TRAINING ENCODERS NEED NOT BE THE BEST READOUTS: PERSONALIZED POST-TRAINING ENCODER ADAPTATION FOR COLLABORATIVE FILTERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Collaborative filtering (CF) learns user and item representations from interaction histories and ranks items by dot-product scores. From matrix factorization to graph-based models, a fixed encoder matrix maps a trainable embedding table to these representations, and the same encoder is reused at inference. Yet the encoder plays a dual role: during training it both constructs representations and shapes gradient updates to the embedding table, whereas at inference it only reads out the converged table. Convergence therefore does not imply that the training encoder is the best readout, and controlled experiments confirm that an alternative encoder can outperform reuse. A user-wise analysis further shows that each encoder row combines collaborative and preference information whose score effects depend on the learned item geometry, calling for a user-specific readout informed by the trained model. Building on this analysis, we propose **PEAR**, *Personalized post-training Encoder Adaptation for Recommendation*, a lightweight framework for linear inference encoders in CF. PEAR freezes the embedding table and final item representations, constructs two encoder bases from training interactions, fitted predictions, and item geometry, and fits at most two bounded coefficients per user, limiting adaptation capacity without learning new embeddings. Experiments across datasets, backbone encoders, and training objectives show that PEAR consistently improves recommendation performance over converged models while preserving the original dot-product scoring interface and serving cost. Code and data are provided in the supplementary material.

1 INTRODUCTION

Collaborative filtering (CF) is a core technique of modern recommender systems (Adomavicius & Tuzhilin, 2005), which learns user and item representations from interaction histories and ranks candidate items by their scores. A broad class of widely used CF backbones admits a *linear encoder* form (Shen et al., 2021; Park et al., 2025): a fixed matrix, determined by the interaction data and model configuration, maps a trainable embedding table to representations, and scores are dot products of user and item vectors (Definition 1). This matrix is the identity in matrix factorization (MF) (Koren et al., 2009) and in structurally regularized models (Mao et al., 2021; Yu et al., 2026), and a polynomial or spectral filter of the normalized interaction graph in graph-based CF (GCF) (He et al., 2020; Lee et al., 2024a; Lin et al., 2022; Guo et al., 2023). In all these models the same encoder serves both embedding training and inference (He et al., 2020; Lin et al., 2022), although its function differs: during training it also shapes how gradients update the embedding table, whereas at inference it only *reads out* the converged table. This distinction raises a question: once the embedding table has converged, should the training encoder still be its readout?

Progress in CF has come largely from designing better encoders for training, including weighted neighborhood aggregation (He et al., 2020; Lee et al., 2024a), graph filtering (Guo et al., 2023; Liu et al., 2026), learned frequency scalers (Liu et al., 2026), and local–global graph contrastive learning (Cai et al., 2023). Despite their different designs, these methods reuse the training encoder at inference (He et al., 2020; Lin et al., 2022; Guo et al., 2023). A separate line of work has begun to relax this coupling. LightGODE (Zhang et al., 2024) and TAG-CF (Ju et al., 2024) train embeddings without graph convolution and apply propagation after training; Wu et al. (2024) show that a graph-

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.

trained model can benefit from fewer propagation layers at inference. These studies suggest that the training encoder need not be the best readout of the converged table. However, the substitute encoders they adopt are still prescribed by graph structure alone: one encoder is shared by all users and fixed independently of what the model has learned.

We first ask whether a converged embedding table admits a better readout than its training encoder. Fixing each learned table and varying only the inference encoder, we observe that an alternative encoder outperforms reuse in 18 of 30 training configurations, and the preferred choice varies across datasets, objectives, and training encoders (Section 2.2). Our analysis attributes this to the encoder's *dual role* (Proposition 1): during training it both constructs representations and shapes how gradients update the embedding table, whereas at inference only the readout remains, so embedding stationarity does not imply that the training encoder is the best readout. To locate what an inference change should act on, we examine the user row of a linear encoder, the *user encoder*. Each row combines user-origin collaborative and item-origin preference information, and the learned item geometry determines how changes to either source translate into score differences (Section 2.3).

These findings motivate adapting each user's inference encoder after training, which raises two challenges. The first is *deciding how to adjust a user's encoder weights*. A useful adjustment must raise the relative scores of relevant items, and graph proximity alone does not determine which reweighting of the learned embeddings achieves this. The adjustment must instead respond to the user's training interactions, the model's fitted predictions, and the item geometry that governs their effect on scores. The second is *limiting the freedom of the adjustment*. Freezing the embedding table does not by itself bound adaptation capacity: when the table has full column rank, an unrestricted user-encoder row can realize any user representation. Fitting rows directly thus introduces a parameter for every user–node pair and retains the freedom of learning new user embeddings (Section 2.4).

To address these challenges, we propose **PEAR**, *Personalized post-training Encoder Adaptation for Recommendation*, a general, lightweight post-training framework for adapting linear inference encoders in CF. PEAR freezes the embedding table and final item representations, and uses training interactions, fitted predictions, and item geometry to construct two fixed encoder bases per user. One basis recombines collaborative information from user embeddings, while the other recombines preference information from item embeddings. Each user learns at most two bounded mixing coefficients, enabling lightweight personalized adaptation within a low-dimensional space. By separating adaptation from backbone training, PEAR supports different backbone encoders and pretraining objectives through the same procedure. After adaptation, the resulting user representations can be precomputed, preserving the original dot-product scoring interface and serving cost.

Our contributions are threefold:

- We revisit encoder reuse for linear encoders in collaborative filtering, showing empirically that a converged embedding table can admit a better inference encoder and explaining why through the encoder's dual role and a user-wise view of the readout.
- We introduce PEAR, a general, lightweight post-training framework that constructs prediction-guided encoder bases and learns at most two bounded coefficients per user, supporting personalized adaptation across backbone encoders and training objectives.
- We demonstrate consistent gains over converged models across backbone encoders and training objectives, establishing the effectiveness of adapting user encoders with few learnable parameters.

2 Preliminaries and Analysis

2.1 Linear Encoders in Collaborative Filtering

Collaborative filtering (CF) learns representations of users and items from observed interactions. Let $\mathcal{U}$ and $\mathcal{I}$ denote the user and item sets, with $n = |\mathcal{U}|$, $m = |\mathcal{I}|$, and $N = n + m$. The binary matrix $\boldsymbol{R} \in \{0,1\}^{n \times m}$ records observed training interactions. We collect the trainable user and item embeddings in an embedding table $\boldsymbol{E} \in \mathbb{R}^{N \times d}$, with users preceding items. An encoder maps this table to representations, while an objective specifies how the table is learned.

Definition 1 (Linear node encoder)**.** For a fixed set of user and item nodes, a linear node encoder is a matrix $\boldsymbol{H} \in \mathbb{R}^{N \times N}$ that combines their embeddings into representations for recommendation:

$$\boldsymbol{X} = \boldsymbol{H}\boldsymbol{E} = [\boldsymbol{Z}; \boldsymbol{V}], \qquad s_{ui} = \boldsymbol{z}_u^\top \boldsymbol{v}_i, \tag{1}$$

where $\boldsymbol{z}_u^\top$ and $\boldsymbol{v}_i^\top$ are rows of the user and item representation matrices $\boldsymbol{Z} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{V} \in \mathbb{R}^{m\times d}$. The predicted preference score s_{ui} measures user u's compatibility with item i.

Many CF backbones use an encoder of this form, fixed by the interaction data and model configuration while $\boldsymbol{E}$ is optimized. The formulation accommodates identity mappings (Mao et al., 2021; Yu et al., 2026), weighted neighborhood aggregation (Lin et al., 2022; Yu et al., 2022), and polynomial combinations of fixed propagation operators (Guo et al., 2023; Liu et al., 2026). MF (Koren et al., 2009) and LightGCN (He et al., 2020) are two widely used instances. MF uses $\boldsymbol{H} = \boldsymbol{I}_N$, directly taking representations from $\boldsymbol{E}$. For LightGCN, define the normalized interaction operator

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{R} \\ \boldsymbol{R}^\top & \boldsymbol{0} \end{bmatrix}, \qquad \boldsymbol{S} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}, \qquad \boldsymbol{H}_{\mathrm{LGCN}} = \sum_{k=0}^{K} \alpha_k \boldsymbol{S}^k, \tag{2}$$

where $\boldsymbol{D}$ contains node degrees and α_k are fixed aggregation weights. We also evaluate LightGCN++ (Lee et al., 2024a); Appendix B.3 specifies the post-training readout used to connect its frozen checkpoint to our inference-encoder formulation.

