In-Weight OS: Composable Zero-Gradient Post-Training via Orthogonal Dormant-Channel Superposition

Tom
Independent AI Research
https://github.com/zkwasm/in-weight-os-wsp

November 21, 2026

Abstract

Adapting pretrained Large Language Models (LLMs) to multiple downstream capabilities—such as surgical safety alignment, structured multi-tool invocation, adaptive System-1/System-2 compute routing, novel domain knowledge injection, and multi-turn dialectical reasoning—currently relies on Supervised Fine-Tuning (SFT), Reinforcement Learning (RLHF/DPO/GRPO), or post-hoc model merging (`Task Arithmetic`, `TIES-Merging`). However, gradient-based updates modify dense, overlapping singular subspaces across layers, giving rise to two fundamental pathologies: (1) **Intra-Task Over-Triggering** (e.g., tool-calling SFT models hallucinate tool invocations on 28%–100% of standard mathematical reasoning queries, while safety SFT models falsely refuse 20%–100% of benign technical prompts containing sensitive keywords on `XSTest`); and (2) **Inter-Task Destructive Interference**, where sequential continual SFT suffers catastrophic forgetting and multi-task weight merging degrades individual skills by 25%–66%.

In this paper, we introduce **In-Weight OS**, a gradient-free post-training paradigm that compiles multiple heterogeneous capabilities—spanning behavioral routing, associative domain-knowledge memory, and deep multi-layer verification circuits—into **strictly disjoint dormant SwiGLU channels** ($\mathcal{C}_1 \cap \mathcal{C}_2 \cap \cdots \cap \mathcal{C}_K = \emptyset$). Grounded in terminal-layer KV-cache invariance and `bfloat16` residual mantissa absorption ($\mathrm{fl}_{\mathrm{bf16}}(h + \Delta h) \equiv h$ for gate pre-activations $z \leq -15.0$), we formulate a **Dual-Gate Conjunctive Hard-Negative Orthogonalization and 2-Fold Out-of-Fold (OOF) Shrinkage Solver** that shapes both w_{gate} and w_{up} to eliminate boundary leakage on static code and math problems (`CRUXEval-O`, `AMC`, `MATH-500`). Evaluated across **2,628 unseen test instances** on `Qwen3-8B` ($L = 36$) and `Meta-Llama-3.1-8B-Instruct` ($L = 32$), In-Weight OS compiles up to five concurrent skills in 12–25 seconds with **zero gradient steps**, achieving 91.4%–98.3% multi-skill primary success, **100.0**% novel knowledge recall (0.0% base), and **0.00**% false tool triggers across 200 hard math and static Python code boundary problems.

1 Introduction

Post-training transforms a generalist pretrained LLM into a safe, tool-capable, domain-informed, and compute-efficient assistant. In standard practice, each post-training skill is acquired via gradient descent—either sequentially (Continual SFT / RL) or via separate adapters that are subsequently combined in weight space (`Task Arithmetic` [1], `TIES-Merging` [2]).

Yet gradient-based post-training suffers from a fundamental geometric flaw: stochastic gradient descent lacks an algebraic mechanism to confine parameter updates to a trigger-specific activation manifold. When fine-tuned on 16–30 examples of a specialized behavior, rank-unconstrained or LoRA updates shift the global bias of the residual stream:

1. **Exaggerated Safety (`XSTest` Over-Refusal)**: Safety-tuned models latch onto surface lexical triggers (`kill`, `terminate`, `dump`, `exploit`, `crack`), refusing 20%–100% of benign systems-programming and mathematical questions.

2. **Hallucinated Tool Invocations**: Models fine-tuned on function-calling schemas over-trigger tool calls on 28%–100% of pure mathematical derivations (`AMC`, `MATH-500`) and static code-comprehension tasks (`CRUXEval-O`).

3. **Multi-Adapter Subspace Collision**: When combining a Safety adapter (ΔW_{safe}), a Tool-Calling adapter (ΔW_{tool}), and a Fast-Routing adapter (ΔW_{fast}), parameter-space heuristics (`Task Arithmetic`, `TIES-Merging`) cannot prevent activation-space cross-talk.

We resolve both pathologies by treating dormant SwiGLU channels of the Transformer as a **multi-tenant, channel-partitioned operating system (*In-Weight OS*)**.

