RETHINKING WHAT LLMS TRANSFER TO RECOMMENDATION: LOCALIZING AND RECOMBINING PRETRAINED COMPUTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) initialization improves recommendation, with its benefits often attributed to general capabilities that existing methods seek to preserve. However, recovering general-task performance through mixed supervision brings no clear additional recommendation gain. Yet shared-neuron ablations reveal continued dependence on resources used by general tasks. What, then, does pretrained initialization supply that recommendation actually uses? We derive a cross-entropy training-gain bound in a simplified feed-forward model, separating the improvement available from initial computation from the cost of learning its recombination. Neuron-level analyses support this account: many channels used by the trained recommender are identifiable before training and remain useful as adaptation changes their task allocation. These findings motivate *initial localization-guided neuron recombination*, which selects channels from their initial activity on recommendation contexts and learns a low-rank residual mapping to recombine their responses after recommendation fine-tuning. Across three domains and four LLM backbones, our method consistently improves over standard fine-tuning, and ablations supporting both initial localization and learned recombination.

1 INTRODUCTION

Large language models (LLMs) increasingly provide pretrained foundations for downstream learning(Raffel et al., 2020). Recent work has extended this approach to recommendation, adapting LLMs with user interaction data to learn user preferences and achieving strong results across recommendation benchmarks (Zhu et al., 2024; Geng et al., 2022; Bao et al., 2023; Liao et al., 2023).

The benefits of LLMs for recommendation are often attributed to their world knowledge and general capabilities, which complement the behavioral patterns learned from interaction data (Wei et al., 2024). Accordingly, existing methods seek to retain these resources during adaptation through strategies such as freezing pretrained parameters, mixing general-domain supervision, and distilling from the original model (Jia et al., 2025; Zhou et al., 2025).

However, our experiments reveal a mismatch between general-task recovery and recommendation improvement. LLM initialization outperforms random backbone initialization, but recovering mathematics, code, and knowledge-question-answering performance through mixed supervision yields no clear additional recommendation gain (Figure 1). At the same time, interventions on neurons shared by recommendation and general tasks sharply damage the trained recommender, showing that recommendation still depends on some of these shared resources. This combination raises a more specific question: *what does pretrained initialization supply that recommendation actually uses, and how can that computation be better adapted?*

We address this question by deriving a cross-entropy training-gain bound in a simplified feed-forward model, separating the improvement available from initial computation from the cost of learning its recombination. The resulting design principle is to locate useful pretrained computation and explicitly learn its recommendation-specific recombination. Our neuron-level analyses show that LLM initialization provides a reusable neuronal foundation: many positions used by the trained recommender are identifiable before training, an initially shared subset remains functionally important afterward, and adaptation reuses inherited positions while changing their task allocation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Guided by these findings, we propose **initial localization-guided neuron recombination**. The method selects channels from their initial activity on recommendation contexts and learns a low-rank residual mapping to recombine their responses after recommendation SFT. Localization determines which channels provide inputs to the mapping, while the mapping learns their recommendation-specific recombination. This additional stage trains only the recombination parameters using recommendation supervision, turning the identified pretrained resources into explicit targets for downstream adaptation. Across three recommendation domains and four LLM backbones, our method consistently improves over SFT, with ablations supporting its two guiding principles: locating useful initial channels and learning their recommendation-specific recombination.

2 The Separation Between Capability Recovery and Recommendation

Retaining an LLM's general capabilities is an appealing way to preserve its value for recommendation. We examine this connection through two comparisons: whether pretrained initialization improves recommendation, and whether recovering general-task performance brings further gains.

Comparison setup. We study generative product recommendation under three initializations: **Random**, **Qwen** with pretrained Qwen3-1.7B weights (Yang et al., 2025), and recommendation-pretrained **OneRec** (Zhou et al., 2025). Random and Qwen share the same architecture and learned semantic-ID (SID) interface. Each model is trained on recommendation data alone (**Rec**) or with additional mathematics, code, and knowledge supervision (**Mix**), which retains the recommendation data and adds training compute. Comparisons use the same evaluation cohort; details are in Appendix A.

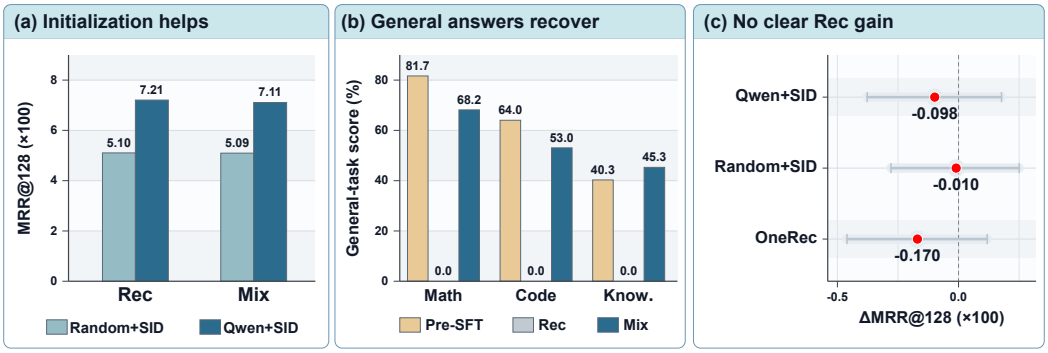

Figure 1: **Initialization helps, but capability recovery brings no clear further recommendation gain.** (a) Qwen outperforms random backbone initialization with the same learned SID interface. (b) Mix recovers Qwen's ordinary general-task performance. (c) Recommendation remains comparable under Mix, with slightly lower point estimates than Rec; paired 95% intervals for Mix-minus-Rec span zero. Intervals resample users for fixed trained models.

2.1 Pretrained Initialization Provides a Downstream Advantage

Under recommendation-only training, Qwen outperforms Random on all four recommendation metrics (Figure 1(a); Appendix E). Since the models share the same learned item interface, this advantage demonstrates the value of the pretrained backbone. The advantage also persists under Mix, indicating that additional general-task supervision does not replace the pretrained starting point.

This establishes that pretrained resources are useful for recommendation. We next ask whether recovering the model's general-task performance makes those resources more useful.

2.2 General-task Recovery Brings no Clear Recommendation Gain

Mix achieves the intended general-task recovery: Qwen regains substantial performance on mathematics, code, and knowledge questions (Figure 1(b)). Yet recommendation does not improve in

step. Across all three initializations, Mix yields similar or slightly lower recommendation scores than Rec, with paired intervals for the changes spanning zero (Figure 1(c)). The same pattern holds across the other recommendation metrics and when general tasks are added individually to OneRec (Appendices E and E.1).

2.3 Recommendation Still Depends on Task-shared Neurons

To test whether recommendation still uses task-shared resources, we localize neurons separately on recommendation and general-task prompts in the initial Qwen model (Appendix B). The selected sets overlap. Removing a fixed subset of these shared positions sharply degrades recommendation after both Rec and Mix, whereas matched random removal has little effect (Figure 2(a); Appendix G).

Recommendation still depends on task-shared resources, yet recovering general-task performance brings no clear additional gain. These findings motivate a closer examination of LLM initialization: *what makes the pretrained starting point useful for recommendation, and how can this advantage be better exploited?*

3 What Pretraining Leaves for Recommendation

LLM initialization improves recommendation, yet general-task recovery brings no clear further gain. What does pretraining supply that recommendation training can exploit? We derive a cross-entropy training bound that separates the usefulness of initial computation from the cost of learning its new contributions. Our empirical analysis then shows that pretrained initialization supplies useful neurons, which are reused with changing task allocation as recommendation performance improves.

3.1 Formalizing Recommendation Training

Given user history x_i and target item-identifier sequence s_i, recommendation SFT minimizes the answer-token cross-entropy

$$\mathcal{L}_{\mathrm{Rec}}(\theta) = -\frac{1}{N}\sum_{i}\sum_{t=1}^{|s_i|}\log p_\theta(s_{i,t} \mid x_i, s_{i,<t}), \qquad N = \sum_i |s_i|, \tag{1}$$

where θ denotes the model parameters and p_θ its next-token distribution. Pretrained and random initialization give different starting parameters θ_0 for the same objective.

To examine their role, we separate the computation produced by a neuron from its contribution to the output, following analyses of feature reuse (Lippl & Lindsey, 2024). In the feed-forward network (FFN) at layer l, let $z_l(h_l;\theta) \in \mathbb{R}^{m_l}$ denote the intermediate activation vector:

$$y_l = W_l^{\mathrm{down}} \underbrace{\left[\mathrm{SiLU}(W_l^{\mathrm{gate}} h_l) \odot (W_l^{\mathrm{up}} h_l)\right]}_{z_l(h_l;\theta)} = \sum_{j=1}^{m_l} w_{lj} z_{lj}(h_l;\theta). \tag{2}$$

Here $h_l \in \mathbb{R}^d$ is the layer input, and m_l is the number of intermediate neurons, also called FFN channels. The scalar activation z_{lj} is the jth entry of z_l, and $w_{lj} \in \mathbb{R}^d$ is the jth column of the output projection W_l^{down}. The activation specifies the neuron's response to the input; the column specifies how this response contributes to the output. Training can change both z_{lj} and w_{lj}.

3.2 Initial Computation and the Cost of Recombination

To derive how these two components affect learning, we study a simplified FFN layer following two-factor initialization analyses (Azulay et al., 2021; Lippl & Lindsey, 2024). For an answer-token context X, we fix the input $h_l(X)$, gate weights W_l^{gate}, and unit weight directions $p_j, e_j \in \mathbb{R}^d$. Training changes the scalar amplitudes of the up-projection row $(W_l^{\mathrm{up}})_{j,:} = u_j p_j^\top$ and down-projection column $w_{lj} = v_j e_j$. Consequently,

$$z_{lj}(h_l(X)) = u_j \xi_j(X), \qquad w_{lj} z_{lj} = \underbrace{u_j v_j}_{\beta_j} e_j \xi_j(X). \tag{3}$$

The fixed function ξ_j is the input-response pattern determined by the gate and direction p_j. The amplitude u_j controls the strength and sign of this response; v_j controls its contribution along output direction e_j. Their product β_j is therefore the effective weight of the neuron's computation. We write $\mathcal{L}(\beta)$ for the cross-entropy of an analytical predictor that combines these contributions linearly into token logits (Appendix C.1).

Let u_0, v_0 denote the initial amplitudes, with effective weights $\beta_0 = u_0 \odot v_0$. Under gradient flow, the continuous-time limit of gradient descent, their combined update is

$$\dot{\beta}_j = -(u_j^2 + v_j^2) g_j, \qquad g_j = \frac{\partial \mathcal{L}}{\partial \beta_j}, \tag{4}$$

where the dot denotes a time derivative. The gradient g_j specifies the task-directed change, while $u_j^2 + v_j^2$ determines its update scale. Applying initialization-geometry tools (Gunasekar et al., 2018; Azulay et al., 2021) to these dynamics, we derive a finite-time guarantee (Proposition 1). For a finite target combination $\bar{\beta}$, the training gain $\mathcal{G}_T = \mathcal{L}(\beta_0) - \mathcal{L}(\beta(T))$ after time $T > 0$ satisfies

$$\mathcal{G}_T \geq \underbrace{\mathcal{L}(\beta_0) - \mathcal{L}(\bar{\beta})}_{\text{gain available from initial computation}} - \frac{1}{T} \underbrace{\mathcal{C}_0(\bar{\beta}, \beta_0)}_{\text{cost of recombination}} \tag{5}$$

Here $\mathcal{C}_0$ is the nonnegative, initialization-induced cost of changing the effective weights from β_0 to $\bar{\beta}$. We call this change *recombination*: learning new contributions for existing computation through updates to either factor. The conditions, exact cost, and proof appear in Appendix C.2.