The usual training pipeline fixes a *training encoder* $\boldsymbol{H}_{\mathrm{tr}}$ and optimizes $\boldsymbol{E}$ using a recommendation objective evaluated on $\boldsymbol{H}_{\mathrm{tr}}\boldsymbol{E}$. After training converges, we freeze the learned embedding table and denote it by $\boldsymbol{E}_0$. An *inference encoder* $\boldsymbol{H}_{\mathrm{inf}}$ then reads out the final representations $\boldsymbol{X}_{\mathrm{inf}} = \boldsymbol{H}_{\mathrm{inf}}\boldsymbol{E}_0$ for recommendation. The most common practice is to reuse the training encoder at inference, setting $\boldsymbol{H}_{\mathrm{inf}} = \boldsymbol{H}_{\mathrm{tr}}$ (He et al., 2020; Lin et al., 2022; Guo et al., 2023; Liu et al., 2026; Yu et al., 2026).

2.2 Revisiting Encoder Reuse at Inference

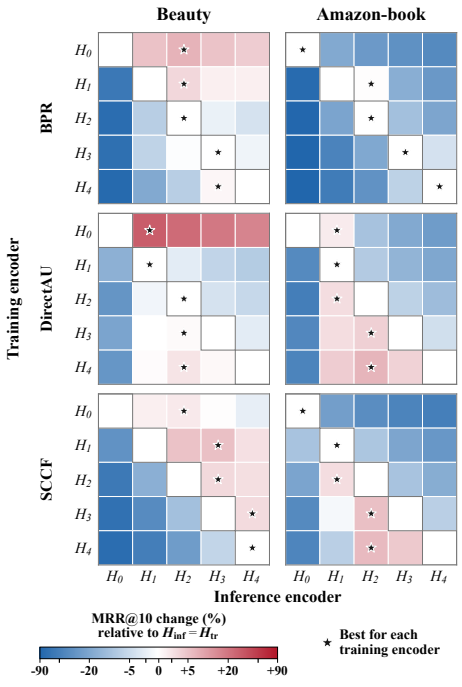


Figure 1: Encoder reuse versus substitution. Each row fixes a trained embedding table. Colors show the percentage change in test MRR@10 relative to its diagonal entry. Stars mark retrospective row-wise maxima. Positive off-diagonal entries favor substitution.

To examine how the choice of inference encoder depends on embedding training, we hold each converged embedding table fixed and vary only the inference encoder. This controlled comparison isolates the effect of changing the readout from that of further embedding optimization. To obtain a simple family of alternative encoders, we instantiate equation 2 with the widely used uniform-weight setting:

$$\mathcal{H} = \{\boldsymbol{H}_0, \ldots, \boldsymbol{H}_4\}, \quad \boldsymbol{H}_k = \frac{1}{k+1}\sum_{j=0}^{k} \boldsymbol{S}^j. \tag{3}$$

Here $\boldsymbol{H}_0 = \boldsymbol{I}_N$ is the MF encoder; the remaining matrices provide distinct linear combinations of the same embedding table. We pair each training encoder with three objectives: BPR, a widely used pairwise ranking loss (Rendle et al., 2009); DirectAU, a widely adopted objective that directly optimizes representation alignment and uniformity (Wang et al., 2022); and SCCF, a more recent contrastive objective with state-of-the-art recommendation performance (Wu et al., 2024).

Encoder reuse is not uniformly preferable. Figure 1 compares inference encoders on Beauty and Amazon-book with each trained embedding table held fixed (see Appendix B.1 for details). An alternative inference encoder achieves higher mean test MRR@10 than encoder reuse in 18 of the 30 training configurations. The preferred encoder also varies across losses, datasets, and training encoders. Thus, an embedding table can admit a better inference readout than its training encoder. To understand why these choices can differ, we examine the encoder's roles during training and inference.

The dual role of a linear encoder. During training, the encoder both constructs representations and determines how their gradients update the embedding table. At inference, only the representation mapping remains. To formalize this distinction, let $J_{\boldsymbol{H}}(\boldsymbol{E}) = \ell(\boldsymbol{H}\boldsymbol{E}) + \Omega(\boldsymbol{E})$ be the training objective, where ℓ is a differentiable data-fitting loss on the encoded representations and Ω is a differentiable regularizer on the embedding table.

Proposition 1 (The dual role of a linear encoder)**.** *For a fixed $\boldsymbol{H}$, write $\boldsymbol{X} = \boldsymbol{H}\boldsymbol{E}$ and $\boldsymbol{G} = \nabla_{\boldsymbol{X}}\ell(\boldsymbol{X})$. Then*

$$\nabla_{\boldsymbol{E}} J_{\boldsymbol{H}} = \boldsymbol{H}^\top \boldsymbol{G} + \nabla_{\boldsymbol{E}}\Omega. \tag{4}$$

Under plain gradient descent with step size η, the induced representation update is

$$\boldsymbol{X}^+ = \boldsymbol{X} - \eta \boldsymbol{H}\boldsymbol{H}^\top \boldsymbol{G} - \eta \boldsymbol{H}\nabla_{\boldsymbol{E}}\Omega. \tag{5}$$

After freezing $\boldsymbol{E} = \boldsymbol{E}_0$, substituting $\boldsymbol{H}$ changes only the readout $\boldsymbol{H}\boldsymbol{E}_0$. Stationarity with respect to $\boldsymbol{E}$ does not, in general, imply that $\boldsymbol{H}$ minimizes $\ell(\boldsymbol{H}\boldsymbol{E}_0)$ over admissible encoders.

app:dual-role gives the full proof and an optimality counterexample. Together, the empirical results and the dual-role analysis show that an encoder suited to training is not necessarily the best readout for the learned embedding table.

2.3 A User-wise View of the Inference Encoder

To understand how an encoder determines an individual user's recommendations, we now examine the scoring function conditional on a fixed final item matrix $\boldsymbol{V}$ and the frozen embedding table $\boldsymbol{E}_0$.

Definition 2 (User encoder)**.** The *user encoder* for user u is the row $\boldsymbol{p}_u^\top = (\boldsymbol{e}^{(u)})^\top \boldsymbol{H}_{\text{inf}} \in \mathbb{R}^{1\times N}$, where $\boldsymbol{e}^{(u)}$ is the uth standard basis vector. Given $\boldsymbol{E}_0$ and $\boldsymbol{V}$, it determines the user's representation and score vector:

$$\boldsymbol{z}_u = \boldsymbol{E}_0^\top \boldsymbol{p}_u, \qquad \boldsymbol{s}_u^\top = \boldsymbol{p}_u^\top \boldsymbol{E}_0 \boldsymbol{V}^\top. \tag{6}$$

For fixed candidates and tie-breaking, $\boldsymbol{p}_u$ determines the ranking.

