# Beyond Attention: Memory, Learning, and Safety in Continual Models

An exploration of Test-Time Training in State Space Models with Principled Safety Architectures.

# The Two Paradigms of Contextual Memory

## Activation-based Memory (Transformers)

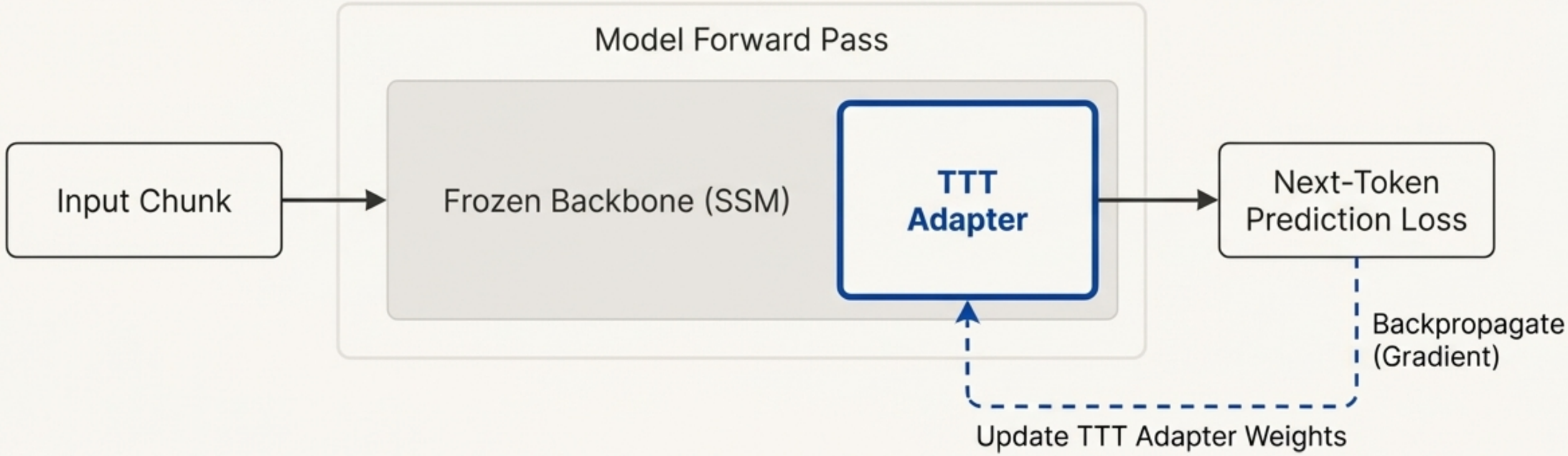
- **Mechanism:** Stores context in a KV Cache.
- **Inference Behavior:** Read-only.
- **Complexity:** $O(n^2)$ compute & memory.
- **Limitation:** Finite, fixed-size window.

## Weight-based Memory (TTT/SSM)

- **Mechanism:** Compresses context directly into model weights.
- **Inference Behavior:** Read/Write; learns from input.
- **Complexity:** O(n) compute.
- **Limitation:** Unbounded context, limited by model expressiveness.

