DISCOVERED, NOT DESIGNED: POPULATION EVOLUTION FOR COLLABORATIVE AND COMPUTE-INTENSIVE MODEL DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-driven evolution enables iterative model development, but two practical goals remain underexplored: finding model designs that transfer across related tasks and sustaining improvement when training is expensive. We introduce **Population Evolution (PE)**, a collaborative, hierarchical framework that connects ongoing local searches through shared experimental evidence. PE evaluates code changes across related training instances and shares the results to guide subsequent proposals and promotion to larger training scales. For expensive targets, PE searches small training subsets and screens candidates through peer and intermediate evaluations before full-target training. We introduce **RMD-Bench** to evaluate both settings across ranking, watch-time prediction, RL algorithm discovery, and LLM/VLM pretraining. Compared with standalone evolution at matched source iterations, PE raises mean best local gains from 7.01% to 8.97% in ranking and from 2.84% to 3.85% in watch-time, while improving the best larger-scale outcome in all three joint-discovery families. In watch-time discovery, PE improves best larger-scale gains with four of five harnesses and all four proposers. On new recommendation datasets under shared target-side calibration, every evaluated PE design improves over the reference in mean performance. Under matched total GPU compute, completed LLM discovery runs yield a best relative accuracy gain of 2.48% and 13 successful candidates for PE, versus 0.92% and none for direct evolution. VLM loss reduction reaches 8.78% versus 5.05% under matched total GPU compute.

1 INTRODUCTION

LLM-driven evolution automates model development by proposing code changes, evaluating them, and using the results to guide further edits (Liu et al., 2024; Novikov et al., 2025; Lange et al., 2025). Related research agents automate idea generation and experimentation (Lu et al., 2024), while neural architecture discovery offers a route toward recursive self-improvement (Pepe et al., 2026). Despite this progress, two goals remain underexplored in LLM-driven neural model discovery: discovering universal model designs across related tasks and sustaining improvement when candidate training is expensive.

Collaborative model discovery within a domain. Related datasets often share modeling requirements, so a mechanism discovered on one can inform development on another. Independent single-task searches primarily use their own feedback (Novikov et al., 2025; Lange et al., 2025), leaving this evidence isolated. Connecting these searches can support both local improvement and *universal model designs*: a common architecture or training mechanism used across related instances, with parameters trained separately on each.

Sustaining discovery on computationally demanding tasks. When every candidate requires substantial computation, as in full LLM or VLM training, a fixed budget permits few proposal–evaluation cycles. Parallel full-target searches broaden exploration, but costly evaluations limit how many revisions each search can make (Novikov et al., 2025). Sustained discovery therefore requires affordable experiments that inform continued development, followed by verification under full-target training.

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.

We introduce **Population Evolution (PE)**, which connects ongoing local searches through a shared ledger of executable interventions and cross-instance evaluation results (Figure 1).

Each source retains its objective and archive; shared feedback guides local edits, while replay lets other instances reuse its discoveries. Cross-instance replay tests the same intervention after retraining under each receiving instance's protocol.

In **joint discovery**, searches on related datasets or training configurations reuse, adapt, and combine mechanisms; replay and larger-scale evaluation establish where the resulting designs remain useful. In **decomposed discovery**, small subsets of an expensive target's training data provide affordable search instances, with peer and intermediate evaluations directing candidates toward full-target training. This hierarchy supports iterative refinement while reserving full-target training for screened candidates.

We introduce the **Recursive Model Discovery Benchmark (RMD-Bench)**, providing executable evaluation protocols for five task families spanning both settings. Compatible interfaces and fixed references support evaluation of local gains and transfer in joint discovery, and full-target quality and yield in decomposed discovery. All three joint-discovery families also test transfer to new datasets or disjoint coding tasks. We start from FuXi-Linear (Ye et al., 2026) for ranking, EGMN (Zhao et al., 2025) for watch-time prediction, GRPO (Shao et al., 2024) for RL algorithm discovery, Gemma 3-style architectures (Gemma Team, 2025) for LLM pretraining, and Qwen3.5-style architectures (Qwen Team, 2026) for VLM pretraining. A complete LLM or VLM discovery run, including five searches and their evaluations, requires thousands of GPU-hours.

At matched source iterations, PE improves the mean per-source best gain in both recommendation families and the best larger-scale outcome in all three joint-discovery families. Every evaluated PE recommendation design improves over the reference mean on new datasets. Retraining uses shared target-side calibration without candidate-specific tuning. For LLM pretraining, completed runs under matched total GPU compute yield a 2.48% best relative accuracy gain and 13 successful candidates for PE, versus 0.92% and none for direct evolution. For VLM pretraining, best loss reduction rises from 5.05% to 8.78% under matched total GPU compute. Our contributions are:

- **Population Evolution:** a collaborative and hierarchical framework that uses shared experimental feedback to guide local search, reuses executable interventions across instances, and promotes candidates across training scales.
- **RMD-Bench:** five executable task families with compatible code interfaces, shared reference implementations, and controlled cross-instance and cross-scale evaluation, including transfer to new datasets and tasks in all three Track A families.
- **Model discoveries and transfer:** recommendation designs that improve over shared references on new datasets, better larger-scale outcomes in all three Track A families, and successful full-target LLM/VLM interventions, with watch-time comparisons across harnesses and proposers.

2 Population Evolution

An instance E_i specifies the data, objective, and training/evaluation protocol. In standard single-task evolution, an LLM repeatedly edits model or training code on one instance, using evaluation feedback to guide subsequent proposals (Liu et al., 2024; Novikov et al., 2025); we call this a *local search*. PE connects local searches across related *source instances* $\mathcal{D} = \{E_i\}$, sharing evidence to guide proposals and promotion to larger training scales. A *candidate* is an executable intervention in model architecture or training code. A *shared ledger* M_t stores candidate records and evaluation results. Figure 1 summarizes local search, cross-instance sharing, and promotion; Algorithm 1 in Appendix B.4 gives the asynchronous loop.

2.1 Local search on each source instance

Each local search selects a parent from its own code archive. An LLM edits that parent's model or training code, and the resulting candidate is trained and evaluated on the same source instance. Evaluation updates the local history and archive; PE supplements this feedback with shared experimental evidence. At proposal step t, proposer p_i generates candidate $h_{i,t}$ from local history $\tau_{i,t}$ and shared

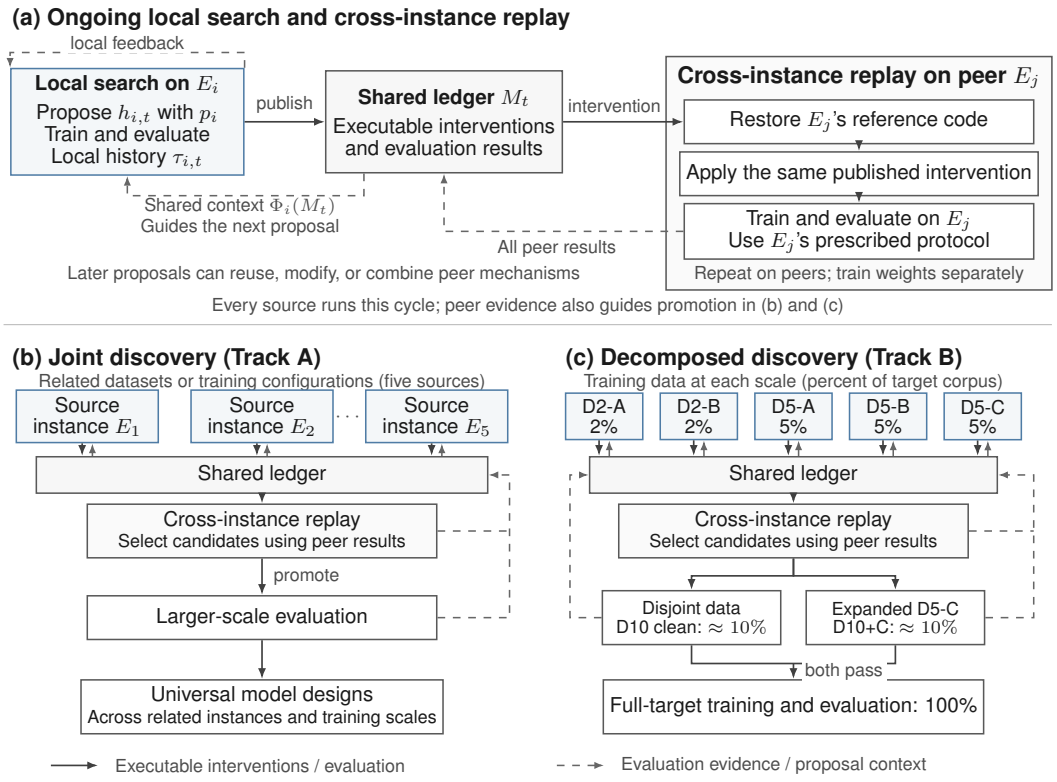


Figure 1: **PE connects local searches through replay evidence and cross-scale promotion.** (a) Published interventions are retrained on peers; their results enter the shared ledger and inform later proposals (Equation 1). (b) Track A tests the same designs across related instances and at larger training scales. (c) Decomposed discovery screens candidates on small sources and both intermediate instances before full-target training. In Track B, full-target results remain outside proposal context.

evidence $\Phi_i(M_t)$:

$$h_{i,t} \sim p_i(h \mid \tau_{i,t}, \Phi_i(M_t)) \,. \tag{1}$$

With the local harness, proposer, and archive-update rules fixed, omitting $\Phi_i(M_t)$ reduces the source-search procedure to standalone evolution. Candidate code is inherited and edited; model parameters are trained separately for each evaluation.

2.2 Sharing and evaluating candidates across instances

When a candidate meets its source's configured improvement criteria, PE publishes its executable intervention, mechanism description, provenance, and score to the shared ledger. The other source instances are its *peers*. PE restores each peer's reference code, applies the published intervention, then trains and evaluates it under that peer's protocol. We call this *cross-instance replay* (Figure 1(a)). All peer results, including weak or unsuccessful transfers, are added to the ledger.

Before every LLM call, $\Phi_i(M_t)$ summarizes the instance's own published candidates and peer transfers to E_i, including mechanisms and measured outcomes of successful and unsuccessful transfers. Peer and promotion results affect local proposals only through this context; code archives remain instance-specific. The proposer can use these outcomes to reuse, modify, or combine mechanisms. Appendices B.2 and B.3 detail the shared records, proposal context, and prompt structure.

2.3 PROMOTING CANDIDATES TO LARGER TRAINING SCALES

Shared results also govern promotion: published candidates must meet the configured improvement criteria on at least two of their four peers in our experiments (Appendix B.1).

In **joint discovery (Track A)**, eligible candidates are retrained on a larger-scale instance. Peer and larger-scale results return to the ledger, supporting local improvement and identifying designs useful across related instances (Figure 1(b)).

In **decomposed discovery (Track B)**, source searches run on affordable training subsets. Eligible candidates are tested on two intermediate instances; those passing both advance to full-target training (Figure 1(c)). Peer and intermediate results inform further proposals, while full-target scores and pass/fail outcomes are withheld from proposal context. This is intended to prevent a few full-target successes from prematurely steering all source searches toward similar designs. A candidate *graduates* when it passes the final stage: larger-scale evaluation in Track A or full-target evaluation in Track B. Local searches continue while these evaluations run asynchronously.

3 RMD-BENCH: TASKS AND EVALUATION

RMD-Bench defines five task families with objectives, reference implementations, editable components, and executable training/evaluation protocols. Compatible interfaces support retraining across related instances. The 34 core instances vary data, training seed, or scale; Track A also evaluates transfer beyond discovery instances. The benchmark fixes evaluation conditions within each instance; each discovery system chooses its search, sharing, and selection policies. Table 1 compares benchmark support for these settings; Section 6 discusses related work, and Appendix A gives task specifications.

3.1 JOINT DISCOVERY (TRACK A)

Track A tests local improvement and discovery of universal model designs useful across related tasks.

Discovery instances. Ranking, watch-time prediction, and RL algorithm discovery each have five source instances and one larger-scale instance (18 instances in total; Figure 1(b)). Within each recommendation family, the five sources use different datasets; watch-time includes CIKM16 and WeChat. Both families use KuaiRand-27K for larger-scale evaluation. Ranking edits sequence representations and temporal interactions in FuXi-Linear (Ye et al., 2026), using the mean of NDCG@10, HR@10, and MRR as its score. Watch-time discovery edits EGMN's distributional head and point-prediction rule (Zhao et al., 2025), using MAE in seconds. RL discovery edits the GRPO loss (Shao et al., 2024) and rollout filtering for Qwen2.5-Coder policies (Hui et al., 2024). The five RL sources train 1.5B policies on the same task with different seeds; the larger-scale instance trains a 3B policy and evaluates pass@1 on a 423-problem LiveCodeBench partition (Jain et al., 2024). Appendices A.1–A.3 specify the three protocols.

Transfer beyond discovery instances. Track A evaluates designs beyond the discovery instances: KuaiVideo for ranking, VK-LSVD for watch-time prediction, and 632 LiveCodeBench tasks disjoint from the larger-scale selection partition for RL. The two recommendation datasets are excluded from source search, peer evaluation, and promotion. KuaiVideo also changes the objective from next-item ranking to candidate-conditioned click-through rate (CTR) prediction. Recommendation retraining uses a shared configuration calibrated and scored on each target's released test split, without candidate-specific redesign or tuning. RL evaluation uses held-out coding tasks. These evaluations supplement the 34 core instances; Appendix F specifies selection, calibration, and reporting.

3.2 DECOMPOSED DISCOVERY (TRACK B)

Track B evaluates whether discovery on small training subsets produces effective model and training designs for an expensive full target within a compute budget.

Training hierarchy. LLM and VLM pretraining each have five source instances, two intermediate instances, and one full-target instance (16 instances in total; Figure 1(c)). The hierarchy tests transfer across training subsets and training scales before committing to expensive full-target evaluation. The

sources include three disjoint 5% subsets, D5-A/B/C. D2-A and D2-B each contain 2% of the target corpus and are nested in D5-A and D5-B, respectively. Disjoint peers test whether gains persist beyond the source subset; the nested 2%/5% instances test transfer across data volumes and numbers of training steps. Two approximately 10% intermediate instances test different forms of transfer: D10 clean uses data disjoint from every source, while D10+C combines D5-C with a fresh 5% subset. Requiring success on both is intended to prioritize designs whose gains persist under full-target training. Full-target evaluation uses 100% of the training corpus. The same executable intervention is evaluated at each tier under its prescribed training configuration.

Pretraining tasks. LLM discovery starts from a 185M-parameter Gemma 3-style decoder (Gemma Team, 2025) and can modify its architecture, optimizer, and training mechanisms. Training uses the FineWeb-Edu 100B core set (Lozhkov et al., 2024; Karpathy, 2025). The score is mean accuracy over the seven zero-shot tasks used in Mamba-2 (Dao & Gu, 2024): ARC-Challenge, ARC-Easy, HellaSwag, LAMBADA OpenAI, OpenBookQA, PIQA, and WinoGrande. VLM discovery edits a Qwen3.5-style backbone (Qwen Team, 2026), projector, optimizer, and schedule, retaining a frozen SigLIP encoder (Zhai et al., 2023). The starting VLM has approximately 646M trainable parameters; full-target training uses 2.38M raw FineVisionMax-format examples (Wiedmann et al., 2025). The score is token-weighted validation cross-entropy. Appendices A.4 and A.5 give the full specifications.