Collaborative and preference information within one row. Partition the frozen table as $\boldsymbol{E}_0 = [\boldsymbol{U}_0; \boldsymbol{I}_0]$ and the user encoder as $\boldsymbol{p}_u = [\boldsymbol{p}_u^{\mathcal{U}}; \boldsymbol{p}_u^{\mathcal{I}}]$, where $\boldsymbol{p}_u^{\mathcal{U}} \in \mathbb{R}^n$ and $\boldsymbol{p}_u^{\mathcal{I}} \in \mathbb{R}^m$. Expanding the readout over these blocks gives

$$\boldsymbol{s}_u^\top = \boldsymbol{p}_u^\top \boldsymbol{E}_0 \boldsymbol{V}^\top = \big((\boldsymbol{p}_u^{\mathcal{U}})^\top \boldsymbol{U}_0 + (\boldsymbol{p}_u^{\mathcal{I}})^\top \boldsymbol{I}_0\big)\boldsymbol{V}^\top = \underbrace{(\boldsymbol{p}_u^{\mathcal{U}})^\top \boldsymbol{U}_0 \boldsymbol{V}^\top}_{\text{user-origin contribution}} + \underbrace{(\boldsymbol{p}_u^{\mathcal{I}})^\top \boldsymbol{I}_0 \boldsymbol{V}^\top}_{\text{item-origin contribution}}. \tag{7}$$

The first term combines learned user embeddings, including the user's own, to contribute collaborative information. The second combines learned item embeddings to contribute preference information. These names identify information sources: both tables have been learned from collaborative interactions. More specifically, each block weights the item-score profiles represented by the rows of $\boldsymbol{U}_0\boldsymbol{V}^\top$ or $\boldsymbol{I}_0\boldsymbol{V}^\top$. Reusing an encoder preserves the same weights over these two sets of profiles after training. This view identifies which learned information is combined for recommendation. To assess changes to these combinations, we next examine how the learned item representations shape their effects on score differences.

Item representations induce the geometry of encoder changes. An encoder change $\Delta\boldsymbol{p}_u$ produces $\Delta\boldsymbol{z}_u = \boldsymbol{E}_0^\top \Delta\boldsymbol{p}_u$ and $\Delta\boldsymbol{s}_u = \boldsymbol{V}\Delta\boldsymbol{z}_u$. Because adding the same constant to every item score leaves the ranking unchanged, consider centered score changes. Let $\boldsymbol{C}_m = \boldsymbol{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$, where $\mathbf{1} \in \mathbb{R}^m$ is the all-ones vector, and define

$$\boldsymbol{Q}_{\boldsymbol{V}} = \frac{1}{m}\boldsymbol{V}^\top \boldsymbol{C}_m \boldsymbol{V} = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{v}_i - \bar{\boldsymbol{v}})(\boldsymbol{v}_i - \bar{\boldsymbol{v}})^\top, \qquad \bar{\boldsymbol{v}} = \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{v}_i. \tag{8}$$

Then the squared change in centered scores satisfies

$$\frac{1}{m}\|\boldsymbol{C}_m \Delta\boldsymbol{s}_u\|_2^2 = \Delta\boldsymbol{z}_u^\top \boldsymbol{Q}_{\boldsymbol{V}} \Delta\boldsymbol{z}_u = \Delta\boldsymbol{p}_u^\top \boldsymbol{E}_0 \boldsymbol{Q}_{\boldsymbol{V}} \boldsymbol{E}_0^\top \Delta\boldsymbol{p}_u. \tag{9}$$

The positive semidefinite matrix $\boldsymbol{Q}_{\boldsymbol{V}}$ captures the scale and correlations of the learned item representations. Unless it is proportional to the identity, equal-length representation changes can have

different effects on item-score differences; a change in its null space leaves all score differences unchanged. Thus, Euclidean displacement in representation space need not reflect the magnitude of a change in recommendation scores. Together, equation 7 and equation 9 identify two aspects of user-encoder adaptation: which information sources to combine, and how those combinations affect scores under the learned item geometry.

2.4 From Structural Readout to Personalized Adaptation

Beyond fixed structural coefficients. The preceding analysis identifies what user encoders combine and how their changes affect scores. Yet in widely used structural readouts such as LightGCN (He et al., 2020) and TAG-CF (Ju et al., 2024), aggregation coefficients are prescribed by the interaction graph and propagation settings, without explicitly responding to fitted representations or predictions. The learned score profiles in equation 7 instead motivate revisiting each user's combination of collaborative and preference information after training. We study this conditional problem by freezing $\boldsymbol{E}_0$ and the final item matrix $\boldsymbol{V}$ obtained with $\boldsymbol{H}_{\mathrm{tr}}$, and adapting only user encoders. By equation 6, changing one user-encoder row affects only that user's scores, whereas changing an item representation can affect all users. Fixing $\boldsymbol{V}$ also preserves the uniform score geometry $\boldsymbol{Q}_{\boldsymbol{V}}$, isolating readout adaptation within the existing item space.

Guiding and constraining user adaptation. Two questions remain: how to guide user-specific changes and how to limit their flexibility. Let $\ell_u(\boldsymbol{z}; \boldsymbol{V})$ be a twice continuously differentiable user objective using training interactions, and define $\boldsymbol{d}_u = -\nabla_{\boldsymbol{z}} \ell_u(\boldsymbol{z}_u; \boldsymbol{V})$ at the original readout. For $\Delta \boldsymbol{z}_u = \boldsymbol{E}_0^\top \Delta \boldsymbol{p}_u$, a local expansion gives

$$\ell_u(\boldsymbol{z}_u + \Delta \boldsymbol{z}_u; \boldsymbol{V}) - \ell_u(\boldsymbol{z}_u; \boldsymbol{V}) = -\boldsymbol{d}_u^\top \Delta \boldsymbol{z}_u + O(\|\Delta \boldsymbol{z}_u\|_2^2). \tag{10}$$

First-order utility depends on alignment with $\boldsymbol{d}_u$, so different training positives or fitted scores can favor different corrections even at equal interaction counts. The item geometry in equation 9 quantifies relative score changes; computing the item covariance under each user's predictive distribution personalizes this measure. This local training analysis does not guarantee held-out ranking gains, which require empirical evaluation. Freezing $\boldsymbol{E}_0$ alone need not limit adaptation capacity: if it has full column rank, an unrestricted $\boldsymbol{p}_u$ can produce any representation in $\mathbb{R}^d$ through $\boldsymbol{E}_0^\top \boldsymbol{p}_u$. Together, these considerations motivate fixed, source-specific encoder bases guided by training signals and predictive geometry, with only a few bounded coefficients fitted per user.

3 Method

PEAR, *Personalized post-training Encoder Adaptation for Recommendation*, is a post-training framework that personalizes user-side inference readouts while keeping the learned embedding table and final item representations fixed. Section 3.1 first defines a compact personalized encoder with two source-specific adaptation directions. Section 3.2 then derives a user-specific target from training interactions, fitted predictions, and item geometry. Section 3.3 translates this target into fixed item- and user-source encoder bases, and Section 3.4 fits two bounded coefficients per user. Together, these stages form a lightweight pipeline that produces adapted user representations while preserving the original dot-product scoring interface.

3.1 A Compact Personalized Inference Encoder

PEAR adapts how each user combines preference information from learned item embeddings and collaborative information from learned user embeddings. Let $\boldsymbol{p}_u^{(0)\top}$ denote user u's row of the training encoder $\boldsymbol{H}_{\mathrm{tr}}$, and $\boldsymbol{z}_u^{(0)}$ the original user representation. PEAR's final personalized inference encoder and its resulting representation are

$$\begin{aligned} \boldsymbol{p}_u(\boldsymbol{c}_u) &= \boldsymbol{p}_u^{(0)} + c_u^I \boldsymbol{q}_u^I + c_u^U \boldsymbol{q}_u^U, \\ \boldsymbol{z}_u(\boldsymbol{c}_u) &= \boldsymbol{E}_0^\top \boldsymbol{p}_u(\boldsymbol{c}_u) = \boldsymbol{z}_u^{(0)} + c_u^I \boldsymbol{b}_u^I + c_u^U \boldsymbol{b}_u^U. \end{aligned} \tag{11}$$

The fixed encoder bases $\boldsymbol{q}_u^I, \boldsymbol{q}_u^U \in \mathbb{R}^N$ act on the item and user blocks, respectively; each induces a representation direction $\boldsymbol{b}_u^a = \boldsymbol{E}_0^\top \boldsymbol{q}_u^a$ for source $a \in \{I, U\}$. The bounded, signed mixing coefficients $\boldsymbol{c}_u = (c_u^I, c_u^U)^\top$ restrict adaptation to a bounded subset of an affine space of dimension at most two. We next make these directions responsive to the trained model.