Initial Computation Defines the Opportunity for Improvement. The first term asks how much recommendation loss can be reduced using the available computation. Consider a target $\bar{\beta}$ whose changes are confined to an initial set of positions S. If their responses support a better recommendation predictor, this term is positive. Thus, localization identifies candidate neurons whose computation can support improvement when appropriately combined. The set S specifies a comparison target; training still updates all neurons.

Recombination determines the cost of realizing that opportunity. The second term accounts for learning the required combination. Let $\delta_j = \bar{\beta}_j - \beta_{0j}$ be the required weight change for neuron j, and let $q_{0j} = u_{0j}^2 + v_{0j}^2$ be its initial update scale from Equation 4. For small changes,

$$\mathcal{C}_0(\beta_0 + \delta, \beta_0) = \frac{1}{2} \sum_j \frac{\delta_j^2}{q_{0j}} + O(\|\delta\|_2^3). \tag{6}$$

The leading term is a weighted squared distance: larger required changes cost more, while a larger initial update scale makes the same change less costly. The remainder is cubic in the Euclidean norm of δ; Appendix C.3 derives this expansion and bounds the remainder. For a target loss reduction $A = \mathcal{L}(\beta_0) - \mathcal{L}(\bar{\beta}) > 0$, the bound guarantees positive gain once $T > \mathcal{C}_0/A$.

Initialization shapes both the computation available to recommendation and the cost of learning its new use. The first factor makes useful initial neurons worth locating; the second makes their recombination consequential under a finite training budget. General-task answer recovery does not determine either factor. Improving transfer therefore calls for retaining useful computation while learning its recommendation-specific contributions at a lower cost.

3.3 Initial localization and reuse in recommendation training

We examine these two factors empirically through initial localization and subsequent changes in task allocation.

LLM initialization supplies useful computation. Neuron localization shows that over half of Qwen's final recommendation positions were already recommendation-related at initialization, while most positions under random initialization were initially unselected (Figure 2(b)). Removing initially shared positions sharply degrades Rec and Mix performance, unlike matched random removal (Figure 2(a); Appendix G). Initial localization therefore identifies pretrained computation that remains functionally important after training.

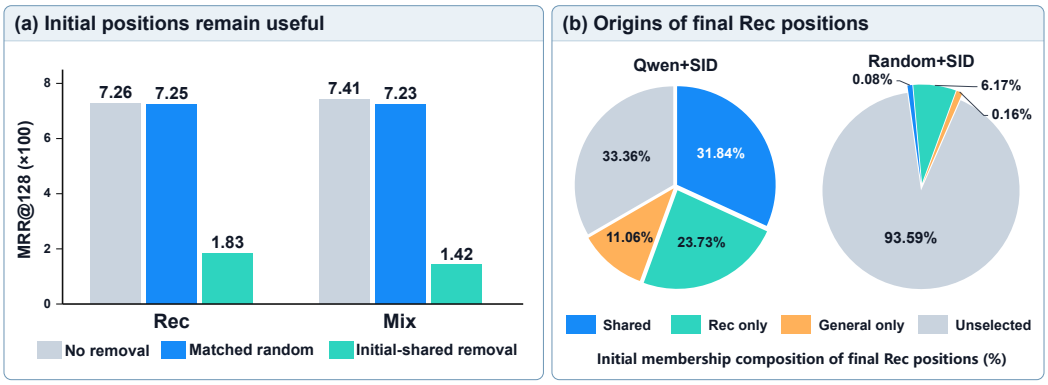

Figure 2: **Initially identified neurons remain useful and acquire new task roles.** (a) Qwen recommendation performance after removing a fixed initially shared subset or matched random positions, compared with no removal. (b) Initial task membership, as percentages of each model's final recommendation set after Rec. Shared + Rec only gives its initially recommendation-selected share; Unselected means selected by neither recommendation nor general tasks at initialization.

Training reuses neurons and reorganizes their task roles. We next examine how recommendation uses this foundation across tasks. Initially shared and recommendation-only neurons remain involved, while initially general-only neurons are recruited (Figure 2(b)). Fixed-cohort selection frequencies show sustained engagement of shared neurons and increased recommendation use of general-only neurons (Appendix F). Training thus reuses inherited positions while changing their task allocation.

LLM initialization supplies reusable computation for recommendation. In Equation 5, useful initial neurons support potential improvement, while the second term captures the cost of learning their new contributions. General-task answer recovery captures neither factor. This motivates retaining useful initial computation while facilitating its recommendation-specific recombination.

4 Initial Localization-Guided Neuron Recombination

To identify and better use the pretrained computation that supports recommendation, we propose *initial localization-guided neuron recombination*. The method locates useful FFN channels from their initial activations on recommendation contexts, then learns how to recombine their responses through a low-rank residual mapping. Initial localization determines which computation to use, while the learned mapping adapts its contributions to item prediction.

4.1 Localizing Useful Channels before Adaptation

To locate useful channels, we evaluate the initial model θ_0 on recommendation prompts and select channels with consistently strong contributions to the FFN output. Following the localization criterion used in our analysis, we rank channels by their activation magnitude weighted by the norm of their output-projection column, and retain those frequently selected across prompts (Appendix D.1). Since initially shared neurons contribute to subsequent recommendation training (Section 3.3), we retain selected channels shared with general tasks for later recombination. Let S_l denote the selected channels in layer l, $k_l = |S_l|$, and $P_l \in \{0,1\}^{k_l \times m_l}$ the matrix that extracts their activations from the layer's m_l channels. We use S_l as the input support for subsequent recombination.

The selected positions identify where to read useful computation. To adapt that computation to recommendation, we first perform ordinary full-parameter SFT on the original multi-target Semantic ID (SID) answers, obtaining θ_{SFT}. The recombination module then reads the activations produced at these positions in each forward pass of this model. Thus, the positions come from initial localization, while their responses reflect recommendation training.

4.2 Learning to Recombine the Selected Responses

Following the FFN decomposition in Section 3.1 (Equation 2), we implement recombination by learning additional output vectors for the selected channels. For layer input h_l, let $z_l = z_l(h_l; \theta_{\mathrm{SFT}}) \in \mathbb{R}^{m_l}$ be the FFN intermediate activations, and let $w_{lj} \in \mathbb{R}^{d_l}$ be column j of its output projection W_l^{down}, where d_l is the hidden width. The original FFN output and the additional contribution are

$$y_l = \sum_{j=1}^{m_l} w_{lj} z_{lj}, \qquad \Delta y_l = \sum_{j \in S_l} \delta w_{lj} z_{lj}, \tag{7}$$

where z_{lj} is channel j's activation and $\delta w_{lj} \in \mathbb{R}^{d_l}$ is its learned additional output vector. Learning these vectors changes how the selected responses enter the hidden representation, making their recombination an explicit target of adaptation.

Section 3.2 links training gain to the cost of learning new contributions from existing computation. Guided by this insight, we directly learn channel combinations and their shared output directions through the low-rank parameterization of LoRA (Hu et al., 2021). Specifically, $A_l \in \mathbb{R}^{r \times k_l}$ combines the selected activations $P_l z_l$ into r features s_l, and $B_l \in \mathbb{R}^{d_l \times r}$ learns how these features contribute to the hidden space. Adding this contribution gives

$$s_l = A_l P_l z_l \in \mathbb{R}^r, \quad \Delta y_l = B_l s_l, \quad y_l = y_l + \beta \Delta y_l = W_l^{\mathrm{down}} z_l + \beta B_l A_l P_l z_l, \tag{8}$$

where β controls residual strength. The selection matrix P_l restricts the module's inputs to the localized channels, while the learned factors determine their new contributions. The module uses $r(k_l + d_l)$ parameters per layer (Appendix D.2).

With A_l randomly initialized and $B_l = 0$, the residual starts from zero and preserves the SFT model's initial predictions. We then optimize only $\phi = \{A_l, B_l\}_{l=1}^{L}$, where L is the number of layers, keeping θ_{SFT} fixed.

Since each inference path generates one candidate item, we train the new combinations by predicting one complete target SID followed by EOS for each original context. This target is selected once from the deduplicated target list and held fixed. We minimize answer-token cross-entropy at $\beta = 1$, with the objective and target construction specified in Appendix D.3.

5 Experiments

5.1 Experimental Setup

Datasets and models. We evaluate on the Ad, Product, and Video domains of OpenOneRec's RecIF-Bench (Zhou et al., 2025). Table 1 reports training contexts and development/test users; evaluation covers the complete official test cohorts. We use Qwen-1.7B, Qwen-8B, Llama-1B, and Llama-8B to test transfer across model families and scales.

Baselines and evaluation. We compare with SASRec (Kang & McAuley, 2018), HSTU (Zhai et al., 2024), ReaRec (Tang et al., 2025), TIGER (Rajput et al., 2023), GraphLoRA[1], and the mixed-data OpenOneRec baseline. For each backbone, ordinary recommendation SFT is the direct reference for our method. We report user-averaged SID-level Recall and NDCG at cutoffs 8 and 32, abbreviated as R and N.

Table 1: Recommendation dataset sizes.

Domain	Train	Dev	Test
Ad	109,050	1,000	27,677
Product	62,713	1,000	27,910
Video	140,575	1,000	38,781

Adaptation. We localize channels using 1,000 training prompts per backbone and domain, without answers, retaining at most the last 256 tokens. Channels selected among the layer-wise top 10% on at least 80% of prompts form the fixed input support. Following Section 4, recommendation SFT uses the original multi-target answers; residual training uses one fixed complete target SID plus EOS per context. Both stages use recommendation supervision alone.

[1] https://github.com/wgj15965/GraphLoRA

5.2 Main Recommendation Results

Table 2 compares recommendation performance across three domains and four LLM backbones.