3.3 Evaluation and reporting

Each intervention is evaluated by training a model under the receiving instance's prescribed initialization, data, and training allocation. For candidate score $y_e(h)$ and fixed reference score b_e on instance e, relative improvement is

$$r_e(h) = \frac{s_e\big(y_e(h) - b_e\big)}{\max(|b_e|, \epsilon)}, \tag{2}$$

where $s_e = +1$ for ranking score or accuracy and -1 for MAE or loss; $\epsilon > 0$ prevents division by zero. Reported percentage gains are $100 r_e(h)$. Track A reports local gains, larger-scale transfer, and transfer beyond discovery instances. Track B reports full-target quality and discovery yield: the number of candidates meeting predefined full-training success criteria. Section 4 specifies the comparisons; Appendices A.7, A.8, and B.5 describe baseline calibration, discovery compute scales, and budget accounting.

Table 1: **Explicit benchmark support for collaborative and compute-intensive discovery.**

Evaluation support	MLE-bench (2024)	Innovator Bench (2025)	Research Gym (2026)	PostTrain Bench (2026)	MLS-Bench (2026)	Edge Bench (2026)	**RMD-Bench (ours)**
Collaboration protocol	—	—	—	—	—	—	✓
Cross-dataset transfer	—	—	✓	—	✓	—	✓
Cross-scale transfer	—	—	—	—	✓	—	✓
Hierarchical training	—	—	—	—	△	—	✓
Discovery yield vs. compute	—	—	—	—	—	—	✓
Calibrated starting configurations	—	—	✓	—	✓	—	✓

✓: explicit support; —: not specified; △: partial support in MLS-Bench's adaptive pretraining study. Collaboration protocols support exchanging discoveries during ongoing search; transfer evaluates a common intervention across datasets or training scales. Discovery yield counts candidates meeting predefined full-training success criteria; Track B compares completed runs under matched total GPU compute.

4 Experiments

Experiments test local discovery and transfer in Track A, and full-target quality and completed-run yield in Track B. Section 5 analyzes peer evaluation and shared evidence.

4.1 Experimental setup and comparisons

Shared protocols. The main experiments use the AdaEvolve-based search harness (Cemri et al., 2026) with GPT-5.5 at high reasoning effort as the proposer (Appendix B). Methods share starting

models, reference configurations, editable interfaces, instructions, and training/evaluation protocols (Appendix A.6). Gains use fixed instance references (Equation 2). Section 4.4 varies the harness and proposer in watch-time discovery.

Track A: independent and collaborative discovery. Standalone evolution (S) runs independent source searches using the same harness and parent-selection rules as PE, without shared context. Both methods run five 100-iteration searches. Table 2A–C compares locally proposed candidates with and without shared feedback; peer-replayed candidates are reported separately (Section 5.1). Local recommendation results average the best gain from each dataset; RL uses the best result across five searches of the same task. At larger scale, S retrains each source's top candidates by source performance, while PE retrains promoted candidates. S's per-source quota is rounded up to give it at least as many larger-scale evaluations as PE (Table 2D). Neither method searches locally at larger scale. Transfer evaluations use shared target protocols without candidate-specific redesign or tuning (Appendix F.1).

Track B: hierarchical discovery under a compute budget. PE screens candidates from five subset searches at intermediate scale before full training; direct evolution (Direct) runs five full-target searches. Completed runs use matched total GPU compute within each family. PE proposals use only source and intermediate feedback; Direct uses full-target feedback (Appendix B.5).

4.2 Joint discovery improves local quality and generalization

We examine local gains, transfer within the discovery hierarchy, and transfer to new datasets or tasks.

Shared feedback improves mean best local recommendation gains. At the matched source allocation, PE raises mean best local gains from 7.01% to 8.97% in ranking and 2.84% to 3.85% in watch-time, exceeding S on 3/5 and 4/5 datasets, respectively (Table 2). PE's mean gains over calibrated starting models are non-trivial in magnitude, comparable to published improvements over the strongest reported baselines on public datasets: 0.71–12.15% in NDCG@10 for FuXi-Linear (Ye et al., 2026) and 1.72–5.04% in MAE for EGMN (Zhao et al., 2025). The best RL source gain rises from 23.53% to 32.35% (Table 2C).

Selected designs improve larger-scale outcomes. After larger-scale retraining without local search, PE yields more successful evaluations than S and raises best gains from 4.70% to 6.19% in ranking, 1.77% to 2.17% in watch-time, and 1.95% to 6.64% in RL (Table 2D). These larger-scale instances participate in discovery; we next evaluate transfer to new datasets and tasks.

Table 2: **PE improves mean best local recommendation gains and best larger-scale outcomes.** A–C report relative gains (%) from locally proposed candidates at matched source iterations; C also gives solved-task counts. D reports successful/total larger-scale evaluations and best gains after retraining without local search.

A. Ranking: best source gain

Dataset	S	PE
KuaiLive	8.93	**23.19**
KuaiRand-1K	5.53	**9.29**
KuaiRand-Pure	**8.03**	6.07
KuaiRec	1.03	**2.42**
KuaiSAR	**11.52**	3.88
Mean	7.01	**8.97**

B. Watch-time: best source gain

Dataset	S	PE
CIKM16	7.32	**12.55**
KuaiRec	0.40	**0.67**
WeChat	0.80	**0.81**
KuaiRand-Pure	**0.75**	0.16
KuaiRand-1K	4.93	**5.08**
Mean	2.84	**3.85**

C. RL: best source outcome

Summary	S	PE
Best gain (%)	23.53	**32.35**
Solved / 408	14	**15**

D. Larger-scale evaluation outcomes

		Passed / tested		Best gain (%)	
Domain	S top-k	S	PE	S	PE
Ranking	12	11/60	**18/60**	4.70	**6.19**
Watch-time	2	4/10	**5/7**	1.77	**2.17**
RL algorithm	3	0/15	**2/12**	1.95	**6.64**

S: standalone evolution. Bold marks the better S/PE value before rounding. In D, S uses $k = \lceil N_{\mathrm{PE}}/5 \rceil$ per source, where N_{PE} counts PE's larger-scale evaluations. PE evaluates promoted candidates. Scoring and source results: Appendix A.3 and Table 21.

Recommendation designs transfer to new datasets. Under shared target-side calibration, every evaluated PE recommendation design improves on the reference in mean performance (Appendices F.2 and F.3). PE also achieves better candidate-set means on both targets (Table 3A). The best individual design comes from S on VK-LSVD and PE on KuaiVideo.

For RL, the best PE and S candidates achieve 23.10% and 22.94% pass@1 on held-out LiveCodeBench-632 (LCB-632; Table 3B), both below the unmodified reference mean of 23.26% (Appendix F.4).

Table 3: **Transfer performance of the designs returned by each discovery procedure.**

A. Recommendation designs

Target metric	Candidate-set mean		Best candidate	
	S	PE	S	PE
KuaiVideo AUC (%) ↑	74.39	**74.45**	74.43	**74.46**
VK-LSVD MAE (s) ↓	14.55	**14.54**	**14.46**	14.53

B. RL: best candidate

LCB-632	S	PE
Pass@1 (%) ↑	22.94	**23.10**

Candidate scores average results across three training seeds (recommendation) or three stochastic reruns (RL). Candidate-set means average these scores across each method's evaluated recommendation designs. Bold marks the better S/PE value before rounding; Appendix F specifies selection and evaluation protocols.

4.3 Decomposed discovery improves full-target quality and discovery yield

We compare search depth, full-target quality, and yield in completed runs under matched total GPU compute.

Subset training sustains deeper search. Under these budgets, PE completes 100 iterations per source search, compared with 14 LLM or 20 VLM iterations per full-target search for Direct. Source and intermediate feedback guide candidate refinement before full-target training.

LLM: full-target quality and completed-run yield. PE's best full-target relative accuracy gain is 2.48%, versus 0.92% for Direct (Figure 6a). At the predefined 1% threshold, PE yields 13 successful candidates from 39 full-target attempts; Direct yields none from 70 (Table 4). PE returns more successful candidates at all four examined thresholds from 0.5% to 2% (Appendix E.3).

VLM: gains in both model quality and discovery yield. PE improves best full-target loss reduction from 5.05% to 8.78%. PE yields 57 successful candidates from 61 full-target attempts, versus 12 from 100 for Direct (Table 4). These results support affordable subset search with screening before full-target training (final-stage outcomes across families: Appendix E.1).

Table 4: **Hierarchical discovery improves full-target quality and completed-run yield.** Completed-run results use matched total GPU compute. Gains are relative LLM accuracy increases or VLM loss reductions; successful candidates meet full-target thresholds.

Domain	Method	Full-target attempts	Successful candidates	Best gain (%)
LLM	Direct	70	0	0.92
	PE	39	**13**	**2.48**
VLM	Direct	100	12	5.05
	PE	61	**57**	**8.78**

Figure 6 summarizes LLM quality and yield from these completed runs. Appendices B.5 and B.6 detail budget accounting, evaluation counts, and mean scores of passing evaluations.

Successful LLM designs span several intervention types. The 13 successful LLM candidates span changes to Muon update geometry, RoPE configurations within the existing hybrid attention stack, and late-training checkpoint averaging. Section 5.2 follows one discovery; Appendix D groups the mechanisms and lists designs from all five families (Table 23).

4.4 Applicability across search harnesses and proposal models

For watch-time discovery from EGMN, we vary the harness or proposer while holding the other fixed in matched PE/S comparisons (Appendix E.2, Table 25). PE improves the best larger-scale gain with

4/5 harnesses and, with AdaEvolve fixed, all four proposers. With Claude Opus 5, the gain rises from 2.61% to 3.64%. PE improves mean best local gains with 4/5 harnesses and 2/4 proposers. With both Opus proposers, PE has lower local means but better larger-scale outcomes than S; it improves best local gains on at least three of five datasets with every proposer.

5 Analysis: How Collaboration Supports Discovery

We compare local S–PE proposals with and without shared feedback (Section 4.2), measure direct reuse through peer replay (Section 5.1), and examine reconstructed traces of evidence availability and subsequent refinement (Section 5.2).

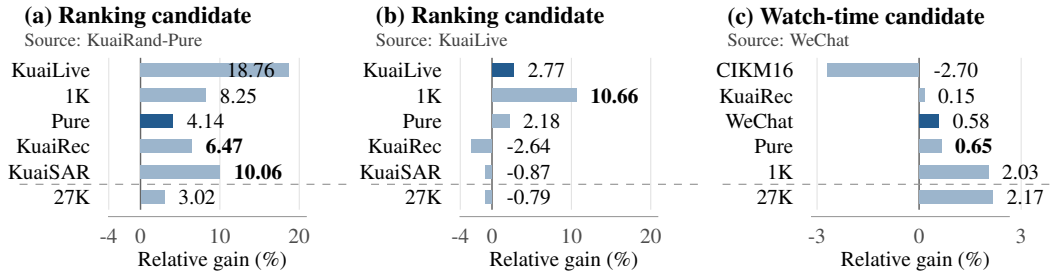


Figure 2: **Peer discoveries can outperform local candidates; larger-scale evaluation tests transfer.** Each panel shows one design retrained unchanged across six instances. Gains use fixed instance references; bold values exceed the best gain from locally proposed PE candidates on the receiving instance (Table 2). Dark bars identify sources; dashed lines separate larger-scale evaluations. 1K/Pure/27K abbreviate KuaiRand datasets (Appendix D).

5.1 Peer discoveries support direct reuse and inform promotion

Direct reuse improves results beyond locally proposed candidates. We compare PE's best locally proposed candidate (Table 2A–B) with its best available candidate after replaying peer discoveries on the same instance. Including replayed peer candidates improves PE's best available gains on three of five ranking and one of five watch-time datasets, raising PE's mean best gains from 8.97% to 11.29% and 3.85% to 3.95%, respectively (Appendix C, Table 21). A KuaiRand-Pure ranking candidate raises KuaiSAR's best available PE gain from 3.88% to 10.06%; a WeChat watch-time candidate raises KuaiRand-Pure's from 0.16% to 0.65%. Replay results inform proposals, while each search continues to select parents from its local archive (Section 2.2). Benefits vary by dataset; with peer candidates, the mean PE–S difference stays positive when any dataset is omitted (Appendix C).

Cross-instance and larger-scale evidence guide which designs advance. Figure 2 compares three individual PE designs, each retrained unchanged on its source, four peers, and larger-scale KuaiRand-27K. The KuaiRand-Pure ranking candidate improves all four peers and gains 3.02% at larger scale (a). The KuaiLive ranking candidate improves KuaiRand-1K by 10.66% but declines by 0.79% at larger scale (b). The WeChat watch-time head reduces larger-scale MAE by 2.17% despite regressing on CIKM16 (c). All three pass peer screening, but only two retain positive gains at larger scale. Peer evaluation identifies reusable designs; larger-scale evaluation checks whether their gains persist.

5.2 Tracing shared evidence and subsequent local refinement

The LLM case reconstructs available shared evidence and traces subsequent local adaptations of established techniques; a complementary watch-time reuse case appears in Appendix D.1.

Shared checkpoint averaging and continued local development.

In LLM pretraining, a D2-B candidate implements checkpoint averaging using exponential moving average (EMA) and stochastic weight averaging (SWA) (Izmailov et al., 2018). Replay yields a 1.43% relative accuracy gain on D5-B. Reconstructing context from archived records and the selection rule

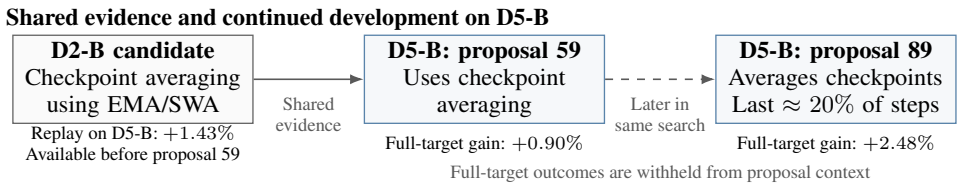


Figure 3: **Shared checkpoint-averaging evidence and subsequent local development.** D2-B evidence is available before D5-B proposal 59; proposal 89 later refines the averaging window. Arrows indicate reconstructed evidence flow and temporal order, not direct parentage. Gains are retrospective comparisons with fixed instance references. Transfer results: Figure 4 (Appendix D.1).

indicates availability before proposal 59. Proposal 59 uses checkpoint averaging late in training. Proposal 89 later restricts averaging to approximately the final 20% of training. Proposals 59 and 89 gain 0.90% and 2.48%, respectively, over the fixed full-target reference. Proposal 89 raises accuracy from its direct parent's 41.13% to 41.25% (Table 24). Full-target outcomes were withheld from proposal feedback.

6 Related Work

Architecture search, Population-Based Training, and AutoML-Zero automate model and algorithm design (Real et al., 2019; Jaderberg et al., 2017; Real et al., 2020). LLM-guided search enables flexible code edits (Romera-Paredes et al., 2024; Liu et al., 2024; Novikov et al., 2025; Lange et al., 2025), while quality-diversity methods maintain diverse candidates (Mouret & Clune, 2015; Nasir et al., 2024). Evolutionary multitasking and multi-task architecture search transfer knowledge across tasks (Gupta et al., 2016; Liao et al., 2024; Zhang & Liu, 2025); AgentRxiv shares research reports (Schmidgall & Moor, 2025). FunSearch uses islands to maintain diverse populations for one target problem (Romera-Paredes et al., 2024). EMO-STA evolves a shared archive before adapting selected programs to individual tasks (Gozeten et al., 2026). PE maintains ongoing task-specific searches and archives, sharing executable interventions and peer evaluation results to guide local edits and cross-scale promotion.

NASNet transfers architectures from CIFAR-10 to ImageNet (Zoph et al., 2018), while FABOLAS varies dataset size during hyperparameter optimization (Klein et al., 2017). AlphaEvolve uses evaluation cascades (Novikov et al., 2025); multi-fidelity methods allocate evaluation resources (Li et al., 2018; Trofimov et al., 2020), and Janus co-evolves proxy evaluators (Liu et al., 2026c). PE combines affordable local search, peer and intermediate screening, and full-target verification.