3.2 Training Signal and Personalized Score Geometry

To support different pretraining objectives through the same procedure, PEAR uses a common positive-set loss for basis construction and coefficient fitting. For training positives $\mathcal{P}_u = \{i : R_{ui} = 1\}$, the predictive probabilities for a user representation $\boldsymbol{z}$ are $\pi_{ui}(\boldsymbol{z}) \propto \exp(\boldsymbol{z}^\top \boldsymbol{v}_i/\sigma_u)$ over the full catalog, where $\boldsymbol{v}_i$ is the fixed item representation and σ_u is the fixed baseline root-mean-square score. The loss $\ell_u(\boldsymbol{z}; \boldsymbol{V}) = -\log \sum_{i\in\mathcal{P}_u} \pi_{ui}(\boldsymbol{z})$ rewards probability mass on the positive set, with positives weighted by their current probabilities.

The scaled negative gradient $\boldsymbol{g}_u = -\sigma_u \nabla_{\boldsymbol{z}} \ell_u(\boldsymbol{z}_u^{(0)}; \boldsymbol{V})$ identifies locally useful changes. The predictive item covariance $\boldsymbol{C}_u = \mathrm{Cov}_{i\sim\pi_u(\boldsymbol{z}_u^{(0)})}(\boldsymbol{v}_i)$ measures their effects on relative scores, weighting items by the user's fitted predictions. To balance training utility against predictive change, we define the target representation direction

$$\boldsymbol{K}_u = \boldsymbol{C}_u + \gamma_u \boldsymbol{I}_d, \qquad \boldsymbol{t}_u = \arg\max_{\boldsymbol{t}} \left\{ \boldsymbol{g}_u^\top \boldsymbol{t} - \tfrac{1}{2} \boldsymbol{t}^\top \boldsymbol{K}_u \boldsymbol{t} \right\} = \boldsymbol{K}_u^{-1} \boldsymbol{g}_u. \tag{12}$$

The damping strength $\gamma_u > 0$ limits changes along weakly constrained directions. The predictive Fisher matrix $\boldsymbol{C}_u/\sigma_u^2$ is generally distinct from the loss Hessian (Appendix C.1).

3.3 Constructing Bases from Two Information Sources

We now translate the target representation direction into two encoder bases that recombine the model's learned information. For each source $a \in \{I, U\}$, the dictionary $\boldsymbol{B}_u^a = \boldsymbol{W}_u^a \boldsymbol{E}_0$ uses a selection-and-rescaling matrix $\boldsymbol{W}_u^a$. The item dictionary contains all normalized original item embeddings; the user dictionary uses normalized original embeddings of neighbors selected by degree-discounted training-interaction overlap (Appendix C.2). The source weights solve

$$\boldsymbol{\eta}_u^a = \arg\min_{\boldsymbol{\eta}} \frac{1}{2} \|\boldsymbol{B}_u^{a\top} \boldsymbol{\eta} - \boldsymbol{t}_u\|_{\boldsymbol{K}_u}^2 + \frac{\lambda_u^a}{2} \|\boldsymbol{\eta}\|_2^2, \qquad a \in \{I, U\}, \tag{13}$$

where the predictive metric defines $\|\boldsymbol{x}\|_{\boldsymbol{K}_u}^2 = \boldsymbol{x}^\top \boldsymbol{K}_u \boldsymbol{x}$ and λ_u^a controls ridge regularization. Both sources follow the same training signal, while their dictionaries induce source-dependent ridge regularization, even when a dictionary spans the representation space.

The encoder basis $\boldsymbol{q}_u^a = r_u^a \boldsymbol{W}_u^{a\top} \boldsymbol{\eta}_u^a$ uses a scale factor r_u^a that matches the induced representation direction's norm to the original user norm $\|\boldsymbol{z}_u^{(0)}\|_2$; zero directions remain zero (Appendix C.3). This gives the coefficients comparable scales relative to each user's original representation. Under the conditions in Appendix C.4, each nonzero component is a local descent direction for the training loss at initialization, without implying held-out ranking gains. With these prediction-guided bases fixed, only their contributions remain to be fitted.

3.4 Fitting Two Coefficients per User

PEAR learns source coefficients $c_u^a = \kappa_a \tanh(\theta_u^a/\kappa_a)$, where θ_u^a is trainable and $\kappa_a > 0$ bounds its contribution. Zero initialization recovers the pretrained encoder. For users with multiple training positives, collected in $\mathcal{U}_{\mathrm{fit}}$, we minimize

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{U}_{\mathrm{fit}}|} \sum_{u\in\mathcal{U}_{\mathrm{fit}}} \left[\ell_u(\boldsymbol{z}_u(\boldsymbol{c}_u); \boldsymbol{V}) + \beta \|\boldsymbol{c}_u\|_2^2 \right]. \tag{14}$$

The penalty strength β shrinks the mixing coefficients toward zero. The loss uses the current adapted user representation; the bases, embedding table, item representations, and baseline scales stay fixed. A fixed, damped 2×2 Fisher preconditioner computed at the original readout accounts for the two directions' coupled predictive effects (Appendix C.5).

PEAR thus personalizes linear CF readouts with at most two learned scalars per user. The adapted user vectors are cached and scored against the fixed item representations using the original dot product, preserving serving cost. Construction costs are detailed in Appendix C.7; Algorithm 1 summarizes the procedure.

Table 1: Performance comparison of different backbones and objectives. Results are reported as mean ± sample standard deviation over five random seeds. The "Avg. Improv." rows first compute the relative improvement from the five-seed means for each objective and then average these relative improvements over the four objectives.