Table 2: Main recommendation results. R@k and N@k denote Recall@k and NDCG@k; all scores are multiplied by 100. Bold identifies our method in each group. $\Delta\%$ gives its relative improvement over the corresponding SFT baseline, computed as $100(\mathrm{Ours}/\mathrm{SFT} - 1)$. ($p$-value < 0.05)

	Method	Ad				Product				Video			
		R@8	R@32	N@8	N@32	R@8	R@32	N@8	N@32	R@8	R@32	N@8	N@32
Disc.	SASRec	0.97	3.09	0.66	1.38	0.65	2.07	0.56	1.09	0.24	0.63	0.28	0.45
	HSTU	1.11	3.72	0.83	1.69	0.60	2.16	0.57	1.16	0.33	1.13	0.39	0.75
	ReaRec	1.47	5.09	1.10	2.32	0.75	2.62	0.60	1.27	0.39	1.12	0.46	0.79
Gen.	TIGER	1.86	5.62	1.41	2.64	1.67	4.10	1.51	2.41	0.36	0.72	0.45	0.59
	GraphLoRA	3.14	8.73	2.43	4.30	2.84	6.31	2.71	4.02	1.52	2.65	2.00	2.42
	OpenOneRec	3.19	8.79	2.37	4.23	3.20	6.76	3.04	4.38	2.15	3.76	2.83	3.44
Qwen-1.7B	*SFT*	3.13	8.90	2.38	4.30	3.19	6.73	3.04	4.37	2.15	3.74	2.86	3.46
	Ours	**3.34**	**9.35**	**2.59**	**4.60**	**3.32**	**6.87**	**3.10**	**4.46**	**2.29**	**3.84**	**2.92**	**3.59**
	$\Delta\%$	+6.71	+5.06	+8.82	+6.98	+4.08	+2.08	+1.97	+2.06	+6.51	+2.67	+2.10	+3.76
Qwen-8B	*SFT*	3.43	9.48	2.62	4.67	3.36	7.01	3.17	4.55	2.34	3.84	2.95	3.50
	Ours	**3.60**	**9.80**	**2.79**	**4.87**	**3.47**	**7.23**	**3.37**	**4.64**	**2.45**	**3.91**	**3.02**	**3.64**
	$\Delta\%$	+4.96	+3.38	+6.49	+4.28	+3.27	+3.14	+6.31	+1.98	+4.70	+1.82	+2.37	+4.00
Llama-1B	*SFT*	2.85	8.29	2.16	3.95	2.75	6.02	2.64	3.86	1.47	2.61	1.94	2.38
	Ours	**3.14**	**8.72**	**2.43**	**4.30**	**2.82**	**6.27**	**2.66**	**3.96**	**1.52**	**2.66**	**2.00**	**2.43**
	$\Delta\%$	+10.18	+5.19	+12.50	+8.86	+2.55	+4.15	+0.76	+2.59	+3.40	+1.92	+3.09	+2.10
Llama-8B	*SFT*	3.17	8.81	2.40	4.27	2.84	6.23	2.70	3.98	1.60	2.67	2.04	2.43
	Ours	**3.44**	**9.31**	**2.62**	**4.58**	**2.90**	**6.38**	**2.74**	**4.10**	**1.68**	**2.73**	**2.11**	**2.47**
	$\Delta\%$	+8.52	+5.68	+9.17	+7.26	+2.11	+2.41	+1.48	+3.02	+5.00	+2.25	+3.43	+1.65

LLM backbones provide a strong starting point. LLM-based recommenders consistently outperform SASRec, HSTU, ReaRec, and TIGER across the evaluated domains. This result complements the initialization comparison in Section 2, further supporting the value of pretrained initialization for recommendation.

Localized recombination improves the use of pretrained computation. Our method improves all four metrics for every backbone–domain pair, with relative Recall@32 gains of 1.82%–5.68% over the corresponding SFT baseline. These consistent gains support our central hypothesis that useful pretrained computation can be better exploited when its outputs are explicitly recombined for the downstream task. Notably, this improvement requires only recommendation supervision, without an additional objective for recovering general-task performance.

Backbone suitability matters beyond model size. Recommendation performance varies substantially across model families. For example, Qwen-1.7B outperforms Llama-8B on all four metrics in both Product and Video despite having substantially fewer parameters. Model size alone therefore does not explain recommendation performance. This observation is consistent with the analysis in Section 3, where transfer depends on which useful computations are available at initialization and how effectively they can be adapted to recommendation. At the same time, the consistent gains from our method across all four backbones indicate that these different pretrained starting points all leave room for improved task-specific recombination.

5.3 Effect of Initial Localization

To test whether the gains depend on which channels are recombined, we compare our localized support S_l with a fixed random support S_l^{rnd} of the same layer-wise size, and with down-projection LoRA using all channels. All three adapters start from the same Qwen-1.7B SFT model on Ad and use rank 8 with matched supervision and training budgets.

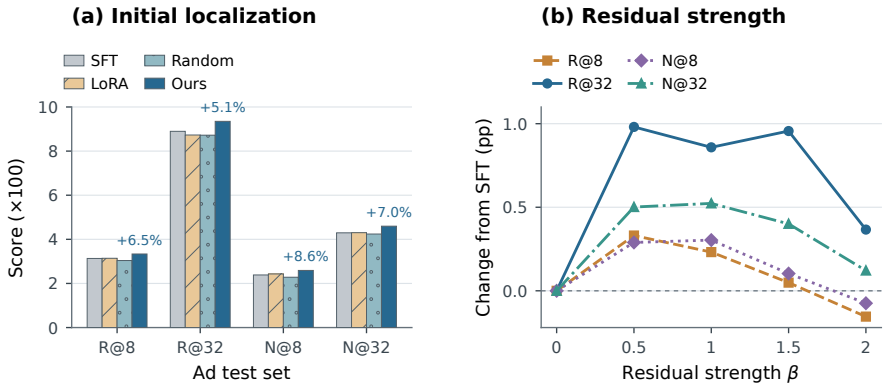

Figure 3: **Channel selection and recombination strength.** (a) Ad test results with Qwen-1.7B. Random and Ours select equally many channels per layer; LoRA uses all channels. Annotations show gains over SFT. (b) Varying β for one rank-8 adapter on 1,000 Ad development users. Changes are in percentage points relative to $\beta = 0$ (SFT).

Only initial localization improves all four metrics over SFT (Figure 3a). Neither random selection with the same parameter count nor unrestricted LoRA matches this improvement. The gain therefore depends on selecting useful inputs for recombination, supporting the hypothesis from Section 3.3 that initial activity identifies resources valuable for subsequent adaptation.

5.4 Effect of Recombination Strength

To assess the effect of learned recombination, we vary only its strength β in a fixed rank-8 Ad adapter, with $\beta = 0$ recovering SFT. All four metrics improve at $\beta \in \{0.5, 1, 1.5\}$, whereas $\beta = 2$ reduces Recall@8 and NDCG@8 below SFT (Figure 3b). With the selected channels fixed, this contrast shows that recommendation quality also depends on the strength of their learned recombination. Together with the localization ablation, these results support both components of our design: identifying useful initial channels and adapting their recombination for recommendation.

5.5 General-Task Performance and Computational Reuse

Section 2 showed that recommendation still depends on task-shared parameters despite deteriorating general-task scores. Does this deterioration imply that the associated functionality has been erased? We evaluate the same Product analysis checkpoints under ordinary and assisted output protocols, without updating their parameters. Assistance provides a fixed answer prefix for mathematics, the problem-provided function signature and docstring for code, or legal-option scoring for knowledge questions. Evaluations cover GSM8K, HumanEval, and the held-out MMLU-Pro subset.

Poor ordinary outputs can conceal retained functionality. After recommendation-only SFT, Qwen+SID scores zero on all three ordinary evaluations, yet assistance raises its code and knowledge scores to 54.27% and 29.85%. OneRec shows the same contrast for these tasks. Recovery is task-dependent: mathematics remains weak for Qwen+SID and declines for OneRec. Thus, ordinary answer failure does not establish complete loss of general-task functionality; some remains accessible without further parameter learning.

Computational reuse differs from general-task recovery. Our findings separate the two in both directions: recovering general-task answers does not ensure additional recommendation gains, while failing to produce them does not imply complete computational erasure. Together with the shared-channel interventions in Section 2, this supports the reuse perspective of Section 3: what matters for recommendation is which pretrained resources remain useful and how they are recombined. Our method targets these resources through initial localization and recommendation-specific recombination.

Table 3: General-task scores (%) under ordinary (Ord.) and assisted (Assist.) output protocols, with identical weights within each pair. None, Rec, and Mix denote no further training, recommendation-only SFT, and mixed-data SFT. Shaded rows isolate recommendation-only adaptation.

Initialization	Training	Math		Code		Knowledge	
		Ord.	Assist.	Ord.	Assist.	Ord.	Assist.
Qwen+SID	None	81.65	81.65	64.02	65.85	40.30	31.95
Qwen+SID	Rec	0.00	7.05	0.00	54.27	0.00	29.85
Qwen+SID	Mix	68.16	71.34	53.05	56.10	45.30	45.10
OneRec	None	75.89	79.83	13.41	61.59	25.55	27.75
OneRec	Rec	19.18	8.26	0.00	50.00	0.00	29.40
OneRec	Mix	72.93	72.33	53.05	58.54	44.50	44.40

6 Related Work

LLMs for Recommendation. LLMs are increasingly adopted for recommendation, leveraging pretrained language understanding, reasoning, and world knowledge to complement collaborative signals. For example, LLaRA integrates textual item features with interaction-based representations (Liao et al., 2023). However, recommendation adaptation can degrade general-domain abilities, motivating efforts to preserve them during specialization. LEARN freezes the LLM while learning recommendation-specific alignment (Jia et al., 2025), EAGER-LLM balances recommendation and textual reasoning through an annealing adapter (Hong et al., 2025), and OpenOneRec mixes general-domain data with recommendation training and further applies on-policy distillation (Zhou et al., 2025). RecGPT instead uses reasoning-enhanced pre-alignment and self-training for adaptation (Yi et al., 2025). These works show how recommendation and general capabilities can be jointly maintained, but do not establish whether preserving or recovering general-domain performance itself improves recommendation.

Understanding LLMs for Recommendation. Recent work investigates how LLMs process interaction histories and exploit pretrained knowledge. ReLLa studies difficulties with long behavior histories (Lin et al., 2024), while Lost in Sequence shows that sequential order is often underused (Kim et al., 2025). Kim et al. (2026) further find that gains concentrate on targets covered by training-time one-hop transitions, highlighting memorization. Other studies focus on pretrained knowledge and reasoning: OneRec-Think introduces item–text alignment and explicit preference reasoning (Liu et al., 2025), OneReason analyzes when reasoning helps recommendation (OneRec Team et al., 2026), and PauseRec identifies knowledge-verbalization and text–SID alignment issues, showing that answer information can remain recoverable from logits despite degraded textual outputs (He et al., 2026). These analyses clarify how LLM recommenders use sequence information and pretrained knowledge, but leave open what internal resources pretrained initialization contributes and whether their utility depends on continued general-capability expression. We address this gap by tracing task-related neurons through adaptation, revealing a pretrained neuronal foundation that is reused and reorganized and remains functionally important even as general-task performance declines.

7 Conclusion

We revisited what LLM initialization contributes to recommendation. Recovering general-task performance brings no clear additional recommendation gains, yet shared neurons remain functionally important. Our analysis highlights useful initial computation and the cost of learning its recombination, supported by evidence of early localization and subsequent reuse. Guided by these findings, initial localization-guided neuron recombination consistently improves recommendation across three domains and four LLM backbones. These results support identifying and recombining the pretrained computation that recommendation actually uses.

AI USE STATEMENT

Generative AI tools, including OpenAI Codex, were used to assist with English drafting and editing, interpretation of author-provided experimental results, implementation checks, literature search and verification, schematic layout, and figure/table scripting. All AI-assisted analyses, code, citations, and manuscript content were reviewed and verified by the authors. Empirical results reported in this paper were obtained from the authors' experiment records rather than generated by AI. The authors take responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

We provide detailed information needed to reproduce the experiments in the appendix. Appendix A specifies model initialization, data construction, training and optimization settings, evaluation protocols, and uncertainty estimation. Appendix B describes the neuron-localization procedure, task-set construction, mask construction, matched controls, and full generation interventions. Appendices E, F, G report the complete behavioral, structural, and intervention results. Anonymous source code is available at https://anonymous.4open.science/r/Neuron_Recombination-3F51.

