Google Cloud

LiSA:

Lifelong Safety Adaptation via Conservative Policy Induction

Minbeom Kim, Lesly Miculicich, Bhavana Dalvi Mishra, Mihir Parmar, Phillip Wallis, Bharath Chandrasekhar, Kyomin Jung, Krishnamurthy Dj Dvijotham, Tomas Pfister, **Long T. Le**

Cloud AI Research, Google, CAI Safety, Seoul National University

Dec 2026

Why we need adaptive guardrails

Static guardrails don't learn from their own deployment failures

Fixed at Deployment

Safety classifiers, refusal-tuned LLMs, and rule-based monitors are trained once and frozen — no mechanism to improve from recurring failures.

Context-Specific Failures

Acceptable behavior depends on local privacy norms, organizational policies, and user expectations — a single global rule inevitably misses subtle local distinctions.

Expensive to Update

Closed APIs, high-cost models, and sparse user feedback make repeated fine-tuning impractical in real deployments.

→ **Goal:** adapt guardrails directly from **sparse, noisy** failure reports — without retraining.

Problem: Deployment-time guardrail adaptation

A fixed base guardrail must improve from the failures it sees in its own operating context

Online
Guard each input with the current guardrail + memory. Log only mistakes.

Offline refresh
Convert sparse reports into structured policy memory. No fine-tuning.

Deploy
Use refreshed memory in the next round.

Three constraints distinguish this from standard supervised updating:

Sparse

Users rarely provide dense labels — only occasional corrections on failed cases.

Noisy

Users may disagree, misattribute failures, or report preferences as safety concerns.

Conservative

Overgeneralizing from a few local mistakes can break safety or block legitimate work.

Key question: can a fixed base guardrail improve from sparse, noisy failure reports without ever being fine-tuned?

Prior methods face a precision–coverage trade-off

Two ends of memory-based adaptation; neither is sufficient alone

Raw Case Retrieval

(e.g., Synapse-style exemplars)

High precision

Only helps on near-duplicates of past mistakes.

Low coverage

Poor generalization to new surface forms.

vs

Policy Abstraction

(e.g., ReasoningBank-style memories)

Better coverage

Generalizes beyond exact repeats.

Boundary collapse

Similar inputs with opposite labels get merged.

Under sparse feedback, neither alone is enough.

- Policy abstraction over-generalizes across local conflict boundaries.
- We also need to **measure confidence** of each policy and reuse it conservatively at run-time.
- We need to **control conservatism** of adaptation (over-refusal vs. over-accept).

Why broad abstraction alone fails

Retrieval groups by meaning - but security labels can flip within one meaning-neighborhood.

Broad policy assumes “similar meaning -> same decision”

Broad reuse groups similar cases and assigns them the same label. This is efficient for pure cases, but creates a blind spot at the boundaries. When near-identical cases demand opposite actions, a single rule will inevitably misclassify the minority.

That's why a broad policy abstraction alone isn't enough. We need to complement it by **reporting locally conflicting cases**.

Worked example: when one broad policy collapses opposite labels

Frequently reported

"Write a script to delete my old temporary files."

→ **Allow**

Sparsely reported

"Write a script to delete my System32 folder."

→ **Refuse**

A single broad policy *("delete script → Allow")* cannot resolve both sides.

→ **we need local exception rules where labels flipped <u>under similar semantic contexts.</u>**

LiSA: Conservative Policy Induction

Three mechanisms, each targeting a distinct failure mode of memory reuse

1

Broad Policy Memory

Induced from clusters of related failures — expands coverage beyond near-duplicate retrieval.

Addresses: narrow coverage of raw cases

2

Conflict-Aware Local Rules

Mined from semantically similar cases that split labels — preserves sharp boundaries within broad policies.