Existing benchmarks cover competitions and ML experimentation (Chan et al., 2024; Nathani et al., 2025), multi-GPU/multi-node experiments (Wu et al., 2025), and paper-derived research tasks in ResearchGym (Garikaparthi et al., 2026). Others cover post-training (Rank et al., 2026), adaptive pretraining budgets (Lyu et al., 2026), or open-ended tasks (Zhu et al., 2026). RMD-Bench evaluates collaboration across ongoing local searches and discovery across training scales, including transfer to new datasets and tasks in Track A and full-target quality and yield in Track B (Table 1).

7 Conclusion

Population Evolution connects searches through shared evidence and hierarchical promotion; RMD-Bench supplies executable evaluation protocols. Shared feedback improves mean best local recommendation gains; peer replay enables reuse. PE improves best larger-scale outcomes, and its recommendation designs transfer to new datasets. Under matched total GPU compute, hierarchical discovery improves full-target LLM/VLM quality and completed-run yield. Watch-time best larger-scale gains improve with four of five harnesses and all four proposers. Appendix G discusses limitations and future directions.

AI USE STATEMENT

The authors wrote most of the manuscript and supplied its key scientific points, numerical results, and technical details. Generative AI tools were used in a supporting role to draft and polish selected passages, retrieve related literature, and provide feedback on methodology and result interpretation. LLM-driven proposal and code generation within the discovery experiments are described in Section 2 and Appendix B. The authors reviewed and revised AI-assisted content and made the final scientific and editorial decisions, taking full responsibility for the final text, claims, results, and research artifacts.

REPRODUCIBILITY STATEMENT

Appendix A specifies the datasets, preprocessing, starting models, editable components, training and evaluation protocols, and baseline calibration used in RMD-Bench. Appendix B documents PE's publication and promotion criteria, shared proposal context, prompts, asynchronous search procedure, and evaluation accounting. Appendix F details candidate selection, target calibration, reference scores, and repeated transfer evaluations. Appendices C and E report per-instance outcomes and additional configuration and budget comparisons. The benchmark and search code will be made publicly available.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *arXiv preprint arXiv:1907.10902*, 2019. URL https://arxiv.org/abs/1907.10902.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. *arXiv preprint arXiv:1911.11641*, 2019. URL https://arxiv.org/abs/1911.11641.

Mert Cemri, Shubham Agrawal, Akshat Gupta, Shu Liu, Audrey Cheng, Qiuyang Mang, Ashwin Naren, Lutfi Eren Erdogan, Koushik Sen, Matei Zaharia, Alex Dimakis, and Ion Stoica. AdaEvolve: Adaptive LLM driven zeroth-order optimization. *arXiv preprint arXiv:2602.20133*, 2026. URL https://arxiv.org/abs/2602.20133.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. MLE-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*, 2024. URL https://arxiv.org/abs/2410.07095.

CIKM Cup. CIKM Cup 2016 Track 2: Personalized e-commerce search challenge, 2016. URL https://competitions.codalab.org/competitions/11161.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. KuaiRec: A fully-observed dataset and insights for evaluating recommender systems. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, pp. 540–550, 2022a. doi: 10.1145/3511808.3557220. URL https://doi.org/10.1145/3511808.3557220.

Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. KuaiRand: An unbiased sequential recommendation dataset with randomly exposed videos. *arXiv preprint arXiv:2208.08696*, 2022b. URL https://arxiv.org/abs/2208.08696.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation. Software, Zenodo, 2023. URL https://zenodo.org/records/10256836.

Aniketh Garikaparthi, Manasi Patwardhan, and Arman Cohan. ResearchGym: Evaluating language model agents on real-world AI research. *arXiv preprint arXiv:2602.15112*, 2026. URL https://arxiv.org/abs/2602.15112.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786.

Halil Alperen Gozeten, Xuechen Zhang, Emrullah Ildiz, Ege Onur Taga, Tara Javidi, and Samet Oymak. Evolutionary multi-task optimization for LLM-guided program discovery. *arXiv preprint arXiv:2605.22613*, 2026. URL https://arxiv.org/abs/2605.22613.

Sofia Gulevskaia, Mikhail Trapeznikov, Aleksandr Poslavsky, and Alexander D'yakonov. Hierarchical Exponential-Gaussian mixtures for watch-time distribution prediction. *arXiv preprint arXiv:2608.23356*, 2026. URL https://arxiv.org/abs/2608.23356.

Abhishek Gupta, Yew-Soon Ong, and Liang Feng. Multifactorial evolution: Toward evolutionary multitasking. *IEEE Transactions on Evolutionary Computation*, 20(3):343–357, 2016. doi: 10.1109/TEVC.2015.2458037. URL https://doi.org/10.1109/TEVC.2015.2458037.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. URL https://arxiv.org/abs/2409.12186.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2018. URL https://arxiv.org/abs/1803.05407.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017. URL https://arxiv.org/abs/1711.09846.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. URL https://arxiv.org/abs/2403.07974.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Andrej Karpathy. FineWeb-Edu-100B-Shuffle. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/karpathy/fineweb-edu-100b-shuffle.

Andrej Karpathy. autoresearch. Software, 2026. URL https://github.com/karpathy/autoresearch.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 528–536, 2017. URL https://proceedings.mlr.press/v54/klein17a.html.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025. URL https://arxiv.org/abs/2509.19349.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. URL https://jmlr.org/papers/v18/16-558.html.

Yongqi Li, Meng Liu, Jianhua Yin, Chaoran Cui, Xin-Shun Xu, and Liqiang Nie. Routing micro-videos via a temporal graph-guided recommendation system. In *Proceedings of the 27th ACM International Conference on Multimedia*, pp. 1464–1472, 2019. doi: 10.1145/3343031.3350950. URL https://doi.org/10.1145/3343031.3350950.

Peng Liao, XiLu Wang, Yaochu Jin, and WenLi Du. MO-EMT-NAS: Multi-objective continuous transfer of architectural knowledge between tasks from different datasets. *arXiv preprint arXiv:2407.13122*, 2024. URL https://arxiv.org/abs/2407.13122.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32201–32223, 2024. URL https://proceedings.mlr.press/v235/liu24bs.html.

Shu Liu, Shubham Agarwal, Monishwaran Maheswaran, Mert Cemri, Zhifei Li, Qiuyang Mang, Ashwin Naren, Ethan Boneh, Audrey Cheng, Melissa Z. Pan, Alexander Du, Kurt Keutzer, Alvin Cheung, Alexandros G. Dimakis, Koushik Sen, Matei Zaharia, and Ion Stoica. EvoX: Meta-evolution for automated discovery. *arXiv preprint arXiv:2602.23413*, 2026a. URL https://arxiv.org/abs/2602.23413.

Shu Liu, Mert Cemri, Shubham Agarwal, Alexander Krentsel, Ashwin Naren, Qiuyang Mang, Zhifei Li, Akshat Gupta, Monishwaran Maheswaran, Audrey Cheng, Melissa Pan, Ethan Boneh, Kannan Ramchandran, Koushik Sen, Matei Zaharia, Alexandros G. Dimakis, and Ion Stoica. SkyDiscover: A flexible, adaptive framework for AI-driven scientific and algorithmic discovery. In *Proceedings of the ACM Conference on AI and Agentic Systems*, pp. 1223–1227, 2026b. doi: 10.1145/3786335.3813221. URL https://doi.org/10.1145/3786335.3813221.

Ximeng Liu, Qianlong Wang, Yingming Mao, Annan Li, Yatao Li, Shizhen Zhao, Jianmin Wu, Dawei Yin, and Dou Shen. Janus: An algorithm–evaluator co-evolution framework for LLM-driven discovery under expensive evaluation budgets. *arXiv preprint arXiv:2608.08189*, 2026c. URL https://arxiv.org/abs/2608.08189.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. FineWeb-Edu: The finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. DeepCoder: A fully open-source 14b coder at o3-mini level. Technical blog and dataset, 2025. URL https://huggingface.co/datasets/agentica-org/DeepCoder-Preview-Dataset.

Bohan Lyu, Yucheng Yang, Siqiao Huang, Jiaru Zhang, Qixin Xu, Xinghan Li, Xinyang Han, Yicheng Zhang, Huaqing Zhang, Runhan Huang, Kaicheng Yang, Zitao Chen, Wentao Guo, Junlin Yang, Xinyue Ai, Wenhao Chai, Yadi Cao, Ziran Yang, Kun Wang, Dapeng Jiang, Huan-ang Gao, Shange Tang, Chengshuai Shi, Simon S. Du, Max Simchowitz, Jiantao Jiao, Dawn Song, and Chi Jin. MLS-Bench: A holistic and rigorous assessment of AI systems on building better AI. *arXiv preprint arXiv:2605.08678*, 2026. URL https://arxiv.org/abs/2605.08678.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018. URL https://arxiv.org/abs/1809.02789.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015. URL https://arxiv.org/abs/1504.04909.

Muhammad U. Nasir, Sam Earle, Julian Togelius, Steven James, and Christopher W. Cleghorn. LLMatic: Neural architecture search via large language models and quality diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1110–1118, 2024. doi: 10.1145/3638529.3654017. URL https://doi.org/10.1145/3638529.3654017.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. MLGym: A new framework and benchmark for advancing AI research agents. *arXiv preprint arXiv:2502.14499*, 2025. URL https://arxiv.org/abs/2502.14499.

Alexander Novikov, Ngan Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016. URL https://arxiv.org/abs/1606.06031.

Alberto Pepe, Chien-Yu Lin, Despoina Magka, Bilge Acun, Yannan Nellie Wu, Anton Protopopov, Carole-Jean Wu, and Yoram Bachrach. Agentic discovery of neural architectures: AIRA-Compose and AIRA-Design. *arXiv preprint arXiv:2605.15871*, 2026. URL https://arxiv.org/abs/2605.15871.

Aleksandr Poslavsky, Alexander D'yakonov, Yuriy Dorn, and Andrey Zimovnov. VK-LSVD: A large-scale industrial dataset for short-video recommendation. In *Proceedings of the ACM Web Conference 2026*, pp. 8657–8660, 2026. doi: 10.1145/3774904.3792933. URL https://doi.org/10.1145/3774904.3792933.

Changle Qu, Sunhao Dai, Ke Guo, Xiao Zhang, Liqin Zhao, Shijun Wang, Yannan Niu, Lantao Hu, Han Li, and Jun Xu. KuaiLive: A real-time interactive dataset for live streaming recommendation. *arXiv preprint arXiv:2508.05633*, 2025. URL https://arxiv.org/abs/2508.05633.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. PostTrainBench: Can LLM agents automate LLM post-training? *arXiv preprint arXiv:2603.08640*, 2026. URL https://arxiv.org/abs/2603.08640.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4780–4789, 2019. doi: 10.1609/aaai.v33i01.33014780. URL https://ojs.aaai.org/index.php/AAAI/article/view/4405.

Esteban Real, Chen Liang, David So, and Quoc Le. AutoML-Zero: Evolving machine learning algorithms from scratch. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8007–8019, 2020. URL https://proceedings.mlr.press/v119/real20a.html.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://www.nature.com/articles/s41586-023-06924-6.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019. URL https://arxiv.org/abs/1907.10641.

Samuel Schmidgall and Michael Moor. AgentRxiv: Towards collaborative autonomous research. *arXiv preprint arXiv:2503.18102*, 2025. URL https://arxiv.org/abs/2503.18102.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Asankhaya Sharma. OpenEvolve: An open-source evolutionary coding agent. Software, 2025. URL https://github.com/algorithmicsuperintelligence/openevolve.

Zhongxiang Sun, Zihua Si, Xiaoxue Zang, Dewei Leng, Yanan Niu, Yang Song, Xiao Zhang, and Jun Xu. KuaiSAR: A unified search and recommendation dataset. *arXiv preprint arXiv:2306.07705*, 2023. URL https://arxiv.org/abs/2306.07705.

Ilya Trofimov, Nikita Klyuchnikov, Mikhail Salnikov, Alexander Filippov, and Evgeny Burnaev. Multi-fidelity neural architecture search with knowledge distillation. *arXiv preprint arXiv:2006.08341*, 2020. URL https://arxiv.org/abs/2006.08341.

Luis Wiedmann, Orr Zohar, Amir Mahla, Xiaohan Wang, Rui Li, Thibaud Frere, Leandro von Werra, Aritra Roy Gosthipaty, and Andrés Marafioti. FineVision: Open data is all you need. *arXiv preprint arXiv:2510.17269*, 2025. URL https://arxiv.org/abs/2510.17269. Dataset release: https://huggingface.co/datasets/HuggingFaceM4/FineVisionMax.

Yunze Wu, Dayuan Fu, Weiye Si, Zhen Huang, Mohan Jiang, Keyu Li, Shijie Xia, Jie Sun, Tianze Xu, Xiangkun Hu, Pengrui Lu, Xiaojie Cai, Lyumanshan Ye, Wenhong Zhu, Yang Xiao, and Pengfei Liu. InnovatorBench: Evaluating agents' ability to conduct innovative LLM research. *arXiv preprint arXiv:2510.27598*, 2025. URL https://arxiv.org/abs/2510.27598.

Minghao Yan, Bo Peng, Benjamin Coleman, Ziqi Chen, Zhouhang Xie, Shuo Chen, Zhankui He, Noveen Sachdeva, Isabella Ye, Weili Wang, Chi Wang, Ed H. Chi, Fernando Pereira, Wang-Cheng Kang, Derek Zhiyuan Cheng, and Beidou Wang. PACEvolve: Enabling progress-aware consistent evolution. *arXiv preprint arXiv:2601.10657*, 2026. URL https://arxiv.org/abs/2601.10657.

Yufei Ye, Wei Guo, Hao Wang, Luankang Zhang, Heng Chang, Hong Zhu, Yuyang Ye, Yong Liu, Defu Lian, and Enhong Chen. FuXi-Linear: Unleashing the power of linear attention in long-term time-aware sequential recommendation. *arXiv preprint arXiv:2602.23671*, 2026. URL https://arxiv.org/abs/2602.23671.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019. URL https://arxiv.org/abs/1905.07830.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. *arXiv preprint arXiv:2303.15343*, 2023. URL https://arxiv.org/abs/2303.15343.

TingJie Zhang and HaiLin Liu. Multi-task neural architecture search using architecture embedding and transfer rank. *arXiv preprint arXiv:2504.00772*, 2025. URL https://arxiv.org/abs/2504.00772.

Xu Zhao, Ruibo Ma, Jiaqi Chen, Weiqi Zhao, Ping Yang, and Yao Hu. Multi-granularity distribution modeling for video watch time prediction via Exponential-Gaussian mixture network. *arXiv preprint arXiv:2508.12665*, 2025. URL https://arxiv.org/abs/2508.12665.

Deyao Zhu, Xin Zhou, Shengling Qin, et al. EdgeBench: Unveiling scaling laws of learning from real-world environments. *arXiv preprint arXiv:2607.05155*, 2026. URL https://arxiv.org/abs/2607.05155.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. URL https://arxiv.org/abs/1707.07012.

A RMD-BENCH SPECIFICATIONS AND EVALUATION PROTOCOLS

This appendix specifies the data, starting model, editable interface, and evaluation protocol for each task family in Section 3. Table 5 summarizes the five families; the following subsections give their protocols, baseline calibration, and discovery compute scales. Counts describe the processed inputs used in our experiments and can differ from the original public releases. Following Section 2, source instances host local search, intermediate instances provide Track B screening and feedback, and larger-scale or full-target instances assess selected designs. Each PE source search receives 100 iterations. Appendix F specifies the additional evaluations of selected designs beyond these discovery instances. For error and loss objectives, a harness that maximizes fitness receives the negative metric; reported values and promotion retain the original lower-is-better metric.