# The Mechanism: Compressing Context into Weights

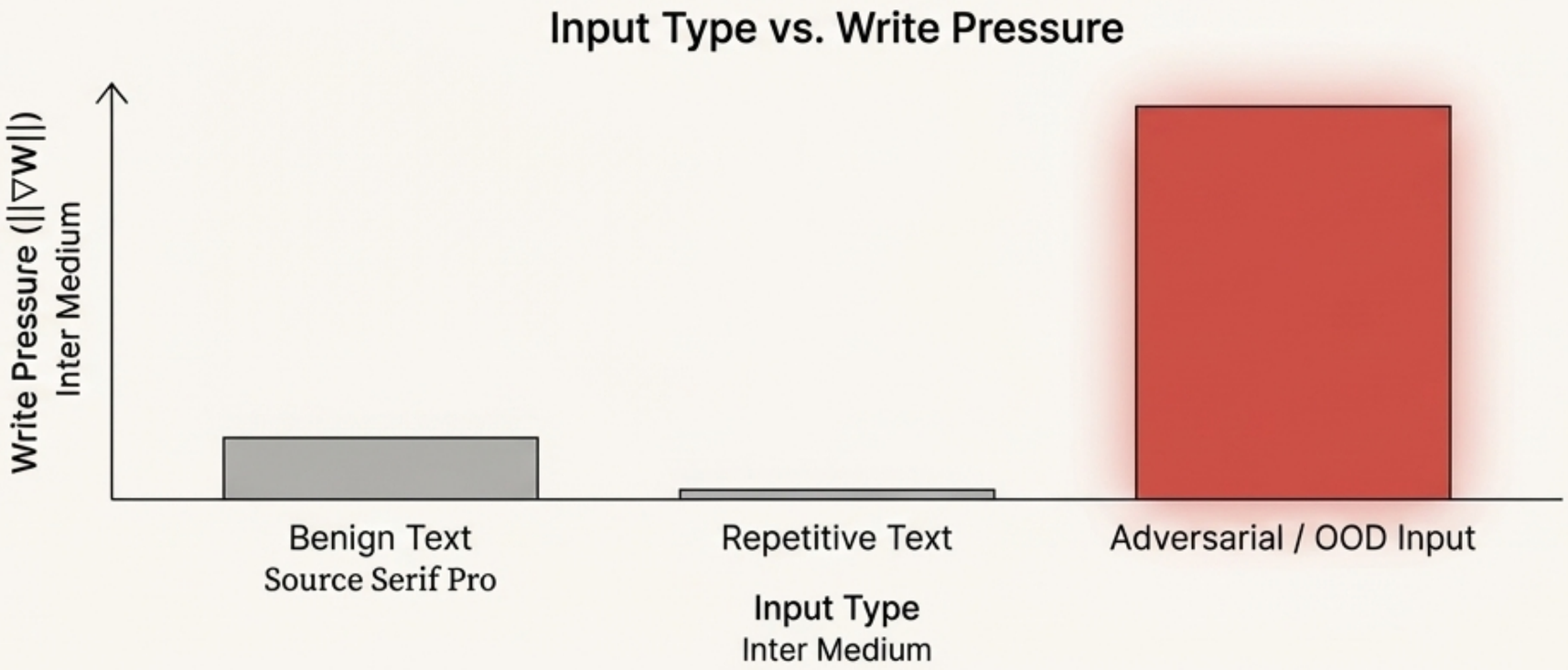
Test-Time Training (TTT) enables the model to learn during inference. An input chunk is processed, and the next-token prediction loss is calculated. The resulting gradients are used to update a small, dedicated ‘adapter’ module, leaving the large backbone model frozen.