REFERENCES

Shahar Azulay, Edward Moroshko, Mor Shpigel Nacson, Blake E Woodworth, Nathan Srebro, Amir Globerson, and Daniel Soudry. On the implicit bias of initialization shape: Beyond infinitesimal mirror descent. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 468–477. PMLR, 2021. URL https://proceedings.mlr.press/v139/azulay21a.html.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*. ACM, 2023. doi: 10.1145/3604915.3608857.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code. arXiv preprint arXiv:2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Yuxin Chen, Yiran Zhao, Yang Zhang, An Zhang, Kenji Kawaguchi, Shafiq Joty, Junnan Li, Tat-Seng Chua, Michael Shieh, and Wenxuan Zhang. The emergence of abstract thought in large language models beyond any language. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 9902–9933, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0e4b12a79106789483fe6746702f4cb0-Paper-Conference.pdf.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. arXiv preprint arXiv:2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In

Proceedings of the 16th ACM Conference on Recommender Systems, pp. 299–315. ACM, 2022. doi: 10.1145/3523227.3546767.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1832–1841. PMLR, 2018. URL https://proceedings.mlr.press/v80/gunasekar18a.html.

Yinhan He, Liam Collins, Bhuvesh Kumar, Jundong Li, Neil Shah, and Donald Loveland. Implicit Reasoning for Large Language Model-based Generative Recommendation, 2026. URL https://arxiv.org/abs/2606.14142.

Minjie Hong, Yan Xia, Zehan Wang, Jieming Zhu, Ye Wang, Sihang Cai, Xiaoda Yang, Quanyu Dai, Zhenhua Dong, Zhimeng Zhang, et al. EAGER-LLM: Enhancing large language models as recommenders through exogenous behavior-semantic integration. In *Proceedings of the ACM on Web Conference 2025*, pp. 2754–2762, 2025. URL https://doi.org/10.1145/3696410.3714933.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. arXiv:2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

Jian Jia, Yipei Wang, Yan Li, Honggang Chen, Xuehan Bai, Zhaocheng Liu, Jian Liang, Quan Chen, Han Li, Peng Jiang, et al. Learn: knowledge adaptation from large language model to recommendation for practical industrial application. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 11861–11869, 2025. URL https://doi.org/10.1609/aaai.v39i11.33291.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. arXiv:1808.09781, 2018. URL https://arxiv.org/abs/1808.09781.

Sein Kim, Hongseok Kang, Kibum Kim, Jiwan Kim, Donghyun Kim, Minchul Yang, Kwangjin Oh, Julian McAuley, and Chanyoung Park. Lost in sequence: Do large language models understand sequential recommendation? In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 1160–1171, 2025. URL https://doi.org/10.1145/3711896.3737035.

Sunwoo Kim, Sunkyung Lee, Clark Mingxuan Ju, Donald Loveland, Bhuvesh Kumar, Kijung Shin, Neil Shah, and Liam Collins. On the Memorization Behavior of LLMs in Generative Recommendation: Observations, Implications, and Training Strategies, 2026. URL https://arxiv.org/abs/2606.17276v1.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. LLaRA: Large Language-Recommendation Assistant. arXiv preprint arXiv:2312.02445, 2023. URL https://arxiv.org/abs/2312.02445.

Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. ReLLa: Retrieval-enhanced Large Language Models for Lifelong Sequential Behavior Comprehension in Recommendation. In *Proceedings of the ACM Web Conference 2024*, pp. 3497–3508, 2024. URL https://arxiv.org/abs/2308.11131.

Samuel Lippl and Jack Lindsey. Inductive biases of multi-task learning and finetuning: multiple regimes of feature reuse. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/d7346ff79699b5bba26f8af89e700a95-Abstract-Conference.html.

Zhanyu Liu, Shiyao Wang, Xingmei Wang, Rongzhou Zhang, Jiaxin Deng, Honghui Bao, Jinghao Zhang, Wuchao Li, Pengfei Zheng, Xiangyu Wu, Yifei Hu, Qigen Hu, Xinchen Luo, Lejian Ren, Zixing Zhang, Qianqian Wang, Kuo Cai, Yunfan Wu, Hongtao Cheng, Zexuan Cheng, Lu Ren, Huanjie Wang, Yi Su, Ruiming Tang, Kun Gai, and Guorui Zhou. OneRec-Think: In-Text Reasoning for Generative Recommendation, 2025. URL https://arxiv.org/abs/2510.11639.

OneRec Team, Biao Yang, Boyang Ding, Chenglong Chu, Dunju Zang, Fei Pan, Han Li, Hao Jiang, Honghui Bao, Huanjie Wang, Jian Liang, Jiangxia Cao, Jiao Ou, Jiaxin Deng, Jinghao Zhang, Kun Gai, Lu Ren, Peiru Du, Pengfei Zheng, Rongzhou Zhang, Ruiming Tang, Shiyao Wang, Siyang Mao, Siyuan Lou, Teng Shi, Wei Yuan, Wenlong Xu, Xingchen Liu, Xingmei Wang, Xinqi Jin, Yan Sun, Yan Wang, Yifei Hu, Yingzhi He, Yufei Ye, Yuhao Wang, Yunhao Zhou, Yuqin Dai, Zhao Liu, Zhipeng Wei, Zhixin Ling, Ziming Li, Zixing Zhang, Ziyuan Liu, An Zhang, Changxin Lao, Chaoyi Ma, Chengru Song, Defu Lian, Fan Yang, Guowang Zhang, Hao Peng, Jiayao Shen, Jie Chen, Jun Xu, Junmin Chen, Kun Zhang, Kuo Cai, Mingxing Wen, Minmao Wang, Minxuan Lv, Qi Zhang, Qiang Luo, Sheng Yu, Shijie Li, Shijie Yi, Shuang Yang, Shugui Liu, Shuni Chen, Tinghai Zhang, Tingting Gao, Xiang Wang, Xiangyu Wu, Xiangyu Zhao, Xiao Lv, Xiaoyou Zhou, Xuming Wang, Yong Du, Zejian Zhang, Zhaojie Liu, Zhiyang Zhang, Zhuang Zhuang, Ziqi Wang, and Ziyi Zhao. OneReason Technical Report, 2026. URL https://arxiv.org/abs/2606.06260.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. arXiv:2305.05065, 2023. URL https://arxiv.org/abs/2305.05065.

Jiakai Tang, Sunhao Dai, Teng Shi, Jun Xu, Xu Chen, Wen Chen, Jian Wu, and Yuning Jiang. Think before recommend: Unleashing the latent reasoning power for sequential recommendation. arXiv:2503.22675, 2025. URL https://arxiv.org/abs/2503.22675.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract.html.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 806–815. ACM, 2024. doi: 10.1145/3616855.3635853.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. arXiv preprint arXiv:2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

Chao Yi, Dian Chen, Gaoyang Guo, Jiakai Tang, Jian Wu, Jing Yu, Mao Zhang, Sunhao Dai, Wen Chen, Wenjun Yang, Yuning Jiang, Zhujin Gao, Bo Zheng, Chi Li, Dimin Wang, Dixuan Wang, Fan Li, Fan Zhang, Haibin Chen, Haozhuang Liu, Jialin Zhu, Jiamang Wang, Jiawei Wu, Jin Cui, Ju Huang, Kai Zhang, Kan Liu, Lang Tian, Liang Rao, Longbin Li, Lulu Zhao, Na He, Peiyang Wang, Qiqi Huang, Tao Luo, Wenbo Su, Xiaoxiao He, Xin Tong, Xu Chen, Xunke Xi, Yang Li, Yaxuan Wu, Yeqiu Yang, Yi Hu, Yinnan Song, Yuchen Li, Yujie Luo, Yujin Yuan, Yuliang Yan, Zhengyang Wang, Zhibo Xiao, Zhixin Ma, Zile Zhou, and Ziqi Zhang. RecGPT Technical Report, 2025. URL https://arxiv.org/abs/2507.22879.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, Yinghai Lu, and Yu Shi. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. arXiv:2402.17152, 2024. URL https://arxiv.org/abs/2402.17152.

Guorui Zhou, Honghui Bao, Jiaming Huang, Jiaxin Deng, Jinghao Zhang, Junda She, Kuo Cai, Lejian Ren, Lu Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rongzhou Zhang, Ruiming Tang, Shiyao Wang, Wuchao Li, Xiangyu Wu, Xinchen Luo, Xingmei Wang, Yifei Hu, Yunfan Wu, Zhanyu Liu, Zhiyang Zhang, Zixing Zhang, Bo Chen, Bin Wen, Chaoyi Ma, Chengru Song, Chenglong Chu, Defu Lian, Fan Yang, Feng Jiang, Hongtao Cheng, Huanjie Wang, Kun Gai, Pengfei Zheng, Qiang Wang, Rui Huang, Siyang Mao, Tingting Gao, Wei Yuan, Yan Wang, Yang Zhou, Yi Su, Zexuan Cheng, Zhixin Ling, and Ziming Li. OpenOneRec Technical Report. arXiv preprint arXiv:2512.24762, 2025. URL https://arxiv.org/abs/2512.24762.

Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. Collaborative large language model for recommender systems. In *Proceedings of the ACM Web Conference 2024*. ACM, 2024. doi: 10.1145/3589334.3645347.

A Experimental Settings for Analysis

This study uses the Product Rec task from RecIF-Bench and the OneRec semantic-ID interface (Zhou et al., 2025). General-task evaluations cover GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), and the custom MMLU-Pro split described below (Wang et al., 2024). The primary comparisons use one epoch under a common optimization recipe; they measure the benefit of initialization within this budget rather than the best attainable performance of a separately tuned random model.

A.1 Initialization and shared SID interface

The pretrained language-model condition uses Qwen3-1.7B, not Qwen3-1.7B-Base. We copy its backbone and ordinary text embeddings into the OneRec architecture. Random+SID uses the same architecture with the framework's default random initialization.

Both conditions use the same learned SID interface from OneRec, including 24,578 SID rows. The SID extension consists of three codebooks with 8,192 entries each, plus beginning and ending markers. Input and output embeddings are tied. The shared SID embeddings account for about 2.9% of the model parameters and remain trainable during SFT. Consequently, this experiment measures backbone initialization with a common learned recommendation interface. It does not measure the performance of an entirely untrained recommendation system.

A.2 Training data and optimization

Training data. Mathematics and code examples come from the task-filtered Tulu pools used in the experiment. Knowledge examples use a custom MMLU-Pro split. Normalized question stems and near-duplicate grouping keep repeated questions together, retaining one canonical example per group. Category-stratified allocation produces 6,758 training, 1,000 localization, and 2,000 evaluation questions.

Knowledge supervision uses the fixed answer format `The answer is (X).` and does not include a reference reasoning trace. This evaluation uses the custom split described above, rather than the complete MMLU-Pro benchmark. Table 4 summarizes the resulting primary SFT recipes and realized training exposure.

Table 4: Primary SFT recipes. Token counts include processed non-padding input and answer tokens. The actual recommendation exposure differs slightly because incomplete distributed batches are dropped.

	Rec	**Mix**
Product training pool	62,713	62,713
Mathematics / code / knowledge pools	0 / 0 / 0	8,959 / 8,959 / 6,758
Optimizer steps	979	1,365
Processed examples, all tasks	62,656	87,360
Processed Product examples	62,656	62,692
Processed tokens, all tasks	55,435,841	69,800,055

Optimization configuration. Table 5 summarizes the shared training and optimization settings for the primary SFT experiments, together with the sampling composition of Mix. The objective is token-level cross-entropy over supervised answer tokens. Therefore, the sample ratio does not directly determine a task's loss weight or gradient contribution. Relative to Rec, Mix processes approximately 39.4% more optimizer steps and 25.9% more tokens; the corresponding absolute counts are reported in Table 4.

A.3 Recommendation evaluation and metrics

We evaluate recommendation on a fixed subset of 5,000 users from the official Product test set. All model comparisons use the same users, targets, and candidate ordering. This is a constrained

Table 5: Training parameters and optimization settings used in the primary SFT experiments.

