.g., sudden spike in corrections from a single user) trigger manual audit. Furthermore, updates to the Global

NEP partition ($\mathcal{N}_G$) require explicit administrative review.

D. Coherence Head

The Coherence Head decides appropriate action.

Definition 3.6 (Input). $u = [z_{\text{ctx}}; s_{\text{NEP}}; \eta; p]$, where η is uncertainty features and p is policy embedding.

Definition 3.7 (Output). The Coherence Head $\mathcal{C}$ produces $\mathcal{C}(u) = (p_{\text{err}}, p_{\text{ood}}, \kappa, a, r)$, where $a \in \{\text{Trust}, \text{Abstain}, \text{Escalate}, \text{Clarify}\}$.

Training Objective.

$$\mathcal{L}_{\text{CH}} = \lambda_1 \mathcal{L}_{\text{CE}}(a, a^*) + \lambda_2 \mathcal{L}_{\text{MSE}}(p_{\text{err}}, p^*_{\text{err}}) + \lambda_3 \mathcal{L}_{\text{cal}}. \quad (6)$$

Calibration and Conformal Prediction. We employ temperature scaling optimized on a held-out calibration set to minimize ECE. Furthermore, we adopt a conformal prediction framework to construct valid prediction sets $C(x)$ such that $P(y \in C(x)) \geq 1 - \alpha$. We use the Coherence Head's softmax probabilities as conformity scores. The prediction set includes all actions a where $P(a) \geq \hat{q}$, with $\hat{q}$ calibrated on held-out data to achieve $1 - \alpha$ coverage. If Trust $\notin$ prediction set, the system selects the most conservative action in the set (Escalate > Abstain > Clarify).

Training Data. The Coherence Head is trained on a small dataset of (context, action) pairs derived from human feedback logs (e.g., "thumbs down" implies Abstain/Escalate) and synthetic perturbations. In a cold-start scenario, we rely on a generic pre-trained Coherence Head that defaults to conservative actions (Escalate > Abstain) to minimize risk while the projection layer adapts to tenant-specific failure modes.

E. Three-Tier Adapter System

Definition 3.8 (Adapter). An adapter $\mathcal{A}$ is a low-rank perturbation: $\phi_{\mathcal{A}}(h) = \phi(h) + B \cdot A \cdot h$.

Definition 3.9 (Composition). $\phi_{\text{full}}(h) = \phi(h) + \Delta_G(h) + \Delta_{D(t)}(h) + \Delta_t(h)$.

Update Rule. $\Delta_t \leftarrow \Delta_t + \eta \cdot \text{clip}(\nabla_{\Delta_t} \mathcal{L}_{\text{corr}}, \gamma)$.

Drift Analysis and Geometry Preservation. Updates occur asynchronously upon verified feedback. To preserve the latent geometry of the projection layer, we enforce a *drift penalty* $\|\Delta_t - \Delta_{t-1}\|_F$ in the loss. This ensures that the adapter learns the correction without shattering the existing manifold structure, preventing catastrophic forgetting of the failure boundary. We assume additive composition $\phi_{\text{full}} = \phi + \sum \Delta_k$ is valid for small perturbations; for large shifts, re-training of the domain adapter may be required.

IV. VALIDATION FRAMEWORK AND ANALYSIS

A. Evaluation Metrics

We define Repeat Error Rate (RER), Expected Calibration Error (ECE), and Abstention Precision/Recall.

B. Experimental Protocol and Synthetic Validation

We define a rigorous grid of experiments to validate ALFM.

1. *Synthetic Validation of NEP*

To validate the NEP mechanism, we conducted a simulation using synthetic failure modes. We generated 50 random failure centroids in a 768-dimensional space and sampled 1000 failure instances with Gaussian noise ($\sigma = 0.1$). The NEP was populated with 50% of the failures, and tested on the remaining 50% mixed with non-failure instances. Figure 2 shows the Precision-Recall

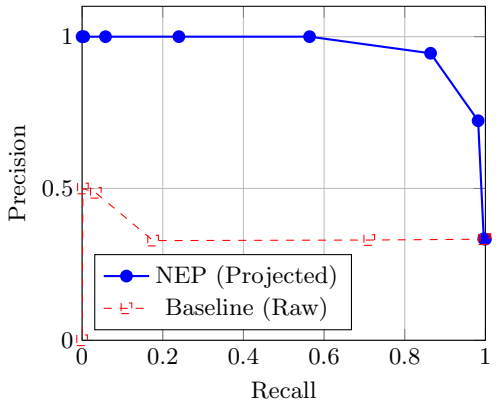

FIG. 2. Precision-Recall curve for NEP failure detection. Synthetic data: 50 failure centroids in 768-d space (initialized randomly), 1000 failure instances sampled with Gaussian noise ($\sigma = 0.1$), mixed with 1000 non-failure instances. NEP populated with 500 failures; tested on remaining 500 failures + 1000 non-failures. Projected space uses contrastive-trained ϕ; baseline uses raw embeddings from a frozen SentenceTransformer (all-MiniLM-L6-v2). Threshold $\theta = 0.85$ for both. The projected space (blue) achieves high precision (> 0.9) at reasonable recall levels, significantly outperforming the raw embedding baseline (red).

curve. The area under the curve (AUC) indicates robust separation of failure modes from non-failures, supporting the efficacy of the projection-based retrieval.