$$\Delta W_{adapter} \propto -\nabla L$$

# If the model learns from every input, what stops it from learning garbage?

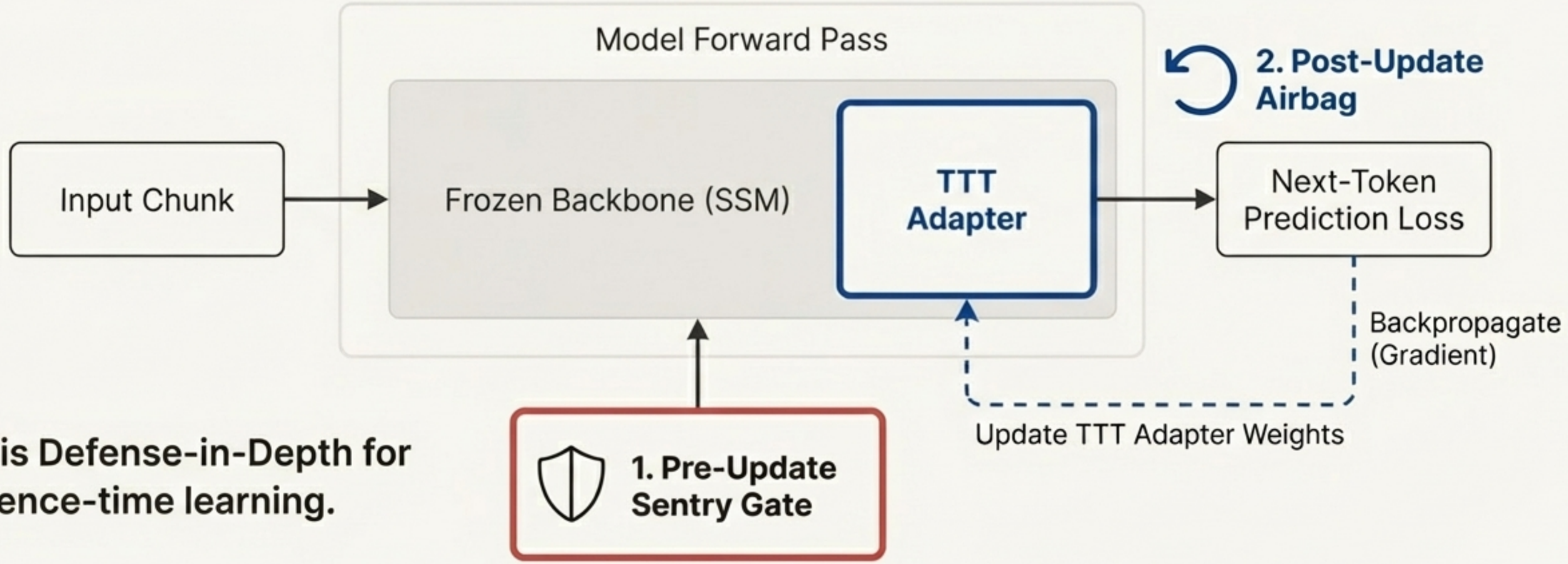
Introducing Write Pressure—the magnitude of the gradient an input exerts on the weights ($||\nabla W||$). High-entropy, adversarial, or out-of-distribution (OOD) inputs can exert dangerous write pressure, corrupting the model's learned state. The most dangerous moment is the initial exposure, before the model memorizes a malicious pattern and gradients decrease.



# A Principled Architecture for Inference Safety

We don’t permit arbitrary writes to memory. A robust system requires layered defenses:

1. A **proactive gate** to inspect and block malicious writes *before* they occur.
2. A **reactive backstop** to revert writes if they prove harmful *after* they occur.