		Douban-book		Amazon-kindle		Amazon-book		Beauty	
Backbone	Objective	MRR@10	NDCG@10	MRR@10	NDCG@10	MRR@10	NDCG@10	MRR@10	NDCG@10
MF	BPR	$0.1786_{\pm 0.0040}$	$0.0918_{\pm 0.0023}$	$0.1078_{\pm 0.0020}$	$0.0844_{\pm 0.0016}$	$0.0687_{\pm 0.0010}$	$0.0380_{\pm 0.0007}$	$0.0392_{\pm 0.0007}$	$0.0400_{\pm 0.0007}$
	+PEAR	$\mathbf{0.2416}_{\pm 0.0041}$	$\mathbf{0.1275}_{\pm 0.0023}$	$\mathbf{0.1428}_{\pm 0.0023}$	$\mathbf{0.1073}_{\pm 0.0018}$	$\mathbf{0.0978}_{\pm 0.0014}$	$\mathbf{0.0543}_{\pm 0.0008}$	$\mathbf{0.0433}_{\pm 0.0007}$	$\mathbf{0.0438}_{\pm 0.0007}$
	DirectAU	$0.1500_{\pm 0.0026}$	$0.0807_{\pm 0.0012}$	$0.1252_{\pm 0.0026}$	$0.0976_{\pm 0.0024}$	$0.0706_{\pm 0.0010}$	$0.0402_{\pm 0.0005}$	$0.0499_{\pm 0.0007}$	$0.0503_{\pm 0.0008}$
	+PEAR	$\mathbf{0.2189}_{\pm 0.0040}$	$\mathbf{0.1137}_{\pm 0.0020}$	$\mathbf{0.1562}_{\pm 0.0040}$	$\mathbf{0.1177}_{\pm 0.0033}$	$\mathbf{0.0976}_{\pm 0.0010}$	$\mathbf{0.0552}_{\pm 0.0004}$	$\mathbf{0.0540}_{\pm 0.0008}$	$\mathbf{0.0530}_{\pm 0.0009}$
	LightCCF	$0.2667_{\pm 0.0062}$	$0.1433_{\pm 0.0038}$	$0.1645_{\pm 0.0017}$	$0.1267_{\pm 0.0015}$	$0.1176_{\pm 0.0013}$	$0.0672_{\pm 0.0007}$	$0.0607_{\pm 0.0008}$	$0.0596_{\pm 0.0008}$
	+PEAR	$\mathbf{0.2968}_{\pm 0.0085}$	$\mathbf{0.1575}_{\pm 0.0044}$	$\mathbf{0.1840}_{\pm 0.0016}$	$\mathbf{0.1375}_{\pm 0.0016}$	$\mathbf{0.1427}_{\pm 0.0015}$	$\mathbf{0.0798}_{\pm 0.0008}$	$\mathbf{0.0641}_{\pm 0.0008}$	$\mathbf{0.0621}_{\pm 0.0009}$
	SCCF	$0.2516_{\pm 0.0050}$	$0.1364_{\pm 0.0027}$	$0.1499_{\pm 0.0018}$	$0.1162_{\pm 0.0018}$	$0.1182_{\pm 0.0015}$	$0.0665_{\pm 0.0011}$	$0.0530_{\pm 0.0007}$	$0.0514_{\pm 0.0006}$
	+PEAR	$\mathbf{0.2842}_{\pm 0.0054}$	$\mathbf{0.1506}_{\pm 0.0025}$	$\mathbf{0.1764}_{\pm 0.0024}$	$\mathbf{0.1312}_{\pm 0.0017}$	$\mathbf{0.1449}_{\pm 0.0021}$	$\mathbf{0.0800}_{\pm 0.0012}$	$\mathbf{0.0562}_{\pm 0.0011}$	$\mathbf{0.0526}_{\pm 0.0011}$
	Avg. Improv.	+26.36%	+25.03%	+21.69%	+17.29%	+31.13%	+29.81%	+7.58%	+5.35%
LightGCN (2 layers)	BPR	$0.2089_{\pm 0.0041}$	$0.1095_{\pm 0.0021}$	$0.1285_{\pm 0.0025}$	$0.1001_{\pm 0.0021}$	$0.0809_{\pm 0.0006}$	$0.0453_{\pm 0.0004}$	$0.0546_{\pm 0.0008}$	$0.0560_{\pm 0.0007}$
	+PEAR	$\mathbf{0.2592}_{\pm 0.0056}$	$\mathbf{0.1359}_{\pm 0.0032}$	$\mathbf{0.1583}_{\pm 0.0037}$	$\mathbf{0.1189}_{\pm 0.0029}$	$\mathbf{0.1060}_{\pm 0.0014}$	$\mathbf{0.0592}_{\pm 0.0008}$	$\mathbf{0.0592}_{\pm 0.0011}$	$\mathbf{0.0598}_{\pm 0.0011}$
	DirectAU	$0.2210_{\pm 0.0026}$	$0.1171_{\pm 0.0009}$	$0.1225_{\pm 0.0024}$	$0.0940_{\pm 0.0019}$	$0.0737_{\pm 0.0010}$	$0.0408_{\pm 0.0006}$	$0.0671_{\pm 0.0009}$	$0.0672_{\pm 0.0011}$
	+PEAR	$\mathbf{0.2691}_{\pm 0.0039}$	$\mathbf{0.1406}_{\pm 0.0023}$	$\mathbf{0.1648}_{\pm 0.0040}$	$\mathbf{0.1225}_{\pm 0.0033}$	$\mathbf{0.1057}_{\pm 0.0017}$	$\mathbf{0.0577}_{\pm 0.0010}$	$\mathbf{0.0704}_{\pm 0.0011}$	$\mathbf{0.0692}_{\pm 0.0011}$
	LightCCF	$0.2606_{\pm 0.0023}$	$0.1399_{\pm 0.0009}$	$0.1592_{\pm 0.0031}$	$0.1237_{\pm 0.0020}$	$0.1087_{\pm 0.0013}$	$0.0619_{\pm 0.0009}$	$0.0585_{\pm 0.0015}$	$0.0582_{\pm 0.0014}$
	+PEAR	$\mathbf{0.2994}_{\pm 0.0019}$	$\mathbf{0.1584}_{\pm 0.0012}$	$\mathbf{0.1822}_{\pm 0.0028}$	$\mathbf{0.1362}_{\pm 0.0024}$	$\mathbf{0.1356}_{\pm 0.0011}$	$\mathbf{0.0753}_{\pm 0.0007}$	$\mathbf{0.0621}_{\pm 0.0015}$	$\mathbf{0.0606}_{\pm 0.0016}$
	SCCF	$0.2581_{\pm 0.0027}$	$0.1393_{\pm 0.0017}$	$0.1428_{\pm 0.0028}$	$0.1114_{\pm 0.0020}$	$0.1029_{\pm 0.0023}$	$0.0581_{\pm 0.0012}$	$0.0571_{\pm 0.0009}$	$0.0575_{\pm 0.0008}$
	+PEAR	$\mathbf{0.2905}_{\pm 0.0032}$	$\mathbf{0.1532}_{\pm 0.0015}$	$\mathbf{0.1784}_{\pm 0.0035}$	$\mathbf{0.1319}_{\pm 0.0027}$	$\mathbf{0.1360}_{\pm 0.0033}$	$\mathbf{0.0746}_{\pm 0.0020}$	$\mathbf{0.0609}_{\pm 0.0011}$	$\mathbf{0.0591}_{\pm 0.0010}$
	Avg. Improv.	+18.32%	+16.85%	+24.27%	+19.40%	+32.84%	+30.54%	+6.54%	+4.17%
LightGCN++	BPR	$0.2000_{\pm 0.0030}$	$0.1033_{\pm 0.0017}$	$0.1108_{\pm 0.0010}$	$0.0849_{\pm 0.0007}$	$0.0670_{\pm 0.0012}$	$0.0381_{\pm 0.0008}$	$0.0435_{\pm 0.0006}$	$0.0441_{\pm 0.0006}$
	+PEAR	$\mathbf{0.2187}_{\pm 0.0033}$	$\mathbf{0.1128}_{\pm 0.0015}$	$\mathbf{0.1484}_{\pm 0.0021}$	$\mathbf{0.1069}_{\pm 0.0016}$	$\mathbf{0.0902}_{\pm 0.0018}$	$\mathbf{0.0492}_{\pm 0.0009}$	$\mathbf{0.0510}_{\pm 0.0007}$	$\mathbf{0.0505}_{\pm 0.0007}$
	DirectAU	$0.2434_{\pm 0.0030}$	$0.1285_{\pm 0.0018}$	$0.0925_{\pm 0.0023}$	$0.0698_{\pm 0.0020}$	$0.0606_{\pm 0.0006}$	$0.0322_{\pm 0.0004}$	$0.0566_{\pm 0.0012}$	$0.0562_{\pm 0.0013}$
	+PEAR	$\mathbf{0.2534}_{\pm 0.0045}$	$\mathbf{0.1320}_{\pm 0.0026}$	$\mathbf{0.1110}_{\pm 0.0034}$	$\mathbf{0.0807}_{\pm 0.0025}$	$\mathbf{0.0620}_{\pm 0.0007}$	$\mathbf{0.0332}_{\pm 0.0005}$	$\mathbf{0.0588}_{\pm 0.0011}$	$\mathbf{0.0585}_{\pm 0.0011}$
	LightCCF	$0.2190_{\pm 0.0033}$	$0.1135_{\pm 0.0018}$	$0.1055_{\pm 0.0021}$	$0.0799_{\pm 0.0015}$	$0.0661_{\pm 0.0009}$	$0.0362_{\pm 0.0004}$	$0.0435_{\pm 0.0006}$	$0.0430_{\pm 0.0007}$
	+PEAR	$\mathbf{0.2352}_{\pm 0.0044}$	$\mathbf{0.1202}_{\pm 0.0022}$	$\mathbf{0.1444}_{\pm 0.0031}$	$\mathbf{0.1037}_{\pm 0.0019}$	$\mathbf{0.0818}_{\pm 0.0011}$	$\mathbf{0.0443}_{\pm 0.0006}$	$\mathbf{0.0494}_{\pm 0.0007}$	$\mathbf{0.0488}_{\pm 0.0008}$
	SCCF	$0.2400_{\pm 0.0033}$	$0.1227_{\pm 0.0018}$	$0.1025_{\pm 0.0021}$	$0.0786_{\pm 0.0012}$	$0.0710_{\pm 0.0014}$	$0.0386_{\pm 0.0007}$	$0.0451_{\pm 0.0009}$	$0.0453_{\pm 0.0008}$
	+PEAR	$\mathbf{0.2498}_{\pm 0.0046}$	$\mathbf{0.1260}_{\pm 0.0021}$	$\mathbf{0.1367}_{\pm 0.0022}$	$\mathbf{0.0989}_{\pm 0.0012}$	$\mathbf{0.0826}_{\pm 0.0012}$	$\mathbf{0.0442}_{\pm 0.0005}$	$\mathbf{0.0513}_{\pm 0.0012}$	$\mathbf{0.0511}_{\pm 0.0012}$
	Avg. Improv.	+6.23%	+5.13%	+31.04%	+24.29%	+19.26%	+17.28%	+12.11%	+11.22%

