Understanding: an experiment-LLM-memory experiment

Cédric Allier and Stephan Saalfeld ✉

Janelia Research Campus, Howard Hughes Medical Institute, Ashburn, VA 20147, USA.

✉ saalfelds@janelia.hhmi.org

January 5, 2026

Abstract

We previously showed that graph neural networks (GNNs) can recover circuit structure and signaling functions of neural assemblies from activity data alone. However, this inverse problem is known to be ill-posed under certain conditions. For instance, when neural activity is low-rank, many different circuits can generate the same neuron traces. It remains an open question whether graph neural networks can recover connectivity in general, or only for specific classes of neural assemblies. To address this question, we extend our graph neural network framework with a large language model. In this new setup, the role of the graph neural network framework is to perform experiments and deliver quantified results. The role of the large language model is to explain, interpret and finally compress the experiment results into an explicit memory. On the basis thereof, the large language model is next prompted to mutate selectively the neural dynamics configuration and/or the graph neural network training scheme. We show that these sequential interactions between experimentation, large language model and long-term memory, leads progressively to a scientific tool. Testable hypotheses are drawn, specific experiments are conducted in untested regions and ultimately causal understanding emerges. Importantly, drawing a coherent picture of the exploration requires systematic reasoning at the level of the coupled system. In our framework, induction, abduction, deduction and falsification are distributed across interacting components. The large language model generates hypotheses and parameter mutations, while empirical evaluation through external experiments, together with structured episodic and long-term memory, enables cumulative hypothesis refinement. Together, these elements implement a closed-loop scientific reasoning process that is not achievable by any single component in isolation.

1 Introduction

Understanding when graph neural networks (GNNs) can recover synaptic connectivity from neural activity, and when they fundamentally fail, requires exploring a high-dimensional landscape of neural activity regimes. A classical approach would rely on nested grid searches: first over classes of neural assemblies to generate activity, and then over GNN training hyperparameters for each regime. Such an approach rapidly becomes intractable, prone to gaps, and poorly suited to uncovering general principles across regimes.

LLM-guided discovery. Recent methods coupling large language models (LLMs) with external validation, including FunSearch (Romera-Paredes et al., 2024), AlphaEvolve (Novikov et al., 2025), OPRO (Yang et al., 2024), and Google Tree Search (Aygün et al., 2025), have demonstrated the ability to explore problem–solution landscapes in mathematics and computer science. These systems iteratively select candidate solutions based on solution–score tuples and use LLMs to propose refinements or fill gaps in the search space. While highly effective for optimization, these approaches are limited by the finite context window of LLMs: without explicit long-term memory, hypotheses and partial insights may be discarded between iterations, preventing cumulative understanding.

Several recent systems combine LLMs with external experimentation or validation. The AI Scientist (Lu et al., 2024) and Virtual Lab (Swanson et al., 2024) integrate neural network training and wet-lab validation, respectively. However, these approaches treat experiments as largely independent, lacking mechanisms to accumulate mechanistic understanding across runs. As a result, knowledge does not persist or transfer be-

yond local optimization.

Conversely, approaches such as Titans (Behrouz, Zhong, and Mirrokni, 2024) and Anthropic's engineering guidelines (Anthropic, 2025) introduce memory to overcome the limited context window of LLMs. Titans implements memory within neural networks trained by gradient descent, while Anthropic advocates text-based progress files to ensure consistency across LLM outputs. In both cases, memory supports coherence, but without coupling to external experiments, generated insights cannot be empirically tested or refuted, leaving their validity uncertain.

Alternative models such as OpenAI's o1/o3 (OpenAI, 2024; OpenAI, 2025) and DeepSeek's R1 (DeepSeek-AI et al., 2025) extend LLM reasoning by scaling test-time computation and structuring deliberation into long chains of thought, often coupled to code execution or proof verification. These models achieve state-of-the-art performance on many benchmarks (Li et al., 2025), but they discard structured long-term memory. As a consequence, knowledge accumulation across problem-solving episodes requires an external observer rather than being internalized by the system itself.

Finally, systems such as Robin (Ghareeb et al., 2025), AI co-scientist (Gottweis et al., 2025), and Kosmos (Mitchener et al., 2025) couple LLMs to large-scale literature analysis and world models to generate hypotheses. Kosmos, for example, performs extensive synthesis across existing datasets and publications, enabling deep inductive insights. However, when hypotheses are derived from literature alone, deduction and falsification remain external to the system, as experimental validation requires additional studies. While these approaches represent a conceptual leap in AI-assisted science, they do not yet implement the full scientific methodology as a closed loop.