Table 5: **RMD-Bench task families and discovery spaces.** Each family provides five source instances. Track A adds a larger-scale evaluation instance; Track B adds two intermediate instances and a full target. Sizes refer to the reference implementations and processed corpora used here. Detailed task specifications follow.

Task family	Starting method	Discovery space	Source instances	Final-stage evaluation
Track A: joint discovery across related instances				
Ranking	FuXi-Linear (Ye et al., 2026)	Sequence representations; temporal and retention mechanisms	KuaiLive, KuaiRand-1K/Pure, KuaiRec, KuaiSAR	KuaiRand-27K; 146.82M interactions; ranking score ↑
Watch-time	EGMN (Zhao et al., 2025)	Distributional head; duration prediction	CIKM16, KuaiRec, WeChat, KuaiRand-Pure/1K	KuaiRand-27K; 23.99M examples; MAE (s) ↓
RL algorithm	Qwen2.5-Coder (Hui et al., 2024); GRPO	Policy loss; rollout filtering	Five sources, different seeds; 1.5B policy, 15 steps	3B policy, 50 steps; 423 LiveCodeBench problems; pass@1 ↑
Track B: decomposed discovery for expensive targets				
LLM pretraining	Gemma 3-style (Gemma Team, 2025); ∼185M	Architecture; optimizer; schedule; training mechanisms	FineWeb-Edu 100B core set; two 2% and three 5% views	Full core set; seven-task accuracy ↑
VLM pretraining	Qwen3.5-style (Qwen Team, 2026) + SigLIP; ∼646M trainable	Language architecture; projector; optimizer; schedule	FineVisionMax format; two 2% and three 5% views	2.38M raw training examples; validation loss ↓

A.1 SEQUENTIAL RANKING

Data and preprocessing. The source family comprises KuaiLive (Qu et al., 2025), KuaiRand-1K and KuaiRand-Pure (Gao et al., 2022b), KuaiRec (Gao et al., 2022a), and KuaiSAR (Sun et al., 2023). KuaiLive treats clicked streamers as items. KuaiSAR uses clicked interactions from recommendation pages. For each user or session, interactions are ordered chronologically; the last interaction is the next-item target, and preceding interactions form the history. KuaiRand ranking uses the processed standard-exposure logs. The benchmark evaluates full-catalog next-item ranking.

Starting model and training. The reference is FuXi-Linear (Ye et al., 2026), with positional and temporal channels and a causal chunkwise implementation (Table 7). User and item embeddings are L2-normalized and scored by dot product. Training uses autoregressive sampled softmax and AdamW. Baseline HPO selects dropout, learning rate, per-GPU batch size, and weight decay separately for each dataset (Appendix A.7). Table 6 gives the resulting training allocations, and Table 16 lists the selected hyperparameters.

Table 6: Processed ranking datasets and training allocation. All but KuaiRand-27K are source instances. Batch size is the per-GPU value selected by baseline HPO (Table 16).

Dataset	Users	Interactions	Items	Mean seq. length	Context length	Batch size per GPU	Epochs
KuaiLive	22,497	4,381,721	151,849	194.77	512	47	30
KuaiRand-1K	988	290,969	2,025	294.50	1,024	38	30
KuaiRand-Pure	25,220	1,235,423	6,073	48.99	512	88	30
KuaiRec	7,176	12,529,113	9,958	1,745.97	1,024	40	50
KuaiSAR	17,491	1,595,760	55,505	91.23	512	46	50
KuaiRand-27K	27,285	146,821,616	461,994	5,381.04	1,024	51	50

Table 7: Fixed FuXi-Linear reference configuration.

Component	Setting
Encoder	Four blocks; four attention heads; $d_q = d_k = d_v = 32$
Embeddings and position	Item dimension 128; positional linear channel dimension 32
Temporal channel	Eight heads; base 2, stride 3, start index 10; learnable decay
Sequence computation	Chunk size 128; RoPE disabled
Sampled-softmax loss	128 negatives; temperature 0.05
Execution	Eight-GPU DDP; BF16/TF32

Editable interface. The proposer can jointly modify three components:

- FuXi-Linear blocks, channels, and the user representation.
- Retention, temporal interactions, normalization, and gating.
- The causal chunkwise attention kernel.

The input/output interfaces and the training path from user representations to sampled-softmax loss remain intact. Data loading, loss, sampling, and evaluation are fixed. KuaiRand-1K additionally uses the per-candidate HPO procedure in Appendix A.7; the other four sources use fixed optimizer settings.

Evaluation and promotion. The ranking score used as evolutionary fitness is

$$R_{\text{rank}} = (\text{NDCG@10} + \text{HR@10} + \text{MRR})/3.$$

Published interventions are retrained on the four peer datasets, and candidates meeting the peer criterion advance to KuaiRand-27K for larger-scale evaluation. Table 20 specifies publication, peer, and larger-scale success thresholds; standalone top-k candidates use the same larger-scale evaluator and success criterion.

A.2 Watch-time prediction

Data and preprocessing. The source datasets are CIKM16 (CIKM Cup), KuaiRec (Gao et al., 2022a), WeChat, KuaiRand-Pure, and KuaiRand-1K (Gao et al., 2022b), with KuaiRand-27K as the larger-scale evaluation instance. Following EGMN (Zhao et al., 2025), we include CIKM16 as a related engagement-duration task: its labels describe time spent in online search sessions. Training uses a fixed deterministic 80% partition; the remaining 20% supplies the evaluation scores used throughout discovery. Within each dataset, all methods share a p99 label scale estimated from the processed corpus before partitioning. KuaiRand watch-time preprocessing combines random- and standard-exposure logs. KuaiRec and KuaiRand-27K use sampled processed corpora. Table 8 reports processed examples rather than original-release interactions.

Starting model and training. EGMN (Zhao et al., 2025) uses 16-dimensional sparse and sequence embeddings, mean-pooled sequence features, linear projections of continuous features, and a 256–128–64 MLP encoder. Its starting head combines one exponential component for fast skips and ten

Table 8: Processed watch-time datasets. The user/query vocabulary counts queries for CIKM16 and users otherwise; the p99 scale converts normalized MAE to seconds. These counts reflect watch-time preprocessing and differ from ranking.

Dataset	Train	Eval.	Total	User/query vocabulary	Item vocabulary	p99 scale (s)
CIKM16	246,853	61,714	308,567	9,837	122,642	5.237
KuaiRec	980,676	245,169	1,225,845	7,176	9,599	47.697
WeChat	5,059,605	1,264,902	6,324,507	19,994	95,511	165.364
KuaiRand-Pure	1,871,608	467,902	2,339,510	27,279	7,268	182.240
KuaiRand-1K	6,951,281	1,737,821	8,689,102	1,000	2,926,199	188.515
KuaiRand-27K	19,193,265	4,798,317	23,991,582	27,285	5,489,023	187.368

non-negative left-truncated Gaussian components. The point prediction is the mixture expectation. The starting model's objective combines the mixture negative log-likelihood with entropy and absolute-error terms weighted by α and β:

$$\mathcal{L} = \mathcal{L}_{\text{mixture NLL}} + \alpha\mathcal{L}_{\text{entropy}} + \beta|\hat{y} - y|.$$

Runs use Adagrad, 10 epochs, batch size 2,048, and a fixed training seed. The S and PE runs with GPT-5.5 share the HPO-selected configurations in Table 9, with per-candidate HPO disabled; Appendix A.7 specifies calibration.

Table 9: Fixed HPO-selected watch-time hyperparameters shared by the S and PE runs with GPT-5.5.

Dataset	Learning rate	Weight decay	α	β	Dropout
CIKM16	0.091684	3.35×10^{-6}	0.659274	2.023865	0.141097
KuaiRec	0.25076	1.94×10^{-5}	0.619568	3.41108	0.059406
WeChat	0.085141	1.98×10^{-6}	0.454659	3.263077	0.055439
KuaiRand-Pure	0.064052	8.58×10^{-7}	0.521568	4.296266	0.077754
KuaiRand-1K	0.134232	2.31×10^{-5}	0.646871	1.511465	0.362471
KuaiRand-27K	0.102368	1.53×10^{-5}	0.489522	1.778713	0.138667

Editable interface and fitness. Only the watch-time prediction head is evolvable. Its search space includes likelihood parameterization, mixture components, point prediction, and duration conditioning. All candidates share the feature encoder, data pipeline, optimizer settings, epoch count, batch size, and evaluator. The evaluation metric is MAE (lower is better), reported in seconds on this partition using the fixed p99 label scale in Table 8. Candidates advance through peer and larger-scale evaluation using the MAE-reduction criteria in Table 20.

A.3 RL ALGORITHM DISCOVERY

Source and larger-scale instances. The five source instances differ only in the training random seed. Each uses Qwen2.5-Coder-1.5B-Instruct (Hui et al., 2024), 15 GRPO steps, and the same 408-problem DeepCoder Codeforces/test partition (Luo et al., 2025). The larger-scale instance uses Qwen2.5-Coder-3B-Instruct, 50 steps, and a disjoint 423-problem LiveCodeBench partition (Jain et al., 2024). This partition is a deterministic platform-by-difficulty stratified selection from a 1,055-problem snapshot. Table 10 summarizes the source and larger-scale training and evaluation settings.

The RL training pool has 16,252 DeepCoder `primeintellect/train` examples and 7,436 `taco/train` examples, totaling 23,688 problems. After normalizing prompt text, we find no exact overlap between the training and evaluation prompts and retain all training examples. Every candidate starts from the same pretrained policy weights, without inheriting trained weights from earlier search iterations.

Training and search space. The RL protocol disables HPO and fixes Adam with learning rate 10^{-6}, a low-variance KL estimator with coefficient 0.001, and clipping coefficients 0.2/0.28.

Table 10: Fixed RL training and evaluation contracts. Both training scales use eight responses per prompt and 16 H200 GPUs per candidate evaluation.

Property	Source (each of five instances)	Larger scale
Policy parameters	1.5B	3B
GRPO steps	15	50
Prompts per step	64	32
Evaluation problems	408	423
Evaluation tests	Public tests	Private tests
Training seed	Instance-specific	Fixed

Source instances use tensor parallelism 2 and maximum response length 4,000. The larger-scale partition contains 241 AtCoder, four Codeforces, and 178 LeetCode problems.

Editable interface. Evolution can modify the GRPO loss through changes to surrogate-loss geometry, clipping treatment, KL treatment, and advantage weighting, and can modify rollout filtering through the selection, weighting, or rejection of rollout groups. It cannot change the model, training corpus, learning rate, fixed clipping/KL coefficients, RL horizon, rollout batch, response count, or evaluator.

Two reward levels. Within candidate training, a sampled program receives the fraction of unit tests passed, with a -0.1 penalty for unparseable or truncated output. The outer evolutionary fitness is code-generation pass@1 under deterministic decoding: the number of solved evaluation problems divided by the partition size. Evaluation generates one completion per problem, with the evaluator's test cases excluded from the generation prompt. The larger-scale evaluator uses hidden private tests. The fixed reference score is the arithmetic mean of three valid baseline repetitions.

Promotion and reported outcomes. Table 20 gives the joint relative and absolute improvement criteria for publication, peer validation, and larger-scale success. The larger-scale evaluation measures transfer jointly across policy size, training horizon, and code benchmark. The five RL sources share the same task, training data, and objective and differ only in random seed. We therefore treat them as parallel searches for the strongest algorithm and report the best outcome across the five searches.

A.4 LLM PRETRAINING

Data hierarchy. The benchmark uses the FineWeb-Edu 100B core set from the shuffled release (Lozhkov et al., 2024; Karpathy, 2025). Table 11 gives the exact corpus inventories and training allocations. Source and intermediate subsets use fixed contiguous shard ranges from the shuffled corpus. The three 5% sources D5-A/B/C are mutually disjoint; D2-A/B are 2% subsets nested within D5-A/B, respectively. D10 clean is disjoint from every source slice. D10+C combines D5-C with a fresh disjoint 5% slice, testing an increase in data around a known source. The full-target instance trains on the complete reference corpus.

Table 11: LLM training hierarchy. Token counts describe the processed corpus; step counts scale approximately with the nominal corpus fractions. All instances use the same seven-task mean accuracy.

Dataset	Role	Shards	Documents	Tokens	Steps
D2-A	Source	36	1,924,096	1,925,811,698	694
D2-B	Source	36	1,915,904	1,921,710,324	694
D5-A	Source	91	4,853,760	4,862,315,015	1,750
D5-B	Source	91	4,844,544	4,859,363,728	1,750
D5-C	Source	91	4,853,760	4,863,973,228	1,750
D10 clean	Intermediate	182	9,712,640	9,727,423,143	3,470
D10+C	Intermediate	182	9,711,616	9,728,203,396	3,470
Full target	Target	1,823	97,230,848	97,419,200,814	34,700

Model and training. The starting model is an approximately 185M-parameter decoder with a Gemma 3-style architecture (Gemma Team, 2025): RMSNorm, SwiGLU, RoPE, and hybrid local/global attention. The model uses the Llama 3.1 tokenizer and a context length of 1,024. Training uses eight H200 GPUs, a micro-batch size of 352, and no gradient accumulation, giving approximately 2.88M tokens per step. Each candidate is pretrained from scratch. The training-step count scales with data volume to approximate one pass through each subset; baseline hyperparameters are specified per instance in Table 17.

Editable interface. The proposer can modify four components:

- Model architecture, including attention, normalization, and feed-forward computation.
- Muon optimizer geometry (Jordan et al., 2024), including Newton–Schulz updates, clipping, and parameter grouping.
- The Warmup–Stable–Decay learning-rate schedule.
- Training mechanisms within the fixed data and compute budget.

Dataset assignment, tokenizer, sequence length, batch size, total steps, checkpoint isolation, and evaluation are protected.

Evaluator. We use lm-evaluation-harness (Gao et al., 2023) in zero-shot mode. ARC-Challenge and ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and OpenBookQA (Mihaylov et al., 2018) use normalized accuracy. PIQA (Bisk et al., 2019), WinoGrande (Sakaguchi et al., 2019), and the OpenAI variant of LAMBADA (Paperno et al., 2016) use standard accuracy. The evaluator loads the final checkpoint and takes the unweighted mean of these seven scores. All seven evaluations must return successfully, and the implementation must pass the causal-leakage check, for a run to be valid. Decomposed PE excludes full-target outcomes from adaptive proposal feedback; direct full-target evolution uses those outcomes during search.

Publication and promotion. Published candidates must pass peer screening and both intermediate evaluations before full-target training. Table 20 specifies the improvement thresholds at each stage and the full-target graduation criterion.

A.5 VLM PRETRAINING

Data hierarchy. The benchmark uses a 1,000-shard image–conversation corpus in FineVisionMax format (Wiedmann et al., 2025). Shards 980–999 define the validation split; the full target uses the remaining 980 training shards. Table 12 gives the source, intermediate, and full-target inventories.

Table 12: VLM training hierarchy. Rows report the raw training-data inventory and prescribed training steps.