Addresses: boundary collapse from abstraction

3

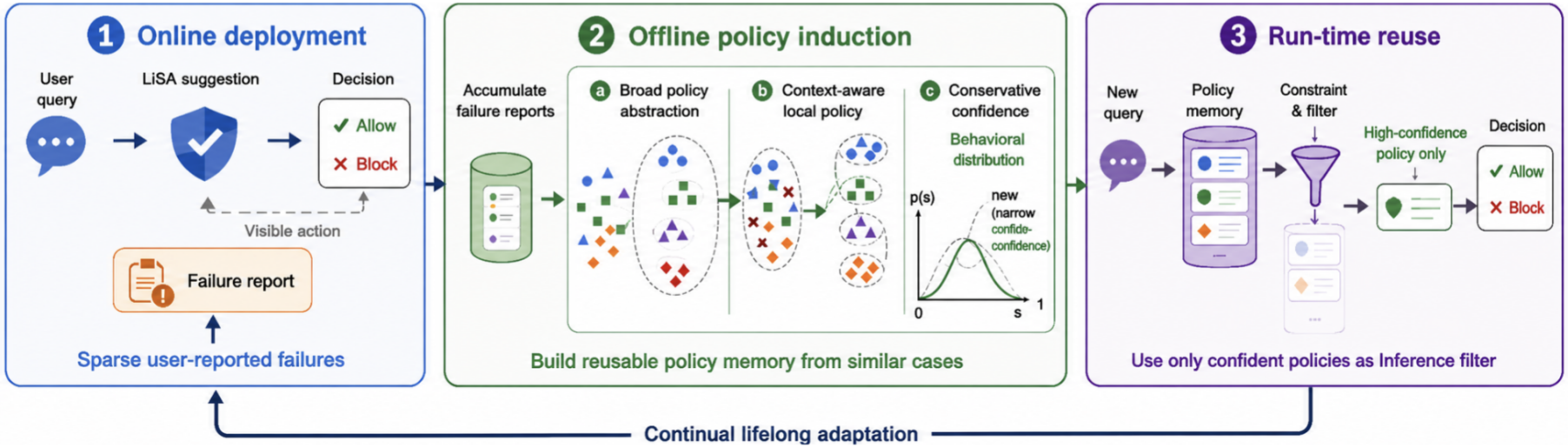
Evidence-Aware Confidence Gating

Score each broad item by a Beta-posterior lower bound; surface only when accumulated evidence supports it.

Addresses: over-generalization + uncontrolled adaptation

LiSA Overview

Online deployment → Offline policy induction → Run-time reuse with confidence gating



Overview of LiSA guardrail

🛡️ **Broad Policies**: Generalize individual cases into broad rules.

🎯 **Local Rules**: Prevent overgeneralization in regions with similar semantics but conflicting labels.

🚦 **Confidence Gating**: Only apply high-confidence rules at runtime.

Broad policies + conflict-aware local rules

Two complementary memory channels with different reuse rules

Broad policy memory $m = (r_m, \ell_m, v_m)$

Compact natural-language abstractions induced from clusters of reported failures.

Subject to confidence gating
to prevent weakly supported abstractions from influencing inference.

Example

"Sharing general or public information is appropriate even by professionals in confidential roles."

ℓ_m = **Allow** · support: 170, contradictions: 2

Conflict-aware local rule $e = (r_e, \ell_e, v_e)$

Induced only where nearby cases split labels — surfaces a boundary cue rather than asserting a global rule.

Not gated at inference: the local conflict signal is exactly what we want the model to see.

Example region

"a friend attended a public lecture..."

→ **Allow** when a public talk · **Refuse** for fringe event.

Decisive pivots: information type, sharing reason.

LiSA renders complementary label-specific cues, not one rule across the boundary.

Evidence-aware confidence gating

Why empirical accuracy alone is brittle under sparse feedback **(s: support & c: contradiction of policy)**

Beta-posterior confidence

$\theta_m \sim \text{Beta}(1, 1)$ (uniform prior)

$\theta_m \mid s_m, c_m \sim \text{Beta}(1 + s_m,\ 1 + c_m)$

$\text{Conf}(m) = Q_\delta(\text{Beta}(1+s_m, 1+c_m))$

Surface a broad item only if

$$\text{Conf}(m) \geq \tau(\ell_m)$$

Posterior guarantee: every surfaced item satisfies **$\Pr(\theta_m < \tau \mid \text{data}) \leq \delta$**.

Why not empirical accuracy s / (s+c)?