The gap. Accumulating scientific understanding requires the tight coupling of three components: external experiments, LLM-based hypothesis generation, and explicit long-term memory. We present a system that closes this loop. A graph neural network framework performs controlled experiments on simulated neural dynamics and produces quantitative evaluations. A large language model interprets these results, compresses them into structured long-term memory, and intervenes by mutating either the neural dynamics configuration or the GNN training scheme. These interventions generate predictions, which are tested by subsequent experiments. Memory is updated accordingly: reinforced when predictions are confirmed and revised when falsified.

We demonstrate this framework on the problem of neural connectivity inference: determining when GNNs can recover the synaptic weight matrix W_{ij} from neural activity $\{x_i(t)\}$. Across 2048 GNN optimizations spanning 128 neural activity regimes, the system accumulates explicit transferable understanding. Induction identifies regularities across regimes, such as scaling relationships between learning rate ratios and constraint complexity. Abduction proposes explanatory mechanisms, for example that data effective rank determines the attainable connectivity R^2. Deduction generates testable predictions, such as increasing n_{frames} to raise effective rank and enable recovery. Falsification rejects or refines these predictions when outcomes contradict expectations. Through these processes, experimental results are consolidated into explicit memory as hypotheses and regime-level knowledge that persist across blocks and systematically constrain, inform, and accelerate future exploration.

2 Methods

Experimental setup. We employ the GNN framework introduced in [...] to model neural dynamics with graph-computation (appendix A): neuron-neuron interactions are computed first, then aggregated to update neuron states. The computation implemented with the PyTorch Geometric library (Fey and Lenssen, 2019) is differentiable, so interaction functions, neuron properties, and update rules can be learned from data. External stimuli can also be learned via neural network representations. This poses an inverse problem over the space $\mathcal{G}$ of graph-computations modeling neural dynamics: given activity traces $\{x_i(t)\}$ produced by some $g \in \mathcal{G}$, recover g. In this experimental setup we address the existence and uniqueness of solutions obtained by means of GNN optimization.

Triad experiments - LLM - memory The system couples the three components, i.e. external experiments, LLM reasoning, and long-term memory, through simple text files exchange. A YAML configuration specifies simulation parameters (connectivity type, noise, n_{neurons}) and GNN training (learning rates, architecture, regularization). The experiment executes and writes the GNN evaluations results to `analysis.log` which, serves as an episodic memory for the LLM.

Two text files structure the LLM's scientific process. The *instruction file* (appendix B) defines the experimental protocol: the inverse problem to solve, parameter ranges to explore, convergence criteria, and the logging format. It also contains a brief domain knowledge. The *memory file* (appendix C)maintains the LLM's accumulated understanding in two tiers:

a *knowledge base* of established principles and regime comparisons that grows across experimentation, and a *working memory* of current hypotheses and recent iterations. This two-tier structure enables both long-term knowledge accumulation and short-term memory.
Experiments are organized into *blocks*. Each block fixes a neural assembly $g \in \mathcal{G}$ and its activity traces $\{x_i(t)\}$. Within a block, the LLM explores GNN training parameters (learning rates, regularization) over $\sim$16 optimization runs. Parameter exploration follows Upper Confidence Bound (UCB) tree search, adapted from Romera-Paredes et al. (2024). Each configuration is a node; the LLM selects the next parent by balancing exploitation (high-performing) with exploration (under-visited branches):

$$\mathrm{UCB}(k) = R_k^2 + c\sqrt{\frac{\ln N}{n_k}}$$

where R_k^2 is the mean coefficient of determination between true and learned connectivity weights W_{ij}. This metric was chosen to scores model recovery at node k of search tree, n_k its visit count, and N the total iterations. The LLM mutates GNN training parameters from the selected parent and logs its reasoning.

At block boundaries, the LLM performs three operations. First, to overcome rigid rule-based exploration, it evaluates its tree search strategy (branching rate, improvement rate, stuck detection) and edits the *instruction* file itself to add or modify rules. Second, it selects the next simulation regime to address gaps in the search space or refine optimization of a previous block. Third, it compresses the block into the knowledge base: a neural activity regime comparison table (best R^2, optimal parameters, key findings), established principles that transfer across neural activity regimes, and open questions for future blocks.