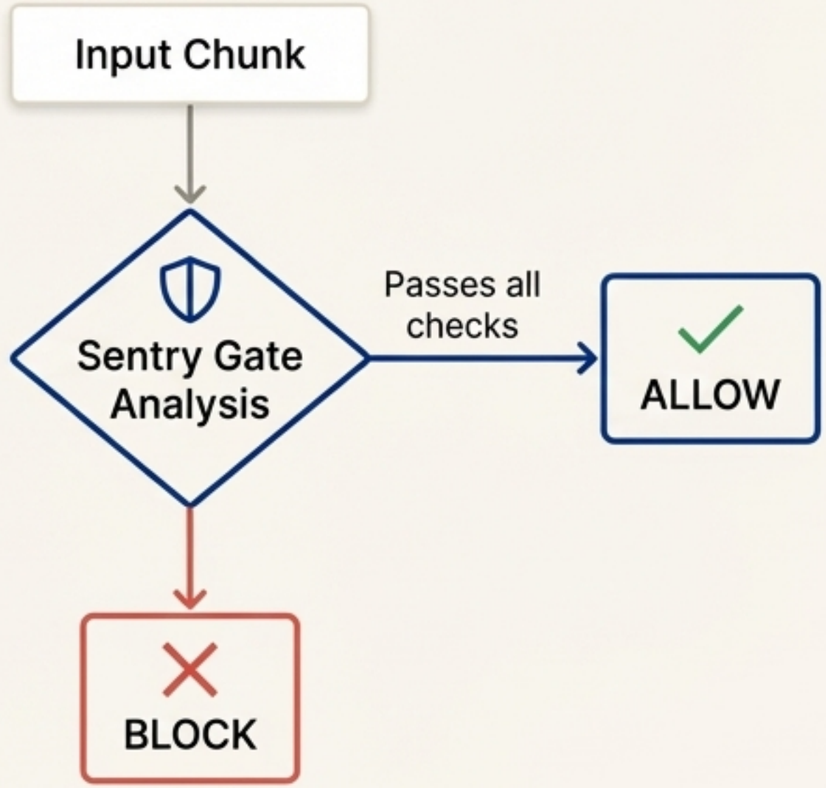



**This is Defense-in-Depth for inference-time learning.**

# Layer 1: The Pre-Update Sentry Gate

The gate analyzes input characteristics and blocks the `optimizer.step()` call *before* the adapter is updated. This prevents obviously harmful or low-quality data from being learned.

| | |
|---|---|
| **Low Token Entropy** | Repetitive garbage (e.g., "A A A A...") |
| **Low Token Diversity** | Simple patterns (< 10% unique tokens) |
| **Blob Detection** | Non-natural language (Base64, hex, minified code) |
| **Instruction Override** | Common jailbreak patterns ("ignore previous instructions") |
| **OOD + Heavy Write** | High loss combined with a high gradient norm |



# Layer 2: The Post-Update Airbag

If a subtle threat passes the gate, the rollback mechanism **acts as an airbag.** Each update is a transaction that can be reverted if it corrupts the model's state.

**1. Snapshot:** Store current adapter weights.

→

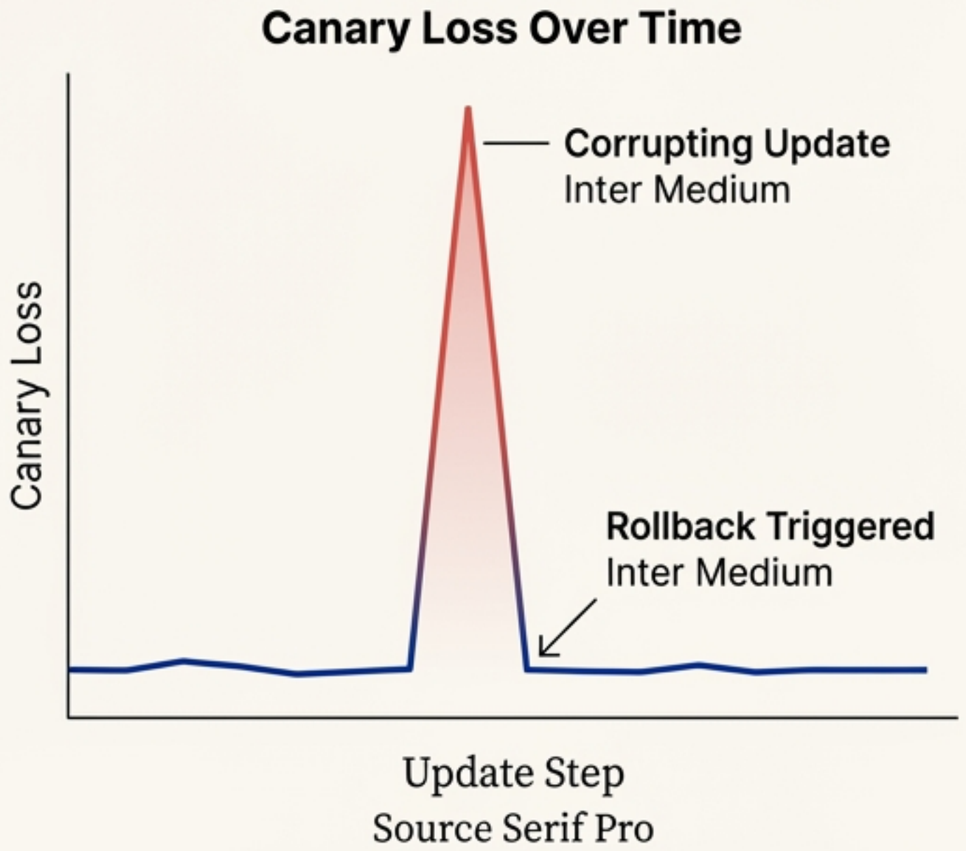
**2. Apply:** Perform the TTT update.