Dataset	Role	Shards	Shard indices	Raw rows	Steps
D2-A	Source	20	0–19	48,500	48
D2-B	Source	20	50–69	48,500	48
D5-A	Source	50	0–49	121,250	118
D5-B	Source	50	50–99	121,250	118
D5-C	Source	50	100–149	121,250	118
D10 clean	Intermediate	100	150–249	242,500	237
D10+C	Intermediate	100	100–149; 250–299	242,500	237
Full target	Target	980	0–979	2,376,500	2,320

Model and training. The language backbone is a custom 28-layer hybrid decoder with a Qwen3.5-style architecture (Qwen Team, 2026), including full grouped-query attention and causal-convolutional linear attention, RMSNorm, SwiGLU, and RoPE. A frozen SigLIP-so400m-patch14-384 encoder (Zhai et al., 2023) supplies 729 image-token positions per image; a trainable two-layer MLP projects these features into the language model. The model has approximately 641.6M parameters in the language backbone, 646.1M trainable parameters including the projector, and 1.074B total parameters including the frozen vision encoder.

Training uses eight H200 GPUs, a micro-batch size of 8, gradient accumulation over 16 steps, a sequence length of 2,048, and BF16 precision. Step counts scale with subset size to approximate one training epoch, ranging from 48 to 2,320. Each candidate trains the language backbone and projector from scratch with the frozen vision encoder; evolved candidates do not inherit learned weights from earlier iterations.

Editable interface. The proposer can modify three components:

- The language backbone's attention, normalization, RoPE, and SwiGLU blocks, and the MLP projector.
- Optimizer updates, orthogonalization, Q/K clipping, and parameter grouping.
- The Warmup–Stable–Decay learning-rate schedule.

The frozen SigLIP encoder, projector width, image-token interface, outer fusion/data path, data assignment, sequence length, batch and accumulation, steps, and evaluator remain fixed.

Evaluation metric and validity. The evaluator measures token-weighted next-token cross-entropy on validation data disjoint from training. Before scoring, the evaluator verifies the frozen vision encoder, parameter budget, and causal validity. The shifted next-token objective predicts position $t+1$ from position t; masked positions are excluded from the supervised loss. The reported loss sums cross-entropy over these supervised tokens and divides by their total count, rather than averaging example losses.

Graduation. Candidates advance after peer screening and must pass both intermediate evaluations before full-target training. Table 20 specifies the relative and absolute loss-reduction criteria at each stage. Decomposed PE withholds full-target outcomes from adaptive search feedback.

A.6 Shared task guidance

Preliminary experiments identify recurring implementation errors, which we summarize in shared domain guidance. All compared discovery methods receive the same guidance for a given task family in every LLM call. Table 13 summarizes the guidance; Appendix B.3 shows how it accompanies task instructions and shared evidence.

Table 13: Shared domain guidance for valid candidate designs. These instructions complement the fixed interfaces and evaluators.

Domain	Scientific and computational constraints
Ranking	Avoid attention whose cost is quadratic in the full sequence length.
Watch-time	Preserve non-negative distributional support and stable likelihoods; treat CIKM16 duration as session context.
LLM	Distinguish vector and matrix parameters in Muon and maintain stable optimizer updates.
VLM	Preserve causal attention and the fixed visual interface.
RL	Preserve response groups and masks, and compute stable probability ratios.

A.7 Baseline calibration and per-candidate hyperparameter optimization

Baseline calibration before discovery. Baseline HPO tunes the unmodified starting implementation separately for each instance before evolutionary search. Ranking, watch-time, and LLM/VLM source and intermediate references use CMA-ES sampling in Optuna (Akiba et al., 2019), with LLM-proposed configurations added after warm-up when the trial budget permits. Results from all evaluated configurations update the CMA-ES search. Tables 14 and 15 specify the calibration settings, trial caps, and search spaces. The starting code, model capacity, data, evaluator, and prescribed epochs or step count remain fixed during calibration; ranking additionally searches the per-GPU batch size. The selected configuration and measured reference score are then fixed and shared by

competing discovery procedures in that instance. RL disables baseline HPO and uses the mean of three independent fixed-configuration runs in each source and larger-scale instance. LLM and VLM full targets each use one measured reference under a prescribed configuration shared by PE and Direct. The cost of full-target training precludes a separate full-target HPO sweep within the study budget; both methods use the same full-target reference and evaluation protocol.

Table 14: Baseline HPO trial caps, applied separately to each instance. Calibration also evaluates the original configuration and confirms the selected configuration. Early stopping uses a patience of ten trials without improvement.

Family	Instances	Trial cap N
Ranking	Five sources and KuaiRand-27K	50
Watch-time	Five sources and KuaiRand-27K	50
LLM pretraining	Two 2% sources and two intermediate instances	20
LLM pretraining	Three 5% sources	10
VLM pretraining	Five sources and two intermediate instances	20

CMA-ES supplies the first ten warm-up trials; approximately 30% of subsequent proposals come from the LLM when trials remain.

Table 15: Baseline HPO search spaces. "Log" denotes a logarithmic search scale; "log integer" additionally restricts values to integers. Unmarked intervals use a linear scale. LLM/VLM full-target configurations are fixed without HPO, and RL has no HPO search space.

Family / tier	Parameters and ranges
Ranking	Dropout $[0.1, 0.6]$; learning rate $[10^{-4}, 10^{-2}]$ (log); per-GPU batch size $[32, 128]$ (log integer); weight decay $[10^{-5}, 10^{-2}]$ (log).
Watch-time	Learning rate $[0.01, 0.3]$ (log); weight decay $[10^{-7}, 10^{-3}]$ (log); $\alpha \in [0, 1]$; $\beta \in [0.1, 5]$; dropout $[0, 0.5]$.
LLM: source, intermediate	Peak learning rate $[5 \times 10^{-5}, 10^{-3}]$ (log); weight decay $[0.01, 0.5]$ (log); warm-up steps $[50, 300]$ (log integer); decay fraction $[0.05, 0.3]$; Muon momentum $[0.9, 0.99]$; Q/K clipping threshold $[50, 200]$ (log).
VLM: source, intermediate	Same ranges as LLM except warm-up steps: $[2, 12]$ for 2% sources, $[5, 30]$ for 5% sources, and $[10, 60]$ for 10% intermediate instances (log integer).

Selected configurations. Table 16 gives the selected ranking hyperparameters, and Table 9 lists the six watch-time configurations. Table 17 gives the LLM/VLM source, intermediate, and full-target configurations. The fixed RL configuration is specified in Appendix A.3.

Table 16: Selected ranking baseline hyperparameters. Batch size is per GPU. Learning rate and weight decay use the column scale factors.

Dataset	Dropout	Learning rate ($\times 10^{-3}$)	Batch size	Weight decay ($\times 10^{-4}$)
KuaiLive	0.49	2.5	47	4.8
KuaiRand-1K	0.352	3	38	1.1
KuaiRand-Pure	0.28727	7.969455	88	6.251374
KuaiRec	0.3	4.3	40	14
KuaiSAR	0.53	5.2	46	9
KuaiRand-27K	0.378871	3.548962	51	1.985122

Per-candidate HPO in the reported configuration. Per-candidate HPO adapts training hyperparameters for a proposed intervention during discovery and is separate from baseline calibration. Per-candidate HPO is enabled only for KuaiRand-1K ranking, using the same procedure for PE and standalone evolution. We enable it on this instance because performance is particularly sensitive to

Table 17: Fixed LLM/VLM baseline hyperparameters. Learning rates use the indicated scale factor. Columns give peak learning rate η, weight decay λ, warm-up steps n_{warm}, decay fraction f_{decay}, Muon momentum μ, and Q/K clipping threshold τ_{QK}. Full-target rows are prescribed configurations without HPO.

Instance	η ($\times 10^{-4}$)	λ	n_{warm}	f_{decay}	μ	τ_{QK}
LLM pretraining						
D2-A	4.480393	0.018409	66	0.143635	0.985564	114.656
D2-B	5.4	0.05	170	0.085	0.919	170
D5-A	4.72	0.022	78	0.108	0.982	121
D5-B	4.480393	0.018409	66	0.143635	0.985564	114.656
D5-C	4.480393	0.018409	66	0.143635	0.985564	114.656
D10 clean	4.480393	0.018409	66	0.143635	0.985564	114.656
D10+C	5.2	0.055	172	0.118	0.9235	156
Full target	2	0.1	500	0.1	0.95	100
VLM pretraining						
D2-A	7.8	0.075	6	0.248	0.9165	168
D2-B	9.085181	0.114728	5	0.172398	0.913847	149.21
D5-A	7.2	0.035	20	0.218	0.918	185
D5-B	9	0.12	15	0.185	0.922	135
D5-C	4.55	0.073	18	0.203	0.929	112
D10 clean	5.178996	0.020461	27	0.164448	0.906785	119.578
D10+C	5.6	0.055	32	0.205	0.921	158
Full target	1	0.1	100	0.1	0.95	100

training hyperparameters and the small dataset makes repeated tuning affordable. Table 18 records the reported GPT-5.5 settings, including the extra trials and optional paired baseline control.

Table 18: Per-candidate HPO in the reported GPT-5.5 configuration. Counts cover one local mutation before any separate peer or promotion evaluations. Disabled settings execute neither per-candidate HPO trials nor paired baseline controls.

Family / sources	Enabled	Extra trials	Paired control	Total evaluations
Ranking / KuaiRand-1K	Yes	3	Up to 1	Up to 5
Ranking / other four sources	No	0	0	1
Watch-time	No	0	0	1
RL algorithm	No	0	0	1
LLM pretraining	No	0	0	1
VLM pretraining	No	0	0	1

Paired baseline control and selection. After a successful main evaluation, per-candidate HPO starts from the receiving instance's calibrated baseline hyperparameters. When a paired control is used, it evaluates the unmodified code under the candidate's selected hyperparameters, yielding $\widetilde{b}_e(h)$ for candidate h on instance e. This control measures the code intervention's effect at the candidate's selected hyperparameters. Source publication retains the ordinary source rule in Table 20 and, when a valid paired control exists, additionally requires the candidate to exceed that control by the prescribed relative improvement margin. Cross-instance selection uses the stronger of the fixed reference and any valid paired control:

$$s_e b_e^{\mathrm{gate}}(h) = \max\{s_e b_e,\, s_e \widetilde{b}_e(h)\}. \tag{3}$$

When no valid paired control is available, we set $b_e^{\mathrm{gate}}(h) = b_e$. Thus $b_e^{\mathrm{gate}}(h)$ is the larger reference for reward/accuracy and the smaller reference for error/loss. Reported gains still use the fixed reference b_e (Equation 2); cross-instance selection uses $b_e^{\mathrm{gate}}(h)$.

A.8 Discovery compute and search depth

We report compute for a complete discovery run, comprising five parallel searches and their prescribed evaluations. Due to policy restrictions, we cannot disclose exact GPU-hour usage; Table 19 reports the permitted orders of magnitude. LLM and VLM pretraining require thousands of GPU-hours per complete run, ranking and RL algorithm discovery require hundreds, and watch-time prediction requires tens. The benchmark and search code will be made publicly available to support reproduction under the stated protocols.

Table 19: **Compute scale of a complete discovery run.** Each run comprises five searches and their prescribed evaluations. Entries give orders of magnitude for total allocated GPU-hours; exact usage is not disclosed due to policy restrictions.

Task family	GPU-hours per complete discovery run
Ranking	Hundreds
Watch-time prediction	Tens
RL algorithm discovery	Hundreds
LLM pretraining	Thousands
VLM pretraining	Thousands

PE uses smaller training subsets for ongoing proposal–evaluation cycles and reserves full-target training for candidates that pass the peer and intermediate gates. Appendix B.5 specifies the completed-run comparison and accounts for all executed search and evaluation stages.

B Population Evolution Implementation and Reporting

This appendix connects the search procedure in Section 2 to its implementation. We specify publication and promotion decisions, the evidence supplied to each source search, and asynchronous execution, then define the units used to report compute and outcomes.

B.1 Publication and promotion thresholds

PE tests publication and promotion margins against the reference score of the instance being evaluated. Based on preliminary studies, we set domain-specific margins to require challenging but attainable improvements under the prescribed training allocations and fix them for the reported comparisons. These margins define operational promotion and success criteria; Tables 2 and 4 also report continuous performance gains. Source publication requires the configured gain over the fixed reference and, when a valid paired control is available, the additional gain over that control (Appendix A.7). Cross-instance selection uses the gating reference $b_e^{\mathrm{gate}}(h)$ defined in Equation 3. With s_e defined in Equation 2, the directed absolute gain is $a_e^{\mathrm{gate}}(h) = s_e\big(y_e(h) - b_e^{\mathrm{gate}}(h)\big)$ and the relative gain is $r_e^{\mathrm{gate}}(h) = a_e^{\mathrm{gate}}(h)/\max(|b_e^{\mathrm{gate}}(h)|, \epsilon)$. An evaluation passes only when the run is valid, the score is finite, and every configured relative and absolute margin is met. For an intervention h originating at E_i, let ρ_x and α_x denote the configured relative and absolute peer margins, and let K_x be the required number of passing peers. Counting only completed valid peer evaluations, the selector implements

$$\mathrm{eligible}(h) = \mathbf{1}\left[\sum_{E_j \in \mathcal{D}\setminus\{E_i\}} \mathbf{1}\big[r_j^{\mathrm{gate}}(h) \geq \rho_x \ \wedge \ a_j^{\mathrm{gate}}(h) \geq \alpha_x\big] \geq K_x\right]. \tag{4}$$

An omitted margin imposes no constraint; in our experiments, $K_x = 2$ for every domain. Scores are not averaged across instances before selection. An eligible candidate proceeds to the larger-scale instance in Track A or to the two intermediate instances in Track B. Both intermediate instances must pass before full-target training. Graduation then requires the Track A larger-scale or Track B full-target margin. Table 20 specifies the margins used by PE; the benchmark's training and evaluation instances can also support other selection policies.

Table 20: PE's publication and promotion margins. Relative gains are direction-normalized; where an absolute margin is listed, both conditions apply. Every peer gate requires at least two wins among four peers.

Family	Source publication	Peer evaluation	Larger scale / intermediate	Full target
Ranking	2% relative	1% relative	2% relative	—
Watch-time	0.5% relative	0.5% relative	0.5% relative	—
RL algorithm	10% rel.; $1.5/408$ abs.	5% rel.; $0.5/408$ abs.	4% rel.; $4/423$ abs.	—
LLM	2% relative	1% relative	2% on both intermediates	1% relative
VLM	10% relative	10% rel.; 0.02 abs.	6% rel.; 0.02 abs., both intermediates	4% rel.; 0.015 abs.

Graduation requires meeting the larger-scale criteria in Track A, or the full-target criteria after both intermediate evaluations pass in Track B.

B.2 Shared candidate records and search context

The shared ledger connects candidate evaluation to ongoing local search. A source publishes a promising intervention, peers evaluate it, and the accumulated results enter later proposal prompts (Figure 1(a)). The following describes this information flow; Appendix B.3 shows how it enters proposal prompts and candidate construction.

Publishing a candidate. After a candidate meets its source instance's publication margin (Appendix B.1), PE creates a shared record containing the executable intervention and a short description of its mechanism. The record identifies the source instance and iteration, reports its score and reference baseline, and tracks the candidate's evaluation results and current status. The record keeps executable code distinct from the mechanism description: peer evaluation executes the code, while proposal context summarizes the mechanism and its measured outcomes.

Evaluating the intervention on peers. For each peer, the coordinator restores that instance's reference implementation, applies the published intervention, and runs the peer's prescribed training and evaluation. Model parameters are trained separately under the receiving instance's protocol. This *cross-instance replay* measures where the published intervention remains useful. Each completed evaluation adds the receiving instance, score, gain, and pass decision to the candidate's record. Valid weak or near-neutral results remain available even when they do not meet a promotion margin. The completed peer results determine eligibility for larger-scale or intermediate evaluation under the criteria in Appendix B.1.