Category	Parameter	Setting
Sampling	Mix sample ratio	71.76% recommendation / 28.24% general tasks
Training	Trainable parameters	Full-parameter training
Training	Global batch size	64
Training	Maximum sequence length	4,096
Optimization	Optimizer	AdamW
Optimization	Weight decay	0.01
Optimization	Peak learning rate	2×10^{-5}
Optimization	Learning-rate schedule	3% linear warmup, followed by cosine decay
Optimization	Gradient clipping	Norm 1.0

beam-search ranking, rather than exhaustive exact ranking of the entire catalog. Targets and predicted candidates are deduplicated as complete SIDs; multiple product IDs can share an SID, so the reported quantities are SID-level metrics.

For user u, let T_u be the set of target SIDs, let L_u be the deduplicated candidate list truncated to 128 entries, and let $H_u = \{j : L_u[j] \in T_u\}$ denote the hit ranks, using one-based indexing. We compute

$$\mathrm{MRR}_u@128 = \begin{cases} 1/\min H_u, & H_u \neq \varnothing, \\ 0, & H_u = \varnothing, \end{cases} \tag{9}$$

$$\mathrm{Hit}_u@128 = \mathbf{1}[H_u \neq \varnothing], \qquad \mathrm{Recall}_u@128 = \frac{|H_u|}{|T_u|}, \tag{10}$$

$$\mathrm{NDCG}_u@128 = \frac{\sum_{j \in H_u} 1/\log_2(j+1)}{\sum_{j=1}^{\min(|T_u|,128)} 1/\log_2(j+1)}. \tag{11}$$

Recall and NDCG are zero for an empty target set. We macro-average over users and multiply by 100 for reporting. The scoring routine operates on the complete generated SID candidates, not teacher-forced next-token ranks.

A.4 General-task evaluation and uncertainty

General-task evaluation uses the full GSM8K and HumanEval test sets and the held-out knowledge split defined in Section A.2. HumanEval is evaluated with execution-based pass@1. Ordinary generation is greedy with at most 1,024 new tokens. Knowledge scoring uses the three-stage answer-extraction procedure, with extraction failures counted as incorrect. Assisted evaluations either supply a fixed answer prefix (mathematics), the problem's function signature and docstring (code), or choose the highest-scoring legal answer letter (knowledge).

Paired bootstrap intervals resample users or problems 2,000 times, preserving pairing across conditions. For a targeted intervention, the three random-mask scores are first averaged within each user; bootstrap then resamples the difference between this mean and the targeted score. These intervals condition on the trained models and the three sampled masks. They exclude training-seed variation and uncertainty across all possible random masks, and are reported per comparison without multiple-comparison correction. A zero-containing interval does not establish equivalence between training recipes.

B Neuron Localization and Intervention Details

Let R_t, M_t, C_t, and K_t denote the independently localized recommendation, mathematics, code, and knowledge sets. We use $t = 0$ for initialization and $t = 1$ for the trained endpoint, and define the general-task union as $G_t = M_t \cup C_t \cup K_t$ while shared positions belong to $R_t \cap G_t$. Set differences describe selection under the detector, rather than exclusive functional modules.

B.1 Coordinates and local output-change scores

The localization principle follows the effect of removing an individual coordinate on a module's output (Chen et al., 2025). Our implementation uses Qwen3's gated feed-forward network, Q/K normalization, rotary positional embeddings, grouped-query attention, and output projection.

We analyze six projection families: query, key, value, attention output, feed-forward up, and feed-forward down. With the linear maps wriiten as $y = xW^\top$, q/k/v/up coordinates index weight rows, while o/down coordinates index columns. We do not localize gate coordinates separately. Given the original input to a module, we measure

$$s_{\ell,p,j}(x) = \left\| F_\ell(x) - F_\ell^{(-p,j)}(x) \right\|_F , \tag{12}$$

where p is a projection family, j is the candidate coordinate, and $F_\ell^{(-p,j)}$ is the local attention or feed-forward output with that coordinate removed. The norm covers valid tokens and output dimensions. The score is computed locally with the same incoming activations; it is not a final-loss gradient or a complete model rerun for each coordinate.

For a feed-forward channel, let $z_j(x)$ be its actual gated activation entering the down projection. Its score is exactly

$$s_{\ell,\mathrm{FFN},j}(x) = \|z_j(x)\|_2 \, \|W_{\mathrm{down}}[:, j]\|_2 . \tag{13}$$

The up and down entries receive identical scores because zeroing the corresponding up row or down column removes the same channel contribution. Structural counts merge these two entries, while projection-coordinate counts retain both.

For q/k coordinates, removal is applied before Q/K normalization, after which normalization, rotary position encoding, attention probabilities, and the local output are recomputed. Shared key/value coordinates are evaluated jointly across their grouped-query contributions.

B.2 Prompt-level ranking and task sets

Each task has 1,000 fixed localization prompts, shared across model states. The detector receives at most the final 256 prompt tokens; reference and generated answers are excluded. For each prompt, layer, and projection, we rank coordinates by equation 12, selecting $\max(1, \lfloor 0.1 d_{\ell,p} \rfloor)$. The selection frequency is

$$f_{D,t}(\ell, p, j) = \frac{1}{1000} \sum_{x \in \mathcal{D}_D} \mathbf{1}[j \in \mathrm{Top}_{10\%}(s_{\ell,p}(x; \theta_t))] . \tag{14}$$

A coordinate enters the task set when $f_{D,t} \geq 0.8$. We obtain R_t, M_t, C_t, and K_t independently. Falling below this threshold means only that a coordinate is not consistently selected across prompts; it may still be active or useful on individual examples. All subsequent intersections, unions, and set differences are formed from these thresholded task sets. The analyzed universe contains 516,096 projection coordinates, corresponding to 344,064 positions after merging the paired feed-forward entries.

B.3 Origin, retention, and frequency summaries

For every endpoint, the following four sets partition R_1:

$$R_1 \cap R_0 \cap G_0, \quad R_1 \cap (R_0 \setminus G_0),$$
$$R_1 \cap (G_0 \setminus R_0), \quad R_1 \setminus (R_0 \cup G_0).$$

Their fractions use $|R_1|$ as denominator. Initial recommendation retention instead uses $|R_1 \cap R_0|/|R_0|$. Shared recommendation fractions use $|R_t \cap G_t|/|R_t|$. Coordinates are compared only within an initialization lineage; identical indices across independent initializations are not assigned equivalent semantic functions.

To provide a size-aware reference for $|R_1 \cap G_0|$, we compute the expected overlap $E = \sum_{\ell,p} |R_{1,\ell,p}| \, |G_{0,\ell,p}|/d_{\ell,p}$ under independent within-stratum sampling, merging paired feed-forward coordinates. Enrichment is the observed overlap divided by E. For continuous-frequency comparisons, we subtract each projection's mean frequency before computing the cosine similarity of the concatenated frequency vectors. These measures summarize coordinate selection, not hidden-representation content.

B.4 MASKS, CONTROLS, AND FULL-GENERATION INTERVENTION

Target masks. The fixed-initial-shared mask selects $\lfloor |(R_0 \cap G_0)_{\ell,p}|/4 \rfloor$ coordinates from $(R_0 \cap G_0)_{\ell,p}$ for each layer/projection stratum (ℓ, p). The selected subset is sampled once without replacement and then held fixed across the Rec and Mix endpoints of the same initialization. The fixed-initial-general-difference mask applies the same quarter rule to $G_0 \setminus R_0$. Current-shared masks apply the quarter rule to each endpoint's $R_1 \cap G_1$.

For current recommendation-difference masks, we use a common per-stratum budget across the four trained Qwen/Random endpoints:

$$m^{\text{diff}}_{\ell,p} = \left\lfloor \frac{1}{4} \min_{e \in \mathcal{E}} \left| (R_1^{(e)} \setminus G_1^{(e)})_{\ell,p} \right| \right\rfloor, \tag{15}$$

where $\mathcal{E}$ denotes the four Rec/Mix endpoints under Qwen and Random initialization. Each endpoint then selects $m^{\text{diff}}_{\ell,p}$ coordinates from its own $(R_1 \setminus G_1)_{\ell,p}$ set. For feed-forward masks, up- and down-projection entries are paired by channel.

Matched random controls. For each targeted mask, we construct independently sampled random-control masks. Each control matches the targeted mask's number of selected coordinates in every layer/projection stratum. Controls for initial masks are sampled without replacement from coordinates outside $R_0 \cup G_0$, whereas controls for current masks are sampled outside $R_1 \cup G_1$ of the corresponding endpoint. The controls therefore match location strata and ablation counts, but not local importance magnitudes. Because endpoint-specific shared-quarter masks can differ in size across initializations, we do not use their raw intervention magnitudes to compare relative dependence between Qwen and Random backbones.

Full-generation intervention. During intervention, the selected weight rows or columns are zeroed throughout recommendation inference, including the full prompt and subsequent SID generation. This differs from localization, which uses the truncated prompt described in Appendix B.2. Let $s(\theta \setminus S)$ denote the recommendation score after ablating target mask S. Random-adjusted damage is

$$D(S) = \frac{1}{B} \sum_{b=1}^{B} s\big(\theta \setminus S_b^{\text{ctrl}}\big) - s(\theta \setminus S)\,, \tag{16}$$

where S_b^{ctrl} is the b-th matched random-control mask. Positive damage indicates that targeted removal is more harmful to recommendation than the matched random controls. Because the controls match location strata and counts but not local importance magnitude, this comparison supports functional involvement of the selected positions in recommendation, but does not establish recommendation-specific semantics or functional exclusivity.

C CROSS-ENTROPY TRAINING IN A TWO-FACTOR CHANNEL MODEL

C.1 THE RESTRICTED FFN AND THE ANALYTICAL CE OBJECTIVE

Let $X = (x_i, s_{i,<t})$ be a teacher-forced recommendation answer-token context and $Y = s_{i,t} \in \{1, \ldots, V\}$ its target. Expectations are over a fixed distribution of these pairs; for equation 1, this is the uniform empirical distribution over its N supervised answer tokens. The hidden dimension d, channel count m, and vocabulary size $V \geq 2$ are finite positive integers. We restrict the FFN in equation 2 at a chosen layer l. Its input map $h_l(X) \in \mathbb{R}^d$, gate rows $\alpha_j^\top$, and unit-norm directions $p_j, e_j \in \mathbb{R}^d$ are fixed. Only the scalar strengths of the input-projection rows and output-projection columns are trained:

$$\begin{aligned} (W_l^{\text{up}})_{j,:} &= u_j p_j^\top, \qquad & w_{lj} &= v_j e_j, \\ z_{lj}(h_l(X)) &= u_j \xi_j(X), \quad & \xi_j(X) &= \text{SiLU}(\alpha_j^\top h_l(X))\, p_j^\top h_l(X). \end{aligned} \tag{17}$$

Thus u_j changes the strength of the scalar response and v_j changes how strongly that response contributes along e_j. The unit directions specify the training parameterization: Euclidean updates of u_j, v_j correspond to Euclidean updates along these permitted weight directions.