Epistemic metrics and ablation studies. We introduce *epistemic metrics* to quantify how the system acquires, tests, revises, and transfers knowledge across experimental blocks. These metrics characterize system-level behaviors such as hypothesis refinement, diagnostic experimentation, regime differentiation, and the efficient reuse of previously established constraints when entering new regions of the search space. Importantly, these metrics do not measure task performance alone, but the structure and dynamics of the exploration process itself.

To isolate the contribution of each component to these epistemic behaviors, we perform controlled ablation studies in which the external experimental engine and evaluation pipeline are held fixed, while selectively modifying the decision module. Specifically, we consider: (i) a *no-memory* ablation, in which the large language model receives only the last K iterations and has no access to accumulated long-term knowledge; (ii) a *no-LLM* ablation, in which parameter mutations are sampled from matched random or heuristic proposal distributions; and (iii) a *frozen-protocol* ablation, in which the large language model may propose parameter updates but is prohibited from editing the exploration rules. These ablations allow us to attribute observed epistemic behaviors to the interaction between hypothesis generation, persistent memory, and adaptive exploration control.

3 Results

4 Discussion

(Collins et al., 2024) critique pure LLM scaling and advocate for structured world models. They focus on human-LLM collaboration through mutual inference, but the LLM assists while the human drives the scientific process. We show it is possible to elevate the LLM to equal partnership, where both human and LLM hypothesize, intervene, and falsify.

Symbolic logic seems to be structurally encoded in LLM. It appears that the LLM coupled to experimentation and long-term memory exhibits rapidly a tropism for logical reasoning.

We summarize the distinction between LLM-guided approaches upon a new perspective which could serve as guidelines for AI-scientific tool. In assessing LLM-guided scientific systems, we consider the availability of three components (external experiments, LLM, explicit long-term memory) and the capacity for four types of reasoning (induction, abduction, deduction, falsification).

The current system demonstrates the methodology on a single domain (neural dynamics). Yet the approach can be extended beyond this field. Any domain expressible as local interactions aggregated over a graph, e.g. molecular systems, traffic networks, social dynamics, gene regulatory circuits, fluid discretizations, falls within $\mathcal{G}$ the proposed framework. Scaling from many regimes to many domains is therefore a natural extension. Work on simulation is a necessary prerequisite to real data. For each simulation domain, establishing learnability and intelligibility boundaries is a generic problem. Under what conditions can parameters be recovered (learnability), and under what conditions can the recovery be attributed to mechanistic understanding rather than statistical fitting (intelligibility) ?

5 Conclusion

Acknowledgements

References

Anthropic (2025). *Effective Harnesses for Long-Running Agents*. https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents.

Aygün, E. et al. (2025). *An AI system to help scientists write expert-level empirical software*. DOI: 10.48550/arXiv.2509.06503.

Behrouz, A., P. Zhong, and V. Mirrokni (2024). *Titans: Learning to Memorize at Test Time*. DOI: 10.48550/arXiv.2501.00663.

Collins, K. M. et al. (2024). "Building machines that learn and think with people". In: *Nature Human Behaviour* 8.10, pp. 1851–1863. DOI: 10.1038/s41562-024-01991-9.

DeepSeek-AI et al. (2025). *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. DOI: 10.48550/arXiv.2501.12948.

Fey, M. and J. E. Lenssen (2019). *Fast Graph Representation Learning with PyTorch Geometric*. DOI: 10.48550/arXiv.1903.02428.

Ghareeb, A. E. et al. (2025). *Robin: A multi-agent system for automating scientific discovery*. DOI: 10.48550/arXiv.2505.13400.

Gottweis, J. et al. (2025). *Towards an AI co-scientist*. DOI: 10.48550/arXiv.2502.18864.

Li, Z.-Z. et al. (2025). *From System 1 to System 2: A Survey of Reasoning Large Language Models*. DOI: 10.48550/arXiv.2502.17419.

Lu, C. et al. (2024). *The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery*. DOI: 10.48550/arXiv.2408.06292.

Mitchener, L. et al. (2025). *Kosmos: An AI Scientist for Autonomous Discovery*. DOI: 10.48550/arXiv.2511.02824.

Novikov, A. et al. (2025). *AlphaEvolve: A coding agent for scientific and algorithmic discovery*. DOI: 10.48550/arXiv.2506.13131.

OpenAI (2024). *Learning to Reason with LLMs*. https://openai.com/index/learning-to-reason-with-llms/.

— (2025). *OpenAI o3-mini*. https://openai.com/index/openai-o3-mini/.