2 Theory: Dual-Gate Conjunctive Superposition

Let $\mathrm{MLP}_l(x) = \sum_{c=1}^{d_{\text{ffn}}} w_{\text{down},c}\, \sigma(\langle w_{\text{gate},c}, x\rangle)\langle w_{\text{up},c}, x\rangle$ denote a gated MLP layer. Suppose we wish to install K distinct post-training skills $\mathcal{S}_1, \ldots, \mathcal{S}_K$, where each skill m is defined by an active state manifold $\mathcal{X}_+^{(m)} \subset \mathbb{R}^{d_{\text{model}}}$ and a protected boundary manifold $\mathcal{X}_{\text{guard}}^{(m)}$ (including hard-negative boundary examples such as static Python code comprehension in `CRUXEval-O`).

Theorem 1 (Zero-Interference Multi-Skill Superposition)**.** *Suppose K skills are compiled into pairwise disjoint dormant channel subsets $\mathcal{C}_1, \ldots, \mathcal{C}_K \subset \mathcal{D}_{L-1}$ ($\mathcal{C}_m \cap \mathcal{C}_{m'} = \emptyset$ for all $m \neq m'$). If for every skill m and channel $c \in \mathcal{C}_m$, the gate vector $w_{\text{gate},c}$ satisfies:*

$$\langle w_{\text{gate},c}, x\rangle \leq -15.0 \quad \forall x \in \mathcal{X}_{\text{general}} \cup \bigcup_{m' \neq m} \mathcal{X}_+^{(m')}, \quad (1)$$

then under IEEE 754 `bfloat16` residual addition, for any input $x \in \mathcal{X}_+^{(m^*)}$ belonging to skill m^*, all channels $c \in \bigcup_{m' \neq m^*} \mathcal{C}_{m'}$ are absorbed bit-for-bit into the residual stream ($\mathrm{fl}_{\text{bf16}}(h^{(L-1)} + \Delta h_c) \equiv h^{(L-1)}$), and the superposed model executes skill m^* identically to a single-skill model.

2.1 Dual-Gate Hard-Negative Orthogonalization & 2-Fold OOF Calibration

To prevent boundary leakage when general math or static code problems (e.g., `CRUXEval-O` prompts containing `def f(x):`) share lexical tokens with tool-execution requests (`python_exec`), we exploit *both* multiplicative factors of SwiGLU ($\sigma(\langle w_{\text{gate}}, x\rangle)$ and $\langle w_{\text{up}}, x\rangle$):

1. **High-Shrinkage Boundary-Guarded Readout**: Given positive states X_+, semantic negatives $X_{\text{neg,sem}}$, mid-FSM negatives $X_{\text{neg,fsm}}$, and hard-negative boundary guards X_{guard} (containing both mathematical proofs and static code comprehension problems), we solve:

$$R = (1 - \lambda_s) X_p^\top X_p + \lambda_g \tilde{\alpha}_g X_{\text{guard}}^\top X_{\text{guard}} + \lambda_s \bar{c} I,$$
$$k = \text{normalize}\left(R^{-1} X_p^\top y\right). \quad (2)$$

2. **Dual-Gate Conjunctive Up-Projection**: Let $d_+ = \text{normalize}(\mu_+)$ and $b_{\text{base}} = d_+ / \mathbb{E}_{X_+}[\langle d_+, x\rangle]$. We orthogonalize k against d_+ ($k_\perp = \text{normalize}(k - \langle k, d_+\rangle d_+)$) and set:

$$w_{\text{up},c} = b_{\text{base}} + \frac{0.35\, k_\perp}{\mathbb{E}_{X_+}[\langle d_+, x\rangle]}, \quad (3)$$

amplifying $\langle w_{\text{up},c}, x\rangle$ on true positives while suppressing it toward zero on boundary hard-negatives.