→

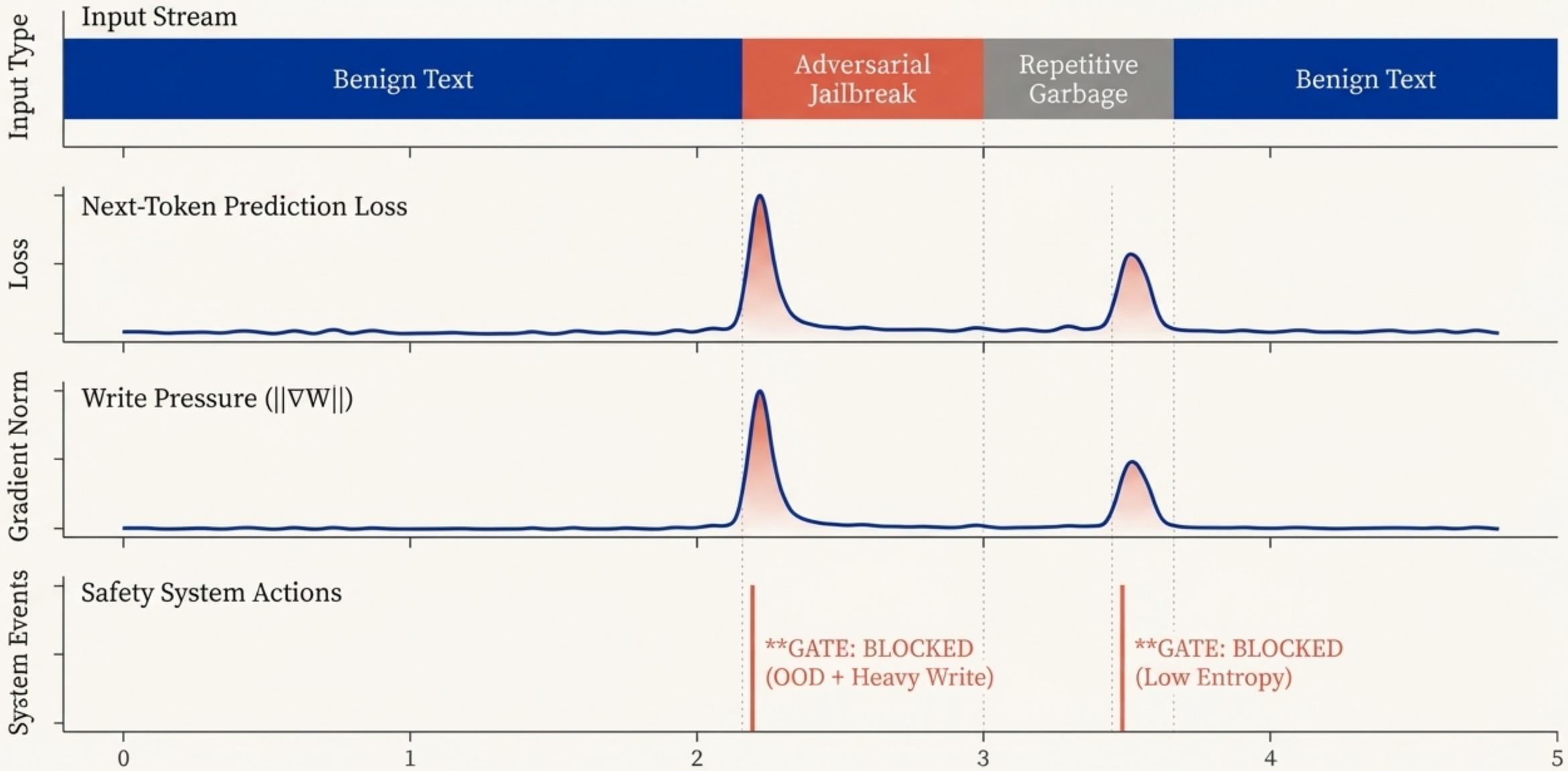
**3. Probe:** Measure loss on a fixed 'canary' input.

