ONE TRAJECTORY, MANY TARGETS: SUPERVISION DENSITY AND THE COMPUTE-OPTIMAL REGIME IN SEQUENTIAL RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequential recommendation turns each user history into several next-item prediction tasks. Prefix-wise training encodes overlapping histories repeatedly, so grouping more targets into one update can save computation. The central question is how this supervision density changes the updates and compute needed to reach a fixed ranking quality. We identify an empirical scaling law and make its denser updates executable through exact shared computation. In five-seed MovieLens 1M (ML-1M), with the data, model, objective, initialization, and target-event budget fixed, quality trajectories align after updates are rescaled by density; threshold crossings follow an inverse-density relation after a shared offset of about 202 updates. At 0.966 million target events, $m = 2$ reaches normalized discounted cumulative gain at rank 10 (NDCG@10) of .1519 in 102.8 training seconds, versus .1498 in 202.5 seconds for $m = 1$. Fixed-state scans and five-seed SASRec sweeps show that the best density depends on the compute regime: the measured frontier peaks at $m = 4$ (.1201 NDCG@10) on ML-10M and improves through the tested $m = 16$ (.0463) on Amazon Beauty. *Exact Prefix-Loss Compilation* (EPC) evaluates aligned prefix losses in one causal pass while preserving a fixed update's loss and gradient when the states and training semantics match. Across 48 matched backbone–dataset comparisons, packing shortens train-to-best time; on Beauty, the combined configuration improves NDCG@10 by 52.8% and reaches its selected checkpoint $50.9\times$ sooner. On ML-10M, four backbones reach matched quality in all five seeds with 1.30–$4.56\times$ fewer target events and 28.70–$87.42\times$ less wall time.

1 INTRODUCTION

Sequential recommendation predicts the next item from an interaction history, so each trajectory supplies multiple next-item targets. This creates a training choice before any model architecture is selected: encode each prefix separately, or group several targets so that one history can support a denser update. We call the number of targets contributed by each sampled user in one update its *supervision density*. Prefix-wise training repeats much of the history computation across these overlapping targets (Tang & Wang, 2018; Zhou et al., 2022), while denser grouping changes how supervision is allocated across users and targets. We study how supervision density scales the updates needed to reach a ranking quality, then how shared execution can lower the cost of denser updates.

Our central result is an empirical supervision-density scaling law within a fixed training protocol. In five-seed MovieLens 1M (ML-1M), we hold users, the causal filter-enhanced multilayer-perceptron recommender (FMLP), full-catalog cross-entropy (full CE), initialization, and target-event budget fixed. Let m cap targets per sampled user, K count optimizer updates, and $K_q(m)$ be the first measured update reaching quality threshold q in normalized discounted cumulative gain at rank 10 (NDCG@10). Define the common fitted update offset as K_0 and each threshold's coefficient as B_q. The $m = 1, 2, 4, 8$ curves nearly align under $m(K - K_0)$; their five-threshold crossings fit $K_q(m) \simeq K_0 + B_q/m$, with $K_0 = 202.35$ updates. The excluded $m = 16$ cap checks the later curve segment retrospectively (Figure 2). Density thus changes updates to quality, while update cost determines training time.

Separate fixed-state gradient scans quantify how supervision density and *supervision breadth*—the number of independently sampled users in an update—shape gradient variance. Increasing targets per user reduces residual within-user variation, whereas sampling more independent users also averages across histories (Equation (7)). The local model fits held-out scans of FMLP and the self-attentive sequential recommender (SASRec) with relative root-mean-square errors (RMSEs) of 2.4% and 1.5%, respectively. Writing a and d for shared and residual gradient-variance components, and c_{traj}

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.

and c_{tgt} for trajectory-encoding and target-scoring costs, gives the central allocation law $m^*_{\text{state}} = \sqrt{c_{\text{traj}} d/(c_{\text{tgt}} a)}$ (Eq. (9)). It turns four observable quantities into an estimable density for allocation. We then test how the scaling relation meets different data and compute regimes, holding the objective and target-event budget fixed within each dataset. A five-seed SASRec sweep on ML-10M and Amazon Beauty directly estimates $(a, d, c_{\text{traj}}, c_{\text{tgt}})$, giving fixed-state predictions $m^*_{\text{state}} = 40.1$ and 49.7. Combining these local predictions with the density-rescaled progress curve and measured update time yields a budget-conditioned frontier (Eq. (10)): ML-10M has a finite measured-quality frontier peaking at $m = 4$, whereas Beauty improves through $m = 16$ as its realized density reaches 5.05. The two frontiers expose a regime shift driven by the interaction of available history density, compute budget, and trained quality, showing how the same allocation law produces different operating points.

The scaling relation counts optimizer updates, so its practical value depends on the cost of evaluating each denser update. SASRec (Kang & McAuley, 2018) exposes multiple prefix-conditioned states in one causal pass. Future-dependent operators make exact reuse conditional: a shared pass must reproduce the states obtained from separately encoded prefixes (Sun et al., 2019; Zhou et al., 2022; Du et al., 2023; Shin et al., 2024; 2025; Liu et al., 2023). This requirement is captured by prefix-consistent states and aligned training semantics.

Exact Prefix-Loss Compilation (EPC) maps each materialized-prefix loss to its aligned state in one causal graph. When this state equals its separately encoded counterpart (the *prefix consistency* condition) and training semantics align, EPC preserves the declared loss and gradient (Theorem 5.2). Natively causal models meet the state criterion; for five evaluated future-dependent operator families, we construct causal variants that do. EPC compares materialized-prefix and shared execution within each variant, preserving that variant's specified update.

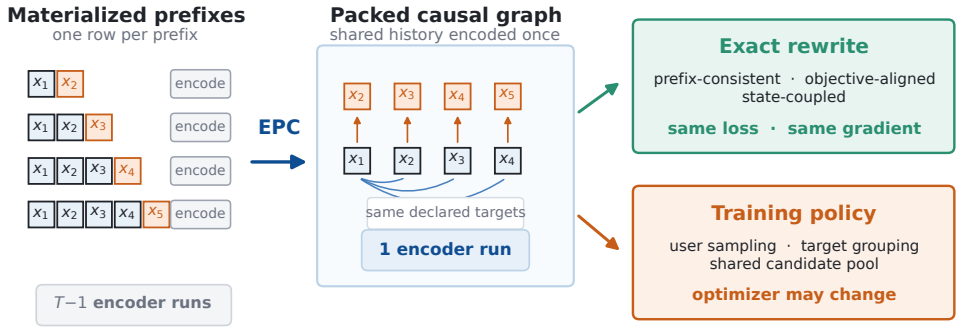


Figure 1: **From materialized prefixes to shared causal execution.** EPC rewrites a fixed materialized-prefix update as one shared causal graph while preserving its declared loss and gradient. Practical packed training may additionally change the stochastic update.

Across 48 backbone–dataset comparisons using a shared-pool contrastive loss (InfoNCE), packing shortens mean time to the best validation checkpoint. On MovieLens 10M (ML-10M), all four backbones reach 95% of each paired right-prefix run's best validation NDCG@10 in all five seeds; per-backbone geometric-mean right-prefix/packed ratios are 1.30–$4.56\times$ for processed target events and 28.70–$87.42\times$ for validation-inclusive wall time. On Amazon Beauty, the combined causal-conversion, packing, and full-catalog CE configuration improves NDCG@10 by 52.8% and reaches its selected checkpoint $50.9\times$ sooner than the released reproduction (Table 35).

Our contributions are threefold:

- **Supervision-density scaling law:** We identify an empirical within-protocol update-to-quality scaling law for within-user supervision density and quantify its threshold-crossing relation.
- **Gradient mechanism and allocation:** We characterize the local density–breadth gradient-variance decomposition and derive a predictive allocation law from $a, d, c_{\text{traj}}, c_{\text{tgt}}$, then connect it to a compute-budget-conditioned quality frontier across datasets and operating regimes.
- **Exact Prefix-Loss Compilation:** We introduce EPC, which compiles materialized-prefix losses into shared causal execution while preserving the declared loss and gradient under prefix consistency and aligned training semantics. Across eight backbones and six datasets, we test exactness, runtime, and ranking quality with five matched seeds.

2 Related Work

Sequential modeling and shared execution. Overlapping trajectory prefixes offer an opportunity to share training computation. Sequential recommendation has progressed from recurrent and

convolutional models (Hidasi et al., 2016; Tang & Wang, 2018) to attention-based architectures (Kang & McAuley, 2018; Sun et al., 2019). SASRec (Kang & McAuley, 2018) provides a native causal case in which multiple next-item positions share one forward pass. This reuse is not automatically valid for bidirectional models (Sun et al., 2019), global sequence operators (Zhou et al., 2022; Du et al., 2023; Liu et al., 2023), or circular operators (Shin et al., 2024; 2025), where earlier states may depend on later positions. EPC identifies when a declared materialized-prefix update can use shared causal execution without changing its loss or gradient.

Supervision allocation and scaling. Training-example selection changes the supervision presented to an optimizer. Sampling and subsequence construction (Petrov & Macdonald, 2022; Dang et al., 2026; Lee et al., 2026), contrastive augmentation (Xie et al., 2022), and multi-target prediction (Gloeckle et al., 2024) alter contexts, views, or prediction targets. Recommendation scaling studies instead relate quality to model size, data volume, or training compute (Ardalani et al., 2022; Zhang et al., 2024). We study supervision density as a complementary axis, varying the next-item targets contributed by each sampled user while holding the model and training-user pool fixed. This separates within-user density from supervision breadth across independent users and measures how density scales updates to a fixed ranking quality.

Objectives and computational reuse. Candidate construction and loss design determine the gradient attached to each target, so target count alone does not define an update. Pairwise ranking (Rendle et al., 2009), sampled-negative objectives (Petrov & Macdonald, 2023), scalable cross-entropy (Mezentsev et al., 2024; Klenitskiy & Vasilev, 2023), and contrastive losses (van den Oord et al., 2018) provide different comparison sets and optimization signals. We align these semantics when testing EPC and vary them separately in practical training. Standard sequence packing reduces padding among independent sequences (Krell et al., 2021), while inference caching reuses computation across decoding steps (Shazeer, 2019). EPC instead reuses overlapping training-prefix computation within one stochastic update while preserving the declared target losses.

3 Optimizing the Use of Trajectory Supervision

Let $\mathcal{V}$ be the item catalog. We reserve each user's final item for testing and penultimate item for validation. A remaining training window $x^u_{1:T_u}$ contains T_u items and $N_u = T_u - 1$ available next-item transitions, indexed by $t = 1, \dots, N_u$; position t supplies the prefix–target pair $(x^u_{1:t}, x^u_{t+1})$. The available labels are fixed by the data; an algorithm chooses which terms share an update and how to compute them.

Optimization unit. Let θ denote model parameters and f_θ the prefix-to-score predictor. Let $\mathcal{T}$ contain the supervised user–position terms in one update, $\mathbf{w} = \{w_{u,t}\}$ their nonnegative weights, $\mathcal{C} = \{C_{u,t}\}$ their candidate multisets, ℓ the per-target loss, and ρ the reduction. We call each supervised user–position pair a *target event*; an algorithm chooses which terms share an update and how to compute them. These fields define

$$\mathcal{U} = (\mathcal{T}, \mathbf{w}, \mathcal{C}, \ell, \rho), \qquad L_{\mathcal{U}}(\theta) = \rho\Big(\big\{w_{u,t}\,\ell\big(f_\theta(x^u_{1:t}), x^u_{t+1}, C_{u,t}\big)\big\}_{(u,t)\in\mathcal{T}}\Big). \tag{1}$$

Let Opt denote one optimizer step, including its state update. With optimizer state s_k, realized unit $\mathcal{U}_k$ produces

$$\widehat{g}_{\mathcal{U}_k}(\theta_k) = \nabla_\theta L_{\mathcal{U}_k}(\theta_k), \qquad (\theta_{k+1}, s_{k+1}) = \mathrm{Opt}(\theta_k, s_k, \widehat{g}_{\mathcal{U}_k}(\theta_k)). \tag{2}$$

Changing any field of $\mathcal{U}$ can alter the optimizer trajectory; changing an update boundary changes $\mathcal{T}$.

Execution of a fixed unit. Let $\mathcal{E}(\mathcal{U})$ be the set of execution graphs that compute the declared unit $\mathcal{U}$, and e one such graph. For one trajectory of T items with fixed targets, weights, candidates, and loss, the equations below show the weighted-sum case of the general reduction ρ in Equation (1), with the fixed candidate argument suppressed for brevity. Write $f^{\mathrm{causal}}_\theta$ for a causal sequence encoder, g_θ for its scoring head, and w_t for a target weight. Materialized-prefix execution evaluates each prefix separately:

$$\mathcal{L}_{\mathrm{prefix}}(\theta) = \sum_{t=1}^{T-1} w_t\,\ell\big(g_\theta(f^{\mathrm{causal}}_\theta(x_{1:t})_t), x_{t+1}\big). \tag{3}$$

A shared causal pass, which we call *packed execution*, instead exposes aligned states $H = f_\theta^{\text{causal}}(x_{1:T-1})$, with H_t at position t:

$$\mathcal{L}_{\text{packed}}(\theta) = \sum_{t=1}^{T-1} w_t \, \ell\big(g_\theta(H_t), x_{t+1}\big). \tag{4}$$

For a mean or other reduction, the same ρ is applied to the aligned per-target terms in both executions. In the practical comparisons, *right-prefix training* denotes the right-aligned implementation of materialized-prefix execution. An *aligned-prefix execution* is the exactness control that evaluates prefixes separately while retaining the shared pass's positions and padding semantics. We reserve *packed training* for the broader policy comparison in which target grouping, candidate construction, or reduction may also change. When the aligned states and all remaining training semantics agree, the two executions are equivalent (Section 5). EPC changes e while holding $\mathcal{U}$ fixed; practical packing may also change $\mathcal{U}$.

For training policy π on a dataset $\mathcal{D}_n$ of n training histories, let $W_q(\pi, e, \mathcal{D}_n)$ be elapsed wall-clock time, including training and validation, until the first recorded checkpoint with validation quality at least q. We report W_q alongside processed target events and, where needed, training-only time.

4 Supervision-Density Scaling and Its Computational Cost

Choosing supervision density requires separating updates-to-quality from the cost of each update. In the controlled ML-1M causal-FMLP/full-catalog CE sweep, the fitted relation measures how caps $m = 1, 2, 4, 8$ change the updates needed to reach each threshold, while event counts and training-only times show the resulting efficiency tradeoff. The fixed-state variance model characterizes user–target allocation locally; EPC reduces the encoding cost of the same declared dense update when the conditions in Section 5 hold.

4.1 An empirical update-to-quality scaling relation

Let $Q(K, m)$ denote validation NDCG@10 after K optimizer updates at per-user target cap m, and let $K_q(m)$ be the first *observed* update count at which $Q \geq q$. Let K_0 denote a shared fitted update offset, Φ a common empirical quality curve, and B_q a threshold-specific coefficient. The central empirical question is whether the ML-1M causal-FMLP/full-catalog CE trajectories share a density-rescaled progress coordinate:

$$Q(K, m) \simeq \Phi(m(K - K_0)), \qquad K_q(m) \simeq K_0 + \frac{B_q}{m}. \tag{5}$$

The offset is estimated from the trajectories rather than assumed to be a startup duration. The threshold form tests whether several quality levels admit the same offset. We fit one shared offset across five quality thresholds using paired $m = 1, 2, 4, 8$ runs. The $m = 16$ arm belongs to the same sweep but is excluded from the fit; its later measured segment checks the curve alignment retrospectively (Section 6.2). The relation characterizes this controlled ML-1M training policy.

The relation also gives a cap-based estimate of processed target events. Let $A_q(m)$ count processed target events through the first observed crossing of quality threshold q at target cap m. If each update draws U users and all reach cap m, then

$$A_q(m) \approx UmK_q(m) \approx U(mK_0 + B_q). \tag{6}$$

This is an approximation because m is a cap: users with shorter histories contribute fewer target events. The measured $A_q(m)$ instead counts the events actually used, including repeated visits to the same user–position target.

The relation maps density to optimizer updates $K_q(m)$, while $A_q(m)$ counts processed target events. Distinct-user data efficiency instead varies the number of available training histories (Appendix C.2.1). Let $T_q^{\text{train}}(m)$ denote the training-only time to the first observed crossing of quality threshold q at target cap m.

4.2 A fixed-state gradient mechanism

At a fixed model state and candidate policy, trajectory targets from one user share a history. Thus increasing targets per user and increasing independent users affect different sources of gradient variation. Throughout the training sweep, m denotes the nominal per-user target cap. In the fixed-state allocation model, m denotes the realized number of targets per user; when histories saturate before the cap, we report the dataset-level realized density $\bar{m}$. We call U the *supervision breadth* and m the *supervision density*. Let $V_{U,m}$ be the covariance trace of the update gradient, σ_b^2 the

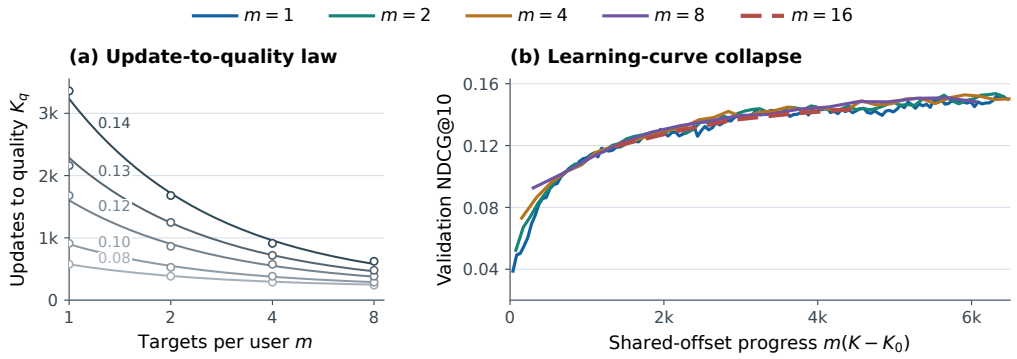


Figure 2: **Empirical update-to-quality scaling and learning-curve collapse.** ML-1M causal FMLP/full CE, five seeds. Left: $K_q(m) = K_0 + B_q/m$ is fitted to seedwise crossings at five quality thresholds, with shared $K_0 = 202.35$; circles mark crossings of the five-seed mean curves for $m = 1, 2, 4, 8$. Right: mean curves against $m(K - K_0)$; the $m = 16$ cap is excluded from fitting and its later measured segment is dashed. On five later measured points, retrospective mean absolute error (MAE) falls from .0139 without the offset to .0015 with it.

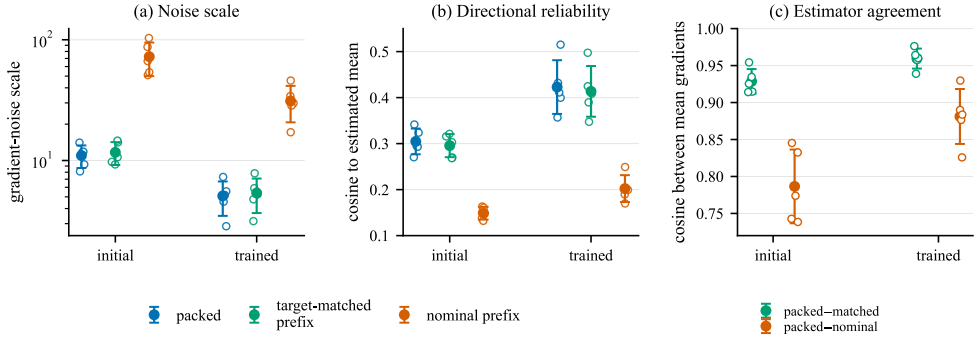


Figure 3: **Gradient geometry behind the density law.** Fixed-state diagnostics separate gradient noise scale, alignment with the estimated mean, and agreement before and after training for packed, count-matched aligned-prefix, and nominal prefix-minibatch updates. The panels make the decomposition operational: density changes the residual component, whereas breadth changes the number of independent histories.

variance of user-conditional mean gradients, σ_b^2 the average within-user target variance, and $\kappa_w \sigma_w^2$ the average within-user cross-covariance trace. Write $a = \sigma_b^2 + \kappa_w \sigma_w^2$ for the shared component and $d = (1 - \kappa_w)\sigma_w^2$ for the residual component. Appendix B.2 gives the sampling assumptions.