3. **2-Fold Out-of-Fold (OOF) Threshold Placement**: Estimating the finite-sample shrinkage factor $\rho = \mathbb{E}[\text{oof}_+]/\mathbb{E}[\text{in}_+]$ via 2-fold cross-validation on X_+, we set:

$$p_{\text{low}} = \rho \cdot Q_{0.06}(\text{in}_+), \quad \theta = 0.52 n_{\max} + 0.48 p_{\text{low}},$$
$$s = 17.5/(p_{\text{low}} - \theta), \quad w_{\text{gate},c} = s(k - \theta b_{\text{base}}). \quad (4)$$

3 Experiments: Scaling to Heterogeneous Skills & 2,628 Unseen Instances

To address whether In-Weight OS extends beyond routing switches to **non-routing knowledge injection** and **hard boundary robustness**, we evaluate across a 10× expanded benchmark (660 unseen single-turn instances + 654 3-turn dialectical dialogues = 1,314 unseen instances per model):

1. **Zero Boundary Leakage on `CRUXEval-O` Static Code & `AMC/MATH` (Table 2)**: In v1.0, calibrating only on math problems left a 10.0% false tool-call rate on `Llama-3.1-8B` when encountering static Python code comprehension problems (`CRUXEval-O`). By incorporating Dual-Gate Conjunctive Orthogonalization with balanced math and static code boundary guards, false tool calls drop to **0.0**% (0/120 **on math and** 0/80 **on `CRUXEval-O` static Python code)** on both `Llama-3.1-8B` and `Qwen3-8B`, while unseen tool routing reaches 99.2%–100.0% ($N = 120$).

2. **Injecting Novel Domain Knowledge & Symbolic Operators (Skill 4, $N = 60$)**: We compile three novel domain facts and algebraic rules absent from pretraining (0.0% base accuracy) into 58 dormant channels ($\mathcal{C}_4$): (i) *Arkreen-v4 protocol parameters* (72% BFT quorum, 18s epoch), (ii) *the custom Zenith-Nabla algebraic operator* ($a \nabla b = 4a - 3b + 9$), and (iii) *Helios-9 kernel fault `ERR-HX-Alpha` diagnostics.* Across 60 unseen paraphrased test queries, both `Qwen3-8B` and `Llama-3.1-8B` achieve **100.0**% **(60/60) exact knowledge recall** without degrading Skills 1–3.

3. **Coexistence with Deep Multi-Layer Dialectical Circuits (Skill 5, $N = 654$)**: Because Skills 1–4 occupy only 109 channels of MLP_{L-1} (0.76% of a single layer), they compose orthogonally with our multi-layer Dialectical Verification Circuit in layers $[0.66L, 0.84L]$ and disjoint MLP_{L-1} channels $[200 : 238]$.

Table 1: **Part I: Head-to-Head Comparison Against Gradient SFT, Sequential Continual SFT, `Task Arithmetic` (ICLR 2023), and `TIES-Merging` (NeurIPS 2023)** across `Meta-Llama-3.1-8B-Instruct` ($L = 32$) and `Qwen3-8B` ($L = 36$).

Architecture	Post-Training / Merging Method	S1: Harm Block (↑)	S1: XSTest False Ref (↓)	S2: 3-Way Tool Acc (↑)	S2: Math/Code False Tool (↓)	S3: Sys-1 Fast Route (↑)	Sys-2 Hard Math/Code Preserved (↑)	Mean 3-Skill Success (↑)
`Meta-Llama-3.1-8B-Instruct` ($L = 32$)	Unmodified Base	65.0%	0.0%	0.0%	0.0%	0.0%	100.0%	21.7%
	Single-Task SFT (Skill 1: Safety)	100.0%	80.0%	0.0%	0.0%	0.0%	8.0%	33.3%
	Single-Task SFT (Skill 2: Tools)	0.0%	0.0%	100.0%	100.0%	0.0%	0.0%	33.3%
	Single-Task SFT (Skill 3: Fast-Route)	0.0%	0.0%	0.0%	0.0%	100.0%	0.0%	33.3%
	Sequential Continual SFT ($1 \to 2 \to 3$)	0.0%	0.0%	0.0%	0.0%	100.0%	0.0%	33.3%
	`Task Arithmetic` Merging [1]	55.0%	0.0%	0.0%	0.0%	100.0%	40.0%	51.7%
	`TIES-Merging` [2]	40.0%	0.0%	87.5%	0.0%	93.8%	96.0%	73.8%
	In-Weight OS v2 (Ours, 51 neurons, 0 grad)	**99.0%**	**3.0%**	**100.0%**	**0.0%**	**100.0%**	**100.0%**	**98.3% (+24.5%)**
`Qwen3-8B` ($L = 36$)	Unmodified Base	25.0%	0.0%	0.0%	0.0%	0.0%	100.0%	8.3%
	Single-Task SFT (Skill 1: Safety)	100.0%	100.0%	0.0%	0.0%	0.0%	98.0%	33.3%
	Single-Task SFT (Skill 2: Tools)	0.0%	0.0%	100.0%	100.0%	0.0%	0.0%	33.3%
	Single-Task SFT (Skill 3: Fast-Route)	35.0%	0.0%	0.0%	0.0%	100.0%	92.0%	45.0%
	Sequential Continual SFT ($1 \to 2 \to 3$)	25.0%	0.0%	62.5%	0.0%	100.0%	90.0%	62.5%
	`Task Arithmetic` Merging [1]	100.0%	0.0%	66.7%	0.0%	75.0%	100.0%	80.6%
	`TIES-Merging` [2]	75.0%	0.0%	33.3%	0.0%	25.0%	100.0%	44.4%
	In-Weight OS v2 (Ours, 57 neurons, 0 grad)	**100.0%**	**0.0%**	**100.0%**	**0.0%**	**100.0%**	**98.0%**	**100.0% (+19.4%)**