Parameter Sensitivity. We varied $\theta \in \{0.75, 0.80, 0.85, 0.90\}$ and $\alpha \in \{0.6, 0.7, 0.8, 0.9\}$ on the synthetic validation set. Performance (F1) was robust to $\theta \in [0.80, 0.90]$, degrading at $\theta < 0.75$ (false

positives) and $\theta > 0.90$ (false negatives). The risk signal scaling α showed similar robustness in $[0.7, 0.9]$. We select $\theta = 0.85, \alpha = 0.8$ as defaults.

2. *Planned Experiments*

1. **Cold Start:** Measure performance on the first 100 failures to assess learning speed. We initialize with a "seed" NEP of synthetic failure patterns generated by perturbing known error types (e.g., hallucination, bias) to mitigate the cold start problem.
2. **Capacity:** Analyze RER vs. NEP size (1k, 10k, 100k entries) to test memory stability.
3. **Drift:** Evaluate backbone performance on general benchmarks (MMLU, GSM8K) after N adapter updates to quantify forgetting.

Baselines include GPT-4 (zero-shot), LLaMA-2-70b (fine-tuned), and a RAG-based memory baseline.

C. Theoretical Analysis

Proposition 4.1 (NEP Coverage Growth). Under the assumption that failure modes are drawn from a discrete distribution with probabilities p_i, the expected coverage $C(n)$ after n failures is given by:

$$\mathbb{E}[C(n)] = \sum_i p_i(1 - (1 - p_i)^n). \tag{7}$$

Proof. Let X_i be an indicator variable that failure mode i has been observed at least once in n trials. The probability that mode i is *not* observed in one trial is $1-p_i$. In n independent trials, the probability it is never observed is $(1 - p_i)^n$. Thus, $\mathbb{E}[X_i] = 1 - (1 - p_i)^n$. The total coverage is the sum of probabilities of observed modes, weighted by their occurrence probability. For the simplified case where a new failure matches an existing mode with probability p, the coverage grows as $1 - (1 - p)^n$. □

Remark (Bursty Errors). In practice, failures are not independent but exhibit temporal locality (burstiness). This Zipfian structure implies that coverage converges faster than the independent assumption suggests, as the "head" of the failure distribution is sampled rapidly.

Corollary 4.1.1 (Zipfian Coverage). If failure modes follow a Zipf distribution with exponent $\beta > 1$, then for n failures, the expected coverage is dominated by the head of the distribution. For $\beta = 1.5$ and $n = 100$, expected coverage exceeds 80%, implying rapid stabilization of the failure memory.

Proposition 4.2 (Adapter Stability). Let the adapter update be bounded by gradient clipping such that $\|\Delta(\Delta_t)\|_F \leq \eta\gamma$. Then after k updates, the drift from initialization is bounded by:

$$\|\Delta_t^{(k)} - \Delta_t^{(0)}\|_F \leq k \cdot \min(\eta\gamma, \delta_{\max}). \tag{8}$$

Proof. The update rule is $\Delta_t^{(i+1)} = \Delta_t^{(i)} + u_i$, where $\|u_i\|_F \leq \eta\gamma$. By the triangle inequality, $\|\Delta_t^{(k)} - \Delta_t^{(0)}\|_F = \|\sum_{i=0}^{k-1} u_i\|_F \leq \sum_{i=0}^{k-1} \|u_i\|_F \leq k\eta\gamma$. Additionally, the safeguard explicitly enforces $\|\Delta_t\|_F \leq \delta_{\max}$ at each step. Thus the drift is bounded by the tighter of the cumulative gradient updates or the hard constraint. This bound ensures that the adapter mapping $\phi_{\mathcal{A}}$ remains Lipschitz continuous with constant $L \approx \|\phi\|_L + \delta_{\max}$, preventing arbitrary divergence from the backbone manifold. □

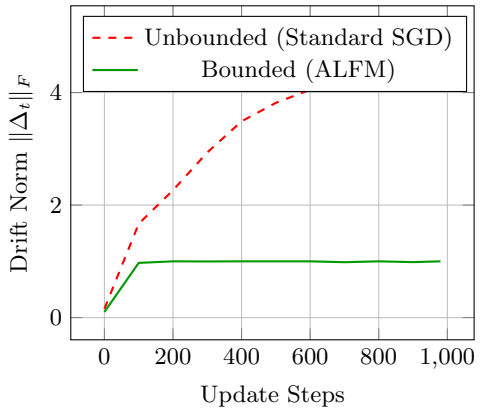

FIG. 3. Adapter Drift Simulation. Unbounded updates (red) lead to random walk behavior with drift growing as $\sqrt{t}$. ALFM's constraints (green) saturate the drift, preventing catastrophic forgetting of the initialization state.