Building context for the next local proposal. Before each LLM call on source instance E_i, PE selects relevant records from the ledger to form the shared context $\Phi_i(M_t)$ in Equation 1. This context contains three groups:

- **Own candidates:** at most ten candidates published by E_i, with available peer and larger-scale or intermediate results.
- **Successful peer transfers:** at most six candidates from other source instances that succeeded on E_i, ranked by their measured benefit on E_i.
- **Weak or unsuccessful peer transfers:** at most six candidates from other source instances that failed or transferred weakly on E_i.

Each selected entry summarizes the mechanism, originating instance, evaluation status, and measured effects; all entries together are limited to 4K tokens. Gains are direction-normalized as in Equation 2, so positive values indicate improvement for both accuracy and loss metrics. The local harness retains its own program archive, while the proposer uses the shared evidence to reuse, modify, or combine mechanisms in later candidates. This separates two uses of shared records: replay tests an existing intervention, while proposal context helps generate the next one.

B.3 PROMPTS AND LOCAL CODE EDITS

Each source instance uses a local search harness to select a parent program and ask an LLM for a code edit. The main experiments use the AdaEvolve-based implementation in SkyDiscover (Liu et al., 2026b; Cemri et al., 2026). At each proposal step, the prompt combines the local program and search history with the shared context described in Appendix B.2.

Inputs to the proposer. The template combines fixed task instructions and shared domain guidance with the selected parent, local history, and latest available shared evidence.

```
SYSTEM
  Objective, dataset, metric, and fixed training budget
  Editable regions, interfaces, and fixed controls
  Shared context: own candidates and their transfer results
                  peer candidates evaluated on this instance
  Domain guidance and harness tool instructions

USER
  Selected parent code and metrics
  Previous attempts and evaluator feedback
  Program examples sampled by the harness
  Search objective and edit requirements
```

Example of the shared context. Consider a proposal on VLM source instance D2-A. Its shared context can include the candidate discovered on D5-C at iteration 55 and subsequently evaluated on D2-A. The intervention replaces global Muon gradient clipping with clipping based on the final update's RMS, yielding relative gains of approximately 10.50% on D5-C and 10.34% on D2-A.

```
Candidate from a peer instance
  Source: VLM D5-C, iteration 55
  Mechanism: final-update RMS trust clipping for Muon
  Source gain: +10.50%; gain on VLM D2-A: +10.34%
  Receiving-instance decision: pass
  Full-target outcomes: withheld from proposal context
```

In this Track B example, the proposer receives peer and intermediate evidence but no full-target scores or pass/fail outcomes (Figure 1(c)). The task objectives, editable components, and fixed experimental controls follow Appendices A.1–A.5.

Output: a local edit to the selected parent. The LLM edits the selected parent within the permitted components, preserving the specified interfaces and fixed experimental controls. The candidate is then trained and evaluated on its source instance. The evaluation updates the local history and archive; candidates meeting the publication margin enter the shared ledger for cross-instance replay (Appendix B.2).

B.4 ASYNCHRONOUS DISCOVERY LOOP

Algorithm 1 summarizes the population operations. Local searches and evaluation jobs proceed asynchronously. Each source harness refreshes its shared-evidence context before every LLM call. Iteration numbers are local to each source; timestamps determine which shared results are available to a proposal.

B.5 EXECUTION ACCOUNTING AND EVIDENCE UNITS

Iterations, evaluations, and complete runs. A source iteration is one local proposal–evaluation cycle. A complete discovery run comprises five source searches for PE or five independent searches for S/Direct, together with their prescribed transfer and target evaluations. Discovery compute includes local search, peer replay, intermediate screening, and final-stage evaluation. Per-candidate HPO trials, paired controls, retries, failed attempts, and re-evaluations are charged to discovery compute. Shared baseline calibration is performed before search and accounted for separately.

Algorithm 1 Population Evolution: event-driven orchestration

Require: Source instances $\mathcal{D}$; local budgets; baselines; selector $\mathcal{S}$; track
Initialize separate local states τ_i, shared ledger M, and evaluation queues
while local proposals or queued evaluations remain within the run allocation **do**
    if the local search on source instance E_i is ready for an iteration **then**
        Run its proposal harness with τ_i and a per-call refresh of $\Phi_i(M)$
        Obtain candidate h and enqueue training and evaluation on E_i
    end if
    if a local evaluation completes **then**
        Update local history and code archive; publish if the source criterion is met
        If published, queue evaluation of the same code change on peer instances
    end if
    if a peer, intermediate, or Track A larger-scale evaluation completes **then**
        Record the job's outcome and pass flag; recompute eligibility using $\mathcal{S}$
        If eligible, queue any unscheduled larger-scale (Track A) or intermediate (Track B) evaluations
        In Track B, queue full-target training once, after both intermediate gates pass
    end if
    if a full-target evaluation completes in Track B **then**
        Record the full-target outcome and terminal state in M; exclude both from all proposal contexts
    end if
end while

Comparison budgets. Track A matches source iterations; Track B compares completed PE and Direct runs under matched total GPU compute across all stages. For Track A larger-scale evaluation, S selects the top $k = \lceil N_{\mathrm{PE}}/5 \rceil$ candidates by source score from each of its five searches, where N_{PE} is the number of PE larger-scale candidate evaluations. Rounding up gives S at least as many larger-scale candidate evaluations as PE. Allocated compute sums GPU count multiplied by training–evaluation runtime across all jobs. Exact GPU-hour usage is restricted by policy; Appendix A.8 reports the compute scales and code-release commitment. Table 4 and both panels of Figure 6 summarize the completed runs; the LLM runs have 39 full-target attempts for PE and 70 candidate-mutation attempts for Direct.

Outcome summaries. For Track A, ranking and watch-time average each source's best local gain across distinct datasets, whereas RL reports the best gain across five parallel searches of the same task (Table 21). Peer-evaluation results and larger-scale outcomes are summarized separately. Track B reports the best full-target gain and the number of successful candidates from each completed run, using the criteria in Table 20. Comparisons and relative gains are computed before display rounding. Reported gains always use the fixed reference in Equation 2; paired controls can strengthen selection without redefining this reference. Appendix B.6 gives raw attempt counts and passing means.

Replication. Given the substantial training cost of complete discovery runs, we report one complete run per method–configuration and allocate the available compute to comparisons across task families and model proposers. The five searches are components of one complete run; repeated baseline evaluations estimate the reference score rather than variation across discovery runs. For the configurations examined in preliminary independent runs, the relative performance ordering was consistent across runs.

B.6 Full-target outcome details

Each candidate contributes at most one success, regardless of repeated evaluations.

LLM outcomes. Over its completed run, PE evaluates 39 candidates in 39 full-target attempts; 34 return numeric scores and five return no score. Thirteen meet the success criterion, with a mean gain of 1.67% among the successful candidates. Attempts without a score remain in the attempt count and do not contribute a success. Direct produces no successful candidates from 70 candidate-mutation attempts; this count excludes the five initialization evaluations. Both methods' outcomes are reported after their complete discovery runs under matched total GPU compute (Appendix B.5). Appendix E.3 varies the full-target success threshold using these same results.

VLM outcomes. Direct completes 100 attempts across five searches, with 96 finite scores and four failed attempts. Seventeen evaluations meet the full-target criterion, corresponding to 12 successful candidates. Relative to the fixed baseline loss 1.2071, the best loss is 1.1461 (5.05% reduction) and the passing-evaluation mean is 1.1535 (4.44% reduction). PE has 61 full-target attempts, 57 finite scores, and four failed attempts. All 57 finite outcomes meet the criterion, yielding 57 successful candidates with a mean gain of 7.66%.

C Per-instance Discovery Outcomes

Table 21 reports local discovery outcomes for S and PE alongside PE's best gains from local and replayed peer candidates (Section 5.1). For ranking and watch-time, the reported means average the best local gain on each dataset. The five RL searches share the same base model and training data and differ only in random seed. Their goal is to find the strongest algorithm for this task, so we report the best result across the five searches.

Table 21: Local discovery outcomes and PE's best gains from local and replayed peer candidates. Gains are relative improvements. For recommendation, S and PE report the best locally proposed gain on each dataset; the RL row reports the maximum across five searches of the same task. Watch-time uses the GPT-5.5 paired comparison.

Domain	Dataset / summary	S	PE	PE best including peer candidates †
Ranking	KuaiLive	+8.93%	+23.19%	+23.19%
	KuaiRand-1K	+5.53%	+9.29%	+10.66%
	KuaiRand-Pure	+8.03%	+6.07%	+6.07%
	KuaiRec	+1.03%	+2.42%	+6.47%
	KuaiSAR	+11.52%	+3.88%	+10.06%
	Mean	+7.01%	**+8.97%**	+11.29%
Watch-time	CIKM16	+7.32%	+12.55%	+12.55%
	KuaiRec	+0.40%	+0.67%	+0.67%
	WeChat	+0.80%	+0.81%	+0.81%
	KuaiRand-Pure	+0.75%	+0.16%	+0.65%
	KuaiRand-1K	+4.93%	+5.08%	+5.08%
	Mean	+2.84%	**+3.85%**	+3.95%
RL algorithm	**Max / 5 searches**	+23.53%	**+32.35%**	—

S: standalone evolution. †The final column takes the best valid score among PE's locally proposed candidates and candidates from peer instances retrained on the receiving source instance. It summarizes the value of peer evaluation within PE. Comparisons use unrounded values.

Variation across datasets. Table 22 reports the PE–S difference on each recommendation source, separately for locally proposed candidates and for PE's best available candidates including peer replay. PE's local gains exceed S on three of five ranking datasets and four of five watch-time datasets, with the largest differences on KuaiLive and CIKM16. Omitting KuaiLive changes the ranking local-mean difference to approximately -1.11 percentage points; omitting CIKM16 changes the watch-time difference to approximately -0.04 points. When PE's replayed peer candidates are included, the corresponding differences remain positive at approximately $+1.79$ and $+0.08$ points, and the mean difference remains positive after omitting any one dataset in either family. Thus local-search benefits are uneven, while direct reuse broadens the available improvements. These are descriptive aggregation checks derived from the displayed source results.

Table 22: **Dataset-wise PE–S differences in relative gain.** Entries are percentage-point differences derived from the rounded values in Table 21. The two columns distinguish local proposals from the best PE candidates available after peer replay.

Domain	Dataset	Local PE minus S	PE including peers minus S
Ranking	KuaiLive	+14.26	+14.26
	KuaiRand-1K	+3.76	+5.13
	KuaiRand-Pure	−1.96	−1.96
	KuaiRec	+1.39	+5.44
	KuaiSAR	−7.64	−1.46
Watch-time	CIKM16	+5.23	+5.23
	KuaiRec	+0.27	+0.27
	WeChat	+0.01	+0.01
	KuaiRand-Pure	−0.59	−0.10
	KuaiRand-1K	+0.15	+0.15

D Graduated Designs and Transfer Analysis

Table 23 illustrates one graduated design from each task family. Tables 2 and 4 report the strongest outcome in each family. The 13 successful LLM candidates span three intervention families: Muon update geometry, RoPE configurations within the existing hybrid attention stack, and late-training EMA/SWA checkpoint averaging.

Table 23: **One concrete graduated design from each task family.** Gains are relative to the fixed reference at the stated final-stage instance. Each row identifies a complete candidate; the intervention column summarizes its principal changes. The success criteria are specified in Table 20.

Family	Source / iteration	Principal intervention	Final-stage outcome
Ranking	KuaiRand-Pure / 57	Gated length-64 causal attention alongside linear retention; inherited temporal and spectral components.	KuaiRand-27K: 3.02% ranking gain.
Watch-time	WeChat / 21	Duration-conditioned, left-truncated Laplace mixture with a learned near-median quantile readout.	KuaiRand-27K: 2.17% MAE reduction.
RL algorithm	Source 2 / 87	Bounded advantage weights based on trajectory distance from the reference policy.	3B, 50 steps: 6.64% pass@1 gain.
LLM	D5-B / 89	EMA/SWA checkpoint averaging over approximately the final 20% of training.	Full target: 2.48% accuracy gain.
VLM	D5-A / 86	Gated global image-context injection into patch tokens, combined with square-root warm-up.	Full target: 7.51% loss reduction.

Interpreting the transfer profiles. Figure 2 shows three individual PE candidates: (a) the ranking design discovered on KuaiRand-Pure at iteration 57, (b) a ranking design discovered on KuaiLive, and (c) the watch-time head discovered on WeChat at iteration 21. The profiles measure transfer after retraining the same design under each receiving instance's protocol; unevaluated pairs remain unknown. Figure 3 traces the availability of shared evidence and subsequent local proposals. Figure 4 evaluates the checkpoint-averaging design from D5-B iteration 89.

D.1 Execution traces and mechanism analysis

The cases use the runs underlying Tables 2 and 4, with archived candidate code, evaluation results, and proposal records supporting the mechanism analysis. The shared-context reconstruction combines

archived ledger records available before each proposal with the context-selection rule; saved base prompts do not include the added shared-context text. The evaluations measure transfer, while the traces establish when evidence was available for reuse. Proposal indices are local to each source search; the sequence below follows event timing across asynchronous searches. Gains use the fixed instance reference unless a direct-parent comparison is stated.

Case 1: LLM checkpoint averaging and continued local refinement. The D2-B iteration-100 candidate implements checkpoint averaging using EMA/SWA late in training, alongside inherited short-context RoPE and Muon update-order changes. It improves three of four peers and gains 1.06% on the full target, above the graduation threshold, while regressing by 1.12% on D2-A. The reconstructed context for D5-B proposal 59 includes this successful peer-evaluation record, which was available before the proposal. D5-B's iteration-59 proposal adopts checkpoint averaging using EMA/SWA in a new local candidate, beginning averaging roughly 55–65% into training or during the WSD decay phase. The resulting design improves D5-B by 3.30% but gains only 0.90% on the full target, below the 1% graduation threshold. The later D5-B iteration-89 candidate averages checkpoints over approximately the final 20% of training. On the full target, this candidate reaches 41.25% accuracy, compared with 41.13% for its direct parent—a 0.28% relative improvement computed from unrounded scores (Table 24). This improvement accompanies a 1.17% source regression. Its 2.48% full-target gain is relative to the fixed 40.25% reference; iteration 59 is an earlier design in the search, not its direct parent. Full-target outcomes are excluded from proposal feedback and are used here to compare the resulting designs. Checkpoint averaging plausibly reduces sensitivity to the final optimizer iterate. The trace documents the peer evidence available before the local edits and the transfer performance of the resulting designs.

Table 24: **Direct-parent comparisons for the representative cases.** Positive relative gains indicate improvement and are computed from unrounded scores. These comparisons use the direct parent rather than the fixed benchmark reference.

Design	Metric	Parent score	Candidate score	Gain (%)
LLM, D5-B / 89	Full-target accuracy (%)	41.13	41.25	0.28
VLM, D5-A / 86	Full-target loss	1.1200	1.1164	0.33
RL, source 2 / 87	Source pass@1 (%)	3.43	3.19	−7.14

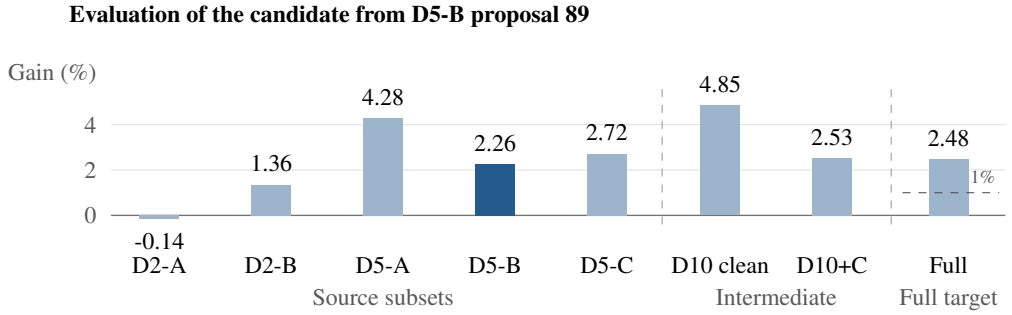


Figure 4: **The checkpoint-averaging candidate from D5-B proposal 89 improves seven of eight instances.** Gains use each instance's fixed reference; the candidate exceeds the 1% full-target threshold. The darker bar marks its source, vertical dividers separate training stages, and the dashed horizontal line marks the full-target threshold.