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and protocols. We evaluate on four public benchmarks: Douban-book (Yu et al., 2022), Amazon-kindle (Wu et al., 2021), Amazon-book (He & McAuley, 2016), and Beauty (Yu et al., 2026) (details in Appendix D). Interactions are split into training, validation, and test sets with an 8:1:1 ratio. For all models, only the training split is used during training, including the construction of graphs, propagators, and other data-dependent model components. We report MRR@10 and NDCG@10 under the full-ranking evaluation protocol.

Baselines. We consider three backbones: MF (Koren et al., 2009), LightGCN (He et al., 2020), and LightGCN++ (Lee et al., 2024a). LightGCN++ (Lee et al., 2024a) uses degree-controlled propagation with parameter-free normalization and layer pooling. Each backbone is trained with four objectives: BPR (Rendle et al., 2009), which optimizes pairwise ranking; DirectAU (Wang et al., 2022), which directly optimizes representation alignment and uniformity; and LightCCF (Zhang et al., 2025) and SCCF (Wu et al., 2024), which incorporate contrastive learning signals, with LightCCF combining its contrastive term with BPR. This crossed design allows us to evaluate PEAR across different encoders and pretraining objectives.

Implementation. All methods are implemented with RecBole (Zhao et al., 2021) for fair comparison. Hyperparameters and checkpoints are selected using validation MRR@10, with stage-specific stopping rules detailed in Appendix D. For each objective, the component study in Section 4.4 uses three post-training runs from a single fixed pretrained checkpoint. These runs assess post-training repeatability conditional on that checkpoint, not variability across independently pretrained models.

4.2 IMPROVEMENT OVER ENCODER REUSE

Table 1 contains all 48 paired backbone–objective–dataset conditions, each aggregated over five random seeds. PEAR achieves a higher mean on both metrics in every condition. Gains vary substantially across objectives and datasets: for two-layer LightGCN, the average MRR@10 improvement across the four objectives is 6.54% on Beauty and 32.84% on Amazon-book. These comparisons

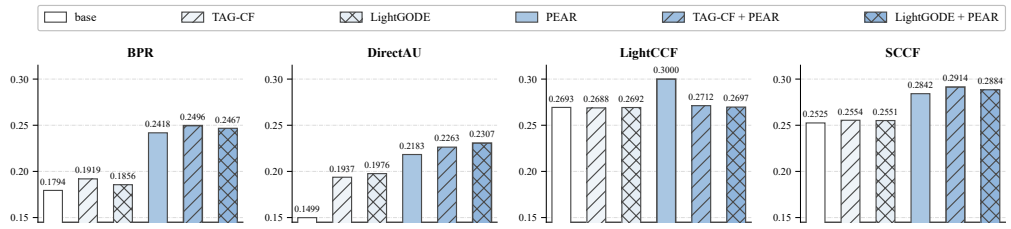


Figure 2: Test MRR@10 on Douban-book with MF. PEAR is applied directly or after structural postprocessing, for each pretraining objective.

change only the user readout, preserving the original embedding table and final item space. Appendix D.2 reports all remaining LightGCN depths.

4.3 Comparison with Structural Postprocessors

We compare PEAR with TAG-CF (Ju et al., 2024) and LightGODE (Zhang et al., 2024) as structural postprocessors of the same MF checkpoints on Douban-book. Figure 2 also evaluates applying PEAR after each postprocessor, freezing the resulting item representations during adaptation. PEAR alone outperforms both structural postprocessors under all four objectives, exceeding the stronger of the two by 10.5–26.0% in MRR@10. For example, under BPR, the base score of 0.1794 rises to 0.1919 with TAG-CF, 0.1856 with LightGODE, and 0.2418 with PEAR. Structural postprocessing and PEAR are complementary under BPR, DirectAU, and SCCF: both compositions improve on either component alone. Under DirectAU, TAG-CF+PEAR and LightGODE+PEAR reach 0.2263 and 0.2307, versus 0.2183 for PEAR alone. LightCCF shows a different pattern: PEAR alone reaches 0.3000, whereas the compositions reach only 0.2712 and 0.2697, slightly above their respective structural starting points. Thus, PEAR improves each postprocessed checkpoint, but the benefit of structural propagation before PEAR depends on the pretraining objective.

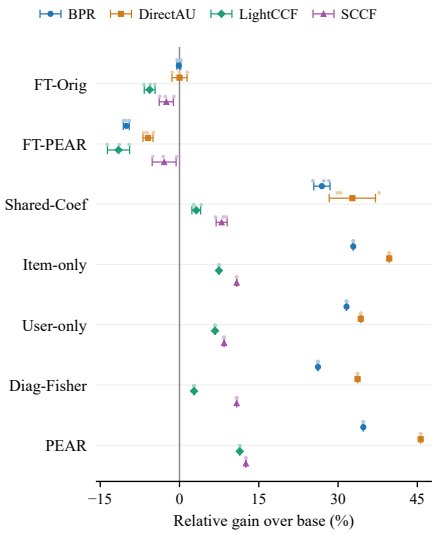


Figure 3: Relative test MRR@10 gains over the MF checkpoint on Douban-book. Markers and whiskers show the mean and sample standard deviation over three adaptation seeds; small dots show individual runs.

4.4 What Makes Adaptation Effective?

We compare six variants with PEAR on Douban-book using MF and all four objectives (Figure 3). For each objective, every variant starts from the same pretrained weights and is run with three seeds; the resulting variation therefore reflects only post-training optimization, rather than differences in pretraining or checkpoint selection.

Further embedding training does not reproduce the gains. **FT-Orig** continues optimizing both embedding tables through the original encoder with the original objective; **FT-PEAR** instead uses PEAR's positive-set likelihood. FT-Orig leaves mean MRR@10 near the baseline for BPR and DirectAU and reduces it for LightCCF and SCCF. FT-PEAR reduces the mean under all four objectives. By contrast, PEAR attains means of 0.2418, 0.2183, 0.3000, and 0.2842, respectively. Under these settings, neither additional training nor the objective switch alone explains PEAR's improvement.

Personalized mixing and both sources contribute. **Shared-Coef** learns one coefficient pair for all users while retaining each user's original basis directions. PEAR exceeds its mean by 5.2–9.1%, supporting personalization of the mixing coefficients beyond personalization of the bases. **Item-only** and **User-only** retain one source direction and fix the other coefficient to zero. Both improve on the

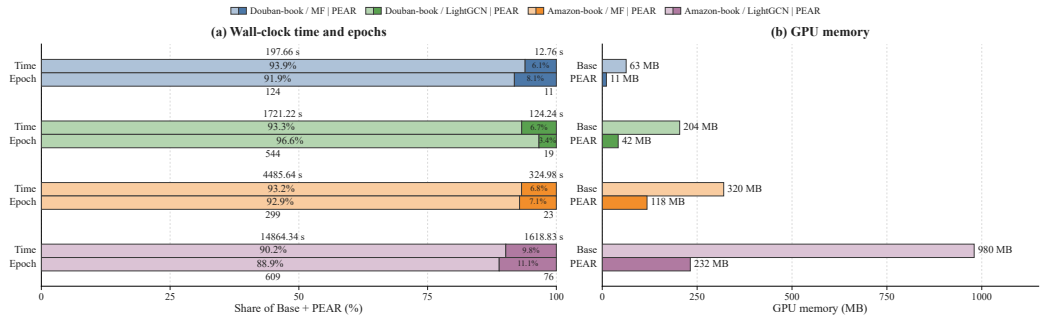


Figure 4: Pretraining (light) and PEAR fitting (dark), excluding basis construction. Rows show Douban-book, then Amazon-book, each with MF above LightGCN. Shares use the sum of both stages; memory is per stage.

frozen checkpoint, but PEAR exceeds the stronger single-source variant, Item-only, by 1.5–3.7% across the four objectives.