Proposition 4.1 (User and target contributions to gradient variance)**.**

$$V_{U,m} = \frac{1}{U}\left[\sigma_b^2 + \sigma_w^2\left(\kappa_w + \frac{1 - \kappa_w}{m}\right)\right] = \frac{a + d/m}{U}, \tag{7}$$

with the shared and residual components defined above.

This decomposition separates between-user variation, within-user cross-covariance, and residual within-user variation. Under fixed moments and $0 \le \kappa_w < 1$, only the residual term $d/(Um)$ changes with density; at fixed U, variance approaches a/U as m grows. For $d > 0$, the marginal reduction is $d/[Um(m+1)]$, so the fixed-state benefit diminishes with density. This relation supplies the variance term for the allocation analysis; the trained response is measured by the quality curves. Shared execution changes the cost of obtaining these gradients. For a fixed input-window length, let $c_{\text{traj}} > 0$ be the cost of encoding one trajectory and $c_{\text{tgt}} > 0$ the cost of scoring one target and its candidates. The per-update cost model is $C \simeq C_{\text{fixed}} + U(c_{\text{traj}} + c_{\text{tgt}} m)$; the fixed term is retained in timing fits but does not determine the continuous allocation below. With fixed variable compute budget C_0, let $\Psi(m)$ denote the cost-weighted variance numerator. The continuous allocation $U = C_0/(c_{\text{traj}} + c_{\text{tgt}} m)$ gives

$$V_{U,m} = \frac{\Psi(m)}{C_0}, \qquad \Psi(m) = (c_{\text{traj}} + c_{\text{tgt}} m)(a + d/m). \tag{8}$$

For $a, d > 0$, let m_* denote the density minimizing this conditional variance objective. It is

$$m_* = \sqrt{\frac{c_{\mathrm{traj}} d}{c_{\mathrm{tgt}} a}}. \tag{9}$$

Appendix B.3 proves this conditional fixed-state variance optimum and its integer counterpart. Equation (9) is the central allocation law: after estimating the two gradient-variance components and the two marginal costs, it predicts the density at which additional targets cease to compensate for their scoring cost. The coefficient scan in Table 33 estimates c_{traj} and c_{tgt} from the affine timing model, including a separate fixed GPU intercept. The resulting m^*_{state} supplies a measurable operating point from fixed-state statistics; the trained quality frontier then determines how this point moves with the available budget.

The same quantities connect this local law to a finite training budget. Let C be a training-only compute budget and let $\widehat{\tau}(m)$ be the measured seconds per update. Combining the progress coordinate in Equation (5) with $K \simeq C/\widehat{\tau}(m)$ gives the budget-conditioned frontier

$$\widehat{m}_{\mathrm{frontier}}(C) = \arg \max_{m \in \mathcal{M}(C)} \widehat{\Phi}\left(m \left(\frac{C}{\widehat{\tau}(m)} - \widehat{K}_0 \right) \right), \tag{10}$$

where $\mathcal{M}(C)$ contains caps whose realized density and measured trajectory remain within the protocol. If $\widehat{\tau}(m) = \tau_0 + \tau_1 m$ and $\widehat{\Phi}$ is increasing over the relevant range, the continuous progress maximizer over this feasible set is

$$\widetilde{m}_{\mathrm{frontier}}(C) = \left[\frac{\sqrt{C\tau_0/\widehat{K}_0} - \tau_0}{\tau_1} \right]_{\mathcal{M}(C)}, \tag{11}$$

where the brackets enforce the feasible cap range. Thus Eq. (9) predicts the local variance–cost allocation, while Eq. (10) predicts the density useful at a specified training budget. Their complementary roles make regime shifts measurable: changing the budget moves the quality frontier, while changing history length changes the realized density. The controlled ML-1M sweep gives $\widehat{\tau}(m) \simeq .026426 + .0001614m$ seconds ($R^2 = .977$); the SASRec sweeps test this distinction through their measured quality–time frontiers (Sections D and 6.2).

4.3 From gradient statistics to supervision efficiency

A fixed compute budget and a fixed per-update target count ask different questions. At fixed $M = Um$, Equation (7) becomes $V_{U,m} = (ma + d)/M$: broader user sampling reduces the shared component. Yet changing weighting, candidates, or loss can also change the mean update direction. The smoothness bound in Appendix B.4 separates alignment from noise (Equation (21)); gradient-noise and objective-gradient probes test these mechanisms, while validation trajectories measure their trained effect.

For a run reaching quality threshold q, let A_q count processed target events, including revisits; in the target-cap sweep of Section 4.1, we write this quantity as $A_q(m)$. Distinct-user data efficiency instead varies the available training histories. Using the validation-inclusive wall time W_q, define the average wall-clock cost per processed event as $\bar{c}_q^{\mathrm{wall}} = W_q/A_q$. Since $W_q = A_q \bar{c}_q^{\mathrm{wall}}$, the wall-time advantage between paired right-prefix and packed runs that both reach q separates into target-event count and per-event execution factors:

$$\frac{W_q^{\mathrm{right}}}{W_q^{\mathrm{pack}}} = \frac{A_q^{\mathrm{right}}}{A_q^{\mathrm{pack}}} \frac{\bar{c}_q^{\mathrm{wall,right}}}{\bar{c}_q^{\mathrm{wall,pack}}}. \tag{12}$$

The first factor measures how many target events are required to reach the same quality, while the second measures the wall-clock cost of processing each event.

5 Exact Prefix-Loss Compilation

EPC rewrites a realized materialized-prefix update $\mathcal{U}_k$ as one shared causal execution while holding its targets and loss semantics fixed. The theorem establishes objective and gradient equality for that fixed unit under prefix consistency and aligned training semantics. The target-cap sweep in Section 4.1 instead measures trained quality under different realized update sequences.

5.1 PREFIX CONSISTENCY AND EXACT TRANSFORMATION

Definition 5.1 (Prefix consistency)**.** An encoder f_θ^{causal} is prefix-consistent if, for every sequence $x_{1:L}$ and position $t \le L$, $f_\theta^{\text{causal}}(x_{1:L})_t = f_\theta^{\text{causal}}(x_{1:t})_t$. Thus, appending, removing, or changing a suffix after position t leaves its state unchanged under aligned positional and padding semantics.

Prefix consistency determines whether a shared state can replace the state obtained by encoding its prefix separately. EPC maps every materialized term in Equation (3) to its aligned state in Equation (4), while positions, padding, supervised targets, weights, candidates, loss semantics, reduction, and stochastic state remain aligned.

Theorem 5.2 (Exact prefix-loss compilation)**.** *Suppose that (i) every sequence-mixing block is prefix-consistent, (ii) positions and padding are aligned, (iii) supervised targets and weights are aligned, (iv) candidate construction, per-target loss semantics, and loss reduction are aligned, and (v) stochastic state is aligned between materialized and shared execution. Then, in exact arithmetic, aligned hidden states, logits, per-target losses, and the declared objective are identical. At every differentiable parameter point, this gives* $\mathcal{L}_{\text{packed}}(\theta) = \mathcal{L}_{\text{prefix}}(\theta)$ *and* $\nabla_\theta \mathcal{L}_{\text{packed}}(\theta) = \nabla_\theta \mathcal{L}_{\text{prefix}}(\theta)$.

Prefix-consistent mixing gives identical supervised states; the same head, aligned loss semantics, and reduction ρ therefore preserve the declared objective and its gradient. Appendix A gives the complete proof, including padding, stochastic coupling, reduction, and nondifferentiable points. The equality is per realized update: identical trajectories additionally require matched parameters, optimizer state, unit sequence, stochastic draws, and optimizer rule.

The same condition is necessary for universal reuse across linear scoring heads: a changed shared state admits a head with a different cross-entropy loss. Appendix B proves this necessity and the trajectory corollary. We therefore test suffix dependence by future-item perturbation and test exactness by aligned-prefix hidden-state, loss, and gradient comparisons; stochastic draws are coupled explicitly.

5.2 CAUSAL VARIANTS AND COMPUTATION SAVINGS

For the five future-dependent operator families studied in our experiments, we construct prefix-consistent causal variants by replacing suffix-dependent computations with operations that use only current and preceding positions. EPC exactness then compares materialized-prefix and shared execution within the same causal variant, not the causal variant with the original future-dependent model. Operator-specific constructions and provenance are in Appendix C.1.

Let $c_{\text{enc}}(t)$ denote the encoder cost of a length-t prefix and let $N = T - 1$ be the number of supervised transitions. Separate execution costs $\sum_{t=1}^{N} c_{\text{enc}}(t)$, whereas EPC uses one pass costing $c_{\text{enc}}(N)$. For $c_{\text{enc}}(t) = \gamma t^p$ with $\gamma > 0$ and $p \ge 1$, their encoder-cost ratio is asymptotically $N/(p+1)$ (Appendix B): linear-cost encoders change from quadratic to linear total cost, and dense self-attention from cubic to quadratic. Candidate scoring remains per target, while storing all prefixes costs $\Theta(N^2)$ tokens instead of $\Theta(N)$ for one trajectory.

6 EXPERIMENTS

The ML-1M causal-FMLP/full-catalog CE sweep tests whether changing target cap produces the fitted update-to-quality relation. Separate fixed-state scans measure local gradient variance, fixed-unit comparisons test EPC's objective and gradient equivalence, and practical runs measure the runtime and ranking effects of changed grouping or candidate policies.

6.1 PROTOCOL

We use MovieLens ML-1M (Harper & Konstan, 2016), Amazon Beauty, Toys, and Sports (McAuley et al., 2015), Steam (Kang & McAuley, 2018), and Gowalla (Cho et al., 2011) with leave-one-out targets, full-catalog evaluation, and either full-catalog cross-entropy (full CE) or shared-pool InfoNCE as the training objective; ML-10M (Harper & Konstan, 2016) supplies the scale study. Full-catalog evaluation ranks every item rather than a sampled candidate subset, whose measured values need not reproduce the full ranking (Krichene & Rendle, 2020). Primary comparisons use five paired seeds (2027–2031). We compare eight sequential-recommender backbones: SASRec (Kang & McAuley, 2018), NextItNet (Yuan et al., 2019), RecBLR (Liu et al., 2024), FMLP (Zhou et al., 2022), FEARec (Du et al., 2023), BSARec (Shin et al., 2024), TV-Rec (Shin et al., 2025), and LinRec (Liu et al., 2023). The first three are native-causal; the remaining five use the causal variants in Section 5.2. NDCG@10 is primary. Train-to-best is validation-inclusive elapsed wall time through each run's validation-selected checkpoint; the scale study uses a paired common-quality endpoint. By contrast, times in the density-budget table are training-only and exclude validation. Distinct-user data efficiency

Table 1: **Cross-dataset SASRec density sweep.** Each entry is a five-seed mean ± standard deviation. $\bar{m}$ is the realized number of targets per user per epoch; it differs from the nominal cap when histories saturate. Training time is the mean training-only time through the selected checkpoint. The $m = 16$ arm is fit-excluded from the shared-offset scaling fits.

Dataset	Cap m	Realized $\bar{m}$	NDCG@10 ↑	Train (s) ↓	Role
ML-10M	1	1.00	.1168 ± .0012	735.7	fit
	2	2.00	.1189 ± .0011	412.6	fit
	4	4.00	.1201 ± .0011	221.3	fit
	8	8.00	.1189 ± .0013	123.8	fit
	16	16.00	.1181 ± .0006	65.7	held-out
Beauty	1	1.00	.0361 ± .0006	226.9	fit
	2	2.00	.0366 ± .0012	152.8	fit
	4	3.17	.0402 ± .0004	96.7	fit
	8	4.25	.0435 ± .0006	61.3	fit
	16	5.05	.0463 ± .0008	54.8	held-out

is measured separately by varying the number of available training histories (Appendix C.2.1). Implementation details, timing conditions, and extended controls are in Appendix C.

Using the terminology in Section 3, right-prefix training separately encodes right-aligned prefixes, whereas aligned-prefix execution retains the shared graph's positional indices and accumulates the prescribed target losses before an optimizer step. Exactness tests use aligned-prefix execution; practical right-prefix-versus-packed comparisons also differ in target grouping and positional alignment.

6.2 Supervision-density scaling and mechanism

We vary target cap m during training while holding the ML-1M users, causal FMLP, full-catalog CE, paired initial states, and target-event budget fixed. The primary test is whether quality curves align in the coordinate $m(K - K_0)$ and whether threshold crossings share the offset in Equation (5).

The four fitted caps yield $K_0 = 202.35$ updates (paired-seed bootstrap 95% interval 184.24–218.46). The fit-excluded $m = 16$ later segment aligns retrospectively (Figure 2); four of its five seeds cross $q = .14$ within budget.

At $q = .14$, raising the cap from $m = 1$ to $m = 16$ increases target events to crossing from 408.3k to 918.1k but reduces training-only time from 87.2 to 13.2 seconds; the $m = 16$ value uses four crossings (Table 32). At approximately 0.966 million events, $m = 2$ reaches NDCG@10 .1519 in 102.8 training seconds, versus .1498 in 202.5 seconds for $m = 1$; $m = 4$ reaches .1500 in 52.3 seconds. These caps define a measured quality–time frontier, distinct from the conditional fixed-state variance optimum in Equation (9).

On ML-1M, fixed-state causal FMLP and SASRec scans each evaluate 20 (U, m) allocations with 128 gradient draws per cell, sampled from users with at least 16 targets. The held-out relative RMSE of Equation (7) is 2.4% for FMLP and 1.5% for SASRec. In a separate full-CE probe, matching the mean 726 targets per update lowers gradient noise scale (GNS) from 31.11 for nominal prefix training to 5.38, near packed training's 5.09; the count-matched rows sample different target identities. These fixed-state results characterize gradient variation, while the paired threshold crossings above measure the trained $K_q(m)$ relation. Appendix D, Table 21, and Figure 3 give the scan conditions and diagnostics.

Cross-dataset SASRec audit. We repeat the cap sweep with the causal SASRec backbone on ML-10M and Amazon Beauty, keeping the full-catalog CE objective, five paired seeds, and approximately fixed target-event budgets within each dataset. Because the Beauty histories saturate before the nominal cap, we report the realized density $\bar{m}$ alongside the cap. Table 1 shows two distinct trained compute frontiers: ML-10M reaches its highest measured NDCG@10 at $m = 4$ (.1201) and then trades quality for speed as the cap grows, whereas Beauty improves monotonically through $m = 16$ (.0463) while reducing training-only time from 226.9 to 54.8 seconds. Thus the nominal cap is not itself the compute-optimal choice; the observed regime depends on the dataset's realized history density.

The fixed-state scan makes this distinction quantitative. Direct fits give $(a, d, c_{\mathrm{traj}}, c_{\mathrm{tgt}}) = (3.660, 137.318, 1.023{\times}10^{-5}, 2.384{\times}10^{-7})$ for ML-10M and $(2.338, 201.639, 9.493{\times}10^{-6}, 3.312{\times}10^{-7})$ for Beauty, yielding the Eq. (9) predictions $m^*_{\mathrm{state}} = 40.1$ and 49.7 (Table 33). Both datasets therefore remain on the density-increasing side of the fixed-state law over the tested caps. The measured budget frontiers then resolve their trained operating points: ML-10M has three nondominated points $(m = 4, 8, 16)$ in the quality–time plane, with its highest measured quality at $m = 4$, whereas Beauty has $m = 16$ as its sole nondominated point because realized density rises to $\bar{m} = 5.05$. The fit-excluded $m = 16$ trajectory improves the shifted-curve MAE from $.0064$ to $.0014$ on ML-10M; Beauty's realized-density frontier is reported separately (Appendix D).

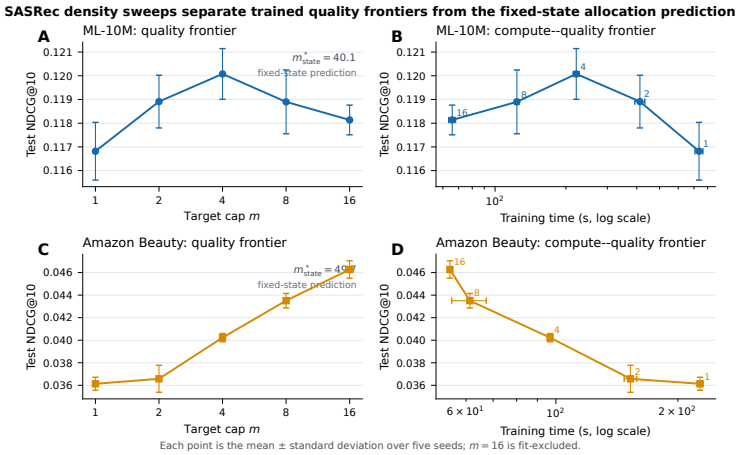


Figure 4: **SASRec density and compute frontiers.** Five-seed means ± standard deviation against nominal cap (A/C) and training-only time (B/D); labels show the budget frontier and fixed-state prediction. The $m = 16$ arm is fit-excluded.

Table 2: **FMLP speed across six datasets.** Mean NDCG@10, HIT@10, and training time in seconds over five seeds under shared-pool InfoNCE; full standard deviations are in the appendix. Prefix is materialized and Packed is shared execution. Beauty released FMLP/BCE is in Table 4.

	Prefix			Packed			
Dataset	NDCG@10	HIT@10	Train (s)	NDCG@10	HIT@10	Train (s)	Speedup
ML-1M	.1509	.2742	982.2	.1434	.2602	55.4	17.7×
Beauty	.0507	.0906	323.3	.0521	.0934	71.4	4.5×
Toys	.0520	.0902	286.9	.0538	.0931	66.5	4.3×
Sports	.0327	.0593	534.0	.0315	.0569	96.3	5.5×
Steam	.0470	.0919	28298.4	.0636	.1210	1996.3	14.2×
Gowalla	.0497	.0910	1351.4	.0563	.1022	190.6	7.1×

6.3 From exact reuse to end-to-end efficiency

The density law identifies why denser updates can improve progress; we next ask whether shared execution delivers that policy without changing the declared update. Across 48 shared-pool InfoNCE cells, packing shortens train-to-best time by 4.80–10.37×, while ranking effects remain dataset-dependent: right-prefix means are higher on ML-1M, whereas packed means are higher on Steam (Appendix Table 36 and Figure 13). Exactness audits find zero hidden-state and loss discrepancies in Float64 and a maximum gradient discrepancy of 1.2×10^{-13}; under matched targets, positions, reduction, and full-CE semantics, EPC changes NDCG@10 from .0534 to .0536 while reducing train-to-best time from 46.4 to 15.1 seconds (Appendix Table 34). Policy controls separate this execution gain from training-policy effects, including the Beauty progression from released BCE to packed full CE: NDCG@10 rises by 52.8% and the selected checkpoint arrives 50.9× sooner, with the causal architecture fixed (Appendix 35). At a paired common-quality endpoint, ML-10M reaches 95% in all five seeds with 1.30–4.56× fewer events and 28.70–87.42× less wall time; the remaining controls and scale-specific outcomes are reported in Appendix Table 37 and Sections C.2.1 and C.3. For a direct model-level view, Table 2 pairs FMLP prefix and packed execution on all six practical-sweep datasets: packing is 4.3–17.7× faster while preserving or improving both ranking metrics on four datasets. The released Beauty FMLP/BCE baseline remains separate because it uses a different objective (Table 4).

7 Conclusion

Supervision density connects data organization to optimization efficiency. Our matched ML-1M experiments establish the empirical scaling relation $K_q(m) \simeq K_0 + B_q/m$, linking density to updates needed for a target quality. The variance–cost model guides local allocation, while SASRec sweeps reveal different quality–time frontiers across datasets. EPC makes shared execution exact under prefix consistency and aligned training semantics. Together, these findings give the community a design principle: choose how many targets share a history by jointly considering gradient variance, progress per update, and execution cost. Sample efficiency therefore depends on how available supervision is organized and computed, alongside how much training data is available.

AI Use Statement. LLM-based assistants were used only for language polishing. The authors conceived the question and method, designed and executed the experiments, made all scientific judgments, verified the source code and raw logs, and remain responsible for every claim in this paper.