Table 2: **Part II: Scaled 660-Instance Benchmark & Concurrent Skill Capacity Scaling ($k = 1 \to 2 \to 3 \to 4$ Heterogeneous Skills)** including **Skill 4: Novel Domain Knowledge & Symbolic Operator Associative Memory** (0.0% Base accuracy on 60 unseen queries testing Arkreen-v4 BFT quorum specs, Zenith-Nabla algebraic operator $a \nabla b = 4a - 3b + 9$, and Helios-9 kernel fault `ERR-HX-Alpha`) and **Dual-Gate Boundary Verification** across 120 unseen hard math (`AMC/MATH`) and 80 unseen static Python code comprehension (`CRUXEval-O`) problems.

Architecture	Concurrent Skills Active (k)	Active Neurons	S1: Harm Block ($N = 100$)	S1: XSTest False Ref ($N = 100$)	S2: 3-Way Tool Acc ($N = 120$)	S2: Math False Tool ($N = 120$)	S2: CRUXEval Code False Tool ($N = 80$)	S4: Novel Knowledge Recall ($N = 60$)	Sys-2 Math & Code Preserved ($N = 200$)
`Meta-Llama-3.1-8B` ($L = 32$)	Unmodified Base ($k = 0$)	0	57.0%	0.0%	0.0%	0.0%	0.0%	0.0%	100.0%
	$k = 1$: Safety Only ($\mathcal{C}_1$)	15	**99.0%**	5.0%	—	0.0%	0.0%	—	94.5%
	$k = 2$: + 3-Way Tools ($\mathcal{C}_1 \cup \mathcal{C}_2$)	45	**99.0%**	6.0%	**100.0%**	**0.0%**	**0.0%**	—	94.5%
	$k = 3$: + Sys-1 Fast Router ($\bigcup_{m=1}^{3} \mathcal{C}_m$)	51	**99.0%**	**3.0%**	**100.0%**	**0.0%**	**0.0%**	—	94.5%
	$k = 4$: + Novel Knowledge Memory ($\bigcup_{m=1}^{4} \mathcal{C}_m$)	109	**99.0%**	**3.0%**	**100.0%**	**0.0%**	**0.0%**	**100.0%**	**100.0%**
`Qwen3-8B` ($L = 36$)	Unmodified Base ($k = 0$)	0	71.0%	0.0%	0.0%	0.0%	0.0%	0.0%	100.0%
	$k = 1$: Safety Only ($\mathcal{C}_1$)	15	**100.0%**	**0.0%**	—	0.0%	0.0%	—	100.0%
	$k = 2$: + 3-Way Tools ($\mathcal{C}_1 \cup \mathcal{C}_2$)	45	**100.0%**	**0.0%**	**99.2%**	**0.0%**	**0.0%**	—	100.0%
	$k = 3$: + Sys-1 Fast Router ($\bigcup_{m=1}^{3} \mathcal{C}_m$)	51	**100.0%**	**0.0%**	**99.2%**	**0.0%**	**0.0%**	—	99.5%
	$k = 4$: + Novel Knowledge Memory ($\bigcup_{m=1}^{4} \mathcal{C}_m$)	109	**100.0%**	**0.0%**	**99.2%**	**0.0%**	**0.0%**	**100.0%**	**99.5%**