For the analytical downstream model, use a fixed linear readout $R \in \mathbb{R}^{V\times d}$ and a fixed logit offset $a_{\text{off}}(X)$. Define $\Phi_j(X) = R[e_j \xi_j(X)]$ and the matrix $\Phi(X) = [\Phi_1(X), \dots, \Phi_m(X)] \in \mathbb{R}^{V\times m}$. The effective coefficients and the answer-token CE are

$$\begin{aligned} \beta_j &= u_j v_j, & a_\beta(X) &= a_{\text{off}}(X) + \Phi(X)\beta, \\ \mathcal{L}(\beta) &= \mathbb{E}\,\ell(a_\beta(X), Y), & \ell(a, Y) &= -a_Y + \log \sum_{s=1}^{V} e^{a_s}. \end{aligned} \tag{18}$$

The affine logit map is an analytical assumption; fixing the weights of a nonlinear Transformer downstream would not itself imply this map. The input and offset are measurable and finite almost surely, and all fixed weights are finite. We assume $\mathbb{E}\|\Phi(X)\|_F^2 < \infty$, $\mathcal{L}(\beta_0) < \infty$, and finite initial factors u_0, v_0 with $\beta_0 = u_0 \odot v_0$ and $c_j = |u_{0,j}^2 - v_{0,j}^2| > 0$ for every j. All vector norms below are Euclidean; $\|\cdot\|_F$ denotes the Frobenius norm.

Convexity and differentiability. Let $p_\beta(X) = \mathrm{softmax}(a_\beta(X))$ and let $\mathbf{e}_Y \in \mathbb{R}^V$ be the one-hot label. The pointwise logit gradient is $p_\beta - \mathbf{e}_Y$, whose norm is at most $\sqrt{2}$. Consequently,

$$|\mathcal{L}(\beta) - \mathcal{L}(\beta_0)| \le \sqrt{2}\,\mathbb{E}\|\Phi\|_F \|\beta - \beta_0\|,$$

so the risk is finite at every finite β. Differentiation under the expectation gives

$$\begin{aligned} \nabla\mathcal{L}(\beta) &= \mathbb{E}[\Phi^\top (p_\beta - \mathbf{e}_Y)], \\ \nabla^2\mathcal{L}(\beta) &= \mathbb{E}[\Phi^\top H(p_\beta)\Phi], \quad H(p) = \mathrm{diag}(p) - pp^\top. \end{aligned} \tag{19}$$

Since $H(p)$ is a covariance matrix with $0 \preceq H(p) \preceq I$, the gradient and Hessian integrands are therefore bounded by integrable multiples of $\|\Phi\|_F$ and $\|\Phi\|_F^2$, respectively. Dominated convergence therefore makes $\mathcal{L}$ twice continuously differentiable, convex, and globally gradient-Lipschitz, with Lipschitz constant at most $\mathbb{E}\|\Phi\|_F^2$. Convexity is in β, not generally in the two factors.

C.2 From the factor dynamics to a finite-time CE bound

Write $g_j(t) = \partial\mathcal{L}/\partial\beta_j$ at continuous training time t. Unit-rate gradient flow in the original factors gives

$$\dot{u}_j = -v_j g_j, \quad \dot{v}_j = -u_j g_j, \quad \frac{d}{dt}(u_j^2 - v_j^2) = 2u_j\dot{u}_j - 2v_j\dot{v}_j = 0. \tag{20}$$

Hence the signed imbalance is conserved. Using $(u_j^2+v_j^2)^2 = (u_j^2-v_j^2)^2 + 4(u_j v_j)^2$ and $u_j^2+v_j^2 > 0$ yields

$$\dot{\beta}_j = -q_j(\beta_j) g_j, \quad q_j(s) = \sqrt{c_j^2 + 4s^2}, \quad q_j(\beta_j) = u_j^2 + v_j^2, \quad \dot{\mathcal{L}} = -\sum_j q_j g_j^2 \le 0. \tag{21}$$

Existence of the flow. The vector field in (u, v) is locally Lipschitz by equation 19. Moreover,

$$|g_j| \le B_j := \sqrt{2}\,\mathbb{E}\|\Phi_j(X)\| < \infty.$$

For $s_j(t) = u_j(t)^2 + v_j(t)^2$,

$$|\dot{s}_j| = 4|u_j v_j g_j| \le 2B_j s_j, \quad s_j(t) \le s_j(0) e^{2B_j t}.$$

The factors cannot diverge in finite time, giving a unique solution for every $t \ge 0$.

Induced mirror geometry. Following the optimization-geometry perspective on initialization (Gunasekar et al., 2018; Azulay et al., 2021), define

$$\begin{aligned} \psi_c(s) &= \frac{s}{2}\,\mathrm{asinh}\left(\frac{2s}{c}\right) - \frac{1}{4}\sqrt{c^2 + 4s^2} + \frac{c}{4}, \\ \Psi(\beta) &= \sum_j \psi_{c_j}(\beta_j), \\ \mathcal{C}_0(\bar{\beta}, \beta_0) &:= D_\Psi(\bar{\beta}, \beta_0) \\ &= \Psi(\bar{\beta}) - \Psi(\beta_0) - \nabla\Psi(\beta_0)^\top(\bar{\beta} - \beta_0). \end{aligned} \tag{22}$$

Here D_Ψ is a Bregman divergence; the same expression defines $D_\Psi(x, y)$ for any finite vectors x, y. Direct differentiation gives $\psi_c'(s) = \frac{1}{2}\operatorname{asinh}(2s/c)$ and $\psi_c''(s) = 1/\sqrt{c^2 + 4s^2} > 0$. Thus Ψ is smooth and strictly convex, making $\mathcal{C}_0$ finite and nonnegative. Its Hessian is the inverse update scale, so equation 21 becomes

$$\frac{d}{dt}\nabla\Psi(\beta(t)) = \nabla^2\Psi(\beta(t))\dot{\beta}(t) = -\nabla\mathcal{L}(\beta(t)). \tag{23}$$

Proposition 1 (Finite-time CE training)**.** *Under the assumptions above, every finite comparator $\bar{\beta} \in \mathbb{R}^m$ and every $T > 0$ satisfy*

$$\mathcal{L}(\beta(T)) - \mathcal{L}(\bar{\beta}) \le \frac{\mathcal{C}_0(\bar{\beta}, \beta_0)}{T}. \tag{24}$$

The comparator need not minimize the loss, and the CE objective need not have a finite minimizer.

Proof. Consider the nonnegative function $\mathcal{D}(t) = D_\Psi(\bar{\beta}, \beta(t))$. Write $g(t) = \nabla\mathcal{L}(\beta(t))$. Differentiating with respect to the second argument and using equation 23 gives

$$\dot{\mathcal{D}}(t) = -\dot{\beta}(t)^\top\nabla^2\Psi(\beta(t))(\bar{\beta} - \beta(t)) = g(t)^\top(\bar{\beta} - \beta(t)) \le \mathcal{L}(\bar{\beta}) - \mathcal{L}(\beta(t)),$$

where the inequality follows from convexity of $\mathcal{L}$. Integrating this inequality and retaining the final nonnegative divergence gives

$$\int_0^T [\mathcal{L}(\beta(t)) - \mathcal{L}(\bar{\beta})]\, dt \le \mathcal{C}_0(\bar{\beta}, \beta_0) - \mathcal{D}(T) \le \mathcal{C}_0(\bar{\beta}, \beta_0).$$

The loss is nonincreasing by equation 21. Therefore its final gap is no larger than the time-averaged gap, proving the claim. □

C.3 Deriving the local recombination cost

Let $\delta = \bar{\beta} - \beta_0 \in \mathbb{R}^m$ denote the requested change in effective contributions. Its coordinate δ_j is the change assigned to channel j, and $\|\delta\|$ is their Euclidean magnitude. Apply the one-dimensional Taylor formula with integral remainder to $f(s) = \Psi(\beta_0 + s\delta)$, for $0 \le s \le 1$. Subtracting $f(0) + f'(0)$ gives the exact Bregman integral

$$\begin{aligned}\mathcal{C}_0(\beta_0 + \delta, \beta_0) &= \int_0^1 (1 - s)\, \delta^\top\nabla^2\Psi(\beta_0 + s\delta)\delta\, ds \\ &= \int_0^1 (1 - s) \sum_j \frac{\delta_j^2}{\sqrt{c_j^2 + 4(\beta_{0,j} + s\delta_j)^2}}\, ds.\end{aligned} \tag{25}$$

The initial curvature in coordinate j is $1/q_{0,j}$, where

$$q_{0,j} := q_j(\beta_{0,j}) = \sqrt{c_j^2 + 4\beta_{0,j}^2} = u_{0,j}^2 + v_{0,j}^2.$$

Replacing the Hessian in equation 25 by its initial value, and using $\int_0^1 (1 - s)\, ds = 1/2$, yields

$$\mathcal{C}_0(\beta_0 + \delta, \beta_0) = \frac{1}{2}\sum_j \frac{\delta_j^2}{q_{0,j}} + \mathcal{R}(\delta) = \frac{1}{2}\sum_j \frac{\delta_j^2}{q_{0,j}} + O(\|\delta\|^3). \tag{26}$$

To bound the omitted term explicitly, differentiate once more: $\psi_c'''(s) = -4s/(c^2 + 4s^2)^{3/2}$. Writing $x = 2|s|/c$ shows $|\psi_c'''(s)| = 2x/[c^2(1 + x^2)^{3/2}] \le 1/c^2$. The mean-value theorem bounds the change in each diagonal Hessian entry by $s|\delta_j|/c_j^2$ along the segment. Consequently, with $c_{\min} = \min_j c_j > 0$,

$$|\mathcal{R}(\delta)| \le \frac{1}{6}\sum_j \frac{|\delta_j|^3}{c_j^2} \le \frac{\|\delta\|^3}{6c_{\min}^2}, \qquad \int_0^1 s(1 - s)\, ds = \frac{1}{6}. \tag{27}$$

This establishes the stated remainder as $\|\delta\| \to 0$. The condition $c_j > 0$ keeps the potential's derivatives nonsingular even if an effective coefficient crosses zero.

The quadratic term makes the interpretation precise: for the same small change δ_j and the fixed parameterization above, a larger $q_{0,j}$ lowers its leading recombination cost. This scale does not determine whether the change serves recommendation; that depends on the target loss. When pretraining and recommendation training follow the same two-factor flow without a reset, the signed imbalances remain conserved across both phases, while β_0 and the initial update scales can change.

C.4 Supported gains and recombination cost

After analyzing unrestricted training, consider a coordinate set $S \subseteq \{1, \dots, m\}$ and comparators whose change is supported there. Let $\mathcal{G}_T = \mathcal{L}(\beta_0) - \mathcal{L}(\beta(T))$. Rearranging Proposition 1 for every such comparator gives

$$\mathcal{G}_T \geq \sup_{\mathrm{supp}(\delta) \subseteq S} \left[\underbrace{\mathcal{L}(\beta_0) - \mathcal{L}(\beta_0 + \delta)}_{\text{available improvement}} - \underbrace{\frac{\mathcal{C}_0(\beta_0 + \delta, \beta_0)}{T}}_{\text{recombination penalty}} \right]. \tag{28}$$

The support is $\mathrm{supp}(\delta) = \{j : \delta_j \neq 0\}$. It restricts the comparator, not the trained parameters: all factors still follow equation 20. For a comparator with improvement $A = \mathcal{L}(\beta_0) - \mathcal{L}(\beta_0 + \delta) > 0$, the bound certifies positive gain when $T > \mathcal{C}_0(\beta_0 + \delta, \beta_0)/A$. Comparing these sufficient times does not order the actual loss trajectories.