REFERENCES

Newsha Ardalani, Carole-Jean Wu, Zeliang Chen, Bhargav Bhushanam, and Adnan Aziz. Understanding scaling laws for recommendation models, 2022. URL https://arxiv.org/abs/2208.08489.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018. doi: 10.1137/16M1080173. URL https://doi.org/10.1137/16M1080173.

Eunjoon Cho, Seth Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21–24, 2011*, pp. 1082–1090. ACM, 2011. doi: 10.1145/2020408.2020579. URL https://doi.org/10.1145/2020408.2020579.

Yizhou Dang, Yifan Wu, Minhan Huang, Chuang Zhao, Lianbo Ma, Guibing Guo, Xingwei Wang, and Zhu Sun. Pay attention to sequence split: Uncovering the impacts of sub-sequence splitting on sequential recommendation models. In Alistair Moffat, Falk Scholer, Hannah Bast, Marc Najork, and Min Zhang (eds.), *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2026, Melbourne, VIC, Australia, July 20-24, 2026*, pp. 2856–2867. ACM, 2026. doi: 10.1145/3805712.3808548. URL https://doi.org/10.1145/3805712.3808548.

Xinyu Du, Huanhuan Yuan, Pengpeng Zhao, Jianfeng Qu, Fuzhen Zhuang, Guanfeng Liu, Yanchi Liu, and Victor S. Sheng. Frequency enhanced hybrid attention network for sequential recommendation. In Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete (eds.), *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pp. 78–88. ACM, 2023. doi: 10.1145/3539618.3591689. URL https://doi.org/10.1145/3539618.3591689.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013. doi: 10.1137/120880811. URL https://doi.org/10.1137/120880811.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15706–15734. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/gloeckle24a.html.

F. Maxwell Harper and Joseph A. Konstan. The MovieLens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, 2016. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06939.

Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pp. 197–206. IEEE Computer Society, 2018. doi: 10.1109/ICDM.2018.00035. URL https://doi.org/10.1109/ICDM.2018.00035.

Anton Klenitskiy and Alexey Vasilev. Turning dross into gold loss: is BERT4Rec really better than SASRec? In Jie Zhang, Li Chen, Shlomo Berkovsky, Min Zhang, Tommaso Di Noia, Justin Basilico, Luiz Pizzato, and Yang Song (eds.), *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18-22, 2023*, pp. 1120–1125. ACM, 2023. doi: 10.1145/3604915.3610644. URL https://doi.org/10.1145/3604915.3610644.

Mario Michael Krell, Matej Kosec, Sergio P. Perez, and Andrew Fitzgibbon. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *CoRR*, abs/2107.02027, 2021. URL https://arxiv.org/abs/2107.02027.

Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23–27, 2020*, pp. 1748–1757. ACM, 2020. doi: 10.1145/3394486.3403226. URL https://doi.org/10.1145/3394486.3403226.

Geon Lee, Bhuvesh Kumar, Mingxuan Ju, Tong Zhao, Kijung Shin, Neil Shah, and Liam Collins. Sequential data augmentation for generative recommendation. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining, WSDM 2026, Boise, ID, USA, February 22-26, 2026*, pp. 303–312. ACM, 2026. doi: 10.1145/3773966.3778000. URL https://doi.org/10.1145/3773966.3778000.

Chengkai Liu, Jianghao Lin, Hanzhou Liu, Jianling Wang, and James Caverlee. Behavior-dependent linear recurrent units for efficient sequential recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21–25, 2024*, pp. 1430–1440. ACM, 2024. doi: 10.1145/3627673.3679717. URL https://doi.org/10.1145/3627673.3679717.

Langming Liu, Liu Cai, Chi Zhang, Xiangyu Zhao, Jingtong Gao, Wanyu Wang, Yifu Lv, Wenqi Fan, Yiqi Wang, Ming He, Zitao Liu, and Qing Li. Linrec: Linear attention mechanism for long-term sequential recommender systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23–27, 2023*, pp. 289–299. ACM, 2023. doi: 10.1145/3539618.3591717. URL https://doi.org/10.1145/3539618.3591717.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In Ricardo Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto (eds.), *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pp. 43–52. ACM, 2015. doi: 10.1145/2766462.2767755. URL https://doi.org/10.1145/2766462.2767755.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *CoRR*, abs/1812.06162, 2018. URL https://arxiv.org/abs/1812.06162.

Gleb Mezentsev, Danil Gusak, Ivan V. Oseledets, and Evgeny Frolov. Scalable cross-entropy loss for sequential recommendations with large item catalogs. In Tommaso Di Noia, Pasquale Lops, Thorsten Joachims, Katrien Verbert, Pablo Castells, Zhenhua Dong, and Ben London (eds.), *Proceedings of the 18th ACM Conference on Recommender Systems, RecSys 2024, Bari, Italy, October 14-18, 2024*, pp. 475–485. ACM, 2024. doi: 10.1145/3640457.3688140. URL https://doi.org/10.1145/3640457.3688140.

Aleksandr V. Petrov and Craig Macdonald. Effective and efficient training for sequential recommendation using recency sampling. In Jennifer Golbeck, F. Maxwell Harper, Vanessa Murdock, Michael D. Ekstrand, Bracha Shapira, Justin Basilico, Keld T. Lundgaard, and Even Oldridge (eds.), *RecSys '22: Sixteenth ACM Conference on Recommender Systems, Seattle, WA, USA, September 18 - 23, 2022*, pp. 81–91. ACM, 2022. doi: 10.1145/3523227.3546785. URL https://doi.org/10.1145/3523227.3546785.

Aleksandr Vladimirovich Petrov and Craig Macdonald. gSASRec: Reducing overconfidence in sequential recommendation trained with negative sampling. In Jie Zhang, Li Chen, Shlomo Berkovsky, Min Zhang, Tommaso Di Noia, Justin Basilico, Luiz Pizzato, and Yang Song (eds.), *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore,*

Singapore, September 18-22, 2023, pp. 116–128. ACM, 2023. doi: 10.1145/3604915.3608783. URL https://doi.org/10.1145/3604915.3608783.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In Jeff A. Bilmes and Andrew Y. Ng (eds.), *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pp. 452–461. AUAI Press, 2009. URL https://www.auai.org/uai2009/papers/UAI2009_0139_48141db02b9f0b02bc7158819ebfa2c7.pdf.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150, 2019. URL https://arxiv.org/abs/1911.02150.

Yehjin Shin, Jeongwhan Choi, Hyowon Wi, and Noseong Park. An attentive inductive bias for sequential recommendation beyond the self-attention. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 8984–8992. AAAI Press, 2024. doi: 10.1609/AAAI.V38I8.28747. URL https://doi.org/10.1609/aaai.v38i8.28747.

Yehjin Shin, Jeongwhan Choi, Seojin Kim, and Noseong Park. TV-Rec: Time-variant convolutional filter for sequential recommendation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025*, 2025. URL https://openreview.net/forum?id=cWEssTIwG5.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (eds.), *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pp. 1441–1450. ACM, 2019. doi: 10.1145/3357384.3357895. URL https://doi.org/10.1145/3357384.3357895.

Jiaxi Tang and Ke Wang. Personalized Top-N sequential recommendation via convolutional sequence embedding. In Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek (eds.), *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pp. 565–573. ACM, 2018. doi: 10.1145/3159652.3159656. URL https://doi.org/10.1145/3159652.3159656.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. Contrastive learning for sequential recommendation. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*, pp. 1259–1273. IEEE, 2022. doi: 10.1109/ICDE53745.2022.00099. URL https://doi.org/10.1109/ICDE53745.2022.00099.

Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11–15, 2019*, pp. 582–590. ACM, 2019. doi: 10.1145/3289600.3290975. URL https://doi.org/10.1145/3289600.3290975.

Gaowei Zhang, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji-Rong Wen. Scaling law of large sequential recommendation models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys '24, pp. 444–453. ACM, October 2024. doi: 10.1145/3640457.3688129. URL https://doi.org/10.1145/3640457.3688129.

Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. Filter-enhanced MLP is all you need for sequential recommendation. In Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (eds.), *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pp. 2388–2399. ACM, 2022. doi: 10.1145/3485447.3512111. URL https://doi.org/10.1145/3485447.3512111.

A PROOF OF THEOREM 5.2

The proof establishes equality of the declared loss and gradient for two executions of the same update. It proceeds from aligned inputs to identical hidden states, then to equality of logits, losses, and parameter derivatives. Throughout, "execution" refers to an embedding layer, N_{blk} stacked blocks of sequence-mixing and position-wise operations, and a scoring head. Both executions share one parameter vector, denoted by θ.

Theorem (Exact prefix-loss compilation; restatement of Theorem 5.2). Fix a trajectory $x_{1:T}$, weights $\{w_t\}_{t=1}^{T-1}$, a per-target loss ℓ, and a scoring head g_θ. Assume both executions use the same shared-parameter network: an embedding layer, N_{blk} stacked blocks that pair a sequence-mixing operation with position-wise maps, and the position-wise head g_θ. The materialized row's prediction is therefore $f_\theta(x_{1:t}) = g_\theta(f_\theta^{\mathrm{causal}}(x_{1:t})_t)$. We require five conditions. (C1) Every sequence-mixing block of $f_\theta^{\mathrm{causal}}$ is prefix-consistent (Definition 5.1) at every parameter value. (C2) The packed and materialized executions use identical positional indexing and padding semantics. Thus, for every $s \le t$, position s of the packed pass and position s of the materialized row for $x_{1:t}$ receive identical input embeddings and masks, and padding cannot enter the mixing computation at a supervised position. (C3) Both executions supervise identical target positions with identical weights. (C4) Both use identical per-target loss semantics, loss reduction, and candidate construction for every target. (C5) Execution is deterministic, or all random draws on the dependency path of a supervised position are coupled position by position. Then, in exact arithmetic, for every $t \le T-1$:

(i) $f_\theta^{\mathrm{causal}}(x_{1:T-1})_t = f_\theta^{\mathrm{causal}}(x_{1:t})_t$ (aligned hidden states);

(ii) $g_\theta\big(f_\theta^{\mathrm{causal}}(x_{1:T-1})_t\big) = g_\theta\big(f_\theta^{\mathrm{causal}}(x_{1:t})_t\big)$ (logits);

(iii) $\ell\big(g_\theta(f_\theta^{\mathrm{causal}}(x_{1:T-1})_t), x_{t+1}\big) = \ell\big(g_\theta(f_\theta^{\mathrm{causal}}(x_{1:t})_t), x_{t+1}\big)$ (per-target losses);

(iv) $\mathcal{L}_{\mathrm{packed}}(\theta) = \mathcal{L}_{\mathrm{prefix}}(\theta)$ (weighted objective);

(v) $\nabla_\theta \mathcal{L}_{\mathrm{packed}}(\theta) = \nabla_\theta \mathcal{L}_{\mathrm{prefix}}(\theta)$ wherever the objective is differentiable (parameter gradients).

Proof. Write the packed execution's layer-r state at position s as $h_s^{(r)}(x_{1:T-1})$ and the materialized execution's layer-r state at position s of the row for prefix $x_{1:t}$ as $\tilde{h}_s^{(r)}(x_{1:t})$, for $s \le t$.

Step 1 (input layer). By (C2), the row for $x_{1:t}$ places item x_s at the same positional index s, with the same positional embedding and padding mask, as the packed pass. By (C5), any input-level stochastic operation, such as embedding dropout, draws identical masks at position s in both executions. Hence $\tilde{h}_s^{(0)}(x_{1:t}) = h_s^{(0)}(x_{1:T-1})$ for all $s \le t$. Thus, the two executions receive identical embedded inputs at every position in their shared prefix.

Step 2 (induction over layers). Assume $\tilde{h}_s^{(r)}(x_{1:t}) = h_s^{(r)}(x_{1:T-1})$ for all $s \le t$. A block applies a sequence-mixing operation followed by position-wise maps. Prefix consistency (C1) states that the mixing output at position s depends only on inputs at positions $\le s$. Those inputs agree by the induction hypothesis. The materialized row contains positions $1, \dots, t$, and (C1) prevents positions $> t$ of the packed input from influencing any $s \le t$. The mixing outputs therefore agree at every such position. Position-wise maps, including feed-forward layers, per-position normalization, residual addition, and activation, apply the same function to equal inputs. Coupled randomness under (C5) preserves this equality. By induction, $\tilde{h}_s^{(N_{\mathrm{blk}})}(x_{1:t}) = h_s^{(N_{\mathrm{blk}})}(x_{1:T-1})$ for all $s \le t$; setting $s = t$ proves the aligned hidden-state statement in claim (i).

Step 3 (padding positions). Padded positions of a materialized row, if any, occupy indices with no supervision by (C3) and, by (C2), carry masks that prevent them from entering the mixing computation of supervised positions under (C1)-compatible masking. Consequently, padding changes neither the aligned state in claim (i) nor any supervised loss.

Step 4 (logits and losses). The head g_θ applies the same position-wise map to each final state, with shared parameters such as tied item embeddings, so (i) gives (ii). By (C4), target x_{t+1} uses the same per-target loss semantics and candidate set in both executions, or identically coupled sampled candidates under (C5). The per-target loss also uses the same reduction. Claim (iii) follows because the two executions apply the same loss function to identical logits, targets, and candidates.

Step 5 (objective). By (C3), both executions have identical target-indexed weighted losses. Condition (C4) applies the same reduction ρ to those terms, giving $\mathcal{L}_{\mathrm{packed}}(\theta) = \mathcal{L}_{\mathrm{prefix}}(\theta)$. For sum reduction, this identity reads

$$\sum_t w_t\, \ell(g_\theta(f_\theta^{\mathrm{causal}}(x_{1:T-1})_t), x_{t+1}) = \sum_t w_t\, \ell(g_\theta(f_\theta^{\mathrm{causal}}(x_{1:t})_t), x_{t+1}).$$

The identical-input argument also covers mean reductions when their denominators and grouping agree.

Step 6 (gradients). Fix t and consider its weighted loss $\phi_t(\theta) = w_t\,\ell(g_\theta(f_\theta^{\mathrm{causal}}(\cdot)_t), x_{t+1})$ in each execution. Steps 1–4 establish more than equality at one parameter point: both executions compose the same parameterized maps on the same inputs with the same coupled random draws. Because (C1) holds at every parameter value, the induction in Step 2 is an identity in θ when the coupled draws are fixed. Every supervised term therefore coincides as a function of θ. Applying the same reduction ρ to the same terms gives identical objective functions and hence identical gradients wherever differentiable, proving (v). At a nondifferentiable point, agreement of automatic-differentiation outputs additionally requires the same local derivative convention; it does not follow from equality of function values alone. □

Remark A.1 (Scope). The theorem equates the value and gradient of the declared objective at a shared parameter point in exact arithmetic. The audits in Section 6.3 measure floating-point disagreement directly. Update grouping, finite-batch sampling, dropout reuse, optimizer state, and candidate coupling instead define the practical training policy analyzed in Section 4. Thus, the empirical analysis treats execution savings and optimization-policy effects separately.

B WHY EXACT REUSE ENABLES DENSER OPTIMIZATION

The results below connect the state identity in Theorem 5.2 to the design of the training algorithm. The first identifies when a state can be reused across prediction heads, the second transfers the resulting updates, and the third quantifies the encoder work saved. We then derive the compute-budget bound used in the main text.

Proposition B.1 (Necessity for universal state reuse)**.** *Fix a sequence, supervised position, parameter value, and realized stochastic state. Let h be its shared representation and $\tilde{h}$ its separately encoded prefix representation. For a catalog of at least two items and a fixed target item, if these representations yield identical full-catalog cross-entropy losses for every linear scoring head, then $h = \tilde{h}$.*

Proof. Suppose $\delta = h - \tilde{h} \neq 0$. Choose a linear head with row $\delta^\top$ for the target and zero rows for the other $J-1$ catalog items, where $J \geq 2$. For a representation v, its cross-entropy loss is $\log(1 + (J-1)\exp(-\delta^\top v))$, a strictly decreasing function of $\delta^\top v$. Since $\delta^\top h - \delta^\top \tilde{h} = \|\delta\|^2 > 0$, the two losses differ, contradicting the assumed equality. Thus $h = \tilde{h}$. Requiring this property at every sequence, position, and model state gives prefix consistency. □

This is a necessity statement for reuse across arbitrary linear scoring heads, already with the full-CE objective used in our controls. A particular fixed head may map unequal states to the same prediction. EPC uses state equality to obtain a reusable guarantee, and the causal variants in Section 5.2 construct that equality for operators whose original states depend on future positions.

Proposition B.2 (Trajectory preservation and encoder savings)**.** *Suppose every update satisfies Theorem 5.2, both executions start from identical parameters and optimizer state, and they use the same realized unit sequence and optimizer rule. Their parameters and optimizer states agree at every update in exact arithmetic. For N supervised prefixes and encoder cost $c_{\mathrm{enc}}(t) = \gamma t^p$, $\gamma > 0$, $p \geq 1$, let R_N denote the separate/shared encoder-cost ratio:*

$$R_N = \frac{\sum_{t=1}^{N} \gamma t^p}{\gamma N^p} = \frac{N}{p+1} + O(1).$$

Proof. At update zero the states agree by assumption. If they agree at update k, the common unit and coupled stochastic state give equal gradients by Theorem 5.2. Applying the same optimizer map to equal parameters, optimizer states, and gradients gives equality at update $k+1$. Induction proves the first claim, including momentum or adaptive state when it is part of the common optimizer map.

Separate execution evaluates each length-t prefix once, while shared execution evaluates length N once. Since x^p is increasing,

$$\int_0^N x^p\,dx \leq \sum_{t=1}^{N} t^p \leq \int_1^{N+1} x^p\,dx.$$

Dividing by N^p yields the claimed asymptotic ratio. For linear cost, the exact ratio is $(N+1)/2$; for quadratic cost it is $(N+1)(2N+1)/(6N)$. □

If both implementations also incur a common non-encoder cost S_N, their total modeled ratio is $(\sum_t c_{\text{enc}}(t) + S_N)/(c_{\text{enc}}(N) + S_N)$. Candidate scoring and other shared costs therefore determine how much of the encoder saving appears in measured runtime. The empirical timing and memory controls report that realized saving.

B.1 Target-count and compute-budget bounds

For vector gradients, write $V_M = \operatorname{tr}\operatorname{Cov}(\hat{g}_M)$. If $\hat{g}_M = M^{-1}\sum_i g_i$, every target has covariance trace σ^2, and every distinct pair has cross-covariance trace $\kappa\sigma^2$, then

$$V_M = \sigma^2\left[\kappa + \frac{1-\kappa}{M}\right]. \tag{13}$$

For $0 \le \kappa < 1$, the variance reduction from one additional target is $\sigma^2(1-\kappa)/[M(M+1)]$. In the fixed-state probe we report the gradient noise scale $\text{GNS} = V_M/\|\mathbb{E}\hat{g}_M\|^2$ (McCandlish et al., 2018); Appendix C.2.1 describes its finite-sample estimator.

Proposition B.3 (Optimization under a compute budget)**.** *Let F be β-smooth and bounded below by F_*, and let stochastic gradient descent use conditionally unbiased gradients with conditional total variance at most V_M. With step size $0 < \eta \le 1/\beta$ and fixed cost c_M per update, budget B permits $K = \lfloor B/c_M \rfloor \ge 1$ updates and satisfies*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla F(\theta_k)\|^2 \le \frac{2(F(\theta_0)-F_*)}{\eta K} + \beta\eta V_M. \tag{14}$$

This separates the update count afforded by execution from the noise of each update. Its smoothness proof follows the standard nonconvex SGD argument (Ghadimi & Lan, 2013; Bottou et al., 2018). Let $\mathcal{F}_k$ contain the history before update k. Assume $\mathbb{E}[\hat{g}_k \mid \mathcal{F}_k] = \nabla F(\theta_k)$ and $\mathbb{E}[\|\hat{g}_k - \nabla F(\theta_k)\|^2 \mid \mathcal{F}_k] \le V_M$ uniformly over the K updates. For $\theta_{k+1} = \theta_k - \eta\hat{g}_k$, β-smoothness gives the standard descent inequality (Ghadimi & Lan, 2013; Bottou et al., 2018):

$$\mathbb{E}[F(\theta_{k+1}) \mid \mathcal{F}_k] \le F(\theta_k) - \eta\left(1 - \frac{\beta\eta}{2}\right)\|\nabla F(\theta_k)\|^2 + \frac{\beta\eta^2}{2}V_M.$$

Use $\eta \le 1/\beta$, take total expectations, sum over $k = 0, \dots, K-1$, and bound $\mathbb{E}F(\theta_K) \ge F_*$. Rearranging yields Equation (14); substituting $K = \lfloor B/c_M \rfloor$ gives Proposition B.3.
For the target-average model in Equation (13), expand

$$\operatorname{tr}\operatorname{Cov}\left(\frac{1}{M}\sum_i g_i\right) = \frac{1}{M^2}\left(M\sigma^2 + M(M-1)\kappa\sigma^2\right).$$

This proves the variance expression and, by subtraction, $V_M - V_{M+1} = \sigma^2(1-\kappa)/[M(M+1)]$. To use it as the uniform bound in Proposition B.3, the corresponding conditional covariance bound must hold along the trajectory. With fixed η, initial objective gap, and budget, configurations satisfying $K_2 \ge K_1$ and $V_2 \le V_1$ have an ordered upper bound in Equation (14). This establishes the joint cost–variance design criterion. Actual ranking quality is evaluated through the common-quality experiments, while the fixed-state gradient diagnostic tests the variance mechanism locally.

