ANUPNET: Label-Free Continual Learning in a Small Language Model with Rooms, Dreams and Sleep

Anup Kumar Baghel LexiNexiAI, Ujjain, India — admin@lexinexiai.com

Technical Report v0 (draft) — September 2026

Abstract

We study whether a small language model can learn from a single unlabeled stream of text — new domain after new domain — without storing old data, without replaying it, and without being told when the domain changes, while keeping most of what it learned before. We propose ANUPNET, a mechanism with four parts: (1) a frozen trunk ("hall") plus small gated side modules ("rooms") in every FFN and attention block; (2) *dreams* — text the model generates itself — used only as a contrast signal to teach gates when to open, not to retrain weights; (3) a *sleep* step that silences a room on its own dreams and re-tunes the keys of older rooms; and (4) a label-free stream controller that opens a new room on *novelty* (loss above the model's own noise band) and reopens an old one on *familiarity* (a frozen gate that wakes up). All decision thresholds are relative to the model's own statistics; no absolute constants are used for decisions.

On a 30M-parameter transformer trained sequentially on English Wikipedia → Python code → children's stories (800 steps per task), ANUPNET reduces average forgetting from 2.74 nats (naive fine-tuning) to 0.33 ± 0.05 (3 seeds), with zero stored data and zero task labels. A capacity-matched control with the same extra parameters but no mechanism is *worse* than naive fine-tuning (3.70), and freezing alone recovers little (2.58) — the gain comes from dreams teaching gates *when to speak*, not from extra parameters. In a fully unlabeled stream with revisits (wiki → code → stories → code → stories), the controller detects all 4 boundaries at the exact step with zero false alarms across 3 seeds, reopens the correct old room on revisit, and improves on the second visit. Acquisition of new tasks costs ~0.45 nats relative to naive fine-tuning; replay (which stores data) still forgets less (0.05). We also report a data-free consolidation step (room → trunk via distillation on dreams) that works but does not yet improve cross-domain mixing, and a mixed-input meter showing that rooms switch at token level within a sequence. We release the code and experiment log and outline a scale-up to a 125M model that is in progress.

---

1. Introduction

Humans learn continuously. A new skill does not erase an old one; details fade while the gist, the skill and the connections stay. Standard language models do not work this way: their weights are frozen after training, and fine-tuning them on a new domain overwrites the old one (catastrophic forgetting). The common defenses — replay buffers, regularization such as EWC, or task-conditioned parameter isolation — either store old data, need task labels, or both. A 2026 survey still lists "smooth learning across tasks and time" as unsolved.

This report asks a narrow, testable question: **can a small LM learn from one unlabeled stream without stored data, and how much does it forget?** We constrain ourselves with three design rules, adopted early and kept throughout:

1. **No stored old data, no replay.** The only memory of the past is in the weights.
2. **No task labels.** Domain names are used only by *meters* (evaluation), never by the mechanism.

3. **No absolute numbers in decisions.** Every threshold (when a boundary is detected, when a room is recognized, when to merge) is a ratio to the model's own running statistics. Engineering constants (side width, gate bias, top-k) are allowed but must show their effect on a meter.

Our contribution is a mechanism that satisfies these rules and an honest, seed-replicated measurement of what it buys and what it costs.

2. Related work (brief)

Parameter isolation. Progressive Networks and Expert Gate add task-specific columns/experts; ANUPNET's rooms are similar in spirit but are opened by the model's own surprise, not by a task id, and are gated by a signal learned from self-generated data. *Pseudo-rehearsal.* Robins (1995) and later "data-free" methods generate samples from the model to rehearse. Our dreams are related but deliberately weaker: they train only the gates (513 parameters per block), never the task weights — the model does not learn from its own hallucinations, it only learns *when a room belongs*. *Complementary Learning Systems and sleep.* Fast hippocampal learning followed by slow cortical consolidation motivates rooms (fast, small) and the consolidation step (room → trunk). *Data-free knowledge distillation and Net2Net-style widening* are used directly in our consolidation v0. *Fast weights, EWC-style importance, and MoE routing* were tested in earlier iterations of this project (EXP-1 to EXP-8) and rejected at this scale; see §5.6.

3. Method

3.1 Architecture: hall and rooms

Base model: a 30M-parameter decoder-only transformer (RoPE, RMSNorm, GELU FFN), vocabulary 10k, trained on the first domain. After the first domain, the trunk (all FFN and attention weights) is **frozen**; embeddings and attention keep a small residual learning rate (0.1×) in early experiments and are handled by attention side modules later.

Each block gets a **room**: in the FFN, a side network 512→512→512 (zero-initialized output); in attention, a low-rank side (rank 16) on the QKV and output projections. Each room has a **gate** `g = σ(w·sg(x) + b)` with bias initialized to −3 (closed), where `sg` is stop-gradient so gate training never flows into the trunk. Block output = trunk(x) + g · room(x). When a room is closed (see §3.4), its weights are frozen and it joins a list of frozen rooms whose gates still run in every forward pass; a new empty room is opened.

