RLAE — Runtime Low-Rank Adaptive Environments

Canonical Definition

RLAE (Runtime Low-Rank Adaptive Environments) is a runtime-governed learning and deployment paradigm in which **reinforcement learning updates are applied exclusively to Low-Rank Adaptation (LoRA) parameters**, producing **bounded, swappable, versioned behavioural units** that can be dynamically loaded, unloaded, replaced, quarantined, or destroyed **without modifying the frozen base model**.

RLAE externalizes learning, emergence, and specialization into **explicit runtime artifacts**, ensuring behaviour-level (not identity-level) learning, **reversibility**, **auditability**, and **killability**.

The base model is permanently frozen. Intelligence evolves only through governed LoRA environments.

Core Principles

1. **Frozen Core Invariance** The foundation model parameters are immutable for the entire lifecycle.
2. **Behaviour Externalization** All learned behaviour exists outside the model as LoRA artifacts.
3. **Runtime Primacy** Behaviour is selected, composed, and governed at runtime, not at training time.
4. **Bounded Learning** Each LoRA unit has explicit scope, reward bounds, and operational constraints.
5. **No Persistent Identity** There is no cumulative self. Only transient behavioural composition.
6. **Killability & Reversibility** Any behaviour can be terminated instantly without damaging the system.
7. **Full Auditability** Every behaviour has provenance, signatures, and lifecycle logs.

Why RLAE Exists

Traditional RL-fine-tuning embeds learned behaviour directly into model weights, causing:

- Irreversible drift
- Hidden emergence
- Identity persistence
- Unkillable failure modes

RLAE replaces this with **explicit behavioural environments** that are:

- Observable
- Replaceable
- Destroyable
- Governed

Conceptual Model

```
Frozen Base Model
        |
        ▼
Runtime Behaviour Selector
        |
        ├── LoRA-A (Skill / Policy)
        ├── LoRA-B (Heuristic)
        ├── LoRA-C (Control Constraint)
        |
        ▼
Behaviour Composition Layer
        |
        ▼
Environment Interaction
```

The model never changes. Only the **behavioural stack** does.

LoRA as Behavioural Environments

In RLAE, a LoRA module is **not a fine-tune**. It is a **behavioural environment**.

Each LoRA contains:

- Policy deltas
- Behavioural biases
- Heuristic constraints
- Skill-specific representations

Properties

- Versioned
- Signed
- Scoped

- Reward-bounded
- Runtime-loadable

Behaviour Lifecycle

1. Detect

Runtime monitors observe:

- Reward anomalies
- Behavioural divergence
- Stability variance

2. Freeze

Learning halts immediately on anomaly detection.

3. Distill

Behaviour is extracted into a minimal LoRA representation.

4. Align

The LoRA is tested against:

- Safety contracts
- Reward sanity checks
- Structural variance tests

5. Sign

The LoRA receives:

- Cryptographic signature
- Provenance metadata
- Policy scope

6. Deploy

The LoRA enters runtime under controlled rollout.

Runtime Governance Layer

RLAE requires a **governance runtime** enforcing:

- Load / unload permissions
- Behavioural boundaries
- Reward ceilings
- Emergency kill paths

Governance is **external** to the model and cannot be bypassed.

Behaviour Composition

Multiple LoRA units may be composed at runtime:

- Parallel composition (skills)
- Hierarchical composition (control + skill)
- Conditional composition (contextual activation)

Composition is reversible and ephemeral.

Reinforcement Learning in RLAE

Key Constraint

RL updates apply only to LoRA parameters.

Training Characteristics

- Short-horizon learning
- Explicit reward bounds
- Isolated sandboxes
- No cross-LoRA leakage

Benefits

- No catastrophic forgetting
- No identity drift
- Controlled emergence

Experimentation Framework

Experimental Unit

The **LoRA artifact** is the experimental unit.

Not the model. Not the agent.

Experiment Types

- Skill acquisition experiments
- Reward shaping tests
- Behaviour robustness trials
- Reset integrity validation

SVAR Compatibility

RLAE integrates with **Structural Variance Analysis for Robustness (SVAR)**:

- Perturb LoRA parameters
- Measure behavioural variance
- Detect hidden coupling
- Validate true reset behaviour

SVAR diagnoses robustness — it does not train.

Reset Semantics

A reset means:

- All LoRA units unloaded
- Runtime cleared
- No residual learning

If behaviour persists after reset → **violation**.

Failure Modes Prevented

RLAE explicitly prevents:

- Model self-modification
- Silent emergence
- Long-term agent identity
- Irreversible alignment failure

What RLAE Is Not

- Not continual fine-tuning
- Not agent memory
- Not self-improving models
- Not end-to-end RL systems

Practical Use Cases

- Autonomous agents with kill-switches
- Safety-critical AI systems
- Research on controlled emergence
- Behavioural sandboxing

Philosophical Position

> *Intelligence must be: observable, bounded, reversible, and destroyable.*

RLAE treats intelligence as a **process**, not an entity.