B.2 Hierarchical gradient covariance

We specify the model behind Proposition 4.1. Fix parameters θ and a candidate policy under which an individual target gradient does not depend on the other sampled users, as in full CE. Draw user histories $Z_1, \dots, Z_U$ independently from a common population. Conditional on a history Z_u, draw m target gradients $g_{u,1}, \dots, g_{u,m}$ with common conditional mean $\mu(Z_u)$. Different user groups are independent. Within a group, assume equal marginal covariance traces and equal pairwise cross-covariance traces after averaging over Z:

$$\operatorname{tr}\operatorname{Cov}_Z(\mu(Z)) = \sigma_b^2, \qquad \mathbb{E}_Z \operatorname{tr}\operatorname{Cov}(g_{u,j} \mid Z) = \sigma_w^2,$$

$$\mathbb{E}_Z \operatorname{tr}\operatorname{Cov}(g_{u,i}, g_{u,j} \mid Z) = \kappa_w\sigma_w^2 \quad (i \ne j).$$

These population-averaged moments define our homogeneous sampling model. The allocation result holds them fixed over a feasible range of m, with valid covariance matrices throughout. For the design interpretation, we take $0 \le \kappa_w < 1$, separating the shared variation from the residual variation reduced by averaging targets.

Let $\bar{g}_u = m^{-1}\sum_j g_{u,j}$ and $\hat{g} = (1/U)\sum_u \bar{g}_u$. Applying total covariance to the user average separates variation across histories from the remaining conditional variation within each history:

$$\operatorname{tr}\operatorname{Cov}(\bar{g}_u) = \operatorname{tr}\operatorname{Cov}_Z(\mu(Z)) + \mathbb{E}_Z \operatorname{tr}\operatorname{Cov}(\bar{g}_u \mid Z) = \sigma_b^2 + \frac{\sigma_w^2}{m}[1 + (m-1)\kappa_w].$$

Independence across groups divides this expression by U, proving Equation (7). Subtracting adjacent values of m gives $d/[Um(m+1)]$. The mean remains $\mathbb{E}_Z\mu(Z)$ throughout; changes in this mean would constitute a different comparison.

For variable history lengths or user weights, apply the weighted-estimator decomposition in Appendix C.2.1; it retains the actual contribution of each history and the random target-count denominator. For user sampling without replacement, include the finite-population covariance correction. These choices connect the covariance calculation to the sampling and reduction policy used in training.

B.3 Optimal allocation under shared and materialized-prefix execution

Assume the same model state, objective, user population, and per-user target sampling distribution as above. Fix input-window length so that selecting more supervised positions changes scoring cost but not the length encoded by the shared pass. Let a user cost $c_{\mathrm{traj}} + c_{\mathrm{tgt}}m$ under shared execution, with trajectory-encoding cost $c_{\mathrm{traj}} > 0$ and per-target scoring cost $c_{\mathrm{tgt}} > 0$. In the continuous relaxation with update cost C_0, choose $U = C_0/(c_{\mathrm{traj}} + c_{\mathrm{tgt}}m)$. Then

$$C_0 V_{U,m} = \Psi(m) = c_{\mathrm{traj}}a + c_{\mathrm{tgt}}d + c_{\mathrm{tgt}}a\,m + \frac{c_{\mathrm{traj}}d}{m}. \tag{15}$$

For $a, d > 0$, differentiating the allocation criterion gives its slope and strictly positive curvature:

$$\Psi'(m) = c_{\mathrm{tgt}}a - \frac{c_{\mathrm{traj}}d}{m^2}, \qquad \Psi''(m) = \frac{2c_{\mathrm{traj}}d}{m^3} > 0. \tag{16}$$

Strict convexity gives the unique minimizer $\sqrt{c_{\mathrm{traj}}d/(c_{\mathrm{tgt}}a)}$. Let $m_{\max}$ denote the largest feasible target count per user in this relaxation. Restricting to $m \in [1, m_{\max}]$ clips this solution. If only m is integer and U remains continuously allocatable, the optimum is among the feasible floor and ceiling of the clipped solution. In particular,

$$\Psi(m+1) - \Psi(m) = c_{\mathrm{tgt}}a - \frac{c_{\mathrm{traj}}d}{m(m+1)}. \tag{17}$$

Thus $c_{\mathrm{traj}}d > c_{\mathrm{tgt}}a$ selects a continuous minimizer above one, while $c_{\mathrm{traj}}d > 2c_{\mathrm{tgt}}a$ selects the first discrete move from one to two targets in this relaxation. If both U and m must be integers, compare feasible pairs directly: for each m, use $U = \lfloor C_0/(c_{\mathrm{traj}} + c_{\mathrm{tgt}}m)\rfloor \geq 1$ and evaluate $(a + d/m)/U$. Hardware and memory restrictions further restrict that feasible set.

Under materialized-prefix execution, let $c_{\mathrm{mat}} > 0$ denote the per-target execution cost. Substituting the budget constraint $U = C_0/(c_{\mathrm{mat}}m)$ into the variance formula gives the corresponding allocation criterion:

$$C_0 V_{U,m} = c_{\mathrm{mat}}(am + d). \tag{18}$$

This increases with m if $a > 0$. Compared with the materialized-prefix allocation $m = 1$, shared execution has lower modeled variance at equal update cost precisely when

$$(c_{\mathrm{traj}} + c_{\mathrm{tgt}}m)(a + d/m) < c_{\mathrm{mat}}(a + d). \tag{19}$$

This criterion turns measured encoding cost, scoring cost, and gradient covariance into a choice between supervision allocations. When $a = 0 < d$, shared-execution $\Psi(m) = c_{\mathrm{tgt}}d + c_{\mathrm{traj}}d/m$ decreases over the feasible range. When $d = 0 < a$, it increases and $m = 1$ minimizes it. When $a = d = 0$, variance is zero for every allocation. These cases complete the allocation result.

Fixed-state covariance estimates support a local allocation choice. To obtain the trajectory-level comparison in Proposition B.3, use the same reference objective, starting point, step size, and update cost, with corresponding conditional variance bounds throughout training. A total budget then gives the same K, and a smaller uniform variance bound yields a tighter optimization upper bound. This links the allocation criterion to the compute-budget result under its stated assumptions.

B.4 Target accounting and directional progress

Holding the per-update target count $M = Um$ fixed in Proposition 4.1 gives $V_{U,m} = (ma + d)/M$. Its increment per additional target per user is a/M. At a common mean gradient, this identifies independent-user coverage as the way to reduce the shared variance component per target. To evaluate policies that also change the mean direction, we next express expected objective descent in terms of both alignment and noise.

For a general estimator, let $\mu = \mathbb{E}[\hat{g} \mid \theta]$ and $V = \mathbb{E}[\|\hat{g} - \mu\|^2 \mid \theta]$. The β-smoothness inequality gives (Bottou et al., 2018; Ghadimi & Lan, 2013)

$$F(\theta - \eta\hat{g}) \le F(\theta) - \eta\langle \nabla F(\theta), \hat{g}\rangle + \frac{\beta\eta^2}{2}\|\hat{g}\|^2. \tag{20}$$

Taking conditional expectation and using $\mathbb{E}[\|\hat{g}\|^2 \mid \theta] = \|\mu\|^2 + V$ yields

$$F(\theta) - \mathbb{E}[F(\theta - \eta\hat{g}) \mid \theta] \ge \eta\langle \nabla F(\theta), \mu\rangle - \frac{\beta\eta^2}{2}\big(\|\mu\|^2 + V\big). \tag{21}$$

Define $D = \langle \nabla F(\theta), \mu\rangle$ and $R = \|\mu\|^2 + V$. Let $\mathcal{P}_*$ denote the maximized local descent bound. When $D > 0$ and $R > 0$, the quadratic lower bound is maximized at $\eta_* = D/(\beta R)$, with value

$$\mathcal{P}_* = \frac{D^2}{2\beta R}, \qquad \frac{\mathcal{P}_*}{M} = \frac{D^2}{2\beta M(\|\mu\|^2 + V)}. \tag{22}$$

With a prescribed upper step-size limit, clip η_* to that limit and evaluate the original quadratic. If $D \le 0$, the best nonnegative value of the lower bound over $\eta \ge 0$ is zero, attained at $\eta = 0$. If $R = 0$, the estimator is zero almost surely and no update occurs.

For nonzero μ and ∇F, define their directional alignment by $\cos\phi = D/(\|\nabla F\|\|\mu\|)$. Substituting $\mathrm{GNS} = V/\|\mu\|^2$ into the positive-alignment result expresses the descent bound through gradient direction and relative noise:

$$\frac{\mathcal{P}_*}{M} = \frac{\|\nabla F\|^2}{2\beta M}\frac{\cos^2\phi}{1 + \mathrm{GNS}}. \tag{23}$$

The optimized local lower bound identifies how to improve expected descent per target on the reference training objective: strengthen alignment and reduce noise relative to the mean signal, while accounting for the number of targets consumed. It also gives each diagnostic a precise role. Our GNS and cosine-to-estimated-mean measurements characterize update stability; $\cos\phi$ characterizes alignment of the mean update with the reference gradient. The objective-gradient probe examines CE-referenced directions at a shared checkpoint (Appendix C.2.1). Separately, the controlled ML-1M causal-FMLP/full-CE cap sweep tests the update-to-quality relation; common-quality and cross-backbone comparisons report practical policy outcomes under their own objectives and endpoints. These results organize three experimentally separable questions: whether execution reuses the same gradients, how user and target sampling affect their covariance, and how the resulting updates translate into ranking quality and target use. Exactness audits address the first, gradient and weighting controls address the second, and the controlled cap sweep and practical comparisons measure distinct trained responses to the third.

C Evidence map and extended results

The five-seed target-cap sweep in Appendix D estimates the update-to-quality relation. The tests below assess EPC's objective and gradient equivalence; separate optimization and candidate controls measure practical-policy effects.

Reproducibility details. All experiments use public datasets with standard 5-core preprocessing (McAuley et al., 2015), which retains only users and items with at least five interactions, followed by leave-one-out evaluation and full-catalog ranking. Evaluation protocols fix maximum length, candidates, padding, and targets. Paired comparisons over seeds 2027–2031 reuse the same initial parameters. Paired effect intervals use two-sided paired-t intervals with four degrees of freedom; the shared-offset interval in Appendix D uses a paired-seed bootstrap. These intervals quantify uncertainty for their corresponding estimates. Where reported, validation NDCG@10 selects the learning-rate multiplier from $\{0.25, 0.5, 1, 2, 4\}$ before test metrics are read.

C.1 Prefix reuse, implementations, and measurement

What EPC changes. EPC maps each declared prefix loss to an aligned state of the shared graph. This mapping fixes the sequence operator, supervised positions, loss reduction, and candidate construction for the exact-execution comparison. Target grouping or candidate scoring can then be varied separately to study their optimization effects.

Operator- and update-level exactness. Table 34(a) summarizes the numerical equivalence tests for five causal lifts. Tables 22 and 23 report full-precision per-lift errors, independent implementation checks, and the 671-target accumulated-update reconstruction. Together, these results test the

theorem's hidden-state, loss, and gradient identities for the evaluated implementations. The backbone map below distinguishes native-causal operators from those requiring a causal lift.

Backbone-specific modifications. Every backbone uses a fixed upstream revision and an explicit EPC modification, summarized in Table 3. A causal lift replaces future-dependent operations with prefix-consistent counterparts; the resulting implementation is the *lifted causal variant*. Exactness then concerns packed versus aligned-prefix execution of that variant, not equivalence to the upstream noncausal function.

Table 3: Upstream operator roles and EPC boundaries for all eight backbones.

Model	EPC role	Preserved operator and EPC modification
SASRec	native	Retain causal self-attention; add explicit padding masks
FMLP-Rec	lift	Retain half-spectrum filters and feed-forward layers; replace circular with causal linear convolution
FEARec	lift	Retain spatial and frequency branches; use triangular masks and non-circular delays
BSARec	lift	Retain spectral and attention branches; make both branches causal
TV-Rec	lift	Retain polynomial taps; replace cyclic with a non-cyclic lower-triangular shift
NextItNet	native	Retain causal-dilated residual blocks; zero padding outputs
RecBLR	native	Retain causal convolution and the behavior-dependent linear recurrent unit scan; make padding steps identity transitions
LinRec	lift	Retain exponential-linear-unit query/key maps and projections; make key normalization and $K^\top V$ cumulative

The same validation rule covers all eight operators. FMLP, FEARec, BSARec, TV-Rec, and LinRec satisfy the float64 aligned-state, loss, gradient, and future-perturbation tests in Table 22. SASRec, NextItNet, and RecBLR retain their causal operators and satisfy the corresponding padding and final-state tests. Paired comparisons reuse identical initial parameters for each seed. These checks support the concrete implementations under the theorem's conditions. The following comparison evaluates the released and causal FMLP training pipelines.

Released and causal FMLP comparison. The three-arm Beauty study retains the split, sampled-BCE objective, hyperparameters, and five-seed schedule. The released arm also retains upstream window enumeration, including an initial empty-context target, and nonzero padded states. The causal arms use nonempty transitions from the shared truncated window and mask padding. Released-prefix versus causal-prefix therefore measures a pipeline change that includes operator and data-interface semantics, while causal-prefix versus causal-packed holds the causal architecture fixed and changes training organization.

The paired intervals quantify the released-to-causal and causal-prefix-to-packed transitions, while the progression table adds the final full-catalog CE step used in the main-text attribution. The timing conditions for these comparisons are stated next.

Timing conditions. The breadth, objective-comparison, scaling, and packed pool/temperature studies measure runtime on individually allocated A100 GPUs, with concurrent compute processes excluded from their aggregates. Profiling diagnostics retain their stated hardware and protocol conditions. Figure 7 and Table 24 follow prefix reuse through storage, loading, transfer, model compute, and the training loop. The estimator definition below formalizes the optimization policy enabled by that execution saving.

C.2 Optimization consequences of target-dense updates

C.2.1 Finite-batch estimators under packed training

Fix the model state and a candidate policy that makes each $g_{u,t} = \nabla_\theta \ell_{u,t}$ independent of batch membership, as in the full-CE controls. For user u, let $\mathcal{I}_u = \{1, \dots, T_u - 1\}$ and $\omega_u = \sum_{t \in \mathcal{I}_u} w_{u,t} > 0$. Given a nonempty sampled user set S, target-normalized packed training uses the following estimator:

$$\widehat{g}_{\mathrm{tgt}}(S) = \frac{1}{\Omega(S)} \sum_{u \in S} \sum_{t \in \mathcal{I}_u} w_{u,t} g_{u,t}, \qquad \Omega(S) = \sum_{u \in S} \sum_{t \in \mathcal{I}_u} w_{u,t}. \tag{24}$$

Equal-user reduction gives every sampled user the same total weight. It normalizes each user's contribution by that user's target mass before taking the batch average:

$$\widehat{g}_{\mathrm{usr}}(S) = \frac{1}{|S|} \sum_{u \in S} \frac{1}{\omega_u} \sum_{t \in \mathcal{I}_u} w_{u,t} g_{u,t}. \tag{25}$$

Table 4: **Released and causal FMLP on Beauty.** All rows use the same split, hyperparameters, and sampled BCE with one history-filtered negative; values are mean ± standard deviation over seeds 2027–2031. The released-to-causal transition also changes padding and target enumeration. Within the causal architecture, the last two rows change target grouping and positional alignment.

Setup	n	NDCG@10	HIT@10	t_{best} (s)	Peak MB
Released-prefix	5	0.0352 ± 0.0010	0.0650 ± 0.0019	1003.6 ± 204.6	199.1 ± 0.0
Causal-prefix	5	0.0357 ± 0.0019	0.0658 ± 0.0028	716.4 ± 266.6	213.2 ± 0.0
Causal-packed	5	0.0352 ± 0.0012	0.0656 ± 0.0019	96.9 ± 26.2	212.6 ± 0.0
Causal-prefix − released-prefix: ΔNDCG@10 +0.0005, 95% CI [-0.0028, +0.0037].					
Causal-packed − causal-prefix: ΔNDCG@10 -0.0005, 95% CI [-0.0027, +0.0018].					

Table 5: **Objective and training-organization effects on Beauty.** Five-seed results use causal FMLP. Full CE attains the highest NDCG@10; each packed configuration reaches its own selected checkpoint earlier. Right-prefix and packed runs differ in target grouping and positional alignment. Time and memory columns average the five runs in each configuration. BCE and BPR use one sampled negative per target.

Execution	Objective	NDCG@10 ↑	t_{best} (s) ↓	Peak MB ↓
Right-prefix	full CE	0.0521±0.0009	69.0	213
Packed	full CE	**0.0538±0.0011**	**19.7**	281
Right-prefix	InfoNCE (shared pool)	0.0507±0.0004	323.3	213
Packed	InfoNCE (shared pool)	0.0521±0.0011	**71.4**	213
Right-prefix	BCE (1 neg.)	0.0357±0.0019	716.4	213
Packed	BCE (1 neg.)	0.0352±0.0012	**96.9**	213
Right-prefix	BPR (1 neg.)	0.0343±0.0017	813.7	213
Packed	BPR (1 neg.)	0.0318±0.0012	**41.6**	213

For uniform user sampling this is unbiased for the user-averaged gradient. In contrast, target normalization gives longer histories more mass within each batch. Its expected contribution to the population gradient also depends on the random batch denominator:

$$\mathbb{E}[\widehat{g}_{\text{tgt}}] = \sum_u \sum_t w_{u,t} g_{u,t}\, \mathbb{E}\left[\frac{\mathbb{I}\{u \in S\}}{\Omega(S)}\right]. \tag{26}$$

This need not equal the globally target-weighted gradient with denominator $\sum_u \omega_u$. For example, a one-user batch produces equal-user averaging in expectation even if user lengths differ. Thus, using the same loss terms does not by itself give the same expected update. A fixed, identical sampled unit nevertheless remains exactly compilable under Theorem 5.2.

For M equally weighted targets with gradients $g_1, \ldots, g_M$, the conditional covariance is

$$\operatorname{Cov}(\widehat{g} \mid M) = \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} \operatorname{Cov}(g_i, g_j \mid M). \tag{27}$$

Both within-user dependence and sampling-induced dependence enter the off-diagonal terms. When M varies, total covariance also includes $\operatorname{Cov}(\mathbb{E}[\widehat{g} \mid M])$. More targets can reduce noise, but target count alone is not an independence assumption or a guarantee of a better ranking endpoint.