3.2 Dreams as contrast

A learned gate cannot learn "old vs new" if it has never seen old inputs — with no stored data, it simply learns "always open" (observed in EXP-9: gates open 81–95% on every domain). We supply the missing contrast from the model itself. When a room opens, the model **dreams**: 256 sequences of 256 tokens are sampled from a random start token (top-k 50). Every second training step adds `λ · mean(g on dreams)` (λ = 1) to the loss. Because the gate sees `sg(x)`, this gradient touches only the gate. Dreams are regenerated per room; with several frozen rooms, each room dreams by being forced open alone ("rooms dreams"), giving each room its own reference distribution.

3.3 Sleep: neend and chaabi

Two small additions at room-closing time:

- **Neend (silencing).** `loss += ν · mean(room(x)² on dreams)` ($\nu = 5$): the room learns to output nothing on old-looking input, so leakage through an imperfect gate does no harm.
- **Chaabi (key refresh).** Gates of *frozen* rooms stay trainable but receive gradient only from a key loss: open on their own room's dreams, closed on other rooms' dreams. This keeps an old room's key aligned with the slowly drifting input distribution (EXP-17 showed that without it, a frozen room's gate on its own domain decays from 62% to 47% across one task).

Neither helps alone (chaabi alone: 1.19; neend alone: 1.23); together they take forgetting from 0.87 to 0.73.

3.4 Stream controller: novelty and familiarity

The model sees one stream of batches; the source changes without notice. Two signals, both computed from the forward pass before the optimizer step:

- **Novelty.** Let `fast`, `slow` be EMAs of the batch loss (0.9, 0.99) and `dev = EMA(|L − fast|)`. If `L > slow + k·dev` (k = 4, a z-score-like rule), a boundary is declared *before* the step: the current room is closed (frozen, neend, chaabi), a new room opened, dreams generated, and the surprising batch is skipped. The detector re-arms after `1/(1−0.99)` steps of slow-EMA warm-up.
- **Familiarity.** On a novelty event, each frozen room's gate on the surprising batch is compared with its gate on its own dreams (`self_ref`). A room claims the batch if `g ≥ 0.5 · self_ref` **and** `g > g_active` (competition rule). If a room claims it, that room is **reopened** instead of creating a new one.

A "trunk key" (a gate on the frozen trunk that never blocks its output, only reports recognition) was implemented and found unnecessary at this stage (§5.5).

3.5 Consolidation v0 (room → hall)

To keep the number of rooms bounded, a room can be merged into the trunk without data: widen each FFN by K units (Net2Net-style: new input rows random, new output columns zero), and distill the per-block output `trunk(x) + g·room(x)` into `trunk_wide(x)` on dreams (MSE, only the new units train). The room is then dropped.

4. Experimental setup

Data. Three real domains: English Wikipedia, Python code, children's stories (~3MB each), tokenized with a shared 10k BPE. **Chain:** train 800 steps per domain in order wiki → code → stories (`exp4`), or as one unlabeled stream with revisits wiki → code → stories → code → stories (`exp5`, 5 × 800 steps).

Meters (evaluation only; never seen by the mechanism): * *Forgetting* — increase in a domain's validation loss between the end of its own training and the end of the chain; averaged over old domains ("AVG BHOOLNA"). * *Final* — mean validation loss over all domains at the end. * *Acquisition* — validation loss on each domain right after its own training. * *Gate meter* — mean gate opening of each room on each domain's validation set. * *Selectivity* — CORE vs DETAIL split of validation tokens (frequent vs rare); ratio of detail forgetting to core forgetting. * *Drift* — relative $\|\Delta\theta\|/\|\theta\|$ per parameter group. * *Mixed-input penalty* — for domain pair (A,B): `L(B | 128 tokens of A context) − L(B | 128 tokens of B context)`. * *Boundary accuracy* — detected vs true change steps, false alarms, rooms opened vs rooms needed.

Baselines. `none` (naive sequential fine-tuning of the full model), `replay` (30% of steps on stored old data — violates rule 1, used as an upper reference), `scratch` (a separate model

per domain), and two controls added after external review: `capacity-matched` (identical extra parameters per task, nothing frozen, no dreams/sleep) and `freeze-only` (frozen trunk + gated sides, no dreams/sleep).

Compute. Single NVIDIA L4 (later RTX Pro 6000) on Google Cloud Run jobs; one chain run takes 20–45 minutes. Seeds: 3 for headline results, 1 for ablations (marked).

5. Results

5.1 Headline: forgetting on the 3-domain chain

Method	Stored data	Labels	Avg forgetting ↓	Final loss ↓	Se
none (naive)	–	–	2.74	5.99	1
scratch (model per task)	–	yes	(4.90)	7.48	1
capacity-matched control	–	–	3.70	6.55	1
freeze-only	–	–	2.58	6.20	1
dream gate (FFN rooms)	–	–	0.87 ± 0.04	5.02	3
+ neend + chaabi	–	–	0.73	4.94	1
+ attention rooms (rank 16) = ANUPNET	**–**	**–**	**0.33 ± 0.05**	**4.70**	**3**
replay 30% (reference)	yes	yes	0.05	4.16	1

Decomposition (single seed unless noted): none 2.74 → +parameters 3.70 → +freeze 2.58 → +dreams/sleep 0.33 → replay 0.05. Extra capacity alone hurts; isolation alone barely helps; the mechanism that helps is dreams teaching gates *when* to open.