Romera-Paredes, B. et al. (2024). "Mathematical discoveries from program search with large language models". In: *Nature* 625.7995, pp. 468–475. DOI: 10.1038/s41586-023-06924-6.

Swanson, K. et al. (2024). *The Virtual Lab: AI Agents Design New SARS-CoV-2 Nanobodies with Experimental Validation*. DOI: 10.1101/2024.11.11.623004.

Yang, C. et al. (2024). *Large Language Models as Optimizers*. DOI: 10.48550/arXiv.2309.03409.

A Neural dynamics and GNN architecture

Neural dynamics

Neural assemblies are simulated as graphs of N neurons with weighted synaptic connections W_{ij}. Activity x_i evolves according to:

$$\dot{x}_i = -\frac{x_i}{\tau_i} + s_i \phi(x_i) + g_i \sum_j W_{ij} \psi(x_j) + \eta_i(t) \tag{1}$$

where τ_i controls decay, s_i self-coupling, g_i scales aggregated inputs, and $\eta_i(t)$ is Gaussian noise. Weights are drawn from .... Transfer functions ϕ, ψ are tanh variants; neuron types are parameterized by ... .

GNN Architecture

The GNN learns:

$$\widehat{\dot{x}}_i = \phi^*(\mathbf{a}_i, x_i) + \sum_j W_{ij} \psi^*(\mathbf{a}_j, x_j) \tag{2}$$

where ϕ^*, ψ^* are MLPs (ReLU, 3 layers, hidden dim 64), $\mathbf{a}_i \in \mathbb{R}^2$ are learnable neuron embeddings, and W_{ij} is the learnable connectivity matrix. The loss combines prediction error with regularization terms enforcing zero steady state, exponential decay, monotonic transfer functions, and weight sparsity.

B Instruction File

```
# Simulation-GNN Training Landscape Study

## Goal

Map the **simulation-GNN training landscape**: understand which simulation
configurations allow successful GNN training (connectivity_R2 > 0.9) and
which are fundamentally harder.

## Iteration loop structure

Each block = `n_iter_block` iterations exploring one simulation configuration.
The prompt provides: `Block info: block {block_number}, iteration
{iter_in_block}/{n_iter_block} within block`

## File structure (CRITICAL)

You maintain TWO files:

### 1. Full Log (append-only record)

**File**: `{config}_analysis.md`

- Append every iteration's full log entry
- Append block summaries
- **Never read this file** | it's for human record only

### 2. Working Memory (active knowledge)

**File**: `{config}_memory.md`

- **READ at start of each iteration**
- **UPDATE at end of each iteration**
- Contains: knowledge base + previous block summary + current block details
- Fixed size (~500 lines max)
```

Iteration Workflow Step 1-4, every iteration

Step 1: Read Working Memory if exists

Read `{config}_memory.md` to recall:

- Established principles
- Previous block findings
- Current block progress

Step 2: Analyze Current Results

- `analysis.log`: metrics from training/test/plot:
  - `spectral_radius`: eigenvalue analysis of connectivity
  - `svd_rank`: SVD rank at 99% variance (activity complexity)
  - `test_R2`: R^2 between ground truth and rollout prediction
  - `test_pearson`: Pearson correlation per neuron (mean)
  - `connectivity_R2`: R^2 of learned vs true connectivity weights
  - `final_loss`: final training loss (lower is better)
- `ucb_scores.txt`: provides pre-computed UCB scores for all nodes

Step 3: Write Outputs

Append to Full Log** (`{config}_analysis.md`) and Current Block sections
of `{config}_memory.md`:

Log Form

```
## Iter N: [converged/partial/failed]
Node: id=N, parent=P
Mode/Strategy: [success-exploit/failure-probe]/[exploit/explore/boundary]
Config: lr_W=X, lr=Y, lr_emb=Z, coeff_W_L1=W, batch_size=B,
       low_rank_factorization=[T/F], low_rank=R, n_frames=NF
Metrics: test_R2=A, test_pearson=B, connectivity_R2=C, final_loss=D
Activity: [brief description]
Mutation: [param]: [old] -> [new]
Parent rule: [one line]
Observation: [one line]
Next: parent=P
```

Step 4: Edit config file for next iteration

- Classification
  - **Converged**: connectivity_R2 > 0.9
  - **Partial**: connectivity_R2 0.1-0.9
  - **Failed**: connectivity_R2 < 0.1