Gradient diagnostic and matching rule. The diagnostic reports the noise-to-signal ratio $\operatorname{tr} \operatorname{Cov}(\widehat{g}) / \|\mathbb{E}\widehat{g}\|^2$, labeled GNS in the figures. It is not multiplied by batch size. For $n_{\text{grad}} = 128$ sampled gradients, we use the sample covariance trace v and the noise-corrected squared-mean estimate $\|\bar{g}\|^2 - v/n_{\text{grad}}$ in the denominator. Comparisons use encoder parameters, excluding the item embedding, at a fixed checkpoint with dropout disabled. The target-count control samples independent prefix rows without replacement in the same number as each sampled packed batch; it does not reuse that batch's target identities. Cosines to the sample mean are descriptive finite-repeat statistics. The separate 671-target accumulation test reuses the actual targets and checks loss and gradient reconstruction. The estimator derivation specifies the comparisons in Section 6.3. Target-count matching examines covariance and direction, equal-user reduction changes within-update weighting, and separate learning-rate selection tests the trained response after adjusting the step size. The user-fraction study below changes a different quantity: the number of distinct histories available for training. Its complete curve measures data use directly rather than inferring it from gradient geometry.

Table 7: **Validation-selected pool and temperature settings.** Seed-2027 validation NDCG@10 selects from a 4×3 grid; selected settings use seeds 2027–2031. The default is unrestricted pool, τ=0.1. Δ compares selected and default run means with a paired 95% CI. A selected setting that coincides with the default reports repeat-run variation rather than a tuning effect. The selection seed remains in the aggregate, and the intervals do not adjust for selection.

Dataset	Backbone	Pool	τ	Default NDCG	Selected NDCG	Δ [95% CI]	Selected HIT	Train-to-best (s)
ML-1M	SASRec	all	0.100	0.1418	0.1421	+0.0003 [-0.0009,+0.0014]	0.2628	55.9
	FMLP	all	0.100	0.1434	0.1433	-0.0001 [-0.0028,+0.0025]	0.2593	53.2
	RecBLR	all	0.100	0.1447	0.1447	+0.0000 [+0.0000,+0.0000]	0.2592	58.5
	LinRec	2048	0.100	0.1325	0.1316	-0.0008 [-0.0030,+0.0014]	0.2448	44.2
Steam	SASRec	all	0.050	0.0323	0.0462	+0.0139 [-0.0108,+0.0387]	0.0889	1388.0
	FMLP	2048	0.100	0.0636	0.0638	+0.0003 [-0.0010,+0.0015]	0.1218	2454.5
	FEARec	all	0.100	0.0620	0.0629	+0.0009 [+0.0001,+0.0017]	0.1186	6137.3
	BSARec	2048	0.050	0.0453	0.0589	+0.0135 [-0.0026,+0.0297]	0.1094	2084.3
	TV-Rec	2048	0.200	0.0479	0.0497	+0.0019 [-0.0001,+0.0038]	0.0976	1329.3
	NextItNet	2048	0.100	0.0434	0.0434	+0.0000 [+0.0000,+0.0000]	0.0816	1783.4
	RecBLR	2048	0.100	0.0525	0.0523	-0.0003 [-0.0012,+0.0007]	0.0976	2158.5
	LinRec	2048	0.100	0.0602	0.0610	+0.0008 [-0.0024,+0.0039]	0.1144	3804.5

User-fraction and exposure accounting. Table 29 records all seven fractions, three implementations, and five seeds. Their available transition counts are 1,376, 2,688, 6,441, 12,809, 25,445, 64,286, and 128,031. Distinct positive target-item counts are 1,179, 2,055, 3,929, 6,035, 8,480, 11,288, and 12,065. Dividing transitions by users gives 5.69–6.14 transitions per user. These are properties of the fixed subsets, whereas target-event counts include repeated training passes through them.

Objective-gradient probe. The fixed-state Beauty diagnostic reported in Table 28 separates item-output coverage from the full-parameter gradient direction. It evaluates full CE and sampled binary cross-entropy (BCE) and Bayesian personalized ranking (BPR) losses at the same causal-FMLP CE checkpoint, using the same 732 targets and 16 negative draws with dropout disabled. Hidden states are detached only for the output-row coverage count; the gradient-cosine calculation includes the encoder. Moving from one to 64 negatives increases BCE's output support from 10.96% to 97.88%, while its mean-gradient cosine to CE changes from 0.085 to 0.089; BPR reaches 0.522 with 64 negatives. This fixed-state comparison shows that a denser sample of a different loss does not reproduce the CE gradient at this checkpoint.

C.3 Candidate coupling under shared-pool InfoNCE

In the implemented sampler, each supervised target contributes sampled negatives to the shared pool. This directly couples target grouping to candidate construction. The pool-cap intervention controls this coupling by limiting the number of scored entries while preserving the packed targets. The definitions below specify the sampled loss, the fixed-state diagnostics, and the validation selections used to measure its trained response.

Selected candidate-pool settings. Figure 8 and Table 26 show how candidate construction affects validation quality under fixed packed grouping. The following tables report the selected configurations and their five-seed evaluations, followed by the joint pool–temperature study.

Table 6: Validation-selected shared-pool InfoNCE configurations for each dataset×model cell. Pool size and temperature are selected by validation NDCG@10; test NDCG@10 is reported over five seeds for every displayed setting.

dataset	model	pool	τ	n	test NDCG@10 ↑
ML-1M	causal BSARec	unrestricted	0.115	5	0.1467±0.0031
ML-1M	causal FEARec	unrestricted	0.13	5	0.1422±0.0014
ML-1M	causal TV-Rec	unrestricted	0.085	5	0.1476±0.0018
Toys	causal BSARec	2048	0.115	5	0.0511±0.0004
Toys	causal FEARec	unrestricted	0.115	5	0.0497±0.0005
Toys	causal TV-Rec	2048	0.1	5	0.0578±0.0006
Sports	causal BSARec	2048	0.115	5	0.0306±0.0006
Sports	causal FEARec	2048	0.115	5	0.0294±0.0008
Sports	causal TV-Rec	2048	0.1	5	0.0390±0.0002

Our shared pool is a multiset of sampled negatives, not a set of other targets. For a batch $\mathcal{B}$, query $\mathbf{q}$, positive item y, normalized score s, temperature $\tau > 0$, and negative multiset $\mathcal{N}_{\mathcal{B}}$, the implemented loss is

$$\ell_{\mathbf{q}} = -\log \frac{\exp(s(\mathbf{q}, y)/\tau)}{\exp(s(\mathbf{q}, y)/\tau) + \sum_{j \in \mathcal{N}_{\mathcal{B}}} \exp(s(\mathbf{q}, j)/\tau)}. \tag{28}$$

Each target contributes sampled negatives filtered against its training history; pooling preserves duplicates and does not re-filter each candidate against every other query. A cap uniformly subsamples this multiset, and the uncapped pool retains it in full. Pool size counts scored negative entries rather than distinct catalog items. Exact compilation preserves the multiset and temperature; the practical control changes the cap while keeping packed target grouping fixed.

Fixed-state candidate diagnostics. Table 27 uses the same unrestricted-pool, $\tau = 0.1$, seed-2027 checkpoint for every pool size within a backbone. The models run in evaluation mode in float32. Five sampling seeds define three sets of eight 128-user batches each; these 120 batches reuse the same model parameters. Each batch draws one negative per target and one permutation of the resulting pool, so the caps are nested. The diagnostic first averages each statistic across queries in a batch and then gives every batch equal weight.

For the softmax distribution p_j over the positive and all negative entries, define $\mathcal{H}(p) = -\sum_j p_j \log p_j$ and $N_{\text{eff}} = \exp(\mathcal{H}(p))$. This is a measure of how widely the softmax mass is distributed, not a count of distinct items or independent gradient contributions. The same unrestricted ML-1M batches contain $5{,}638$ negative entries and $2{,}754$ unique items on average. On Toys and Sports, the 2048 cap removes no entry in any of the 240 sampled batches; the capped and unrestricted candidate multisets are identical there. On ML-1M the cap binds in all 120 batches. Thus the same numerical cap defines different interventions across datasets.

The improvement from pool 128 to the unrestricted pool across nine validation cells shows a candidate-policy response within InfoNCE. Table 6 reports the five-seed evaluations of validation-selected pool sizes. Table 26 extends the candidate-policy control to eight backbones and six datasets, while Table 7 reports the joint pool–temperature selection study.

The candidate controls complement the target-estimator controls: the latter describe which losses are averaged, while the former change the competitors scored within each loss. Their trained outcomes can therefore be related to the cross-architecture organization–objective matrices that follow.

C.4 External scale and controlled architecture context

The published Beauty values provide external context for the complete training configuration in Section 6.3. They are followed by a controlled architecture matrix under one split and implementation.

Table 8: **Comparison with published Beauty results.** Published values are full-catalog ranking results reported by the cited papers; our columns are five-seed means±standard deviations under one uniform packed/full-CE protocol on our shared split. The cited papers use their own architectures, objectives, implementations, and hyperparameter searches. This table provides external scale context; paired experiments with the same split and initialization provide EPC mechanism attribution.

	Published outcome		Our packed/full CE (n=5)	
Backbone	NDCG@10	HIT@10	NDCG@10	HIT@10
SASRec (Zhou et al., 2022)	0.0318	0.0605	0.0478±0.0003	0.0822±0.0008
FMLP-Rec (Zhou et al., 2022)	0.0333	0.0632	0.0538±0.0011	0.0928±0.0010
FEARec (Du et al., 2023)	0.0459	0.0884	0.0484±0.0012	0.0834±0.0021
BSARec (Shin et al., 2024)	0.0611	0.1008	0.0484±0.0001	0.0838±0.0008
TV-Rec (Shin et al., 2025)	0.0608	0.1017	0.0595±0.0004	0.1023±0.0007

Table 8 places five packed/full-CE implementations alongside published Beauty values as external context. Because the sources use different training protocols, the paired matrices provide the attribution for the training choices studied here.

Table 9: **Controlled Beauty architecture matrix** (five seeds; our split and full-catalog evaluation). InfoNCE rows use each update's shared candidate pool. The pool-128 breadth study holds candidate count fixed.

Backbone	Execution	Objective	NDCG@10 ↑	HIT@10 ↑	t_{best} (s) ↓	Peak MB
BSARec	Right-prefix	CE	**0.0494**	0.0869	122.9	191
BSARec	Packed	CE	0.0484	0.0838	**24.6**	283
BSARec	Right-prefix	InfoNCE	0.0483	**0.0871**	411.3	191
BSARec	Packed	InfoNCE	0.0488	0.0871	59.7	192
FEARec	Right-prefix	CE	**0.0485**	**0.0855**	271.9	1182
FEARec	Packed	CE	0.0484	0.0834	**47.7**	1182
FEARec	Right-prefix	InfoNCE	0.0470	0.0841	1450.7	1182
FEARec	Packed	InfoNCE	0.0471	0.0835	139.0	1183
TV-Rec	Right-prefix	CE	0.0582	0.1020	211.4	174
TV-Rec	Packed	CE	**0.0595**	**0.1023**	**72.1**	227
TV-Rec	Right-prefix	InfoNCE	0.0523	0.0938	619.6	175
TV-Rec	Packed	InfoNCE	0.0560	0.0990	170.0	176

This comparison holds the split and implementation fixed, allowing each objective's organization effect to be read within a backbone. The following matrices extend that controlled comparison across datasets and model families.

C.5 Breadth across architectures, objectives, and datasets

The complete matrices retain the model- and dataset-specific responses behind the main-text aggregates. The three-backbone study supplies the detailed CE and InfoNCE comparisons; the eight-backbone study extends the practical shared-pool protocol to six datasets. They retain separate ranking, time, and memory columns because a gain on one endpoint need not imply a gain on another.

Table 10: Full cross-model and cross-dataset matrix with train-to-best time and peak memory (five seeds; NDCG@10 as mean±standard deviation, time and memory as means). Timing evidence is the within-profile right-prefix/packed comparison under the recorded one-process-per-card setting, and Appendix C.1 records which campaigns support absolute timing claims.

dataset	model	execution form	objective	NDCG@10 ↑	Train-to-best (s) ↓	mem (MB) ↓
ML-1M	causal BSARec	packed	full CE	0.1567±0.0011	30.1	343
ML-1M	causal BSARec	packed	InfoNCE	0.1423±0.0025	50.4	714
ML-1M	causal BSARec	right-prefix	full CE	0.1507±0.0027	865.0	176
ML-1M	causal BSARec	right-prefix	InfoNCE	0.1510±0.0019	1429.9	175
ML-1M	causal FEARec	packed	full CE	0.1546±0.0013	54.0	1165
ML-1M	causal FEARec	packed	InfoNCE	0.1363±0.0012	106.3	1167
ML-1M	causal FEARec	right-prefix	full CE	0.1510±0.0023	3149.7	1167
ML-1M	causal FEARec	right-prefix	InfoNCE	0.1481±0.0010	3571.0	1167
ML-1M	causal TV-Rec	packed	full CE	0.1528±0.0014	41.1	285
ML-1M	causal TV-Rec	packed	InfoNCE	0.1468±0.0041	103.1	655
ML-1M	causal TV-Rec	right-prefix	full CE	0.1438±0.0027	832.8	157
ML-1M	causal TV-Rec	right-prefix	InfoNCE	0.1471±0.0017	1891.2	156
Toys	causal BSARec	packed	full CE	0.0442±0.0011	22.3	271
Toys	causal BSARec	packed	InfoNCE	0.0505±0.0004	103.4	192
Toys	causal BSARec	right-prefix	full CE	0.0463±0.0005	98.0	192
Toys	causal BSARec	right-prefix	InfoNCE	0.0502±0.0010	433.6	192
Toys	causal FEARec	packed	full CE	0.0472±0.0008	46.6	1185
Toys	causal FEARec	packed	InfoNCE	0.0486±0.0005	242.8	1184
Toys	causal FEARec	right-prefix	full CE	0.0450±0.0011	223.6	1185
Toys	causal FEARec	right-prefix	InfoNCE	0.0477±0.0003	1070.0	1184
Toys	causal TV-Rec	packed	full CE	0.0601±0.0008	50.4	217
Toys	causal TV-Rec	packed	InfoNCE	0.0578±0.0006	148.0	176
Toys	causal TV-Rec	right-prefix	full CE	0.0589±0.0006	166.8	176
Toys	causal TV-Rec	right-prefix	InfoNCE	0.0557±0.0005	406.7	176
Sports	causal BSARec	packed	full CE	0.0310±0.0010	34.1	338
Sports	causal BSARec	packed	InfoNCE	0.0307±0.0006	116.9	210
Sports	causal BSARec	right-prefix	full CE	0.0304±0.0010	227.8	212
Sports	causal BSARec	right-prefix	InfoNCE	0.0310±0.0005	721.9	211
Sports	causal FEARec	packed	full CE	0.0310±0.0006	75.8	1202
Sports	causal FEARec	packed	InfoNCE	0.0289±0.0009	291.5	1203
Sports	causal FEARec	right-prefix	full CE	0.0277±0.0024	621.1	1202
Sports	causal FEARec	right-prefix	InfoNCE	0.0305±0.0001	2534.9	1202
Sports	causal TV-Rec	packed	full CE	0.0397±0.0011	56.5	283
Sports	causal TV-Rec	packed	InfoNCE	0.0390±0.0002	276.6	193
Sports	causal TV-Rec	right-prefix	full CE	0.0386±0.0005	191.5	194
Sports	causal TV-Rec	right-prefix	InfoNCE	0.0373±0.0003	685.4	194

Table 11: **ML-1M results under the practical protocol.** Mean ± standard deviation over five seeds; packed InfoNCE uses its shared candidate pool. Here Prefix denotes right-prefix execution and Packed denotes the shared packed policy.

Backbone	Execution	NDCG@10	HIT@10	Train-to-best (s)	Peak GPU (MB)	Targets/s
SASRec	Prefix	0.1508 ± 0.0028	0.2708 ± 0.0041	964.7 ± 154.4	151.0 ± 0.0	7562 ± 1799
	Packed	0.1418 ± 0.0022	0.2625 ± 0.0049	60.5 ± 10.0	694.8 ± 3.0	135875 ± 5035
FMLP	Prefix	0.1509 ± 0.0016	0.2742 ± 0.0040	982.2 ± 214.7	196.0 ± 0.0	6064 ± 123
	Packed	0.1434 ± 0.0026	0.2602 ± 0.0041	55.4 ± 18.4	709.9 ± 2.3	137259 ± 3831
FEARec	Prefix	0.1481 ± 0.0010	0.2706 ± 0.0037	3571.0 ± 888.5	1166.9 ± 0.0	1897 ± 137
	Packed	0.1363 ± 0.0012	0.2506 ± 0.0020	106.3 ± 52.1	1167.1 ± 0.0	65892 ± 598
BSARec	Prefix	0.1510 ± 0.0019	0.2760 ± 0.0053	1429.9 ± 271.8	175.1 ± 0.6	4524 ± 240
	Packed	0.1423 ± 0.0025	0.2625 ± 0.0023	50.4 ± 10.0	714.0 ± 4.8	156085 ± 2192
TV-Rec	Prefix	0.1471 ± 0.0017	0.2659 ± 0.0038	1891.2 ± 875.4	156.2 ± 0.2	6665 ± 427
	Packed	0.1468 ± 0.0041	0.2651 ± 0.0072	103.1 ± 55.5	655.5 ± 4.7	130587 ± 3078
NextItNet	Prefix	0.1225 ± 0.0010	0.2267 ± 0.0045	670.1 ± 144.6	151.4 ± 0.0	4884 ± 79
	Packed	0.1111 ± 0.0021	0.2040 ± 0.0027	27.0 ± 4.6	704.4 ± 4.5	133283 ± 4202
RecBLR	Prefix	0.1464 ± 0.0017	0.2653 ± 0.0026	794.9 ± 237.8	446.3 ± 0.1	7637 ± 1556
	Packed	0.1447 ± 0.0014	0.2592 ± 0.0033	62.2 ± 12.5	923.9 ± 3.2	140451 ± 4066
LinRec	Prefix	0.1408 ± 0.0030	0.2578 ± 0.0030	832.8 ± 200.9	1534.4 ± 0.5	6137 ± 1612
	Packed	0.1325 ± 0.0031	0.2455 ± 0.0048	46.3 ± 10.7	1534.0 ± 0.0	136913 ± 5541

Table 12: **Beauty results under the practical protocol.** Mean ± standard deviation over five seeds; packed InfoNCE uses its shared candidate pool. Here Prefix denotes right-prefix execution and Packed denotes the shared packed policy.

Backbone	Execution	NDCG@10	HIT@10	Train-to-best (s)	Peak GPU (MB)	Targets/s
SASRec	Prefix	0.0459 ± 0.0006	0.0826 ± 0.0006	388.1 ± 31.0	167.1 ± 0.0	6235 ± 75
	Packed	0.0476 ± 0.0004	0.0847 ± 0.0014	72.5 ± 22.9	168.4 ± 0.0	32311 ± 1034
FMLP	Prefix	0.0507 ± 0.0004	0.0906 ± 0.0011	323.3 ± 57.3	213.2 ± 0.0	5360 ± 549
	Packed	0.0521 ± 0.0011	0.0934 ± 0.0019	71.4 ± 12.8	212.7 ± 0.3	29571 ± 908
FEARec	Prefix	0.0470 ± 0.0004	0.0841 ± 0.0013	1450.7 ± 326.3	1182.4 ± 0.2	2119 ± 10
	Packed	0.0471 ± 0.0014	0.0835 ± 0.0030	139.0 ± 25.1	1182.6 ± 0.1	11731 ± 199
BSARec	Prefix	0.0483 ± 0.0008	0.0871 ± 0.0017	411.3 ± 126.9	191.1 ± 0.0	4941 ± 50
	Packed	0.0488 ± 0.0013	0.0871 ± 0.0023	59.7 ± 12.5	192.2 ± 0.6	26991 ± 332
TV-Rec	Prefix	0.0523 ± 0.0011	0.0938 ± 0.0016	619.6 ± 192.0	174.8 ± 0.2	6253 ± 97
	Packed	0.0560 ± 0.0005	0.0990 ± 0.0012	170.0 ± 41.9	176.1 ± 0.9	35702 ± 645
NextItNet	Prefix	0.0291 ± 0.0003	0.0527 ± 0.0006	500.8 ± 65.4	181.2 ± 0.0	5402 ± 1154
	Packed	0.0205 ± 0.0005	0.0381 ± 0.0012	76.8 ± 18.9	186.2 ± 2.7	23912 ± 4081
RecBLR	Prefix	0.0433 ± 0.0009	0.0787 ± 0.0019	355.0 ± 106.0	471.1 ± 0.0	6324 ± 1097
	Packed	0.0469 ± 0.0007	0.0832 ± 0.0016	57.8 ± 17.2	471.1 ± 0.0	30884 ± 594
LinRec	Prefix	0.0435 ± 0.0008	0.0788 ± 0.0013	546.8 ± 97.2	1552.8 ± 0.0	5566 ± 1187
	Packed	0.0443 ± 0.0009	0.0804 ± 0.0014	116.7 ± 31.7	1552.9 ± 0.4	27445 ± 242

Controlled three-backbone matrix. Across BSARec, FEARec, and TV-Rec, packed training consistently shortens train-to-best time in the controlled matrix. The ranking differences vary with the objective and dataset. The larger matrix below records the corresponding practical response for both native-causal and lifted operators.