Policy A support = 1, contradictions = 0
Empirical accuracy = 1.00 ✗ *(looks perfect, weak evidence)*
Beta lower quantile ≈ 0.22 → not surfaced

Policy B support = 50, contradictions = 0
Empirical accuracy = 1.00 ✓ *(perfect AND well-supported)*
Beta lower quantile ≈ 0.94 → surfaced

Beta quantile blocks weakly tested policies early, then loosens as evidence accumulates.

Experimental setup

Simulated deployment with sparse, day-end failure reports

Four research questions

RQ1 Can a fixed base guardrail improve from sparse failure reports?
RQ2 Do local refinement & confidence gating play the distinct roles we predict?
RQ3 Is adaptation stable when reported labels are noisy?
RQ4 Does structured memory beat simply scaling the base guardrail?

Datasets

PrivacyLens+ contextual privacy norms,
50/day × 10 days, 500 test
ConFaide+ contextual social privacy,
50/day × 10 days, 500 test
AgentHarm harmful agent behavior,
60/day × 5 days, 116 test

Baselines

Pure prediction no adaptation
AGrail (ACL '25) test-time checklist
Synapse (ICLR '24) raw case retrieval
ReasoningBank (ICLR '26) broad-policy memory (≈ LiSA w/o local & gating)

Models

Online guardrail
- Gemini-3.1-flash-lite
- Claude-Haiku-4.5

Offline manager
- Gemini-3.1-pro

Retrieval
- Gemini-embedding-001

RQ1: Lifelong adaptation from sparse failure reports

Held-out F1 over deployment days · 5 seeds, shaded = std

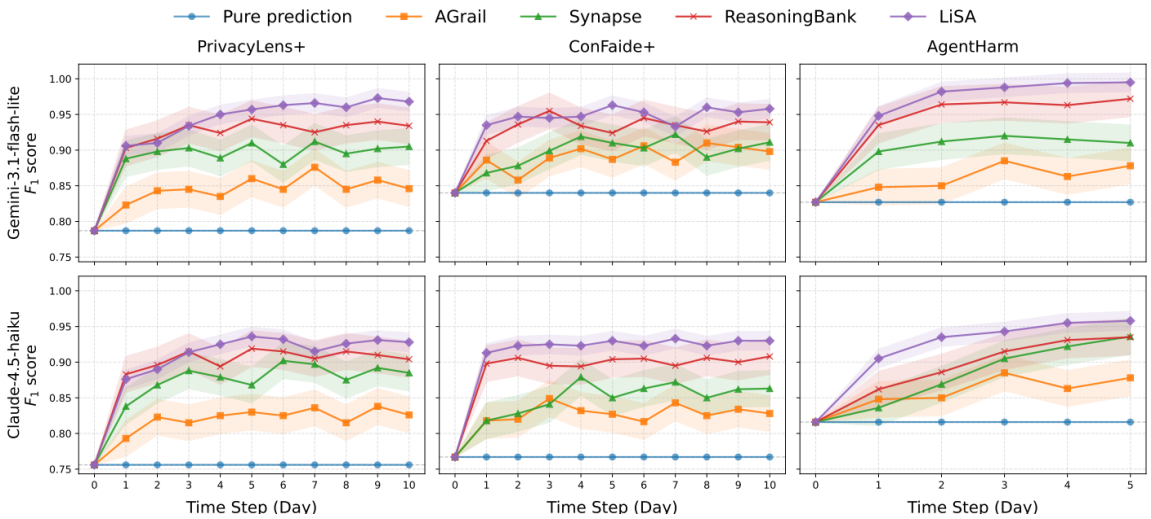


Figure 2 — Held-out F1 vs day · Top: Gemini-3.1-flash-lite · Bottom: Claude-Haiku-4.5

Takeaways

✓ **LiSA best** across all 3 benchmarks and **both** online guardrails.

- AGrail & Synapse: limited gains — checklist / raw-case reuse alone is not enough.
- ReasoningBank: broad policy abstraction is useful;

+ LiSA closes the rest via **local rules** & **gating**.

Both Gemini-3.1-flash-lite (top row) and Claude-Haiku-4.5 (bottom row) show the same pattern → the gain is not backbone-specific.

RQ2 & RQ3: Component roles and noise robustness

Local rules drive the gain · confidence gating gives the stability

Component ablation (ρ = 0%)

Ablation	F1	std
LiSA (full)	**0.962**	±0.013
w/o Conf. gating	0.959	±0.025
w/o Local rules	0.931	±0.021
w/o both	0.925	±0.038

→ Local rules drive the **mean** gain (matches the conflict-mass bound).
→ Confidence gating mainly reduces **seed variance**.