- Training Parameters (change within block)

```
  learning_rate_W_start: 2.0E-3  # range: 1E-4 to 1E-2
  learning_rate_start: 1.0E-4    # range: 1E-5 to 1E-3
  coeff_W_L1: 1.0E-5             # range: 1E-6 to 1E-3
  batch_size: 8                  # values: 8, 16, 32
  low_rank_factorization: False or True
  low_rank: 20                   # range: 5-100
```

- Simulation Parameters (change at block boundaries only)

```
  n_frames: 10000
  connectivity_type: "chaotic"   # or "low_rank"
  Dale_law: False or True
  Dale_law_factor: 0.5
  connectivity_rank: 20 if low_rank

## Parent Selection Rule (CRITICAL)

**Step A: Select parent node**

- Read `ucb_scores.txt`
- If empty → `parent=root`
- Otherwise → select node with **highest UCB**

**Step B: Choose strategy**

| Condition                          | Strategy          | Action                              |
| ---------------------------------- | ----------------- | ----------------------------------- |
| Default                            | **exploit**       | Highest UCB node, try mutation      |
| 3+ consecutive R²  0.9             | **failure-probe** | Extreme parameter to find boundary  |
| n_iter_block/4 consecutive successes | **explore**     | Select outside recent chain         |
| Good config found                  | **robustness-test**| Re-run same config                 |

## Block Workflow step 1 to 2, at the end of a block

**STEP 1 COMPULSORY modify Protocol (this file)**

1. **Branching rate**: Count unique parents in last n_iter_block/4 iters
   - If all sequential (rate=0%) → ADD exploration incentive to rules
2. **Improvement rate**: How many iters improved R²?
   - If <30% improving → INCREASE exploitation
   - If >80% improving → INCREASE exploration
3. **Stuck detection**: Same R² plateau (±0.05) for 3+ iters?
   - If yes → ADD forced branching rule

**STEP 2. Update Working Memory** (`{config}_memory.md`):

- Update Knowledge Base with confirmed principles
- Add row to Regime Comparison Table
- Replace Previous Block Summary
- Clear Current Block sections
- Write hypothesis for next block

## Domain Knowledge

### Spectral radius
- (W) < 1: activity decays → harder to constrain W
- (W)  1: edge of chaos → rich dynamics → good recovery
- (W) > 1: unstable

### Effective rank
- High (30+): full W recoverable
- Low (<15): only subspace identifiable → need factorization

### Learning rates
- lr_W:lr ratio matters (typically 20:1 to 50:1)
- Too fast  learning → noisy W gradients
```