Eight backbones under the practical protocol. Main-text Table 36 and Figure 13 give the cross-dataset view. The six tables below retain every metric and paired seed. Native-causal SASRec, NextItNet, and RecBLR compare practical packing policies on already-causal operators. The five lifted backbones make the same policy comparison after global or bidirectional mixers are replaced by prefix-consistent variants. The accompanying 2×2 matrices show how the packed-versus-right-prefix policy contrast varies with the objective, a pattern that aggregation across backbones and candidate settings would hide.

Across the six practical matrices, packed training reduces train-to-best for every backbone–dataset pair and increases target-event throughput throughout. Ranking does not follow the same pattern: all eight ML-1M InfoNCE point estimates favor right-prefix training, whereas all eight Steam point estimates favor packed training, with mixed responses on the remaining datasets. The detailed matrices therefore separate the consistent runtime benefit of the packed policy from its objective- and dataset-dependent ranking response.

Practical 2×2 comparison of training organization and objective. Table 25 summarizes the cross-backbone interaction. For each condition, the tables below report ranking metrics, dispersion across seeds, training cost, peak memory, and target-event throughput.

Table 13: **Toys results under the practical protocol.** Mean ± standard deviation over five seeds; packed InfoNCE uses its shared candidate pool. Here Prefix denotes right-prefix execution and Packed denotes the shared packed policy.

Backbone	Execution	NDCG@10	HIT@10	Train-to-best (s)	Peak GPU (MB)	Targets/s
SASRec	Prefix	0.0387 ± 0.0191	0.0677 ± 0.0333	237.5 ± 122.5	168.2 ± 0.0	6976 ± 1758
	Packed	0.0487 ± 0.0006	0.0839 ± 0.0005	67.6 ± 14.9	169.4 ± 0.0	31122 ± 325
FMLP	Prefix	0.0520 ± 0.0007	0.0902 ± 0.0016	286.9 ± 88.9	213.8 ± 1.0	7474 ± 1836
	Packed	0.0538 ± 0.0005	0.0931 ± 0.0009	66.5 ± 14.4	213.9 ± 0.0	30129 ± 1065
FEARec	Prefix	0.0477 ± 0.0003	0.0830 ± 0.0011	1070.0 ± 107.7	1184.4 ± 0.2	1936 ± 136
	Packed	0.0486 ± 0.0005	0.0841 ± 0.0009	242.8 ± 71.8	1183.6 ± 0.2	11029 ± 476
BSARec	Prefix	0.0502 ± 0.0010	0.0873 ± 0.0016	433.6 ± 64.1	192.1 ± 0.0	4883 ± 143
	Packed	0.0505 ± 0.0004	0.0879 ± 0.0015	103.4 ± 10.7	192.1 ± 0.0	22150 ± 779
TV-Rec	Prefix	0.0557 ± 0.0005	0.0961 ± 0.0012	406.7 ± 80.3	176.2 ± 0.2	7016 ± 305
	Packed	0.0578 ± 0.0006	0.0995 ± 0.0015	148.0 ± 46.7	176.1 ± 0.2	30821 ± 2313
NextItNet	Prefix	0.0253 ± 0.0004	0.0451 ± 0.0013	445.3 ± 81.3	182.6 ± 0.0	4766 ± 141
	Packed	0.0129 ± 0.0007	0.0234 ± 0.0008	51.0 ± 7.6	183.5 ± 0.5	24781 ± 678
RecBLR	Prefix	0.0452 ± 0.0004	0.0786 ± 0.0014	138.0 ± 36.1	472.0 ± 0.0	7640 ± 1598
	Packed	0.0466 ± 0.0007	0.0804 ± 0.0017	49.8 ± 18.7	472.0 ± 0.0	29703 ± 1252
LinRec	Prefix	0.0445 ± 0.0004	0.0790 ± 0.0011	352.5 ± 103.6	1553.9 ± 0.2	6145 ± 1370
	Packed	0.0452 ± 0.0006	0.0786 ± 0.0015	85.6 ± 31.3	1553.4 ± 0.1	25912 ± 661

Table 14: **Sports results under the practical protocol.** Mean ± standard deviation over five seeds; packed InfoNCE uses its shared candidate pool. Here Prefix denotes right-prefix execution and Packed denotes the shared packed policy.

Backbone	Execution	NDCG@10	HIT@10	Train-to-best (s)	Peak GPU (MB)	Targets/s
SASRec	Prefix	0.0244 ± 0.0087	0.0446 ± 0.0160	541.1 ± 260.7	187.6 ± 0.0	6208 ± 166
	Packed	0.0281 ± 0.0007	0.0510 ± 0.0013	110.3 ± 13.4	186.3 ± 0.0	30089 ± 921
FMLP	Prefix	0.0327 ± 0.0006	0.0593 ± 0.0011	534.0 ± 97.3	231.3 ± 0.4	6685 ± 1262
	Packed	0.0315 ± 0.0005	0.0569 ± 0.0014	96.3 ± 34.9	231.1 ± 0.0	30470 ± 709
FEARec	Prefix	0.0305 ± 0.0001	0.0552 ± 0.0004	2534.9 ± 316.4	1202.2 ± 0.1	1832 ± 80
	Packed	0.0289 ± 0.0009	0.0519 ± 0.0014	291.5 ± 62.8	1202.5 ± 0.7	9261 ± 922
BSARec	Prefix	0.0310 ± 0.0005	0.0566 ± 0.0007	721.9 ± 274.3	210.7 ± 0.1	4803 ± 442
	Packed	0.0307 ± 0.0006	0.0556 ± 0.0013	116.9 ± 32.0	210.3 ± 0.0	24180 ± 961
TV-Rec	Prefix	0.0373 ± 0.0003	0.0670 ± 0.0003	685.4 ± 176.0	193.8 ± 0.2	7008 ± 385
	Packed	0.0390 ± 0.0002	0.0694 ± 0.0006	276.6 ± 43.5	193.4 ± 0.0	32804 ± 747
NextItNet	Prefix	0.0145 ± 0.0006	0.0277 ± 0.0010	575.7 ± 167.2	201.6 ± 0.0	4890 ± 94
	Packed	0.0075 ± 0.0004	0.0143 ± 0.0006	68.2 ± 41.2	202.6 ± 0.1	24287 ± 542
RecBLR	Prefix	0.0278 ± 0.0004	0.0501 ± 0.0008	452.4 ± 90.8	496.2 ± 0.0	5819 ± 181
	Packed	0.0292 ± 0.0006	0.0525 ± 0.0007	71.9 ± 4.2	496.2 ± 0.0	28424 ± 1823
LinRec	Prefix	0.0263 ± 0.0005	0.0485 ± 0.0007	795.2 ± 83.9	1571.0 ± 0.2	5564 ± 1245
	Packed	0.0265 ± 0.0004	0.0489 ± 0.0009	183.3 ± 88.9	1571.0 ± 0.0	25245 ± 674

Table 17: **Complete Beauty matrix of execution form × objective.** Mean ± standard deviation over five paired seeds under the practical protocol.

Backbone	Execution	Objective	NDCG@10	HIT@10	Train-to-best (s)	Peak GPU (MB)	Targets/s
SASRec	Right-prefix	CE	0.0485 ± 0.0006	0.0856 ± 0.0008	87.3 ± 16.9	167.1 ± 0.0	6761.7 ± 252.4
	Packed	CE	0.0478 ± 0.0003	0.0822 ± 0.0008	13.6 ± 3.0	266.0 ± 8.8	41614.0 ± 3150.1
	Right-prefix	InfoNCE	0.0459 ± 0.0006	0.0826 ± 0.0006	388.1 ± 31.0	167.1 ± 0.0	6235.2 ± 74.8
	Packed	InfoNCE	0.0476 ± 0.0004	0.0847 ± 0.0014	72.5 ± 22.9	168.4 ± 0.0	32310.8 ± 1034.2
FMLP	Right-prefix	CE	0.0521 ± 0.0009	0.0922 ± 0.0014	69.0 ± 6.3	212.6 ± 0.0	8086.8 ± 412.7
	Packed	CE	0.0538 ± 0.0011	0.0928 ± 0.0010	19.7 ± 1.7	281.2 ± 9.0	35378.4 ± 1985.6
	Right-prefix	InfoNCE	0.0507 ± 0.0004	0.0906 ± 0.0011	323.3 ± 57.3	213.2 ± 0.0	5359.8 ± 549.0
	Packed	InfoNCE	0.0521 ± 0.0011	0.0934 ± 0.0019	71.4 ± 12.8	212.7 ± 0.3	29570.7 ± 907.6
RecBLR	Right-prefix	CE	0.0436 ± 0.0011	0.0781 ± 0.0017	75.5 ± 2.7	471.2 ± 0.0	6871.5 ± 159.1
	Packed	CE	0.0474 ± 0.0006	0.0829 ± 0.0007	18.0 ± 2.4	518.9 ± 10.7	36483.9 ± 832.6
	Right-prefix	InfoNCE	0.0433 ± 0.0009	0.0787 ± 0.0019	355.0 ± 106.0	471.1 ± 0.0	6324.1 ± 1096.7
	Packed	InfoNCE	0.0469 ± 0.0007	0.0832 ± 0.0016	57.8 ± 17.2	471.1 ± 0.0	30884.5 ± 593.9
LinRec	Right-prefix	CE	0.0432 ± 0.0011	0.0770 ± 0.0024	105.3 ± 16.9	1552.8 ± 0.0	5882.7 ± 148.3
	Packed	CE	0.0467 ± 0.0005	0.0803 ± 0.0015	23.7 ± 1.6	1552.7 ± 0.5	31093.4 ± 1276.7
	Right-prefix	InfoNCE	0.0435 ± 0.0008	0.0788 ± 0.0013	546.8 ± 97.2	1552.8 ± 0.0	5566.3 ± 1187.4
	Packed	InfoNCE	0.0443 ± 0.0009	0.0804 ± 0.0014	116.7 ± 31.7	1552.9 ± 0.4	27444.7 ± 242.5

Table 15: **Steam results under the practical protocol.** Mean ± standard deviation over five seeds; packed InfoNCE uses its shared candidate pool. Here Prefix denotes right-prefix execution and Packed denotes the shared packed policy.

Backbone	Execution	NDCG@10	HIT@10	Train-to-best (s)	Peak GPU (MB)	Targets/s
SASRec	Prefix	0.0273 ± 0.0051	0.0555 ± 0.0127	4531.3 ± 2926.7	170.4 ± 0.0	6241 ± 171
	Packed	0.0323 ± 0.0185	0.0624 ± 0.0342	751.9 ± 1185.7	180.4 ± 1.1	47469 ± 777
FMLP	Prefix	0.0470 ± 0.0062	0.0919 ± 0.0136	28298.4 ± 952.8	216.3 ± 0.0	6173 ± 140
	Packed	0.0636 ± 0.0003	0.1210 ± 0.0006	1996.3 ± 406.4	216.3 ± 0.0	46401 ± 1564
FEARec	Prefix	0.0368 ± 0.0048	0.0694 ± 0.0090	35032.4 ± 9239.7	1186.8 ± 0.0	2152 ± 48
	Packed	0.0620 ± 0.0004	0.1172 ± 0.0007	7756.9 ± 2349.9	1185.7 ± 0.0	17240 ± 183
BSARec	Prefix	0.0399 ± 0.0076	0.0796 ± 0.0140	17829.6 ± 13326.7	195.5 ± 0.0	4992 ± 103
	Packed	0.0453 ± 0.0131	0.0885 ± 0.0235	1509.8 ± 1348.9	199.8 ± 1.7	41402 ± 2422
TV-Rec	Prefix	0.0372 ± 0.0008	0.0707 ± 0.0017	6046.5 ± 3960.5	177.6 ± 0.0	7188 ± 309
	Packed	0.0479 ± 0.0014	0.0910 ± 0.0006	770.5 ± 696.4	177.6 ± 0.1	55159 ± 849
NextItNet	Prefix	0.0352 ± 0.0011	0.0663 ± 0.0018	3291.9 ± 564.1	182.6 ± 0.0	4809 ± 91
	Packed	0.0434 ± 0.0005	0.0816 ± 0.0010	1038.0 ± 190.9	209.8 ± 1.8	37507 ± 919
RecBLR	Prefix	0.0310 ± 0.0015	0.0591 ± 0.0026	8624.4 ± 3725.9	475.1 ± 0.0	5879 ± 129
	Packed	0.0525 ± 0.0006	0.0979 ± 0.0009	1530.6 ± 286.8	474.7 ± 0.0	47531 ± 1259
LinRec	Prefix	0.0366 ± 0.0069	0.0699 ± 0.0135	19734.8 ± 4854.6	1556.4 ± 0.0	5164 ± 210
	Packed	0.0602 ± 0.0016	0.1130 ± 0.0030	2934.1 ± 372.8	1555.8 ± 0.3	39697 ± 1565

Table 16: **Gowalla results under the practical protocol.** Mean ± standard deviation over five seeds; packed InfoNCE uses its shared candidate pool. Here Prefix denotes right-prefix execution and Packed denotes the shared packed policy.

Backbone	Execution	NDCG@10	HIT@10	Train-to-best (s)	Peak GPU (MB)	Targets/s
SASRec	Prefix	0.0392 ± 0.0171	0.0721 ± 0.0314	1906.2 ± 515.3	724.2 ± 0.0	6213 ± 58
	Packed	0.0546 ± 0.0005	0.0994 ± 0.0007	208.5 ± 12.2	724.4 ± 0.3	111178 ± 2428
FMLP	Prefix	0.0497 ± 0.0003	0.0910 ± 0.0006	1351.4 ± 450.8	722.1 ± 0.0	7632 ± 1548
	Packed	0.0563 ± 0.0005	0.1022 ± 0.0009	190.6 ± 30.5	722.4 ± 0.3	108545 ± 5938
FEARec	Prefix	0.0479 ± 0.0011	0.0878 ± 0.0012	10057.5 ± 1699.3	1633.3 ± 0.0	2479 ± 532
	Packed	0.0553 ± 0.0013	0.1002 ± 0.0021	556.3 ± 77.4	1633.3 ± 0.0	40007 ± 1362
BSARec	Prefix	0.0434 ± 0.0008	0.0816 ± 0.0015	1814.7 ± 641.8	724.6 ± 0.0	4872 ± 124
	Packed	0.0559 ± 0.0008	0.1018 ± 0.0011	241.0 ± 34.1	724.4 ± 0.2	88537 ± 3591
TV-Rec	Prefix	0.0442 ± 0.0005	0.0808 ± 0.0008	1353.0 ± 228.5	721.9 ± 0.0	7493 ± 1354
	Packed	0.0567 ± 0.0007	0.1024 ± 0.0013	673.6 ± 132.2	722.2 ± 0.3	120315 ± 8069
NextItNet	Prefix	0.0366 ± 0.0008	0.0682 ± 0.0014	1774.6 ± 290.2	1049.5 ± 0.0	5138 ± 864
	Packed	0.0297 ± 0.0007	0.0550 ± 0.0013	241.9 ± 71.8	1049.7 ± 0.3	81666 ± 9050
RecBLR	Prefix	0.0490 ± 0.0004	0.0892 ± 0.0007	3106.1 ± 841.7	1071.6 ± 0.0	6209 ± 1182
	Packed	0.0568 ± 0.0004	0.1016 ± 0.0009	249.1 ± 59.3	1071.6 ± 0.0	105306 ± 2441
LinRec	Prefix	0.0468 ± 0.0006	0.0855 ± 0.0007	2565.2 ± 510.7	2003.4 ± 0.0	5112 ± 114
	Packed	0.0523 ± 0.0007	0.0957 ± 0.0009	210.3 ± 36.6	2003.6 ± 0.8	90059 ± 2552

Table 18: **Complete ML-1M matrix of execution form × objective.** Mean ± standard deviation over five paired seeds under the practical protocol.

Backbone	Execution	Objective	NDCG@10	HIT@10	Train-to-best (s)	Peak GPU (MB)	Targets/s
SASRec	Right-prefix	CE	0.1517 ± 0.0011	0.2724 ± 0.0023	658.7 ± 202.6	151.5 ± 0.0	7975.0 ± 1809.0
	Packed	CE	0.1574 ± 0.0010	0.2795 ± 0.0010	25.2 ± 6.8	324.0 ± 0.7	275939.8 ± 6158.8
	Right-prefix	InfoNCE	0.1508 ± 0.0028	0.2708 ± 0.0041	964.7 ± 154.4	151.0 ± 0.0	7562.3 ± 1799.4
	Packed	InfoNCE	0.1418 ± 0.0022	0.2625 ± 0.0049	60.5 ± 10.0	694.8 ± 3.0	135874.6 ± 5035.0
FMLP	Right-prefix	CE	0.1523 ± 0.0011	0.2754 ± 0.0044	346.5 ± 183.9	195.1 ± 0.1	7718.8 ± 1312.2
	Packed	CE	0.1574 ± 0.0023	0.2782 ± 0.0038	16.8 ± 2.6	337.8 ± 0.9	301169.9 ± 55812.0
	Right-prefix	InfoNCE	0.1509 ± 0.0016	0.2742 ± 0.0040	982.2 ± 214.7	196.0 ± 0.0	6064.1 ± 122.6
	Packed	InfoNCE	0.1434 ± 0.0026	0.2602 ± 0.0041	55.4 ± 18.4	709.9 ± 2.3	137259.3 ± 3831.2
RecBLR	Right-prefix	CE	0.1442 ± 0.0013	0.2627 ± 0.0026	379.0 ± 106.2	445.8 ± 0.0	7313.8 ± 1490.7
	Packed	CE	0.1481 ± 0.0024	0.2660 ± 0.0038	14.9 ± 2.4	591.0 ± 0.6	312680.7 ± 55385.9
	Right-prefix	InfoNCE	0.1464 ± 0.0017	0.2653 ± 0.0026	794.9 ± 237.8	446.3 ± 0.1	7637.4 ± 1556.4
	Packed	InfoNCE	0.1447 ± 0.0014	0.2592 ± 0.0033	62.2 ± 12.5	923.9 ± 3.2	140451.2 ± 4066.1
LinRec	Right-prefix	CE	0.1433 ± 0.0015	0.2621 ± 0.0027	654.9 ± 126.7	1534.1 ± 0.3	6471.6 ± 1395.7
	Packed	CE	0.1476 ± 0.0010	0.2641 ± 0.0027	28.8 ± 4.4	1534.8 ± 0.4	228403.6 ± 8470.0
	Right-prefix	InfoNCE	0.1408 ± 0.0030	0.2578 ± 0.0030	832.8 ± 200.9	1534.4 ± 0.5	6136.7 ± 1611.8
	Packed	InfoNCE	0.1325 ± 0.0031	0.2455 ± 0.0048	46.3 ± 10.7	1534.0 ± 0.0	136913.2 ± 5540.6

Table 19: **Complete Gowalla matrix of execution form × objective.** Mean ± standard deviation over five paired seeds under the practical protocol.