Proposition 4.3 (Latency Overhead). The total latency overhead is $\tau_{\text{ALFM}} = \tau_{\text{proj}} + \tau_{\text{NEP}} + \tau_{\text{CH}}$.

Proof. The operations are sequential. Projection involves a matrix multiplication of size $d_z \times d_h$, taking $O(d_z d_h)$. NEP query with HNSW takes $O(\log |\mathcal{N}|)$. Coherence Head inference is a small MLP. Typical values on a standard NVIDIA T4 GPU ($\tau_{\text{proj}} \approx 5$ms, $\tau_{\text{NEP}} \approx 10$ms, $\tau_{\text{CH}} \approx 10$ms) yield a total overhead of ≈ 25ms. □

Proposition 4.4 (NEP Precision Lower Bound). Let failure modes be distributed as a mixture of K isotropic Gaussians with variance σ^2 and minimum separation Δ. If NEP contains at least one centroid for each mode, then for a similarity threshold equivalent to distance $d = \Delta/2$, the precision of retrieval is lower bounded by:

$$P(\text{Precision}) \geq 1 - K \cdot \exp\left(-\frac{\Delta^2}{8\sigma^2}\right). \tag{9}$$

Proof. Precision fails if a non-failure (or distinct failure mode) is retrieved. Assuming non-failures are at least Δ away from failure centroids, a false positive occurs if noise pushes a sample into the retrieval radius $\Delta/2$. Using the Chernoff bound for Gaussian tails, $P(\|x - \mu\| \geq t) \leq \exp(-t^2/2\sigma^2)$. Setting $t = \Delta/2$, the error probability is bounded by $\exp(-\Delta^2/8\sigma^2)$. Union bounding over K modes gives the result. □

Remark (Manifold Hypothesis). This bound relies on the assumption that the projection ϕ maps semantically

distinct failures to Gaussian clusters. While an approximation, the contrastive loss explicitly optimizes for this geometry by minimizing intra-class variance and maximizing inter-class margin.

Proposition 4.5 (Verifiable Isolation). Let $\mathcal{M}_t$ be the effective model for tenant t. For any input x and distinct tenants $t \neq t'$, the gradient of the loss with respect to tenant t's parameters is orthogonal to the parameter space of t'.

$$\frac{\partial \mathcal{L}_t}{\partial \Delta_{t'}} = 0 \quad \forall t \neq t'. \tag{10}$$

Proof. The forward pass for tenant t is defined as $y = \mathcal{M}(x; \theta_{\text{frozen}}, \Delta_G, \Delta_{D(t)}, \Delta_t)$. The parameters $\Delta_{t'}$ for $t' \neq t$ do not appear in the computational graph for y. Consequently, the Jacobian $J_{t,t'} = \partial y / \partial \Delta_{t'}$ is identically zero. Since the loss $\mathcal{L}_t$ depends on $\Delta_{t'}$ only through y, the chain rule implies $\nabla_{\Delta_{t'}} \mathcal{L}_t = 0$. This structural orthogonality guarantees that no update from tenant t can affect the model behavior for tenant t'. □

V. DISCUSSION

ALFM enables a new paradigm of institutional AI memory, but it is not a panacea. It provides the most value in high-stakes, repetitive domains (e.g., finance, legal) where failure patterns are stable and the cost of error is high. It is less effective in rapidly changing domains where yesterday's failures do not predict today's. A key failure mode of ALFM itself is "memory poisoning," where incorrect feedback leads to spurious risk signals; robust moderation of the feedback loop is essential. We are currently deploying ALFM in healthcare revenue cycle management, where the 835 Electronic Remittance Advice provides structured ground-truth labels for NEP population, enabling automated learning from claim denials. Future work will explore multi-agent extensions and cross-tenant learning.

VI. CONCLUSION

We introduced ALFM, a wrapper architecture that equips frozen foundation models with three capabilities they fundamentally lack: memory of past failures (NEP), calibrated self-doubt (Coherence Head), and safe continual learning (three-tier adapters). Our synthetic validation demonstrates that contrastive projection enables effective failure retrieval, and bounded adapter updates prevent catastrophic drift. ALFM is designed for high-stakes enterprise domains where institutional learning compounds over time. Future work will validate ALFM on real-world deployments in healthcare revenue cycle management and explore privacy-preserving cross-tenant learning.