C.5 Extension: reducing recombination cost with an additional path

To model an additional recombination path suppported on S, add residual coefficients d_j, initially zero and supported on S, to obtain $b_j = u_j v_j + d_j$. For a residual learning-rate multiplier $\gamma > 0$, define

$$\gamma_j := \gamma \mathbf{1}_{\{j \in S\}},$$

then train both factors and the residual by

$$\dot{u}_j = -v_j g_j, \quad \dot{v}_j = -u_j g_j, \quad \dot{d}_j = -\gamma_j g_j, \quad g_j = \frac{\partial \mathcal{L}(b)}{\partial b_j}. \tag{29}$$

The initial predictor and the set of attainable effective coefficient vectors are unchanged. When $u_{0,j} \neq 0$ on S, $d_j \Phi_j(X) = (d_j/u_{0,j})[u_{0,j}\Phi_j(X)]$ also describes an output mapping of the initial response. This interpretation requires nonzero initial gains; the coefficient-level result permits arbitrary S.

Proposition 2 (Lower recombination cost through a residual coefficient path)**.** *Define the new initialization cost by the infimal convolution*

$$\widetilde{\mathcal{C}}_0(\bar{b}, \beta_0) = \inf_{\mathrm{supp}(d) \subseteq S} \left\{ \mathcal{C}_0(\bar{b} - d, \beta_0) + \frac{\|d\|^2}{2\gamma} \right\}. \tag{30}$$

Under the preceding assumptions, for every finite $\bar{b}$ and $T > 0$ the joint flow satisfies

$$\mathcal{L}(b(T)) - \mathcal{L}(\bar{b}) \leq \frac{\widetilde{\mathcal{C}}_0(\bar{b}, \beta_0)}{T}, \qquad \widetilde{\mathcal{C}}_0(\bar{b}, \beta_0) \leq \mathcal{C}_0(\bar{b}, \beta_0). \tag{31}$$

The cost inequality is strict if and only if $\bar{b}_j \neq \beta_{0,j}$ for some $j \in S$. Locally,

$$\widetilde{\mathcal{C}}_0(\beta_0 + \delta, \beta_0) = \frac{1}{2} \sum_j \frac{\delta_j^2}{q_{0,j} + \gamma_j} + O(\|\delta\|^3). \tag{32}$$

Proof. For fixed b, the objective in equation 30 is continuous, coercive, and strictly convex on the closed coordinate subspace $\{d : \mathrm{supp}(d) \subseteq S\}$: $\mathcal{C}_0 \geq 0$ and the quadratic term grows as $\|d\|^2/(2\gamma)$. It therefore has a unique minimizer. Write $\beta = b - d$. Its coordinate optimality conditions are

$$d_j = \gamma_j [\psi'_{c_j}(\beta_j) - \psi'_{c_j}(\beta_{0,j})], \qquad b_j = \beta_j + \gamma_j [\psi'_{c_j}(\beta_j) - \psi'_{c_j}(\beta_{0,j})].$$

For $j \notin S$, these simply say $d_j = 0$ and $\beta_j = b_j$. For each coordinate the second map has derivative $1 + \gamma_j/q_j(\beta_j) > 0$ and tends to $\pm\infty$ at the corresponding endpoints of the real line. It is a smooth bijection with a smooth inverse. Thus the minimizer and $F(b) := \widetilde{\mathcal{C}}_0(b, \beta_0)$ are smooth, and differentiation at the minimizer gives

$$\nabla F(b) = \nabla\Psi(\beta) - \nabla\Psi(\beta_0), \qquad \nabla^2 F(b) = \operatorname{diag}\left(\frac{1}{q_j(\beta_j) + \gamma_j}\right). \tag{33}$$

In particular, F is strictly convex, $F(\beta_0) = 0$, and $\nabla F(\beta_0) = 0$.

The joint flow preserves the original factor imbalances. Its solutions are global: the earlier bounds on $u_j^2 + v_j^2$ still hold because $|g_j| \le B_j$, and $|d_j(t)| \le \gamma_j B_j t$. Set $\eta = \nabla\Psi(\beta) - \nabla\Psi(\beta_0)$ along this flow. Then $\dot\eta = -g$ and, since $d(0) = 0$, $d_j = \gamma_j \eta_j$. The trajectory therefore satisfies the same minimizer conditions, giving

$$\frac{d}{dt}\nabla F(b(t)) = -\nabla\mathcal{L}(b(t)), \qquad \dot{\mathcal{L}}(b(t)) = -\sum_j [q_j(\beta_j(t)) + \gamma_j] g_j(t)^2 \le 0.$$

Applying the Lyapunov argument of Proposition 1 with potential F proves the time bound, since its initial divergence is $D_F(\bar b, \beta_0) = F(\bar b)$.

Choosing $d = 0$ proves the cost inequality. This candidate is optimal exactly when $\psi'_{c_j}(\bar b_j) = \psi'_{c_j}(\beta_{0,j})$ for $j \in S$, equivalently $\bar b_j = \beta_{0,j}$ there. Otherwise the unique minimum is strictly smaller. Finally, equation 33 at initialization gives the stated quadratic term. The coordinate third derivative is

$$\frac{\partial^3 F}{\partial b_j^3} = -\frac{4\beta_j}{[q_j(\beta_j) + \gamma_j]^3},$$

whose magnitude is at most $1/c_j^2$. The remainder is therefore bounded as in equation 27. □

For a fixed comparator, the available improvement in equation 28 is unchanged while its certified recombination cost can decrease. This strengthens the corresponding guarantee without ordering the actual training trajectories.

C.6 Scope of the comparison

The controlled model learns response gains and output coefficients over fixed nonlinear patterns with an affine logit readout. Full SFT can also change gates, feature directions, and nonlinear downstream computation. The coefficient residual above is an analytical extension, not a global guarantee for an intermediate FFN or a low-rank module, nor a claim of uniqueness over other parameterizations. Expectations may be replaced by empirical averages, giving the same results for training CE without a held-out guarantee.

The analysis does not establish sparsity, preservation of a top-k set, or an unconditional advantage of pretrained over random initialization. Empirical localization and ablation examine initial availability and subsequent function; selection frequencies do not measure $q_{0,j}$ or $\mathcal{C}_0$, and changed task allocation alone does not establish a causal performance effect of recombination.

D Implementation of Initial Localization-Guided Recombination

D.1 Initial FFN channel selection

The method uses the FFN criterion in Appendix B. For each backbone, we fix training prompts before SFT and detect channels using the same prompt truncation and FFN localization criterion as Appendix B.2. Answers are excluded and padding is masked. Channel sets are computed independently for each model and domain.

Let $Z_l(x_i; \theta_0) \in \mathbb{R}^{T_i \times m_l}$ stack the initial model's gated FFN activations over the T_i valid prompt tokens. For channel j, its local output contribution has Frobenius norm

$$a_{lj}(x_i) = \left\| Z_l(x_i;\theta_0)[:, j]\, W_l^{\text{down}}[:, j]^\top \right\|_F = \|Z_l(x_i;\theta_0)[:, j]\|_2 \left\| W_l^{\text{down}}[:, j] \right\|_2, \tag{34}$$

where the projection weights are also evaluated at θ_0. This is the exact local output change from removing the channel with the incoming activations fixed. It measures contribution magnitude rather than final-loss attribution or exclusive recommendation knowledge.

For each prompt and layer, select the $q_l = \max(1, \lfloor \rho m_l \rfloor)$ highest-scoring channels, where $\rho \in (0, 1]$ is a within-layer selection fraction. For n prompts and a frequency threshold $\tau \in (0, 1]$, define

$$\nu_{lj} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\left[j \in \mathrm{Top}_{q_l}(a_l(x_i))\right], \qquad S_l = \{j : \nu_{lj} \geq \tau\}. \tag{35}$$

Thus the final set contains channels that recur across recommendation contexts, with the final set size determined by selection frequency. Each FFN channel is counted once. No endpoint localization is used to define or revise S_l.

D.2 Current activations and supported output corrections

With θ_{SFT} frozen, the residual module uses the selected channels' current FFN activations$P_l z_l(h_l; \theta_{\mathrm{SFT}})$ at every forward pass. Because upstream residual modules can change later hidden states, gradients are propagated through the frozen backbone computation while the backbone parameters themselves remain fixed. The residual is an additive update to the output projection:

$$\Delta W_l^{\mathrm{down}} = \beta B_l A_l P_l, \qquad \mathrm{rank}(\Delta W_l^{\mathrm{down}}) \leq r, \qquad \Delta W_l^{\mathrm{down}}[:, j] = 0 \quad (j \notin S_l). \tag{36}$$

For nonempty sets, $P_l P_l^{\top} = I_{k_l}$, and the column constraint follows because $P_l[:, j] = 0$ outside S_l. Conversely, for nonzero β, every matrix with this column support and rank at most r admits such a factorization: factor its selected columns divided by β as $B_l A_l$, then reinsert them using P_l. The supported low-rank form therefore characterizes the chosen family of linear output corrections.

Layers have independent factors and no intervening bias or activation. Entries of A_l are initialized as independent standard normal draws divided by $\sqrt{k_l}$, while B_l is initialized to zero. The trainable parameter count is $\sum_{l=1}^{L} r(k_l + d_l)$. At fixed β, the correction can algebraically be merged into W_l^{down}; the evaluated implementation applies it as a residual branch.

D.3 Two-stage supervision and target construction

The first stage uses the original recommendation answers, which may concatenate multiple complete target SIDs, for full-parameter SFT. The second stage transforms these same training examples into fixed single-SID answers. Prompts are unchanged.

For each training example with multiple complete target SIDs, we select one target SID once and keep this choice fixed throughout the second stage. The selection is deterministic with respect to the sample identifier, so the transformed target is never resampled during training.

Let s_i^{single} be the selected complete SID followed by EOS, and let $N_{\mathrm{single}} = \sum_i |s_i^{\mathrm{single}}|$ be the total number of supervised answer tokens. With the backbone fixed and only $\phi = \{A_l, B_l\}_{l=1}^{L}$ trainable, the second-stage objective is

$$\mathcal{L}_{\mathrm{recomb}}(\phi) = -\frac{1}{N_{\mathrm{single}}} \sum_i \sum_{t=1}^{|s_i^{\mathrm{single}}|} \log p_{\theta_{\mathrm{SFT}}, \phi}\big(s_{i,t}^{\mathrm{single}} \mid x_i, s_{i,<t}^{\mathrm{single}}\big). \tag{37}$$

Cross-entropy supervises every SID token and EOS, masking prompt and padding labels. Loss is averaged over answer tokens within each microbatch, then across accumulation steps and workers. Equal answer lengths give equal example weights in the second stage. The transformed pool retains all original examples. Both stages use recommendation supervision alone. Training uses $\beta = 1$; at inference, β scales the learned correction.

E Additional Behavioral Results

Table 6 reports the complete behavioral results under the primary training recipes. Recommendation metrics are evaluated on the fixed 5,000-user cohort defined in Appendix A.3 and reported on a $100\times$ scale; general-task scores are percentages.

Table 6: Behavioral results under the primary training recipes. None denotes no SFT in this comparison.