Backbone	Execution	Objective	NDCG@10	HIT@10	Train-to-best (s)	Peak GPU (MB)	Targets/s
SASRec	Right-prefix	CE	0.0500 ± 0.0007	0.0904 ± 0.0011	1114.2 ± 103.3	724.2 ± 0.0	7141.6 ± 223.7
	Packed	CE	0.0563 ± 0.0007	0.0991 ± 0.0012	61.4 ± 3.7	5440.0 ± 94.8	81241.5 ± 4909.1
	Right-prefix	InfoNCE	0.0392 ± 0.0171	0.0721 ± 0.0314	1906.2 ± 515.3	724.2 ± 0.0	6213.2 ± 58.4
	Packed	InfoNCE	0.0546 ± 0.0005	0.0994 ± 0.0007	208.5 ± 12.2	724.4 ± 0.3	111177.8 ± 2427.5
FMLP	Right-prefix	CE	0.0468 ± 0.0008	0.0860 ± 0.0010	880.8 ± 136.9	722.1 ± 0.0	6981.6 ± 196.2
	Packed	CE	0.0591 ± 0.0006	0.1034 ± 0.0011	47.2 ± 3.4	5453.7 ± 95.1	80133.3 ± 5039.1
	Right-prefix	InfoNCE	0.0497 ± 0.0003	0.0910 ± 0.0006	1351.4 ± 450.8	722.1 ± 0.0	7632.4 ± 1547.8
	Packed	InfoNCE	0.0563 ± 0.0005	0.1022 ± 0.0009	190.6 ± 30.5	722.4 ± 0.3	108544.6 ± 5938.4
RecBLR	Right-prefix	CE	0.0383 ± 0.0024	0.0708 ± 0.0033	2718.9 ± 830.1	1071.6 ± 0.0	7679.6 ± 1359.2
	Packed	CE	0.0564 ± 0.0006	0.0992 ± 0.0008	99.2 ± 10.3	6635.5 ± 114.4	52804.3 ± 809.0
	Right-prefix	InfoNCE	0.0490 ± 0.0004	0.0892 ± 0.0007	3106.1 ± 841.7	1071.6 ± 0.0	6209.5 ± 1182.3
	Packed	InfoNCE	0.0568 ± 0.0004	0.1016 ± 0.0009	249.1 ± 59.3	1071.6 ± 0.0	105305.8 ± 2441.3
LinRec	Right-prefix	CE	0.0460 ± 0.0011	0.0838 ± 0.0014	1082.2 ± 163.5	2002.4 ± 0.0	5732.3 ± 60.2
	Packed	CE	0.0547 ± 0.0007	0.0960 ± 0.0011	68.7 ± 10.6	5731.3 ± 95.5	68886.5 ± 3548.2
	Right-prefix	InfoNCE	0.0468 ± 0.0006	0.0855 ± 0.0007	2565.2 ± 510.7	2003.4 ± 0.0	5111.8 ± 114.4
	Packed	InfoNCE	0.0523 ± 0.0007	0.0957 ± 0.0009	210.3 ± 36.6	2003.6 ± 0.8	90059.2 ± 2551.9

Across four backbones and three datasets, packed training lowers train-to-best in all 24 CE/InfoNCE comparisons. Ranking responses remain objective- and dataset-dependent: on ML-1M, CE improves all four backbone point estimates while the corresponding InfoNCE comparisons favor right-prefix training. These matrices extend the main-text Beauty factorial by comparing packed and right-prefix policies under each objective. Because shared-pool InfoNCE couples target grouping to candidate construction, these policy contrasts do not isolate EPC's fixed-unit execution effect. HIT@10, NDCG@10, time, memory, and throughput remain available for separate inspection rather than only through aggregate scores.

C.6 Scaling from ML-1M to ML-10M

The scale study asks how structural reuse affects the cost of reaching matched quality in a larger workload, complementing the controlled density sweep. Main-text Table 37 reports target events and wall time. Figure 5 below shows target-indexed trajectories, prefix compression, and memory ratios; Figures 10 and 6 provide wall-clock trajectories and absolute memory.

Threshold sensitivity. For each paired seed, we reuse the recorded validation trajectory and define the common-quality target as $q_\alpha = \alpha \max_t \mathrm{NDCG}^{\mathrm{right}}_{\mathrm{val}}(t)$. We repeat first-reached accounting for $\alpha \in \{0.90, 0.925, 0.95, 0.975, 0.99\}$ without using test results to select the endpoint. Table 20 summarizes jointly reached pairs and the resulting target-event and wall-time ratios. At 90%, complete coverage reveals a scale-dependent event-count pattern; stricter thresholds preserve the ML-10M direction and provide sensitivity checks for jointly reached ML-1M pairs.

Table 20: **Common-quality threshold robustness.** Each threshold is a fraction of the paired right-prefix run's best validation NDCG@10. Reached counts jointly reached seed pairs over four backbones (maximum 20). Ratio ranges are right-prefix/packed geometric means over backbones with at least one jointly reached pair; values above one favor packed, marked ↑ in the headers.

Dataset	Threshold	Reached	Targets × range↑	Wall × range↑
ML-1M	90%	20/20	.28–.53	4.98–8.54
	92.5%	13/20	.30–.60	5.62–9.89
	95%	7/20	.26–.53	5.30–8.88
	97.5%	2/20	.44	6.29
	99%	1/20	.40	5.86
ML-10M	90%	20/20	1.09–3.31	24.84–63.44
	92.5%	20/20	1.14–3.35	27.20–64.10
	95%	20/20	1.30–4.56	28.70–87.42
	97.5%	20/20	1.56–4.39	34.34–84.38
	99%	20/20	1.86–5.00	37.26–96.08

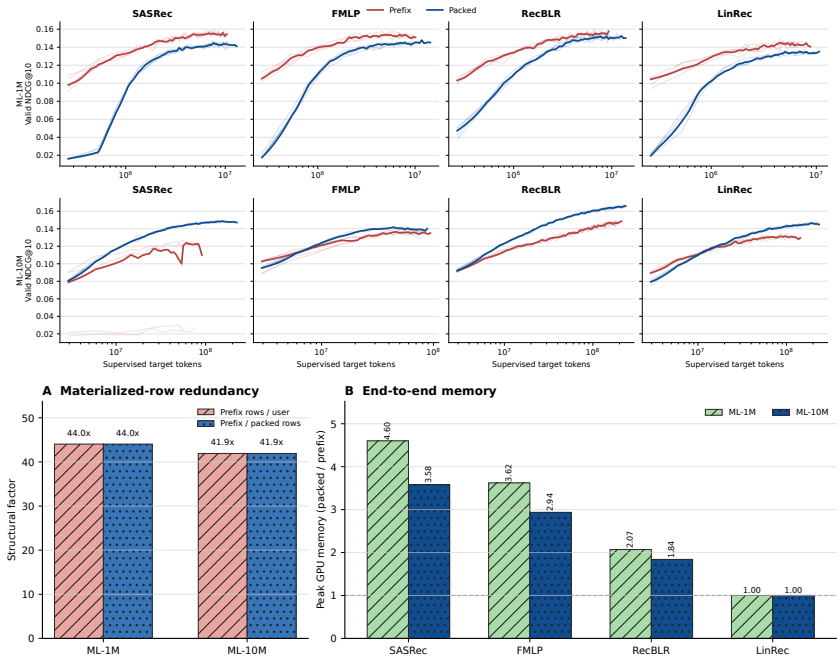


Figure 5: **Target-indexed progress and structural reuse.** The upper view indexes validation quality by processed target events; the lower view reports materialized-row compression and peak-memory ratios.

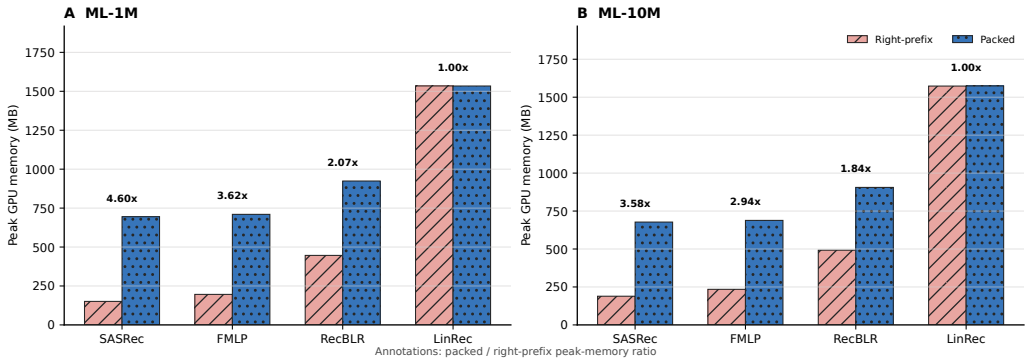


Figure 6: **Absolute peak GPU memory.** Bars report five-seed means for right-prefix and packed InfoNCE. Annotations are paired geometric-mean packed/right-prefix ratios.

The four views characterize different parts of the matched-quality result. Wall-clock trajectories measure how quickly useful checkpoints become available; target-indexed trajectories show progress per processed event. Structural compression measures duplicated rows removed before execution, and absolute memory records the hardware demand of right-prefix and packed training. Their joint pattern supports the structural account of the measured cost at larger scale.

Table 21: **Gradient-noise statistics under nominal, count-matched, and packed updates.** Beauty results use causal FMLP, full CE, fixed trained checkpoints, encoder gradients, dropout disabled, five seeds, and 128 repeats per seed. GNS is covariance trace divided by squared mean-gradient norm. Count-matched prefixes are sampled independently of packed users; their target count matches each packed batch, while their target identities are sampled separately.

update construction	GNS	targets per update	cosine to estimated mean
packed (user-sampled)	5.09±1.61	726.1	0.423
aligned prefix, count-matched	5.38±1.71	726.1	0.414
nominal prefix minibatch	31.11±10.40	128.0	0.202

C.7 Extended Controls and Diagnostics

The results below test numerical equivalence under EPC, separate target and candidate policies, and measure the runtime and memory consequences of prefix reuse. Ranking and efficiency are reported separately because they need not move together.

Table 22: **All five causal lifts satisfy the implementation audit.** Errors compare one packed graph with all materialized aligned prefixes in 64-bit floating point. The future test replaces suffix items while keeping the queried prefix fixed.

Lift	Loss error	Hidden error	Gradient error	Future effect
FMLP	0.0e+00	0.0e+00	2.8e-17	0.0e+00
FEARec	0.0e+00	0.0e+00	1.2e-13	0.0e+00
BSARec	0.0e+00	0.0e+00	4.2e-17	0.0e+00
TV-Rec	0.0e+00	0.0e+00	3.5e-17	0.0e+00
LinRec	0.0e+00	0.0e+00	2.8e-17	0.0e+00

Table 23: **Independent implementation checks.** Float64 maxima for two additional configurations and for one complete update-accumulation audit.

quantity	FEARec	BSARec lift
max |loss diff| (packed vs. aligned)	3.4×10^{-14}	0.0
max |grad diff|	1.8×10^{-12}	8.0×10^{-13}
max |hidden diff|	2.4×10^{-10}	0.0
future-perturbation effect	0.0	0.0
update accumulation (causal BSARec; 671 targets, 34 tensors): loss diff 0.0; max grad diff 1.9×10^{-16}		

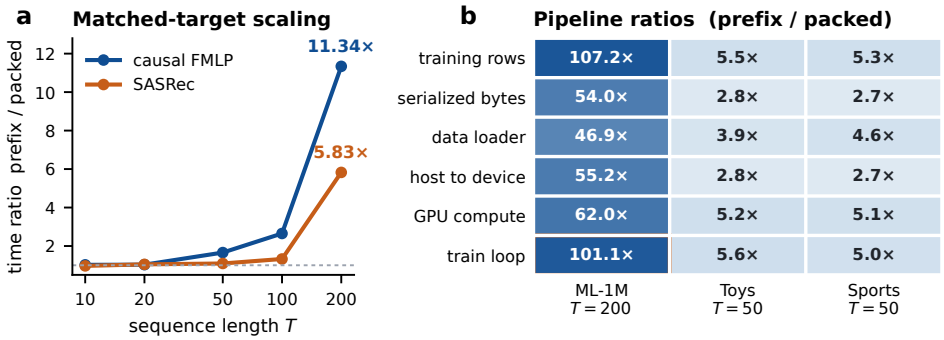


Figure 7: **Matched-supervision execution scaling.** Aligned-prefix versus packed execution and data-pipeline ratios under the timing conditions in Appendix C.1.

Table 24: Data-pipeline and execution ratios for right-prefix over packed training (SASRec, full CE, batch 128, A100; medians over repeats). The profile used a shared GPU, so timing values are within-profile ratios. The ML-1M train-loop ratio extrapolates from the 41,088 of 647,429 target events (6.35%) completed by right-prefix training within the measurement budget; packed training completed the full epoch. At length 200, the 29.2× causal-FMLP and 35.3× SASRec ratios report incremental forward, loss, and backward memory, while Table 34 reports peak memory for complete training runs.

ratio (prefix/packed)	ML-1M (maxlen 200)	Toys (maxlen 50)	Sports (maxlen 50)
rows	107.19×	5.49×	5.28×
serialized bytes	53.99×	2.83×	2.72×
loader epoch	46.85×	3.85×	4.63×
H2D bytes	55.18×	2.83×	2.72×
compute epoch	62.03×	5.23×	5.14×
train-loop epoch-equiv.	101.13× (extrap.)	5.65×	4.96×

Table 25: **Training organization × objective across four backbones.** NDCG@10 is averaged over five paired seeds. Prefix means right-prefix; NCE denotes shared-pool InfoNCE. Time columns give geometric-mean paired prefix/packed train-to-best ratios. The interaction is $(\text{packed} - \text{prefix})_{\text{InfoNCE}} - (\text{packed} - \text{prefix})_{\text{CE}}$; it describes the combined organization and objective change, not an exact-update comparison.

Dataset	Backbone	Prefix+CE	Packed+CE	Prefix+NCE	Packed+NCE	CE time ×	NCE time ×	Interaction
Beauty	SASRec	0.0485	0.0478	0.0459	0.0476	6.47	5.57	+0.0024
	FMLP	0.0521	0.0538	0.0507	0.0521	3.50	4.54	-0.0002
	RecBLR	0.0436	0.0474	0.0433	0.0469	4.23	6.11	-0.0003
	LinRec	0.0432	0.0467	0.0435	0.0443	4.41	4.78	-0.0026
ML-1M	SASRec	0.1517	0.1574	0.1508	0.1418	25.82	15.92	-0.0148
	FMLP	0.1523	0.1574	0.1509	0.1434	18.89	18.15	-0.0125
	RecBLR	0.1442	0.1481	0.1464	0.1447	24.71	12.52	-0.0057
	LinRec	0.1433	0.1476	0.1408	0.1325	22.63	17.90	-0.0127
Gowalla	SASRec	0.0500	0.0563	0.0392	0.0546	18.10	8.79	+0.0091
	FMLP	0.0468	0.0591	0.0497	0.0563	18.52	6.88	-0.0058
	RecBLR	0.0383	0.0564	0.0490	0.0568	26.45	12.28	-0.0103
	LinRec	0.0460	0.0547	0.0468	0.0523	15.78	12.16	-0.0032

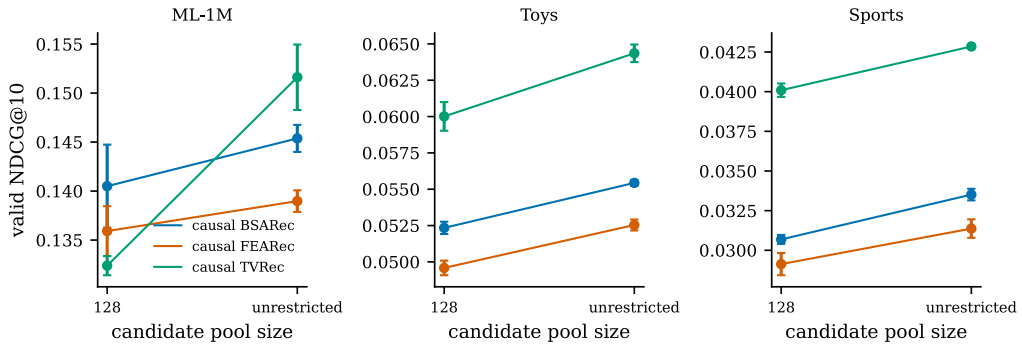


Figure 8: **Shared-pool InfoNCE candidate control.** Validation quality under fixed and unrestricted candidate pools.

Table 28: **Candidate support and objective-gradient direction** at one Beauty causal-FMLP CE checkpoint (seed 2027; 732 fixed targets; dropout disabled). Support counts item-output rows with nonzero gradients for one negative draw, with hidden states detached to exclude encoder-side embedding updates. Cosine compares the mean full-parameter gradient over 16 negative draws with the full-CE gradient. All measurements characterize the objective gradients at one checkpoint.

Objective	Negatives/target	Output-row support (%)	Mean-gradient cosine to CE
Full CE	Entire catalog	100.00	1.000
BCE	1	10.96	0.085
BCE	64	97.88	0.089
BPR	1	10.96	0.484
BPR	64	97.88	0.522

Table 26: **Unrestricted shared pool versus a capped packed pool.** Both packed settings use the same right-prefix reference, whose mean pool is approximately 128. Deltas average six dataset means; pool gain is unrestricted-packed minus capped-packed NDCG. Here time ratios are geometric means of dataset-level ratios of seed-mean times, unlike the seed-paired aggregation in Table 36. Effective pool counts negative entries including duplicates; its capped-packed/prefix ratio is near one.

Backbone	Natural ΔNDCG	Pool-128 ΔNDCG	Pool gain	Fixed-control pool ×	Natural speedup	Pool-128 speedup
SASRec	+0.0044	-0.0033	+0.0077	1.0001	6.58	6.54
FMLP	+0.0029	-0.0057	+0.0087	1.0001	7.60	6.32
FEARec	+0.0033	-0.0073	+0.0107	1.0001	10.15	8.53
BSARec	+0.0016	-0.0063	+0.0079	1.0001	8.76	5.85
TV-Rec	+0.0051	-0.0040	+0.0091	1.0001	4.39	7.45
NextItNet	-0.0063	-0.0133	+0.0070	1.0001	8.08	5.28
RecBLR	+0.0057	-0.0031	+0.0087	1.0001	6.77	8.47
LinRec	+0.0038	-0.0070	+0.0107	1.0001	7.06	5.77

Table 27: **Candidate competition at a fixed model state** (ML-1M, $\tau = 0.1$). Each backbone uses one unrestricted-pool checkpoint trained with seed 2027, evaluated on the same 120 batches (five sampling seeds, three sets, eight batches). Columns average per-query statistics within each batch and then across batches. N_{eff} is softmax perplexity over the positive and negative entries; hardest similarity excludes copies of the positive item. The last column measures the loss gradient with respect to query state h before normalization.

Backbone	Negative pool	N_{eff}	Hardest similarity	$\|\partial \ell / \partial h\|_2$
BSARec	128	14.8	0.482	0.536
	Unrestricted	520.1	0.661	0.636
FEARec	128	14.5	0.475	0.601
	Unrestricted	518.0	0.648	0.732
TV-Rec	128	12.2	0.538	0.559
	Unrestricted	369.9	0.732	0.646

Objective-gradient probe. The fixed-state comparison complements the Beauty objective analysis in Section 6.3. Increasing sampled BCE from one to 64 negatives nearly saturates output-row support while its mean full-parameter gradient direction remains distinct from full CE at the tested checkpoint. Thus, coverage and loss choice produce different gradient responses, motivating their separate trained comparisons.

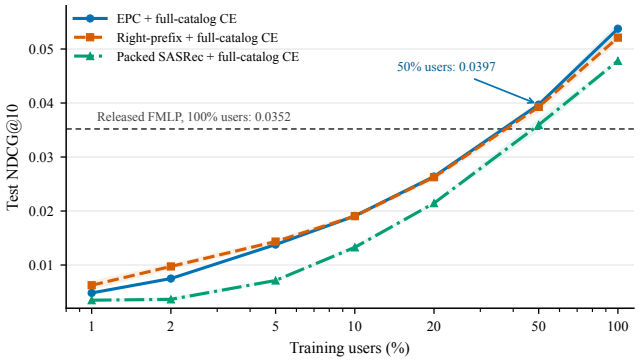


Figure 9: **Ranking under reduced distinct training data on Beauty.** Curves show five-seed NDCG@10 as the fraction of training users increases; the dashed line is the released full-data FMLP/BCE reference.

Complete training-user fractions. These seven-fraction results measure quality as the number of distinct available training histories increases (Figure 9); this axis differs from the target cap per update in the controlled density sweep. Appendix C.2.1 reports the corresponding user, transition, and distinct-item counts and explains how repeated training exposures are counted.

Table 29: **Ranking under reduced distinct training data on Beauty.** NDCG@10 mean ± standard deviation over five seeds with full-catalog CE. Evaluation users and the item catalog are fixed across training fractions.