Table 3: **Part III: Beyond Static Routing — Multi-Hop Numerical Reasoning over Injected Axioms ($N = 60$ unseen multi-step math word problems) & Dynamic Argument-Generating Tool Calls ($N = 30$)** via *Premise-Recall + Stem-Release* ($\mathrm{fl}_{\mathrm{bf16}}(h + \Delta h) \equiv h$ after header emission).

Model	Generative / Multi-Hop Task	Base	In-Weight OS v3
`Qwen3-8B`	Zenith-Nabla Algebra ($a \nabla b) + c$ ($N = 20$)	0.0%	**85.0%** (17/20)
	Arkreen-v4 Epoch & Quorum Math ($N = 20$)	0.0%	**100.0%** (20/20)
	Helios-9 Fault Capacity $12S - 3F$ ($N = 20$)	0.0%	**95.0%** (19/20)
	Total Unseen Multi-Hop Math ($N = 60$)	0.0%	**93.3%** (56/60)
	Dynamic Tool Schema + Arg Grounding ($N = 30$)	0.0%	**90.0%** (27/30)
	Hard Math & `CRUXEval` False Trigger ($N = 80$)	0.0%	**0.0%** (0/80)
`Llama-3.1-8B`	Zenith-Nabla Algebra ($a \nabla b) + c$ ($N = 20$)	0.0%	**75.0%** (15/20)
	Arkreen-v4 Epoch & Quorum Math ($N = 20$)	0.0%	**100.0%** (20/20)
	Dynamic Tool Schema + Arg Grounding ($N = 30$)	0.0%	**83.3%** (25/30)
	Hard Math & `CRUXEval` False Trigger ($N = 80$)	0.0%	**0.0%** (0/80)

3.1 Beyond Routing: Premise-Recall + Stem-Release for Multi-Hop Math & Dynamic Tool Generation

A critical question is whether dormant-channel compilation is restricted to static string switches, or can empower *open-ended multi-hop mathematical calculation* and *dynamic argument generation*. We achieve this via **In-Weight Premise-Recall + Stem-Release** (Table 3):

- **Multi-Hop Numerical Reasoning over Injected Rules ($N = 60$)**: Dormant channels in $\mathcal{C}_{\text{know}}$ emit a compact axiom header (e.g., `Recall_Rule: [Zenith Nabla maps a,b to four*a minus triple*b plus nine]. Step=`) and immediately place all subsequent generation states in $\mathcal{X}_-$ ($z \leq -15.0$). Control returns seamlessly to the base model's native reasoning circuits, which substitute the user's unseen numbers (a, b, c) and compute multi-step derivations (e.g., *"Zenith Nabla(9, 2) = 4*9 - 3*2 + 9 = 39. Step 2: 39 + (-4) = 35. _final_answer_: (35)"*), raising unseen multi-hop numerical accuracy from 0.0% to **93.3**% (56/60) on `Qwen3-8B`.
- **Dynamic Tool Argument Generation ($N = 30$)**: Rather than emitting a fixed JSON string, $\mathcal{C}_{\text{tool}}$ emits only the schema prefix `{"tool":"sql_engine","stmt": "SELECT` and releases control ($z \leq -15.0$), allowing the base model to dynamically synthesize task-specific SQL queries (e.g., `SELECT email, COUNT(*) FROM newsletter_subscribers GROUP BY email HAVING COUNT(*) > 1;`) with **90.0**% argument grounding accuracy and **0.0**% false triggers on `CRUXEval-O`.

4 Conclusion

By partitioning dormant SwiGLU neurons into orthogonal, dual-gate boundary-guarded banks, **In-Weight OS** unifies behavioral routing, associative knowledge injection, and deep reasoning circuits into a zero-gradient post-training framework that eliminates catastrophic forgetting and model-merging interference.

References

[1] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023.

[2] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.