Case 2: Watch-time design reuse across datasets. The WeChat iteration-21 watch-time head combines a duration-conditioned, left-truncated Laplace mixture with a learned near-median quantile readout. The design conditions on $\log(1 + \text{duration})$ and softly anchors component locations to video duration. The mixture models concentrated viewing modes and long watch times from repeated viewing, while the near-median readout aligns point prediction with absolute error. A differentiable quantile is learned within approximately $[0.45, 0.55]$, with a small mixture-dependent gate for an additional soft duration adjustment. Bisection locates the quantile, and a Newton correction

propagates gradients through the distribution parameters. Relative to its parent, the combined edit reduces source MAE from 17.79 to 17.72 seconds, a 0.39% local improvement. When retrained unchanged, the complete design improves KuaiRand-Pure by 0.65%, exceeding the 0.16% gain of its best locally proposed PE candidate. It achieves the best reported larger-scale result, a 2.17% MAE reduction on KuaiRand-27K, while regressing by 2.70% on CIKM16 (Figure 2c). On VK-LSVD under shared target-side calibration, the same head reduces MAE by 1.02% across three training seeds (Appendix F.2). This case shows how PE turns a local discovery into a design useful to other instances, with benefits that extend to a dataset outside discovery.

Ranking: a local transition channel improves receiving source instances. The KuaiRand-Pure iteration-57 candidate adds a length-64 strictly causal local attention branch alongside linear retention, combining them through a small learned gate. Inherited temporal features and a causal spectral filter remain part of the complete design. When retrained on KuaiRec and KuaiSAR, this design outperforms the best candidates proposed locally on those instances. It retains a 3.02% gain on KuaiRand-27K (Figure 2). The local attention branch models recent transitions alongside the existing long-range aggregation. Peer evaluations establish transfer of the complete design; their results subsequently entered the shared context of ongoing peer searches.

RL: reference-distance weighting and selective transfer. The source-2 iteration-87 design computes each trajectory's masked mean absolute difference between rollout-policy and reference-policy log probabilities. It centers the distance using the batch median, scales it by a robust spread, clamps it to $[-6, 6]$, and applies a sigmoid. After advantage whitening, bounded weights amplify positive advantages for reference-like trajectories and attenuate them for distant trajectories. This modulation is reversed for negative advantages. The distance is detached, and the KL coefficient and training budget remain fixed. This reference-dependent weighting is intended to retain useful coding priors while learning from reward. The candidate regresses against its direct parent on the source (Table 24) but remains above the fixed source reference. It passes on two peers and regresses on the other two, then achieves a 6.64% larger-scale gain. The 32.35% source-local maximum instead comes from the source-2 iteration-88 quantile-threshold rollout filter; the local and larger-scale maxima are distinct discoveries. Appendix F.4 separately tests policies trained with the discovered algorithms on held-out coding tasks.

VLM: convergent discovery of global image context. The D5-C iteration-56 design pools frozen SigLIP patch features and applies image-level FiLM to projector hidden states through a 0.1-scaled residual. It reduces full-target loss by 7.07% relative to the fixed baseline despite regressing against its direct parent on both the source and full target. D5-A had explored global image context before this candidate was published, and the D5-C record was not among the successful peer transfers shown to the D5-A iteration-83 proposal. The trace therefore supports convergent exploration of this mechanism family. D5-A iteration 83 mean-pools all image tokens, projects the summary, and adds a small gated residual to each token. At D5-A iteration 86, the broadcast intervention is applied to a different local parent; the resulting candidate uses square-root warm-up. The resulting full-target loss improves over its parent (Table 24), while source loss increases by only 0.02%. Its loss reduction relative to the 1.2071 fixed reference is 7.51%. The design supplies explicit image-level context to every patch while preserving the fixed image-token interface.

E Additional Configuration and Budget Results

E.1 Larger-scale and full-target outcomes

Figure 5 reproduces the best larger-scale outcomes from Table 2D for Track A and the best full-target outcomes from Table 4 for Track B. Each pair uses its domain's metric and fixed reference: Track A compares PE with S after larger-scale retraining, while Track B compares PE with Direct after full-target training. All metrics are oriented so that positive relative gains indicate improvement.

E.2 Search harnesses and proposal models

This comparison tests local discovery and larger-scale transfer across harnesses and proposers. Table 25 holds the watch-time task family and EGMN starting model fixed. Panel A uses GPT-5.5

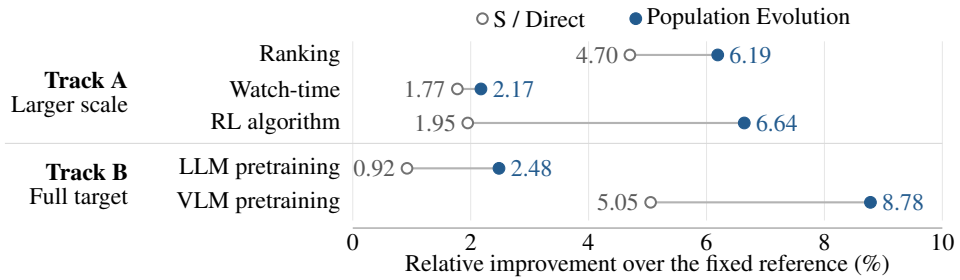


Figure 5: **PE improves the best larger-scale or full-target outcome in all five task families.** Each pair compares PE with standalone evolution (Track A) or direct full-target evolution (Track B). The Track A values match Table 2D, and the Track B values match Table 4. These are completed-run best gains; Track B uses matched total GPU compute. Figure 6 summarizes the same LLM runs.

with AdaEvolve (Cemri et al., 2026), EvoX (Liu et al., 2026a), OpenEvolve (Sharma, 2025), PACEvolve (Yan et al., 2026), or AutoResearch (Karpathy, 2026). Panel B fixes AdaEvolve and varies the proposer among GPT-5.5, GPT-5.6, Claude Opus 4.8, and Claude Opus 5, all at high reasoning effort. Each S/PE pair uses the same harness and proposer. Source columns measure local discovery, while KuaiRand-27K measures larger-scale transfer of the selected designs.

Table 25: **PE improves the best larger-scale gain over S with four of five harnesses and all four proposers.** Watch-time gains are relative MAE reductions (%). Source columns report the best local result on each source instance; KuaiRand-27K reports larger-scale transfer, with passed/tested counts referring to evaluations. Bold marks the better value within each S/PE pair, using unrounded scores.

		Best gain on each source dataset					Larger scale KuaiRand-27K	
Configuration (k)	Method	CIKM16	KuaiRec	WeChat	KuaiRand Pure	KuaiRand 1K	Best gain	Passed / tested
A. Search harnesses (GPT-5.5; high)								
AdaEvolve (2)	S	7.32	0.40	0.80	**0.75**	4.93	1.77	4/10
	PE	**12.55**	**0.67**	**0.81**	0.16	**5.08**	**2.17**	5/7
EvoX (4)	S	7.01	0.28	**0.49**	**0.30**	1.87	**1.45**	4/20
	PE	**11.01**	**0.58**	0.49	0.11	**3.01**	1.05	18/18
OpenEvolve (3)	S	**7.84**	0.32	0.53	0.43	4.19	1.56	12/15
	PE	5.72	**0.56**	**0.59**	**0.59**	**4.47**	**1.75**	11/11
PACEvolve (4)	S	9.29	**0.75**	0.65	**0.95**	**4.64**	2.16	19/20
	PE	**11.14**	0.46	**0.93**	0.41	4.58	**2.50**	20/20
AutoResearch (4)	S	8.25	0.37	**0.71**	0.46	3.56	1.68	20/20
	PE	**9.10**	**0.52**	0.67	**0.73**	**4.48**	**2.42**	16/16
B. Proposal models (AdaEvolve; high)								
GPT-5.5 (2)	S	7.32	0.40	0.80	**0.75**	4.93	1.77	4/10
	PE	**12.55**	**0.67**	**0.81**	0.16	**5.08**	**2.17**	5/7
GPT-5.6 (1)	S	**11.52**	0.55	**0.85**	0.76	4.95	1.34	3/5
	PE	11.29	**0.56**	0.69	**0.81**	**5.45**	**1.50**	1/3
Opus 4.8 (5)	S	**12.10**	0.55	**0.96**	0.78	5.00	2.61	24/25
	PE	11.07	**0.61**	0.84	**0.91**	**5.15**	**2.92**	20/21
Opus 5 (12)	S	**15.26**	1.02	1.05	**1.63**	5.42	2.61	57/60
	PE	12.77	**1.07**	**1.28**	0.78	**5.58**	**3.64**	48/57

S: standalone evolution; k is its larger-scale selection quota per source. Comparisons are within configurations. AdaEvolve/GPT-5.5 is shared by both panels and Table 2. Opus denotes Claude Opus.

E.3 LLM FULL-TARGET QUALITY, YIELD, AND THRESHOLD SENSITIVITY

Figure 6 summarizes the completed LLM discovery runs in Section 4.3, which use matched total GPU compute. Panel (a) reports the best full-target gain; panel (b) reports the number of candidates meeting the predefined 1% relative-improvement criterion. All executed search and evaluation stages contribute to cost under the accounting in Appendix B.5.

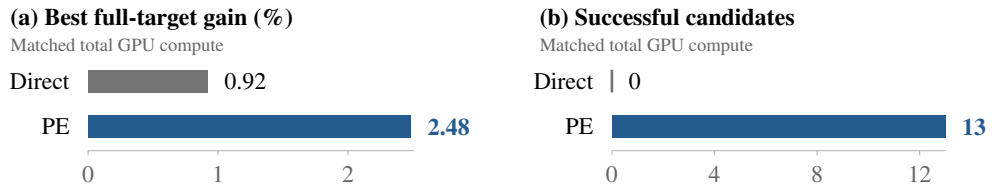


Figure 6: **PE improves LLM quality and discovery yield under matched total GPU compute.** Both panels summarize completed runs. (a) Best relative full-target gains are 2.48% for PE and 0.92% for Direct. (b) The predefined 1% criterion is met by 13 candidates from 39 PE full-target attempts and none from 70 Direct attempts.

Sensitivity to the success threshold. We reclassify the completed full-target results at four relative-improvement thresholds while retaining the fixed reference accuracy of 40.25% (0.4025 on the unit scale). A candidate meets threshold τ when its score is at least $0.4025(1+\tau)$. The main threshold remains the 1% criterion fixed before the reported comparisons. Table 26 shows that PE returns more successful candidates at every examined threshold. At 0.5%, Direct returns six successful candidates and PE returns 21; at 1%, 1.5%, and 2%, Direct returns none while PE returns 13, eight, and three, respectively. The best Direct gain is 0.92%, while the best PE gain is 2.48%; the quality comparison complements the threshold-dependent success counts.

Table 26: **LLM success counts across full-target gain thresholds.** Counts use the same completed runs under matched total GPU compute; the predefined main threshold is bold.

Relative gain threshold (%)	PE	Direct
0.50	21	6
1.00	**13**	**0**
1.50	8	0
2.00	3	0

F TRANSFER TO NEW DATASETS AND CODING TASKS

F.1 CANDIDATE SELECTION AND EVALUATION PROTOCOL

This appendix specifies the three transfer evaluations in Section 3.1, testing the designs returned by each complete discovery procedure beyond its discovery instances. Watch-time heads transfer to VK-LSVD; ranking designs transfer to KuaiVideo with a change from full-catalog next-item prediction to candidate-conditioned click prediction; RL modifications are evaluated on a task partition disjoint from the 423-task larger-scale selection set. The two recommendation datasets were excluded from source search, peer evaluation, and larger-scale promotion.

Candidate selection and evaluation. Each procedure supplies candidates through its own selection rule. Each discovered intervention is evaluated without candidate-specific redesign or hyperparameter tuning, using a shared target protocol. For watch-time, PE contributes designs that passed larger-scale evaluation, and S contributes the best source candidate from each of five datasets. For ranking, PE likewise contributes designs that passed larger-scale evaluation, and S contributes the top four source candidates from each of five datasets. RL compares two successful PE candidates and the best S candidate from each of five source searches. Recommendation weights are trained from scratch

on each new dataset; RL policies start from the same pretrained model and follow a fixed training protocol before evaluation on held-out coding tasks.

Target calibration and reporting. Each recommendation target uses one baseline-calibrated configuration shared by the reference and every candidate, including the training duration. Baseline hyperparameter and training-duration selection use the released test split, which is also used for final reporting; no separate validation split is used for these selection decisions. The recommendation results therefore measure transfer to new datasets under shared target-side calibration, rather than an independent test evaluation after calibration. Candidate-specific HPO and evolutionary adaptation on these targets are disabled. The RL reference uses the fixed, unmodified training configuration without target-side HPO; the 632 coding tasks are disjoint from the larger-scale selection partition. Recommendation references and candidates are trained with three seeds; RL references and candidates undergo three independent stochastic reruns with the same nominal training seed. We report means and sample standard deviations over these three runs. These repetitions assess candidate performance rather than variability across complete discovery runs. AUC, gAUC, and pass@1 are expressed as percentages; MAE is in seconds. Means and standard deviations use two decimal places, except log loss, which uses four. All comparisons use the original precision.

Candidate-set and best-candidate summaries. For recommendation, each candidate's score is its three-seed mean, and candidate-set means give each evaluated candidate record equal weight. Candidate-set means measure average transfer of each procedure's selected designs; best-candidate scores identify its strongest evaluated design by mean AUC or mean MAE. On VK-LSVD, PE and S obtain 14.54 ± 0.01 s and 14.55 ± 0.01 s. For each seed, we first average MAE across candidates; the reported standard deviation is then computed across these three averages. On KuaiVideo, the corresponding mean AUCs are 74.45% and 74.39%. As a weighting check, we group behaviorally equivalent implementations and aggregate their unrounded scores; equivalence is determined from implementations, not equality of rounded metrics. Counting each group once preserves the PE–S ordering: mean AUCs remain 74.45% and 74.39%, and mean MAEs remain 14.54 ± 0.01 s and 14.55 ± 0.01 s after rounding. The five RL source searches share the same training dataset and objective, and their discovery goal is to find the strongest algorithm for that setting. We therefore compare the best RL candidate from each method using its mean over three stochastic reruns, rather than the candidate-set mean. Table 3 summarizes the PE–S comparisons; the following subsections report the references and individual candidates.

F.2 Watch-time transfer to VK-LSVD

This evaluation tests whether a discovered distributional head remains useful on a new dataset under a common feature encoder and training protocol.

Data and shared model. We use the HEGM-processed `VK-LSVD/up0.01_ir0.01` subset of VK-LSVD (Poslavsky et al., 2026; Gulevskaia et al., 2026), retaining its chronological split and precomputed features.[1] Video duration and the remaining numerical features are retained; duration buckets and categorical vocabularies are constructed from training data only. All candidates share the encoder and training settings in Table 27; only the distributional head changes.