Acquisition cost. ANUPNET learns new domains slightly worse than naive fine-tuning: code 4.93 vs 4.49, stories 3.70 vs 3.20 (~0.45 nats). We initially reported "learning intact" by comparing against the wrong baseline; this was caught in external review and corrected.

Where the remaining 0.33 comes from. Attention drift was the largest remaining cause before attention rooms (0.73 → 0.33). Freezing embeddings does not reduce it further (0.34) but kills acquisition, and per-token embedding damping failed in five variants. Replacing sigmoid gates with top-1 hard gating did not help (EXP-34). We attribute the remainder to gate leakage (5–15% opening on old inputs) and leave it open.

5.2 Unlabeled stream with revisits (exp5)

Schedule wiki → code → stories → code → stories, boundaries at steps 801/1601/2401/3201, nothing told to the model.

Seed	Detected	False alarms	Rooms opened / needed	Avg
1	801, 1601, 2401→reopen code, 3201→reopen stories	0	2 / 2	0.4
2	same	0	2 / 2	0.3
3	same	0	2 / 2	0.4

Second visits improve on first: code 5.00 → 4.76, stories 3.72 → 3.46. The competition rule was necessary: a ratio-only recognition rule produced 7 false reopenings in one run (EXP-31b).

Two earlier detector designs failed instructively: a post-step EMA detector fired 4–5 steps late and tripled forgetting (the first steps after a boundary are the most destructive), and a "loss went down" detector for revisits fired on easy batches and missed real revisits when the old and new domains had similar loss. Novelty must be pre-step, and familiarity must come from keys, not from loss.

5.3 Consolidation v0

Merging the code room into the trunk with K = 512 new units, 5000 distillation steps and 32 dream batches costs +0.09 nats at merge time; the final code forgetting after the next domain is 0.69 versus 0.40 with the room kept. K = 128 (4× compression) costs +0.5. Short sleeps (200 steps, 8 dreams) fail outright. Merging works, but at this scale it neither saves much nor improves anything; its role is budget control (merge the least-used room, then drop it). A loss-aware naming mechanism is needed after merging because a merged memory no longer triggers novelty on revisit.

5.4 Mixed inputs

Average mixed-input penalty (lower is better): none 0.19 (meaningless — it has forgotten everything), replay 0.39, rooms 0.65, merged 0.70. Gates switch within a sequence (code gate rises from 18% to 76% across a wiki→code boundary inside one 256-token input). Rooms are 0.26 behind replay on mixing; merging does not close that gap.

5.5 Negative and null results (kept on purpose)

- Per-weight "strength"/importance damping (EWC-like, five variants): 8% gain at 5× dose; direction right, effect too small.
- MoE with 4 experts at this scale: router does not specialize (25/25/25/25), gain is from sparsity only.
- Emotion-weighted gate loss (core tokens ×3, detail ×0.3): within seed noise.
- Balanced dreams (seeding dreams with old-domain tokens): no change — dream composition was not the problem.
- Trunk key / naming: implemented, correct in 2 of 3 runs, but not needed while novelty alone handles revisits; misfires on the first unseen English task.
- Top-1 hard gating: no improvement over sigmoid.

5.6 What we are *not* claiming

- Not competitive with replay when storing data is allowed (0.33 vs 0.05).
- Not shown beyond 30M parameters, 3 domains and 800 steps per domain.
- Not shown on gradual domain mixtures — all boundaries here are sharp.
- Acquisition is ~0.45 nats worse than naive fine-tuning.
- Room count grows with the number of distinct domains; pruning/merging policy is not yet automatic.

6. Scale-up in progress: 125M

A 110M-parameter model (12 layers, d = 768, 12 heads, context 1024, 32k BPE) was pretrained on 1.05B tokens of FineWeb-Edu + Cosmopedia (validation loss 3.30 after 16k steps, ~6 GPU-hours on one RTX Pro 6000). Instruction tuning on SmolTalk and a retrieval/search tool follow; the continual stream with rooms will then be repeated at this scale with the same code. Results will be added in v1.

7. Reproducibility

Code, configs and the full experiment log (EXP-1 … EXP-34, including failures) are in the project repository: `modules.py` (SideFFN, attention side, gates, keys), `exp4_chain.py` (chain), `exp5_stream.py` (unlabeled stream), `probe_side.py`, `m125/` (scale-up). All numbers in this report are read from the saved result JSONs referenced in the log.

Acknowledgements

The experiments were designed and run in an interactive loop with Claude (Anthropic) acting as research assistant and coding partner. Written reviews from several LLM reviewers (Grok, GPT, Gemini, an in-house agent) caught three errors — the acquisition-cost overclaim, a self-reference allocation bug, and the missing capacity-matched control — all fixed before this draft.

Draft status: v0. To do before v1: 125M continual results; gradual-mixture validity test; automatic room budget (merge least-alive room); position-wise mixing meter; related-work citations with full references.