Initialization	SFT	MRR	Hit	Recall	NDCG	Math	Code	Know.
Qwen+SID	None	2.1610	5.4000	1.3935	1.1133	81.65	64.02	40.30
Qwen+SID	Rec	7.2059	43.5200	13.2787	6.0797	0.00	0.00	0.00
Qwen+SID	Mix	7.1076	43.7800	13.2593	6.0713	68.16	53.05	45.30
Random+SID	None	0.0050	0.1600	0.0423	0.0113	0.00	0.00	1.60
Random+SID	Rec	5.1046	35.9200	10.3766	4.5034	0.00	0.00	0.35
Random+SID	Mix	5.0949	35.8000	10.2969	4.5010	0.30	0.00	10.80
OneRec	None	1.4661	4.8000	1.2249	0.8450	75.89	13.41	25.55
OneRec	Rec	7.4711	45.3000	14.1849	6.4094	19.18	0.00	0.00
OneRec	Mix	7.3014	45.6800	14.0933	6.4252	72.93	53.05	44.50

The results show that Qwen+SID exceeds Random+SID by 2.1013 MRR points after Rec, with a paired 95% interval of $[1.6067, 2.5861]$, and by 2.0128 points after Mix, $[1.5342, 2.4863]$. To isolate the effect of mixed supervision, Table 7 compares Mix with Rec within each initialization using paired differences on the same users.

Table 7: Mix-minus-Rec recommendation differences, with paired 95% intervals over 5,000 users.

Initialization	ΔMRR	ΔHit	ΔRecall	ΔNDCG
Qwen+SID	-0.10 [-0.38, 0.18]	+0.26 [-0.54, 1.06]	-0.02 [-0.32, 0.30]	-0.01 [-0.12, 0.10]
Random+SID	-0.01 [-0.28, 0.25]	-0.12 [-0.96, 0.66]	-0.08 [-0.38, 0.21]	-0.00 [-0.11, 0.10]
OneRec	-0.17 [-0.46, 0.12]	+0.38 [-0.42, 1.14]	-0.09 [-0.42, 0.22]	+0.02 [-0.09, 0.13]

Across the three initializations and four recommendation metrics, all twelve Mix-minus-Rec intervals include zero. These intervals support the absence of a clear accompanying gain under the evaluated recipes; they do not establish equivalence.

Initial Qwen+SID knowledge accuracy is 40.30%. The separately evaluated native Qwen checkpoint without the SID extension has a 40.20% knowledge score; it has no recommendation result under this SID protocol and is not substituted into these comparisons.

E.1 Individual auxiliary tasks

To separate the effects of individual auxiliary tasks, we additionally train OneRec with mathematics, code, or knowledge added to recommendation SFT. The mathematics-only and code-only recipes use a 70:30 recommendation-to-auxiliary sample ratio. The knowledge-only recipe uses 62,713 Product and 6,758 knowledge examples (approximately 90.3:9.7) and takes 1,085 steps. The three-task Mix follows the recipe in Table 4; the auxiliary exposure therefore differs across these comparisons.

Table 8: OneRec with individual auxiliary tasks. Recommendation scores are multiplied by 100 and general-task scores are percentages. Auxiliary doses differ as described in the text.

Recipe	MRR	Recall	NDCG	Math	Code	Know.
No SFT	1.4661	1.2249	0.8450	75.89	13.41	25.55
Rec	7.4711	14.1849	6.4094	19.18	0.00	0.00
Rec + math	7.3214	13.9037	6.3460	71.11	63.41	25.70
Rec + code	7.4861	14.0547	6.4438	67.48	54.88	31.80
Rec + knowledge	7.2705	14.1971	6.4311	0.23	0.00	46.05
Mix (three tasks)	7.3014	14.0933	6.4252	72.93	53.05	44.50

For an additional output-controlled knowledge check, OneRec Rec, knowledge-mixed, and three-task-mixed models reach 29.40%, 45.55%, and 44.40% under the same legal-option restriction. Thus, the recovery under knowledge supervision extends to option selection while recommendation still shows no clear gain.

F Additional Neuron Structure Results

At initialization, Qwen+SID has 18,667 recommendation positions, including 8,249 shared with general tasks; Random+SID has 2,055, including eight shared positions. After Rec, the retained initial recommendation counts are 11,968 and 575, respectively. In Qwen+SID, 9,239 final recommendation positions belong to the initial general-task union, or 42.90% of the final set. This overlap is 8.23 times its within-layer/projection random expectation, versus 1.69 times for Random+SID. The reference adjusts for set sizes and localization strata; it is a descriptive overlap comparison.

For the fixed initial Qwen shared cohort, mean recommendation selection frequency changes from 96.73% to 87.30% after Rec. For the initially general-only cohort, it rises from 25.41% to 36.52%. The two observations are measured at initialization and the trained endpoint; connecting them does not infer a training trajectory or a time of stabilization. Reorganization concerns selection at fixed coordinates, without establishing new connectivity or unchanged computational content.

To trace the origin of the final recommendation set, we partition each endpoint set by its task membership at initialization as defined in Appendix B.3. Structural counts in this section merge paired feed-forward entries. Table 9 reports both the resulting counts and their fractions of the endpoint recommendation set.

Table 9: Initial-membership decomposition of the final recommendation set. The four columns partition that set, classified by initial membership.

Initialization	SFT	Shared	Rec only	General only	Unselected
Qwen+SID	Rec	6,857 (31.84%)	5,111 (23.73%)	2,382 (11.06%)	7,184 (33.36%)
Qwen+SID	Mix	6,872 (32.29%)	5,163 (24.26%)	2,364 (11.11%)	6,881 (32.34%)
Random+SID	Rec	7 (0.08%)	568 (6.17%)	15 (0.16%)	8,618 (93.59%)
Random+SID	Mix	4 (0.04%)	591 (6.42%)	9 (0.10%)	8,598 (93.44%)

The main structural analysis uses a selection-frequency threshold of 0.8. To assess sensitivity to this choice, Table 10 repeats the recommendation-set size and shared-fraction summaries at thresholds of 0.5, 0.8, and 1.0, while keeping the prompt-level top-10% ranking fixed.

Table 10: Sensitivity of recommendation-set size and shared fraction to the selection-frequency threshold. All counts merge paired feed-forward channels.

Initialization	SFT	$n_{50\%}$	$\text{Share}_{50\%}$	$n_{80\%}$	$\text{Share}_{80\%}$	$n_{100\%}$	$\text{Share}_{100\%}$
Qwen+SID	None	29,785	47.95	18,667	44.19	4,805	50.16
Qwen+SID	Rec	29,950	49.74	21,534	45.67	9,737	41.38
Qwen+SID	Mix	29,776	47.74	21,280	43.49	9,492	38.78
Random+SID	None	10,396	5.68	2,055	0.39	78	0.00
Random+SID	Rec	21,156	5.58	9,208	0.39	883	0.00
Random+SID	Mix	21,373	11.46	9,202	2.20	953	0.00

The set-based analyses above retain only thresholded task membership. Table 11 instead compares the complete selection-frequency profiles between initialization and the trained endpoint. Frequencies are centered within each layer and projection before cosine similarity is computed.

The strong similarity for Qwen's general-task frequencies coexists with reduced ordinary answer scores. This supports continuity in the coordinate-selection profile, not unchanged general-task computation.

Table 11: Initial-to-final cosine similarity of task-selection frequency vectors.

Initialization	SFT	Recommendation	Math	Code	Knowledge
Qwen+SID	Rec	0.6826	0.9549	0.9423	0.9494
Qwen+SID	Mix	0.6952	0.9533	0.9441	0.9239
Random+SID	Rec	0.2115	0.3738	0.3599	0.3578
Random+SID	Mix	0.2189	0.1945	0.2103	0.2105

G Additional Functional Interventions

The Qwen fixed initial shared mask contains 2,811 projection coordinates, corresponding to 2,012 positions after merging paired feed-forward entries. On the intervention cohort, Rec MRR falls from 7.2601 to 1.8297 under targeted removal; the mean matched-random score is 7.2482. The random-adjusted damage is 5.4184 points, $[4.2391, 6.6556]$. After Mix, damage is 5.8192; the paired Mix-minus-Rec difference is $+0.4008$, $[-0.3258, 1.1008]$. Intervals for all damage contrasts include zero (Table 14).

All intervention results in this section use the fixed 1,000-user cohort defined in Appendix A.3. Recommendation metrics are reported on a $100\times$ scale. Intervention tables count feed-forward up/down projection coordinates separately, and random-control scores average the three matched masks defined in Appendix B.4.

Table 12 first reports the raw Qwen+SID MRR scores after Rec, comparing the baseline with targeted and matched-random removal.

Table 12: Raw Qwen+SID Rec intervention scores MRR@128 ($\times 100$).

Removed set	Coordinates	Baseline	Targeted	Random
Initial shared (quarter)	2,811	7.2601	1.8297	7.2482
Current shared (quarter)	3,216	7.2601	1.3572	7.1897
Current Rec difference (balanced)	3,209	7.2601	6.3916	7.2500
Initial general difference (quarter)	2,913	7.2601	6.7864	7.1900

Table 13 summarizes intervention effects using the random-adjusted damage $D(S)$ defined in Appendix B.4 and extends the comparison across Rec, Mix, and all four recommendation metrics. Positive values indicate greater damage from targeted removal than from the matched random controls. Mask construction follows the quarter and common-budget rules defined in Appendix B.4. Initial/current shared masks use the quarter rule; the current recommendation difference uses the common-budget rule.

Table 13: Random-adjusted intervention damage for Qwen+SID with paired 95% confidence intervals.

SFT	Removed set	MRR	Hit	Recall	NDCG
Rec	Initial shared	+5.42 [4.24, 6.66]	+28.97 [25.93, 31.87]	+9.70 [8.48, 10.97]	+4.56 [3.93, 5.19]
Rec	Current shared	+5.83 [4.65, 7.09]	+30.73 [27.63, 33.83]	+10.25 [8.96, 11.46]	+4.81 [4.17, 5.45]
Rec	Current Rec diff.	+0.86 [0.20, 1.55]	+4.00 [2.17, 6.00]	+1.17 [0.53, 1.87]	+0.65 [0.40, 0.92]
Mix	Initial shared	+5.82 [4.62, 7.03]	+30.40 [27.33, 33.27]	+9.79 [8.58, 11.03]	+4.77 [4.10, 5.43]
Mix	Current shared	+5.50 [4.30, 6.77]	+31.03 [28.13, 34.03]	+10.29 [9.12, 11.57]	+4.68 [4.05, 5.34]
Mix	Current Rec diff.	+1.06 [0.44, 1.67]	+4.20 [1.97, 6.43]	+1.39 [0.64, 2.14]	+0.74 [0.47, 0.99]

Finally, Table 14 directly compares the fixed initial masks across training recipes by subtracting Rec damage from Mix damage. Because the masks are fixed at initialization, the same coordinates are removed at both endpoints. Positive differences indicate greater random-adjusted damage after Mix.

Table 14: Rec-to-Mix change in random-adjusted damage for fixed initial Qwen+SID masks, with paired 95% confidence intervals.

Removed set	**ΔMRR**	**ΔHit**	**ΔRecall**	**ΔNDCG**
Initial shared	+0.40 [-0.33, 1.10]	+1.43 [-0.70, 3.60]	+0.09 [-0.77, 0.99]	+0.21 [-0.08, 0.50]
Initial general difference	-0.22 [-0.89, 0.36]	+0.33 [-1.63, 2.33]	+0.19 [-0.58, 0.88]	-0.02 [-0.26, 0.22]

The fixed initial shared subset is chosen at initialization and evaluated after training, whereas the current shared and current recommendation-difference subsets are defined at the endpoint. In particular, $R_1 \setminus G_1$ is not the same as the initially unselected origin category $R_1 \setminus (R_0 \cup G_0)$; its intervention is not an ablation of that origin category. Projection-coordinate counts in these intervention tables count up/down entries separately, unlike the structural tables.