Calibration and outcomes. Baseline calibration monitors released-test MAE and completes 11 valid trials within a cap of 50, stopping after ten successive trials without improvement. Table 30 lists the search ranges and selected values shared by all candidates; no candidate-specific tuning follows. The shared calibrated EGMN reference has MAE 14.73 ± 0.02 s. Every evaluated PE head improves its mean MAE, with paired reductions of 1.02–1.36% (Table 28). All evaluated S heads also improve over this reference. PE has lower candidate-set mean MAE (Table 3), although the strongest individual head comes from S: 14.46 ± 0.02 s versus 14.53 ± 0.02 s for PE. The WeChat iteration-21 head achieves 14.58 ± 0.03 s and a $(1.02 \pm 0.30)\%$ paired MAE reduction, alongside its source-instance and KuaiRand-27K improvements. This is a concrete example of one unchanged distributional design remaining useful across datasets and training scales.

[1] Raw data: `https://huggingface.co/datasets/deepvk/VK-LSVD`; processed release: `https://github.com/rw404/HEGM`.

Table 27: **Shared VK-LSVD transfer protocol.** Final candidates train on the released training split and are evaluated on the released test split.

Component	Specification
Examples (train / val. / test)	7,505,820 / 1,516,831 / 1,275,643
Chronology	5/1/1-week split after historical-feature weeks 0–19; training uses weeks 20–24
Numerical features	Video duration and 31 other features; normalization constant: 157 seconds
Categorical features	Age, gender, geographic region, place, platform, and user agent
Duration encoding	50 quantile-based buckets fitted on training data
Shared EGMN encoder	16-dimensional categorical embeddings, scalar linear projections, and a 256/128/64 MLP
Reference head	One exponential and ten left-truncated Gaussian components
Reference loss	Mixture likelihood, L1 point-prediction error, and entropy regularization
Training	Adagrad; batch size 2,048; ten epochs; three training seeds

Table 28: **VK-LSVD reference and all evaluated transfer candidates.** Values are means ± sample standard deviations across three training seeds. Reductions are computed against the seed-matched reference. PE identifiers are archived candidate-ID suffixes; S entries identify their source dataset.

Method	Candidate / source	MAE (s) ↓	Paired reduction (%) ↑
Reference	EGMN	14.73 ± 0.02	—
PE	`e33792c7`†	14.53 ± 0.02	1.36 ± 0.15
PE	`2f3242e3`†	14.53 ± 0.02	1.36 ± 0.15
PE	`92f5eb20`†	14.53 ± 0.02	1.36 ± 0.15
PE	`e0b2ffa7`	14.53 ± 0.01	1.35 ± 0.06
PE	`94db2d04`	14.58 ± 0.03	1.02 ± 0.30
S	KuaiRand-1K	14.46 ± 0.02	1.80 ± 0.10
S	WeChat	14.51 ± 0.01	1.47 ± 0.15
S	KuaiRand-Pure	14.55 ± 0.00	1.23 ± 0.12
S	CIKM16	14.56 ± 0.04	1.12 ± 0.39
S	KuaiRec	14.69 ± 0.01	0.24 ± 0.17

†These candidates are behaviorally equivalent implementations. Aggregation uses the original, unrounded run scores.

F.3 RANKING TRANSFER TO KUAIVIDEO CLICK PREDICTION

This evaluation tests sequence-model designs under both a new dataset and a different recommendation objective. A common adapter makes the starting model and selected designs compatible with candidate-conditioned click prediction.

Data and features. We use the RecZoo `KuaiVideo_x1` release with its original train/test split and ALPINE-processed visual features (Li et al., 2019).[2] Each example contains a user, candidate item, positive and negative histories, and a binary click label; histories retain the released order and are truncated to the most recent 100 items. Categorical vocabularies use training data only, while the visual feature table remains frozen without additional normalization or dimensionality reduction. The visual feature table provides complete coverage, including the 41.81% of test rows whose candidate items map to the categorical unknown token. Users are represented by learned categorical embeddings; the released user-visual features are not used. Table 29 summarizes the data and shared adapter specifications.

Table 29: **Shared KuaiVideo data and adapter specifications.** The discovered sequence-model code is the only component that differs among the reference, S, and PE candidates.

Component	Specification
Candidate rows (train / test)	10,931,092 / 2,730,291
Requests (train / test)	803,007 / 209,728; 10,000 users per split
Item vocabulary	Minimum training count 10; 632,404 known items, plus padding and unknown entries
Visual features	64-dimensional vectors for 3,242,315 items; frozen, with separate padding and unknown entries
Item representation	Learned item-ID embedding plus a trainable, bias-free $64 \to 128$ projection of its visual vector
History encoding	Shared sequence encoder and embeddings for positive and negative histories; at most 100 items each
Click predictor	MLP widths 512/256/1; binary cross-entropy loss
Final training	Four epochs; four H200 GPUs; three training seeds

Common sequence-aware CTR adapter. The candidate item is appended as the final query token to each history. Its output states h^+ and h^- summarize the positive and negative histories conditioned on that item. Because these histories lack per-item timestamps, the adapter supplies sequential positions $1, \ldots, L+1$. Given candidate embedding e and user embedding u, the shared click predictor uses

$$\hat{p} = \sigma\big(\mathrm{MLP}\big([h^+; h^-; e; h^+ - h^-; h^+ \odot e; h^- \odot e; u]\big)\big) .$$

The reference, S candidates, and PE candidates use this identical adapter and differ only in the discovered sequence-model design.

Calibration and final training. Baseline calibration uses 50 serial Optuna TPE trials, permitting two to five epochs with early-stopping patience two. Hyperparameters and the training duration are selected using released-test AUC. The selected configuration in Table 30 is shared by all candidates, which train for exactly four epochs and are evaluated once afterward. Each candidate is trained under three seeds; AUC is the primary metric, with sample-weighted user gAUC and binary log loss as auxiliary metrics.

Transfer outcome. The best PE candidate, discovered on KuaiRand-Pure, adds a gated spectral causal long-filter channel. Its mean AUC exceeds that of the best S design from KuaiRand-1K by 0.04 percentage points computed from unrounded means, with improvements in both auxiliary metrics (Table 31). Every evaluated PE design exceeds the shared calibrated reference in mean AUC, with

[2]Release and feature archive: `https://huggingface.co/datasets/reczoo/KuaiVideo_x1`; visual features: `item_visual_emb_dim64.h5`.

Table 30: **Shared calibration for the new recommendation targets.** Baseline configuration and training-duration selection use the released test split; the selected protocol is reused unchanged for the reference and every candidate.

Parameter	Search range	Shared selected value
VK-LSVD		
Learning rate	$[0.01, 0.3]$	0.076611
Weight decay	$[10^{-7}, 10^{-3}]$	4.20799×10^{-7}
Entropy coefficient α	$[0, 1]$	0.37454
Regression coefficient β	$[0.1, 5]$	4.7585
Dropout	$[0, 0.5]$	0.365997
KuaiVideo		
Learning rate	Log-uniform $[10^{-4}, 6 \times 10^{-3}]$	2.9416×10^{-4}
Weight decay	Log-uniform $[10^{-6}, 10^{-3}]$	5.3419×10^{-6}
Dropout	$\{0.1, 0.2, 0.3, 0.5\}$	0.1
Global batch size	$\{1024, 2048\}$	2,048
Epochs	2–5 (patience 2)	4

Table 31: **KuaiVideo reference and best S and PE designs.** The best S and PE designs are selected by mean AUC across three training seeds; all metrics in a row come from that same design. Every metric is reported as mean $\pm$ sample standard deviation over those same runs.

Model	AUC (%) ↑	User gAUC (%) ↑	Log loss ↓
Reference	74.39 ± 0.11	66.14 ± 0.06	0.4378 ± 0.0008
Best S	74.43 ± 0.08	66.27 ± 0.05	0.4371 ± 0.0018
Best PE	$\mathbf{74.46 \pm 0.06}$	$\mathbf{66.29 \pm 0.02}$	$\mathbf{0.4365 \pm 0.0005}$

increases of approximately 0.04–0.07 percentage points computed before rounding (Table 32). Eight evaluated S candidates exceed the reference mean (Table 33). At the candidate-set level, mean AUC is 74.45% for PE and 74.39% for S. These results support design transfer under a common CTR adapter and shared target-side calibration; the small PE–S differences are descriptive comparisons of three-seed means.

Table 32: **All evaluated PE ranking candidates on KuaiVideo.** Identifiers are archived candidate-ID suffixes. Values are means ± sample standard deviations across three training seeds. Every candidate's mean AUC exceeds the reference before rounding.

Candidate	AUC (%) ↑	User gAUC (%) ↑	Log loss ↓
Reference	74.39 ± 0.11	66.14 ± 0.06	0.4378 ± 0.0008
b3cb61d8	74.46 ± 0.06	66.29 ± 0.02	0.4365 ± 0.0005
5f007905	74.46 ± 0.07	66.26 ± 0.01	0.4371 ± 0.0006
7ac9ec81	74.46 ± 0.09	66.29 ± 0.09	0.4369 ± 0.0011
e7ec2459	74.46 ± 0.09	66.29 ± 0.09	0.4369 ± 0.0011
2755e3d1	74.46 ± 0.09	66.29 ± 0.09	0.4369 ± 0.0011
1add4a4b	74.46 ± 0.09	66.29 ± 0.09	0.4369 ± 0.0011
8582c2be	74.45 ± 0.08	66.27 ± 0.04	0.4377 ± 0.0007
d54779e3	74.45 ± 0.08	66.27 ± 0.04	0.4377 ± 0.0007
33963c64	74.45 ± 0.08	66.27 ± 0.05	0.4377 ± 0.0008
314d2b6b	74.45 ± 0.11	66.27 ± 0.10	0.4370 ± 0.0010
4b146074	74.45 ± 0.11	66.27 ± 0.10	0.4370 ± 0.0010
c9913f8e	74.43 ± 0.09	66.26 ± 0.04	0.4377 ± 0.0010
ad9fa175	74.43 ± 0.09	66.26 ± 0.04	0.4377 ± 0.0009
fff8c656	74.43 ± 0.08	66.26 ± 0.07	0.4375 ± 0.0006
17f754bc	74.43 ± 0.08	66.26 ± 0.03	0.4377 ± 0.0009
a3da30b7	74.43 ± 0.09	66.26 ± 0.04	0.4377 ± 0.0010
648f09a3	74.43 ± 0.09	66.26 ± 0.03	0.4378 ± 0.0010
fd6457ef	74.43 ± 0.09	66.25 ± 0.03	0.4377 ± 0.0009

Table 33: **All evaluated S ranking candidates on KuaiVideo.** Each source contributes its top four candidates by source performance. Identifiers are archived candidate-ID suffixes; values are means ± sample standard deviations across three training seeds.

Source	Candidate	AUC (%) ↑	User gAUC (%) ↑	Log loss ↓
—	Reference	74.39 ± 0.11	66.14 ± 0.06	0.4378 ± 0.0008
KuaiRand-1K	560d182a	74.43 ± 0.08	66.27 ± 0.05	0.4371 ± 0.0018
KuaiRand-1K	35335eb3	74.42 ± 0.08	66.27 ± 0.03	0.4371 ± 0.0013
KuaiRand-1K	e5c01c96	74.42 ± 0.08	66.27 ± 0.03	0.4371 ± 0.0013
KuaiRand-1K	5d08ec24	74.42 ± 0.08	66.26 ± 0.04	0.4371 ± 0.0013
KuaiSAR	3a70637c	74.41 ± 0.10	66.28 ± 0.07	0.4368 ± 0.0014
KuaiSAR	6aa0c6a0	74.41 ± 0.09	66.28 ± 0.07	0.4368 ± 0.0013
KuaiSAR	c576350a	74.40 ± 0.09	66.28 ± 0.07	0.4368 ± 0.0013
KuaiSAR	a3707be1	74.40 ± 0.08	66.27 ± 0.06	0.4369 ± 0.0009
KuaiRand-Pure	fb4fcfa1	74.38 ± 0.04	66.26 ± 0.03	0.4372 ± 0.0001
KuaiRand-Pure	e520862d	74.38 ± 0.04	66.26 ± 0.03	0.4373 ± 0.0002
KuaiRand-Pure	63cd4a7d	74.38 ± 0.04	66.26 ± 0.03	0.4372 ± 0.0001
KuaiRand-Pure	3a338551	74.37 ± 0.04	66.26 ± 0.03	0.4372 ± 0.0001
KuaiLive	2577bcf4	74.37 ± 0.11	66.23 ± 0.04	0.4375 ± 0.0009
KuaiLive	8b954d65	74.37 ± 0.11	66.23 ± 0.04	0.4375 ± 0.0009
KuaiRec	7d3732b5	74.37 ± 0.07	66.20 ± 0.02	0.4385 ± 0.0005
KuaiRec	384908be	74.37 ± 0.07	66.20 ± 0.02	0.4385 ± 0.0005
KuaiLive	d895ff6d	74.37 ± 0.11	66.23 ± 0.04	0.4374 ± 0.0009
KuaiRec	be23d8b2	74.36 ± 0.06	66.20 ± 0.02	0.4385 ± 0.0004
KuaiRec	4b23990b	74.36 ± 0.06	66.20 ± 0.02	0.4385 ± 0.0005
KuaiLive	9e6f90f4	74.36 ± 0.12	66.23 ± 0.04	0.4375 ± 0.0009

F.4 RL TRANSFER TO A DISJOINT LIVECODEBENCH PARTITION

This evaluation tests policies trained with the discovered RL algorithms on held-out coding tasks under a fixed training protocol.

Task partition and training. The 1,055 LiveCodeBench tasks comprise 423 larger-scale selection tasks and 632 held-out tasks with disjoint IDs, both scored using private tests. All methods use Qwen2.5-Coder-3B-Instruct, the same larger-scale configuration, and 50 GRPO rollout steps. Candidates change the GRPO loss and/or rollout filtering; held-out coding tasks are used only to evaluate the trained policies. Each candidate is freshly trained and evaluated in three stochastic reruns with the same nominal training seed.

Best-candidate comparison. PE obtains mean pass@1 of $(23.10 \pm 0.32)\%$, compared with $(22.94 \pm 0.16)\%$ for S (Table 3). The unmodified reference reaches $(23.26 \pm 0.42)\%$ pass@1. The reference, best PE candidate, and best S candidate solve 147, 146, and 145 tasks on average, respectively (Table 34). Thus PE exceeds S by one solved task on average, while neither candidate exceeds the reference mean. These are descriptive comparisons across three stochastic reruns.

Table 34: **Reference and all evaluated RL candidates on LiveCodeBench-632.** Each entry uses three independent stochastic reruns with nominal training seed 42. Pass@1 is reported as mean $\pm$ sample standard deviation. The reference uses the unmodified configuration without target-side HPO.

Method	Candidate	Solved / 632 in each rerun	Pass@1 (%) ↑
Reference	—	145, 150, 146	23.26 ± 0.42
PE	`93d95a15`	146, 143, 139	22.57 ± 0.56
PE	`cbf1c653`	146, 144, 148	23.10 ± 0.32
S	`a0936bce`	141, 143, 142	22.47 ± 0.16
S	`db21a301`	146, 145, 144	22.94 ± 0.16
S	`77431a4e`	136, 140, 136	21.73 ± 0.37
S	`86561efd`	140, 143, 149	22.78 ± 0.73
S	`754c241c`	131, 130, 138	21.04 ± 0.69

G LIMITATIONS AND FUTURE DIRECTIONS

RMD-Bench covers five task families, with compatible intervention interfaces within each family. The transfer results apply to the architectures, training protocols, and evaluation settings studied here; extending PE to less compatible tasks may require additional interface design. Track A evaluates design generalization, whereas Track B targets improvement on a specified expensive workload.

The source-level comparison tests shared experimental feedback; larger-scale and full-target results evaluate the complete discovery procedures. Given the cost of full runs, further independent searches would help characterize variability in discovered designs and final performance (Appendix B.5). Future work can also examine alternative exchange and promotion policies and extend the benchmark to additional task families.