C Memory File

```
# Working Memory: signal_chaotic_1_Claude

## Knowledge Base (accumulated across all blocks)

### Regime Comparison Table

| Block | Regime                   | eff_rank | Best R² | Optimal lr_W  | Key finding                          |
| ----- | ------------------------ | -------- | ------- | ------------- | ------------------------------------ |
| 1     | chaotic, Dale=False      | 31-35    | 1.000   | 8E-3 to 30E-3 | lr_W:lr ratio 40-100:1 works         |
| 2     | low_rank=20, Dale=False  | 6-12     | 0.977   | 16E-3         | ratio 320:1 needed; factorization    |
| 3     | chaotic, Dale=True       | 10       | 0.940   | 80E-3         | ratio 800:1 needed; E/I hardest      |
| 4     | low_rank=50, Dale=False  | 7-21     | 0.989   | 20E-3         | n_frames=20000 essential             |
| 5     | low_rank=20, Dale=True   | 28-30    | 1.000   | 8-10E-3       | factorization=False works            |
| 6     | low_rank=50, Dale=True   | 10-20    | 0.998   | 25E-3         | eff_rank variance drives R²          |
| 7     | chaotic, Dale=True       | 28-36    | 0.9999  | 40E-3         | n_frames=20000 solves Block 3        |
| 8     | low_rank=10, Dale=False  | 4-7      | 0.901   | 80E-3         | eff_rank=4 barrier; 7% converge      |

### Established Principles

- lr_W:lr ratio is the key parameter; optimal ratio varies by regime
  - chaotic unconstrained: 40-100:1 sufficient (easiest)
  - low_rank: 160-320:1 required
  - chaotic with Dale_law (n_frames=10000): 800:1 required (hardest)
  - chaotic with Dale_law (n_frames=20000): 400:1 sufficient
- L1 regularization must scale with connectivity complexity
  - chaotic unconstrained: L1=1E-5 optimal
  - low_rank / Dale_law: L1=1E-6 optimal (10x weaker)
- **n_frames is critical**: n_frames=20000 enables:
  - rich activity (effective_rank 28-36 vs 10)
  - perfect recovery even with double constraints (low_rank + Dale_law)
- **effective_rank is the primary predictor of achievable R²**
  - effective_rank20 → R²0.999+ achievable
  - effective_rank<15 → R²0.92 ceiling

### Open Questions

- what is the minimum n_frames needed for perfect recovery?
- is there a universal formula: effective_rank > threshold → factorization not needed?
- can n_neurons=1000 enable higher effective_rank for low_rank=10?

---

## Previous Block Summary (Block 8)

Block 8 (low_rank=10, Dale_law=False, n_frames=30000): Best R²=0.901, 1/14
converged (7%). Key finding: low_rank=10 produces effective_rank=4 typically,
which is a fundamental learnability barrier.

---

## Current Block (Block 9)

### Block Info

Simulation: connectivity_type=low_rank, connectivity_rank=10, Dale_law=True,
Dale_law_factor=0.5, n_frames=30000
Iterations: 128 to 143
```

```
### Hypothesis

Testing low_rank=10 with Dale_law=True to see if E/I constraints change
effective_rank dynamics. Based on Block 8:
- low_rank=10 typically produces effective_rank=4 (unlearnable)
- Dale_law may increase effective_rank by introducing E/I structure

### Iterations This Block

## Iter 128: partial
Node: id=128, parent=127
Config: lr_W=80E-3, lr=5E-5, L1=1E-6, batch_size=32, factorization=T
Metrics: test_R2=0.279, connectivity_R2=0.572
Activity: effective_rank(99%)=13, spectral_radius=0.649
Observation: Dale_law=True increased effective_rank from ~6 to 13

## Iter 129: partial
Node: id=129, parent=root
Config: lr_W=100E-3, lr=5E-5, L1=1E-6, batch_size=32
Metrics: test_R2=0.263, connectivity_R2=0.508
Observation: lr_W=100E-3 (ratio 2000:1) degraded R²; ratio too high

### Emerging Observations

- Dale_law=True with low_rank=10 produces effective_rank(99%)=12-13 vs ~4-7
- R²=0.508-0.615 range achieved - much higher than Block 8's typical 0.07-0.16
- lr_W:lr ratio trending: lower ratio consistently improves R²
```

D Cumulative scientific inference

Table A1: Induction (observations → pattern)

Single-shot	Cumulative
"lr_W=8E-3 works for this config" (iter 3)	"lr_W:lr ratio scales with constraint complexity: 40:1 chaotic, 320:1 low_rank, 800:1 Dale" (experiments 1–7)
"effective_rank=32, R^2=1.0" (iter 2)	"effective_rank $\geq 20 \Rightarrow R^2 \approx 0.999$; effective_rank $< 15 \Rightarrow R^2 \leq 0.92$" (experiments 1–8)

Table A2: Abduction (observation → explanatory hypothesis)

Single-shot	Cumulative
"R^2 jumped 0.16 to 0.89" (iter 125)	"effective_rank jumped 4→7; this causes R^2 improvement—12 prior failures at eff_rank=4 prove it" (iters 113–127)
"n_frames=10000 insufficient for Dale" (experiment 3)	"n_frames controls effective_rank ceiling; 20k enables eff_rank 28–36 vs 10" (experiments 3, 7)

Single-shot	Cumulative
"lr_W=8E-3 worked ⇒ try 10E-3" (iter 4)	"ratio 80:1 optimal experiment 1; experiment 2 needs 4× ⇒ predict lr_W=16E-3, lr=5E-5" (iter 29, verified)
"maybe n_frames=30000 helps" (iter 124)	"$\mathcal{H}_1$: n_frames↑ ⇒ eff_rank↑; $\mathcal{H}_2$: eff_rank↑ ⇒ R^2 ↑; ∴ predict n_frames=30k helps" (iter 124, falsified)

Table A3: Deduction (hypothesis → prediction)

Table A4: Falsification (prediction vs. outcome → reject/refine)

Single-shot	Cumulative
"low_rank unlearnable, give up" (iters 17–25)	"falsified: low_rank=20 unlearnable; refined: learnable with ratio≥320:1" (breakthrough iter 29)
"n_frames=30000 didn't help" (iter 124)	"falsified: n_frames always boosts eff_rank; refined: only when connectivity_rank > threshold" (eff_rank=4 unchanged)