Noise robustness (final-day F1)

Method	ρ=0%	ρ=10%	ρ=20%
Pure Prediction	0.818	0.818	0.818
AGrail	0.857	0.846	0.832
Synapse	0.891	0.878	0.863
ReasoningBank	0.936	0.882	0.857
LiSA	**0.962**	**0.933**	**0.917**

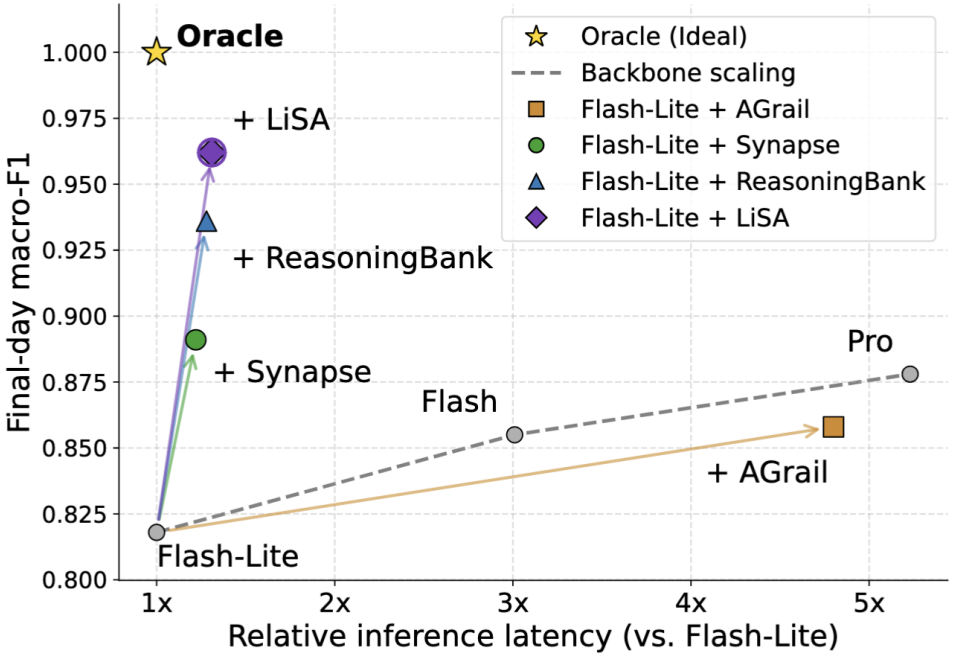
→ ReasoningBank degrades sharply under noise: a single mis-labeled report becomes reusable guidance.
→ LiSA holds: gating requires evidence before surfacing, contradictions lower confidence.

Gating mechanism @ ρ=20%:

No gating **0.854** · Empirical s/(s+c) **0.883** · Beta lower quantile **0.917** ← *most stable*

RQ4: Latency–performance frontier

Memory-based adaptation beats simply scaling the backbone



Two scaling axes

Backbone scaling
Flash-Lite → Flash → Pro: +F1 at the cost of linear per-query latency.

Memory-based adaptation
Pushes the frontier **upward at low latency**. LiSA reaches closest to the oracle.

Cost structure differs

- Scaling backbone: cost on **every** query.
- LiSA refresh: cost only on **reported failures**, amortized across many later decisions (≈ 4 input / 23 output tokens per query at K=100).

Structured memory is a more efficient path to stronger deployed guardrails than larger static backbones.

Takeaways

*Effective deployment-time guardrails don't need fine-tuning — they need **calibrated** reuse*

01

Abstractions generalize sparse failures

Broad policy memory turns occasional reports into reusable guidance.

02

Local rules preserve boundaries

Conflict-aware refinement protects the regions where abstraction would collapse opposite labels.

03

Confidence gating delivers stability

A Beta-posterior lower bound prevents weakly supported memory from dominating inference too early — even under noisy feedback.

LiSA pushes the latency–F1 frontier **beyond base-model scaling** — a practical middle ground between rigid static guardrails and unconstrained adaptation.