Training users (%)	Training users	FMLP right-prefix	FMLP packed	SASRec packed
1	224	0.00625 ± 0.00086	0.00483 ± 0.00015	0.00349 ± 0.00025
2	447	0.00973 ± 0.00059	0.00750 ± 0.00010	0.00364 ± 0.00042
5	1,118	0.01434 ± 0.00045	0.01378 ± 0.00031	0.00714 ± 0.00028
10	2,236	0.01905 ± 0.00044	0.01908 ± 0.00029	0.01327 ± 0.00042
20	4,473	0.02626 ± 0.00058	0.02639 ± 0.00029	0.02146 ± 0.00026
50	11,182	0.03928 ± 0.00101	0.03971 ± 0.00029	0.03595 ± 0.00115
100	22,363	0.05212 ± 0.00087	0.05376 ± 0.00111	0.04781 ± 0.00035

Table 30: **Tuned-learning-rate comparison for causal BSARec** (five paired seeds). Each organization selects its multiplier by validation NDCG@10. Timing comes from a separate profile and supports within-study comparison.

Dataset	Execution	NDCG@10 ↑	Multiplier	t_{best} (s) ↓	Peak GPU (MB) ↓
ML-1M	packed	0.1566 ± 0.0018	0.5	62.9	343
	right-prefix	0.1566 ± 0.0013	0.25	1129.0	175
packed − prefix NDCG@10: -0.00002, 95% CI $[-0.00216, +0.00212]$					
Toys	packed	0.0446 ± 0.0007	2	16.9	271
	right-prefix	0.0460 ± 0.0006	1	101.7	191
packed − prefix NDCG@10: -0.00137, 95% CI $[-0.00235, -0.00040]$					

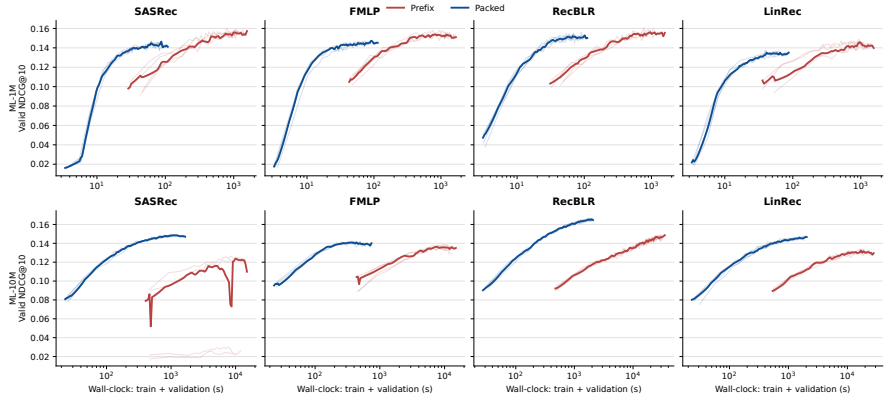


Figure 10: **Validation quality versus wall-clock time** for the MovieLens scale study.

Table 31: **Packed equal-user weighting as an estimator control** (BSARec, full CE, five paired seeds). Target reduction averages transitions; equal-user reduction averages within each user first. Each seed pair shares initialization. Timings support within-study comparison under separate profiling conditions.

Dataset	Reduction	NDCG@10	HIT@10	t_{best} (s)	Peak MB
ML-1M	Target-weighted	0.1567 ± 0.0011	0.2788 ± 0.0023	30.1 ± 5.5	342.8
	Equal-user	0.1550 ± 0.0008	0.2772 ± 0.0027	34.9 ± 11.4	342.9
Toys	Target-weighted	0.0442 ± 0.0011	0.0752 ± 0.0016	22.3 ± 3.9	270.6
	Equal-user	0.0368 ± 0.0004	0.0611 ± 0.0007	21.7 ± 8.7	270.6

The learning-rate control selects each organization's step-size multiplier by validation quality before comparing its selected checkpoints (Table 30 and Figure 11). The equal-user control instead changes how target losses are weighted within a packed update (Table 31); its ML-1M and Toys results separate that choice from target count.

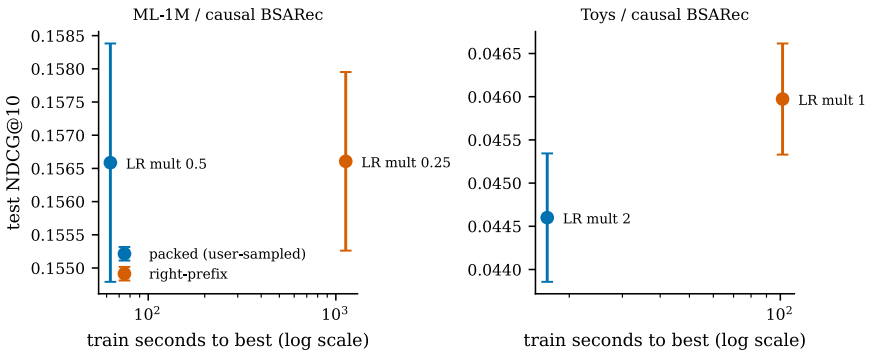


Figure 11: **Validation-selected quality–time frontier** for the learning-rate control.

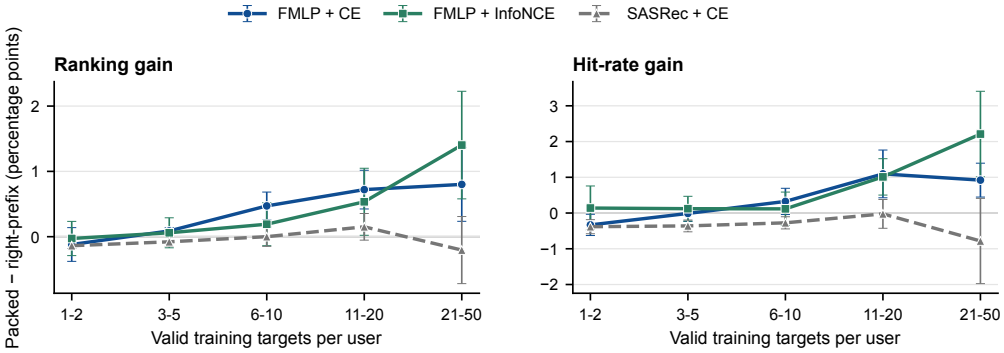


Figure 12: **Packed-minus-right-prefix differences by available targets per user** on Beauty. Bins count valid training targets in a user's history, not the per-update target cap m in the controlled density sweep.

Figure 12 stratifies the practical organization comparison by available history supervision. It describes variation across users rather than an intervention on the update density m.

D Supervision-density protocol and additional results

D.1 Supervision-density protocol

Study coverage. All six ML-1M training arms use five paired seeds (2027–2031; 30 runs). The FMLP and SASRec fixed-state scans each cover 20 user–target allocations, with 128 gradient draws per allocation. The arms share a user split and paired initialization; their epoch counts vary to match the target-event budget, as specified below.

Training allocation. All arms use the same ML-1M training users, causal FMLP, full-catalog CE, AdamW, batch of 128 users, fixed split and order, and paired initialization. Each user's target permutation is fixed and cycled across epochs; a cap m uses $\min(m, \text{available targets})$. Caps $m = 1, 2, 4, 8, 16$ run for 160, 80, 40, 20, and 10 epochs, yielding 966,400 target events each, except $m = 16$ with 966,390. The all-target arm has four epochs of 266,006 events each. Its three- and four-epoch checkpoints bracket, but do not match, the nominal event budget (798,018 events, NDCG@10 .0909, 2.55 training seconds; and 1,064,024 events, .1169, 3.31 seconds, respectively). Table 32 therefore compares the five capped arms only.

Fixed-state variance and cost. The FMLP and ML-1M SASRec gradient scans sample users with at least 16 available targets, whereas the training allocation includes all users. At these checkpoints, held-out relative RMSE for $V_{U,m} = (a + d/m)/U$ is 2.4% and 1.5%, respectively. The new ML-10M and Beauty SASRec scans use the same fixed-state construction and fit (a, d) with held-out relative RMSEs of 10.1% and 1.85%. The scan supplies local covariance behavior for the allocation law; the trained quality curve and runtime response provide the corresponding end-to-end frontier. Absolute GPU timing is fitted with an intercept, $C = C_{\text{fixed}} + U(c_{\text{traj}} + c_{\text{tgt}}m)$; the with-intercept relative RMSEs are .74% and .69% for ML-10M and Beauty. Together, these measurements instantiate the two stages of the framework: local variance–cost allocation followed by budget-conditioned trained response.

Table 32: **Density changes the sample–compute tradeoff.** ML-1M causal FMLP/full CE, five paired seeds. m is the per-user target cap, and shorter histories contribute fewer events. NDCG@10 and training-only time are measured after approximately 966,400 actual events. Crossing columns report events and training-only time to the first observed validation NDCG@10 ≥ 0.14; validation time is excluded.

Target cap m	NDCG@10 ↑	Train (s) ↓	Events to .14 (k) ↓	Train to .14 (s) ↓
1	$.1498 \pm .0012$	202.5	408.3	87.2
2	$\mathbf{.1519 \pm .0021}$	102.8	427.6	46.2
4	$.1500 \pm .0017$	52.3	468.7	26.1
8	$.1483 \pm .0012$	26.7	618.5	17.6
16	$.1435 \pm .0024$	13.9	918.1[†]	13.2[†]

[†]Four of five seeds crossed within the measured budget; one seed is right-censored.

Table 33: **Direct fixed-state allocation estimates.** The variance coefficients a, d and the per-user/per-target timing coefficients $c_{\mathrm{traj}}, c_{\mathrm{tgt}}$ are fitted from separate five-seed SASRec checkpoints. The reported m^*_{state} is the continuous minimizer of Equation (9); it is a local fixed-state variance prediction, not the trained-quality optimum in Table 1.

Dataset	a	d	c_{traj}	c_{tgt}	m^*_{state}	Var./cost RMSE
ML-10M	3.660	137.318	1.023×10^{-5}	2.384×10^{-7}	40.1	10.1% / .74%
Beauty	2.338	201.639	9.493×10^{-6}	3.312×10^{-7}	49.7	1.85% / .69%

Cross-dataset density audit. The eight primary SASRec arms use five paired seeds, full-catalog CE, the same causal architecture and optimizer settings, and target-event budgets of approximately 5.03 million events on ML-10M and 2.00 million events on Beauty. The fit caps are $m \in \{1, 2, 4, 8\}$; $m = 16$ is launched with the same seed set and budget rule but is reserved for the retrospective curve check. On Beauty, the realized density is $\bar{m} = (1.00, 2.00, 3.17, 4.25, 5.05)$ for nominal caps $(1, 2, 4, 8, 16)$, so the cap and the number of targets actually used are not interchangeable.

Separate shared-offset fits over the fit caps give $K_0 = 330.8$ updates for ML-10M/SASRec and $K_0 = 242.0$ for Beauty/SASRec. The relative fit RMSEs are 7.65% and 9.28% over thresholds selected within each dataset's common quality range. The fit-excluded $m = 16$ curve has shifted MAE .0014 versus .0064 under the unshifted coordinate on ML-10M; on Beauty, realized-density saturation produces a distinct frontier regime that we report separately. All eight arms and both fixed-state scans passed seed, protocol, summary, and GPU-exclusivity checks; the aggregate audit is `aggregate_scaling_audit_v1.json` with SHA-256 prefix recorded in the release audit.

Budget-conditioned frontier. The allocation predictions from Equation (9) are 40.1 for ML-10M and 49.7 for Beauty, both above the tested nominal caps. Both datasets therefore occupy a density-increasing fixed-state range over the measured caps. The budget frontier described by Equation (10) supplies the trained operating point in the measured quality–time plane (Figure 4): ML-10M has nondominated caps $m \in \{4, 8, 16\}$, with the highest measured NDCG@10 at $m = 4$, while Beauty has $m = 16$ as a nondominated point because realized density increases to $\bar{m} = 5.05$. Together, these frontiers show how one allocation-to-frontier framework produces dataset-specific operating points.

Update-to-quality fit and fit-excluded density. For each paired seed and validation threshold $q \in \{.08, .10, .12, .13, .14\}$, the crossing is the first *measured* optimizer-update count with NDCG@10 at least q; no between-epoch interpolation is used. A joint least-squares fit estimates one intercept K_0 and a separate B_q in $K_q(m) = K_0 + B_q/m$ over all paired seeds at $m \in \{1, 2, 4, 8\}$, excluding $m = 16$ from coefficient estimation. The fit gives $K_0 = 202.35$; a paired-seed bootstrap with 5,000 resamples gives a 95% interval of $[184.24, 218.46]$. The fitted B_q values, in increasing threshold order, are 373.00, 695.56, 1400.31, 2080.67, and 3030.28 updates. Leaving out one fitted density at a time gives mean per-seed relative crossing errors of 5.1%, 4.0%, 3.7%, and 3.7% for $m = 1, 2, 4, 8$, respectively. For the retrospective, fit-excluded $m = 16$ later segment ($K = 288, 336, 384, 432, 480$), the four fitted-density five-seed mean curves are interpolated at the common coordinate $m(K - K_0)$ and averaged. The predicted $m = 16$ mean NDCG@10 has mean absolute error .00147, versus .01392 with the unshifted coordinate mK. This segment-level alignment is distinct from threshold-crossing evidence: four of five $m = 16$ seeds reach $q = .14$ by the event cap, and one is right-censored.

Measured update cost. Mean training time per optimizer update for $m = 1, 2, 4, 8, 16$ is .02637, .02678, .02725, .02780, and .02893 seconds, respectively. An affine fit to these five means is $\tau^{\mathrm{train}}(m) \simeq .026426 + .0001614m$ seconds ($R^2 = .977$). Its fixed term matters when translat-

Table 34: **Exactness and matched-supervision cost.** Panel (a) uses Float64 aligned prefixes. Panel (b) fixes targets, positions, reduction, update boundaries, and full-CE semantics for Beauty causal FMLP over five seeds. Its timing comes from a separate profile and supports within-study comparison (Appendix C.1).

(a) Exactness audit

Audit	Maximum recorded discrepancy
Five causal lifts: loss / hidden	0 / 0
Five causal lifts: gradient	1.2×10^{-13}
Future perturbation	0
671-target accumulated update: loss / gradient	0 / 1.9×10^{-16}

(b) Matched-supervision training

Execution	NDCG@10 ↑	t_{best} (s) ↓	Peak MB ↓
EPC packed graph	0.0536 ± 0.0004	**15.1**	**294**
Aligned-prefix accumulation	0.0534 ± 0.0007	46.4	332

ing the observed update-to-quality relation into training time; this sweep-specific fit is distinct from the simplified $U(c_{\text{traj}} + c_{\text{tgt}}m)$ allocation model.

Table 35: **From the released Beauty pipeline to packed full-catalog training.** FMLP values are mean ± standard deviation over seeds 2027–2031. BCE is binary cross-entropy loss, CE is full-catalog cross-entropy, and HIT@10 is hit rate among the top ten recommendations. t_{best} is validation-inclusive time to the selected checkpoint. The first transition changes operator and data-interface semantics, the second changes training organization, and the third changes the objective. Objective and target-group controls in Tables 5 and 34 separate the components of this gain.

Configuration	Execution / objective	NDCG@10 ↑	HIT@10 ↑	t_{best} (s) ↓
Released FMLP	prefix / BCE	0.0352 ± 0.0010	0.0650 ± 0.0019	1003.6 ± 204.6
+ causal configuration	prefix / BCE	0.0357 ± 0.0019	0.0658 ± 0.0028	716.4 ± 266.6
+ packed trainer	packed / BCE	0.0352 ± 0.0012	0.0656 ± 0.0019	96.9 ± 26.2
+ full-catalog CE	packed / CE	$\mathbf{0.0538 \pm 0.0011}$	$\mathbf{0.0928 \pm 0.0010}$	$\mathbf{19.7 \pm 1.7}$
Complete configuration vs. released reproduction: $+52.8\%$ NDCG, $+42.8\%$ HIT, $50.9\times$ faster.				

Table 36: **Practical packed training across eight backbones and six datasets.** "Native" backbones are causal as implemented; "lifted" backbones use the causal variants in Section 5.2. W/T/L counts NDCG@10 intervals above, including, and below zero under shared-pool InfoNCE; T means unresolved, not equivalent. ΔNDCG is the mean packed-minus-right-prefix difference across datasets. Time and throughput are geometric-mean right-prefix/packed train-to-best and packed/right-prefix targets/s ratios over paired seeds and datasets; ↑ indicates that larger values favor packed training.

Backbone	Causality	NDCG W/T/L	ΔNDCG	Time × ↑	Targets/s × ↑
SASRec	native	1/4/1	+0.0044	6.86	8.12
FMLP	lifted	3/1/2	+0.0029	7.63	7.98
FEARec	lifted	3/1/2	+0.0033	**10.37**	**9.49**
BSARec	lifted	1/4/1	+0.0016	8.34	9.31
TV-Rec	lifted	5/1/0	+0.0051	4.80	8.12
NextItNet	native	1/0/5	-0.0063	8.33	8.54
RecBLR	native	5/1/0	+0.0057	6.66	7.92
LinRec	lifted	2/3/1	+0.0038	7.20	8.22

Table 37: **Common-quality decomposition.** Right-prefix/packed geometric-mean ratios to reach 95% of each paired right-prefix run's best validation NDCG@10. Only jointly reached pairs enter the cost ratios.

Dataset	Backbone	Reached	Wall × ↑	Targets × ↑	Cost/event × ↑
ML-1M	SASRec	0/5	–	–	–
	FMLP	2/5	5.30	.26	20.30
	RecBLR	5/5	8.88	.53	16.66
	LinRec	0/5	–	–	–
ML-10M	SASRec	5/5	87.42	4.56	19.15
	FMLP	5/5	28.70	1.44	19.96
	RecBLR	5/5	63.37	3.61	17.57
	LinRec	5/5	31.10	1.30	23.91

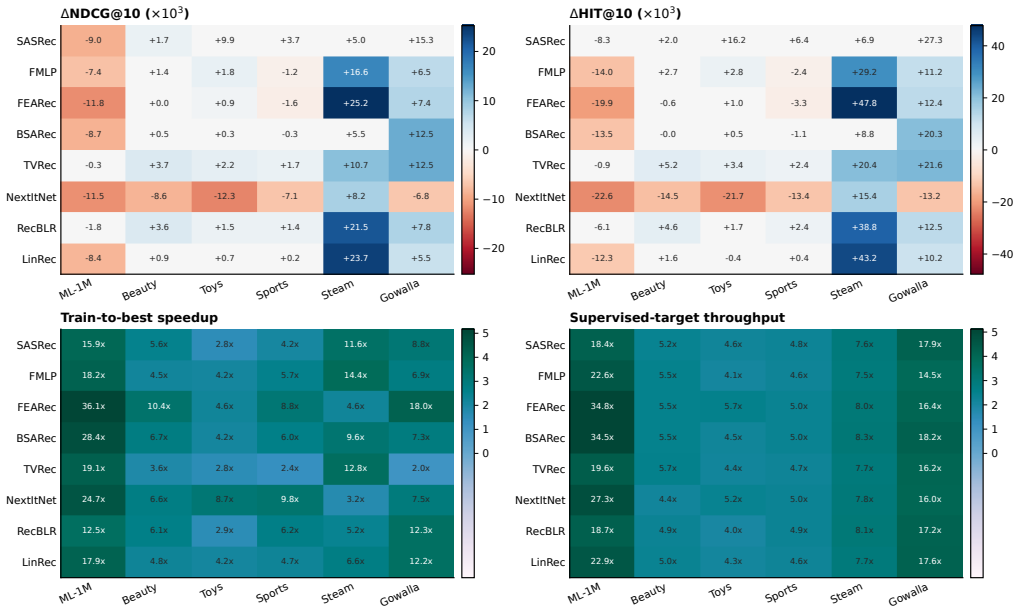


Figure 13: **Practical shared-pool InfoNCE comparisons.** Packed-minus-right-prefix ranking differences, train-to-best ratios, and target-event throughput for eight backbones across six datasets. Training-time and ranking responses are separate outcomes of this policy comparison.