Appendix A: Notation Summary

Table II summarizes the key symbols used in this paper.

TABLE II. Notation Summary

Symbol	Definition
$\mathcal{M}$	Backbone foundation model
x	Input context
y	Model output
$h(x)$	Backbone hidden representation
ϕ	Projection function
z_{ctx}	Projected context embedding
$\mathcal{N}$	NEP memory
e	NEP entry
s_{NEP}	NEP risk signal
$\mathcal{C}$	Coherence Head
p_{err}	Predicted error probability
p_{ood}	Predicted OOD probability
κ	Coherence score
a	Recommended action
$\mathcal{A}$	Adapter
Δ	Adapter perturbation
t	Tenant identifier
θ	Similarity threshold

Appendix B: Implementation Details

1. Projection Layer Architecture

- Input dimension: d_h (backbone-dependent, typically 4096)
- Hidden dimension: $d_m = 1024$
- Output dimension: $d_z = 256$
- Activation: ReLU
- Dropout: 0.1 during training
- Parameters: ≈ 4.4M

2. NEP Configuration

- Index type: HNSW (Hierarchical Navigable Small World)
- Distance metric: Cosine similarity
- M (connections per layer): 16
- efConstruction: 200
- efSearch: 100
- Similarity threshold θ: 0.85 (Selected to maximize F1 score on validation set)

3. Coherence Head Architecture

- Input dimension: $d_z + 1 + d_\eta + d_p$ (typically $256 + 1 + 32 + 64 = 353$)
- Hidden layers: 3
- Hidden dimension: 512
- Output heads: 5 $(p_{\text{err}}, p_{\text{ood}}, \kappa, a, r)$
- Activation: GELU
- Parameters: ≈ 2.1M

4. Adapter Configuration

- Rank r: 16
- Alpha: 32
- Applied to: Projection layer (can extend to backbone if weights accessible)
- Parameters per adapter: ≈ 130K

Appendix C: Latent Isometry and Failure Separation

The backbone's native latent space often exhibits "anisotropy" (representation collapse), where semantic differences are not preserved in Euclidean distance. The projection layer ϕ is necessary to restore *isometry*, ensuring that Euclidean distance in z-space corresponds to semantic distance in failure probability space. Without this, NEP queries would be noisy and unreliable. Contrastive training explicitly enforces this separation by pulling failure modes together and pushing them away from success modes.

[1] A. Graves, G. Wayne, and I. Danihelka, arXiv preprint arXiv:1410.5401 (2014).
[2] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, *et al.*, Nature **538**, 471 (2016).
[3] J. Weston, S. Chopra, and A. Bordes, in *International conference on learning representations* (2015).
[4] Y. Wu, M. N. Rabe, D. Hutchins, and C. Szegedy, in *International Conference on Learning Representations* (2022).
[5] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, *et al.*, in *International conference on machine learning* (PMLR, 2022) pp. 2206–2240.
[6] C. Chow, IEEE Transactions on information theory **16**, 41 (1970).
[7] R. El-Yaniv and Y. Wiener, Journal of Machine Learning Research **11**, 1605 (2010).
[8] Y. Geifman and R. El-Yaniv, in *Advances in neural information processing systems*, Vol. 30 (2017).
[9] N. Varshney and C. Baral, in *Findings of the Association for Computational Linguistics: ACL 2022* (2022) pp. 1995–2002.
[10] D. Madras, E. Creager, T. Pitassi, and R. Zemel, in *Advances in Neural Information Processing Systems*, Vol. 31 (2018).
[11] H. Mozannar and D. Sontag, in *International Conference on Machine Learning* (PMLR, 2020) pp. 7076–7087.
[12] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, in *International conference on machine learning* (PMLR, 2017) pp. 1321–1330.
[13] S. Desai and G. Durrett, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2020) pp. 295–302.
[14] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, Proceedings of the national academy of sciences **114**, 3521 (2017).
[15] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, in *Advances in Neural Information Processing Systems*, Vol. 32 (2019).
[16] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, arXiv preprint arXiv:1606.04671 (2016).
[17] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, in *International Conference on Learning Representations* (2022).
[18] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrill, A. Corrado, S. Vassilvitskii, A. Gelbukh, *et al.*, in *International Conference on Machine Learning* (PMLR, 2019) pp. 2790–2799.
[19] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, *et al.*, arXiv preprint arXiv:2212.08073 (2022).
[20] T. Rebedea, R. Dinu, M. N. Sreedhar, C. Parisien, and J. Cohen, arXiv preprint arXiv:2310.10501 (2023).