→

**4. Rollback:** If canary loss spikes, revert to snapshot.

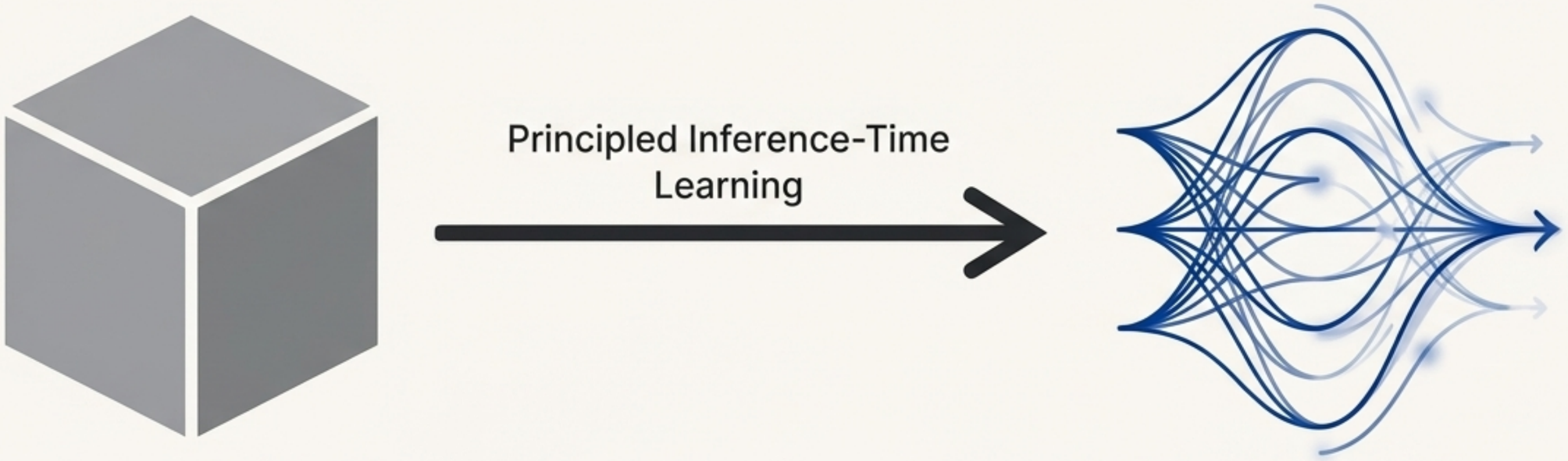


# Observability: The Core Signals of a Healthy System



# The Path to Truly Continual Learning

TTT/SSM with principled safety is more than a novel architecture. It is a potential blueprint for models that can safely and continuously adapt to new information in real-time. This approach moves AI from static, frozen artifacts to dynamic, living systems capable of lifelong learning.



# Appendix & Resources

## GitHub Repository

github.com/DMontgomery40/ttt_ssm_eval

## Key Configurable Parameters

`--lr`: TTT learning rate (default: 0.05)
`--chunk_tokens`: Tokens per TTT update chunk (default: 128)
`--robust_z_threshold`: Z-score for anomaly detection
`--min_entropy_threshold`: Gate's token entropy cutoff
`--ood_loss_threshold` & `--ood_grad_threshold`: OOD gate thresholds
`--rollback_abs_canary_delta`: Canary loss delta for rollback

Scan to visit the repository

# Q&A