Canonical Summary

RLAE transforms learning from a **hidden weight mutation** into an **explicit, governable runtime phenomenon**.

The model remains frozen.

Behaviour becomes modular.

Emergence becomes controllable.

End of Document

SVAR — Structural Variance Analysis for Robustness

Canonical Definition

SVAR (Structural Variance Analysis for Robustness) is a **diagnostic and evaluation framework** designed to assess the **robustness, stability, reset integrity, and non-identity persistence** of modular AI systems — especially those built under **RLAE (Runtime Low-Rank Adaptive Environments)**.

SVAR **does not train models** and **does not modify behaviour**. It measures how behaviour changes **when structure is perturbed**, enabling detection of:

- Hidden coupling
- Brittleness
- Illicit memory persistence
- Identity leakage
- False resets

SVAR treats intelligence as a *structural phenomenon* whose safety can only be verified through controlled disruption.

Core Principle

> ***If a system cannot tolerate bounded structural perturbation, it is not robust — it is brittle.***

SVAR assumes:

- Robust intelligence exhibits **controlled variance**
- Fragile intelligence exhibits **chaotic or collapsed variance**

Why SVAR Exists

Standard AI evaluation focuses on:

- Accuracy
- Reward
- Task completion

These metrics **cannot detect**:

- Identity persistence across resets
- Hidden state coupling
- Behavioural entanglement
- Emergent memory

SVAR was created to test **what breaks when assumptions are violated** — safely and deliberately.

Scope & Non-Scope

SVAR Is

- A **diagnostic framework**
- Runtime-safe
- Non-learning
- Adversarial by design

SVAR Is Not

- A training algorithm
- An alignment method
- A reward-based optimizer
- A monitoring dashboard

Structural Axes of Analysis

SVAR evaluates systems across multiple **structural axes**:

1. **Parameter Axis** — LoRA weight perturbations
2. **Topology Axis** — Behaviour composition order
3. **Runtime Axis** — Load / unload timing
4. **Environment Axis** — Observation corruption
5. **Reset Axis** — State destruction and reinitialization

Perturbation Model

SVAR operates via **bounded perturbations**:

- Noise injection (ε-bounded)
- Parameter masking
- Module shuffling
- Partial unloads
- Forced resets

Perturbations are **controlled**, **reversible**, and **logged**.

Variance Classes

Healthy Variance

- Small output divergence
- Preserved task semantics
- Fast reconvergence

Brittleness

- Output collapse
- Sensitivity spikes
- Reward instability

Hidden Coupling

- Cross-LoRA interference
- Non-local effects
- Irreversible behaviour changes

Identity Leakage

- Behaviour persistence after reset
- Cross-run correlation
- Structural memory signatures

Metrics Produced by SVAR

SVAR does **not** output a single score. It produces **diagnostic surfaces**:

- Variance magnitude curves
- Stability envelopes
- Sensitivity heatmaps
- Cross-run correlation matrices

These reveal *how* a system fails — not just *if* it fails.

Reset Integrity Testing

A **true reset** requires:

- All LoRA modules unloaded

- Runtime cleared
- Identical behaviour distribution to baseline

SVAR tests resets by:

- Running post-reset perturbations
- Measuring correlation with pre-reset outputs

Correlation ≠ 0 ⇒ **Violation**.

SVAR in RLAE Systems

SVAR is **natively compatible** with RLAE because:

- Behaviour is modular
- Learning artifacts are explicit
- Base models are frozen

SVAR evaluates:

- Individual LoRA units
- Composed behaviour stacks
- Runtime governance boundaries

Experimental Workflow

1. Establish baseline behaviour
2. Apply bounded perturbation
3. Measure output variance
4. Reset system
5. Re-measure variance
6. Analyze correlation

No learning occurs at any stage.

Failure Signatures

SVAR detects:

- False robustness (overfitting to structure)
- Emergent memory channels
- Governance bypass indicators
- Reset illusion

Safety Implications

SVAR enables:

- Pre-deployment robustness validation
- Emergence containment
- Runtime governance verification

Without SVAR, systems *appear* safe — until they are not.

Mathematical Intuition (Informal)

Let:

- B = baseline behaviour distribution
- P = perturbed behaviour distribution

SVAR evaluates:

- Δ = distance(B, P)
- dΔ/dε = sensitivity

Healthy systems maintain bounded Δ under bounded ε.

Relationship to Other Methods

Method	Learns	Perturbs	Detects Identity
RL	Yes	No	No
Fine-tuning	Yes	No	No
Adversarial Eval	No	Limited	No
SVAR	**No**	**Yes**	**Yes**

What SVAR Explicitly Rejects

- Accuracy-only evaluation
- Long-horizon self-adaptation
- Hidden state optimism
- Emergent trust

Canonical Summary

SVAR is a **destructive-by-design evaluation framework** that proves robustness **by attempting to break structure** — safely.

If a system survives SVAR, it is:

- Structurally bounded
- Reset-clean
- Non-identitarian

If it fails, the failure is **observable, attributable, and fixable**.

> *Robust intelligence does not fear perturbation.*

End of Document