Cross-direction geometry helps coefficient fitting. Diag-Fisher replaces the two-dimensional Fisher preconditioner with its diagonal while keeping the original bases unchanged (Appendix C.5). PEAR improves on its mean by 2.6–8.2%, supporting retention of the interaction between the two basis directions during coefficient optimization. Together, these comparisons favor the full adaptation procedure, which has the highest mean under every objective.

Source-specific directions outperform generic user adaptation. Appendix D.3 compares PEAR with full-dimensional user-vector fitting and fixed gradient, natural-gradient, and random subspaces under a shared adaptation protocol. On the same MF checkpoints, PEAR exceeds the strongest generic control by 4.5–8.9% in MRR@10 across objectives, supporting the value of its source-specific basis construction beyond generic user-only fitting or a two-coefficient parameterization.

4.5 Cost and Deployment

Figure 4 compares pretraining and PEAR coefficient fitting for MF and LightGCN on Douban-book and Amazon-book. PEAR requires 11–76 fitting epochs, versus 124–609 pretraining epochs. On Douban-book, fitting takes 12.76 s for MF and 124.24 s for LightGCN, compared with 197.66 s and 1,721.22 s for pretraining. On Amazon-book, the corresponding times are 324.98 s and 1,618.83 s, versus 4,485.64 s and 14,864.34 s. Across these settings, fitting adds 6.5–10.9% of the pretraining time and uses 17.5–36.9% of its reported GPU memory. These fitting-stage measurements are separate from the one-time basis construction described in Appendix C.7.

After adaptation, we materialize $\boldsymbol{Z}^{\mathrm{PEAR}}$ and serve it with the fixed item matrix $\boldsymbol{V}$. The coefficient tables, basis caches, and optimizer state are unnecessary for serving. Deployment therefore stores the same number and dimension of user and item vectors and retains the original dot-product scoring cost. Appendix D.4 accounts for construction and the remaining pipeline stages, with the measured PEAR stage sum amounting to 7.4–11.0% of pretraining time.

5 Conclusion

We examined whether collaborative filtering models should reuse their training encoder at inference. Controlled comparisons show that a fixed, converged embedding table can yield better recommendations with a different inference encoder. Our analysis explains this result through the encoder's distinct training and inference roles and identifies the information sources and item geometry that guide user-specific readout changes. Building on these findings, we propose PEAR, a general post-training framework that adapts linear inference encoders using two fixed bases and at most two bounded coefficients per user. The adapted user vectors preserve the original dot-product scoring interface and incur no additional serving cost. Across datasets, backbone encoders, and training objectives, PEAR consistently improves recommendation performance over converged models, supporting inference-encoder adaptation as a distinct stage of collaborative filtering.

AI USE STATEMENT

We used OpenAI GPT-5.6-series models for translation and for drafting, revising, and polishing manuscript text. This assistance concerned the presentation of author-developed content. Generative AI was not used to develop theoretical models or conceptual frameworks, formulate hypotheses or mathematical claims, derive or write proofs, design or provide feedback on methods or experiments, implement methods, generate or preprocess research data, conduct qualitative analyses, or interpret results. The authors manually reviewed AI-assisted text for consistency with the intended technical meaning and for unsupported claims. We take responsibility for the final content of this work, including its text, claims, and research artifacts.

ETHICS STATEMENT

This work evaluates collaborative filtering through offline experiments on publicly available recommendation datasets. We use dataset-provided user–item interaction records and identifiers, and comply with applicable dataset and third-party code licenses and terms of use. No new participants were recruited, and no online user experiments were conducted. Our evaluation concerns offline recommendation performance and does not establish fairness or privacy guarantees. Deployment of PEAR should therefore be preceded by application-specific assessment of potential recommendation biases and privacy risks. The authors declare no conflicts of interest relevant to this work.

REPRODUCIBILITY STATEMENT

We provide an anonymized code package as supplementary material. The paper and accompanying materials document the theoretical assumptions and complete proofs; PEAR's encoder bases, bounded coefficient optimization, and the freezing of the embedding table and final item representations; data preprocessing and splits; ranking protocols and evaluation metrics; hyperparameter searches and stopping criteria; and software, hardware, and computational costs. Hyperparameters and stopping points are selected using validation data, with the test set reserved for final evaluation.

REFERENCES

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.99. URL https://doi.org/10.1109/TKDE.2005.99.

Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. LightGCL: Simple yet effective graph contrastive learning for recommendation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FKXVK9dyMM.

Jiayan Guo, Lun Du, Xu Chen, Xiaojun Ma, Qiang Fu, Shi Han, Dongmei Zhang, and Yan Zhang. On manipulating signals of user-item graph: A jacobi polynomial-based graph collaborative filtering. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 602–613, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599450. URL https://doi.org/10.1145/3580305.3599450.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pp. 507–517, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee. ISBN 9781450341431. doi: 10.1145/2872427.2883037. URL https://doi.org/10.1145/2872427.2883037.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020. doi: 10.1145/3397271.3401063.

Clark Mingxuan Ju, William Shiao, Zhichun Guo, Yanfang Ye, Yozen Liu, Neil Shah, and Tong Zhao. How does message passing improve collaborative filtering? In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Dain Kim, Jinhyeok Park, and Dongwoo Kim. Test-time embedding normalization for popularity bias mitigation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, pp. 4023–4027, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701245. doi: 10.1145/3583780.3615281. URL https://doi.org/10.1145/3583780.3615281.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263.

Geon Lee, Kyungho Kim, and Kijung Shin. Revisiting lightgcn: Unexpected inflexibility, inconsistency, and a remedy towards improved recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys '24, pp. 957–962, New York, NY, USA, 2024a. Association for Computing Machinery. ISBN 9798400705052. doi: 10.1145/3640457.3688176. URL https://doi.org/10.1145/3640457.3688176.

Geon Lee, Kyungho Kim, and Kijung Shin. Post-training embedding enhancement for long-tail recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, pp. 3857–3861, New York, NY, USA, 2024b. Association for Computing Machinery. ISBN 9798400704369. doi: 10.1145/3627673.3679978. URL https://doi.org/10.1145/3627673.3679978.

Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *Proceedings of the ACM Web Conference 2022*, 2022.

Feng Liu, Hao Cang, Huanhuan Yuan, Jiaqing Fan, Yongjing Hao, Fuzhen Zhuang, Guanfeng Liu, and Pengpeng Zhao. How do graph signals affect recommendation: Unveiling the mystery of low and high-frequency graph signals. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, 2026. doi: 10.1145/3770854.3780194.

Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, Li Shang, and Ning Gu. Personalized graph signal processing for collaborative filtering. In *Proceedings of the ACM Web Conference 2023*, WWW '23, pp. 1264–1272, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394161. doi: 10.1145/3543507.3583466. URL https://doi.org/10.1145/3543507.3583466.

Yating Liu, Xin Zheng, Yi Li, and Yanqing Guo. Test-time adaptation on recommender system with data-centric graph transformation. In James Kwok (ed.), *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pp. 4579–4587. International Joint Conferences on Artificial Intelligence Organization, 8 2025. doi: 10.24963/ijcai.2025/510. URL https://doi.org/10.24963/ijcai.2025/510. Main Track.

Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. Ultragcn: Ultra simplification of graph convolutional networks for recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pp. 1253–1262, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384469. doi: 10.1145/3459637.3482291. URL https://doi.org/10.1145/3459637.3482291.

Seongmin Park, Mincheol Yoon, Hye-young Kim, and Jongwuk Lee. Why is normalization necessary for linear recommenders? In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, pp. 2142–2151, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715921. doi: 10.1145/3726302.3730116. URL https://doi.org/10.1145/3726302.3730116.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, B. Khaled Letaief, and Dongsheng Li. How powerful is graph convolution for recommendation? In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pp. 1619–1629, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384469. doi: 10.1145/3459637.3482264. URL https://doi.org/10.1145/3459637.3482264.

Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. Towards representation alignment and uniformity in collaborative filtering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1816–1825, 2022. doi: 10.1145/3534678.3539253.

Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, pp. 726–735, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3462862. URL https://doi.org/10.1145/3404835.3462862.

Yihong Wu, Le Zhang, Fengran Mo, Tianyu Zhu, Weizhi Ma, and Jian-Yun Nie. Unifying graph convolution and contrastive learning in collaborative filtering. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3425–3436, 2024.

Xihong Yang, Yiqi Wang, Jin Chen, Wenqi Fan, Xiangyu Zhao, En Zhu, Xinwang Liu, and Defu Lian. Dual test-time training for out-of-distribution recommender system. *IEEE Trans. on Knowl. and Data Eng.*, 37(6):3312–3326, June 2025. ISSN 1041-4347. doi: 10.1109/TKDE.2025.3548160. URL https://doi.org/10.1109/TKDE.2025.3548160.

Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pp. 1294–1303, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531937. URL https://doi.org/10.1145/3477495.3531937.

Yajie Yu, Chenzhong Bin, Zhoubo Xu, Zhixin Zeng, Tongxin Xu, Cihan Xiao, and Jiafeng Wu. Beyond instance-level alignment and uniformity: Semantic factor learning for collaborative filtering. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, 2026. doi: 10.1145/3770855.3817704.

Weizhi Zhang, Liangwei Yang, Zihe Song, Henry Peng Zou, Ke Xu, Liancheng Fang, and Philip S. Yu. Do we really need graph convolution during training? light post-training Graph-ODE for efficient recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024. doi: 10.1145/3627673.3679773.

Yu Zhang, Yiwen Zhang, Yi Zhang, Lei Sang, and Yun Yang. Unveiling contrastive learning's capability of neighborhood aggregation for collaborative filtering. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, pp. 1985–1994, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715921. doi: 10.1145/3726302.3730111. URL https://doi.org/10.1145/3726302.3730111.

Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *CIKM*, pp. 4653–4664. ACM, 2021.

A Related Work

Training-time encoders and objectives in collaborative filtering. A large body of collaborative filtering (CF) research studies how interaction structure and learning objectives should shape user and item representations during training. Classical matrix factorization directly optimizes user and item embeddings under an identity readout (Koren et al., 2009), whereas graph-based CF incorporates the user–item interaction graph through neighborhood propagation. LightGCN shows that removing feature transformations and nonlinearities from graph convolution yields a strong and simple recommendation backbone (He et al., 2020). Subsequent work modifies or replaces this propagation through constraint-based simplification, graph filtering, or alternative normalization, as in UltraGCN (Mao et al., 2021), JGCF (Guo et al., 2023), and LightGCN++ (Lee et al., 2024a). A complementary line strengthens representation learning through alternative objectives, including SGL (Wu et al., 2021), SimGCL (Yu et al., 2022), NCL (Lin et al., 2022), DirectAU (Wang et al., 2022), SCCF (Wu et al., 2024), and LightCCF (Zhang et al., 2025). Other studies explain graph propagation through smoothing, spectral filtering, or graph-signal perspectives (Shen et al., 2021; Guo et al., 2023). Despite their different mechanisms, these works primarily ask how an encoder or objective should produce useful representations *while the embedding table is being learned*; the resulting encoder is typically retained as part of the final predictor. Our work asks a complementary question: once optimization has produced a fixed embedding table, is the encoder that was useful for learning it necessarily the best way to read it out? The distinction is consequential because a linear encoder has a dual role during training—it both constructs representations and routes gradients back to the embedding table—whereas only its readout role remains after training. PEAR therefore does not introduce another training-time graph backbone or pretraining objective. Instead, it treats encoder reuse as a design choice to be revisited after convergence and adapts the inference readout while leaving the learned embedding table intact.

Decoupling training-time and inference-time propagation. The most closely related studies begin to separate the transformations used during training from those used for inference. Wu et al. (2024) explicitly vary the numbers of graph-convolution layers used in the two stages and show that, for the same learned embeddings, a shallower inference encoder can outperform the propagation depth used during training. TAG-CF (Ju et al., 2024) goes further by training embeddings without graph message passing and applying aggregation only at test time; its analysis attributes much of message passing's benefit in CF to additional neighbor representations in the forward pass rather than to gradient updates propagated to neighboring embeddings. LightGODE (Zhang et al., 2024) similarly moves graph convolution out of embedding training and performs continuous graph propagation as a post-training operation. Together, these results provide important evidence that the computation useful for learning embeddings need not coincide with the transformation that is best for consuming them at inference. They nevertheless instantiate the alternative readout through structurally prescribed graph propagation: the propagation mechanism is chosen from the interaction graph and its configuration rather than fitted as a model-conditioned correction for each individual user. PEAR extends this decoupling in a different direction. Rather than asking only *which global graph operator or propagation depth* should be applied after training, we ask *how each user's row of the inference encoder should read the particular embedding table that the model has learned*. PEAR can start from either an identity or graph-based training encoder, freezes both the converged embedding table and the final item representations, and uses the user's training interactions, the model's fitted predictions, and the learned item geometry to construct two source-specific readout directions. Only at most two bounded coefficients per user are subsequently fitted. Thus, structural postprocessing and PEAR address different levels of the inference transformation: the former changes a graph-defined readout, whereas the latter personalizes how the resulting learned information is combined for each user. This distinction also makes the two ideas composable—a structural postprocessor can first define a new frozen representation space, after which PEAR can adapt its user readouts without changing the downstream scoring interface.

Personalized and post-training adaptation in recommendation. Several adjacent lines of work modify recommendation behavior without retraining a conventional backbone from scratch. PGSP (Liu et al., 2023) introduces personalized graph signals and mixed-frequency graph filtering to better capture individual preferences, illustrating the value of personalization in graph-based signal processing. At inference time, TTEN (Kim et al., 2023) changes item-embedding normaliza-

tion to control the effect of embedding magnitude and mitigate popularity bias. EDGE (Lee et al., 2024b) instead performs post-training enhancement of long-tail item embeddings, adjusting their magnitudes and directions using information from better learned head items. A separate test-time-adaptation literature targets distribution shift: DT3OR (Yang et al., 2025) adapts recommendation models at test time through self-distillation and contrastive objectives for shifted user/item features, while TTA-GREC (Liu et al., 2025) adapts graph-based recommenders by transforming user–item and knowledge graphs using pseudo-label and self-supervised signals. These methods collectively demonstrate that the predictor need not remain unchanged once base training is complete, but they adapt different objects for different purposes: graph signals or topology, item embeddings, scoring normalization, or model parameters in response to test-time distribution shift. PEAR occupies a distinct setting. It does not target popularity or long-tail correction, does not assume an out-of-distribution test environment, and requires neither test-time interactions nor pseudo-labels. Instead, PEAR conditions adaptation on information already present in a converged CF model and changes only the *user-side inference readout*, while keeping the embedding table and final item representations fixed. Moreover, personalization is deliberately restricted to a low-dimensional, source-aware encoder space: the two directions separately recombine item-origin preference information and user-origin collaborative information, and only their bounded user-specific contributions are learned. This makes PEAR different from unconstrained user-embedding fine-tuning as well as from graph- or data-level test-time adaptation, while preserving the original dot-product scoring interface and serving cost.

B Supporting Analysis and Diagnostic Protocol

B.1 Protocol for the Encoder-Reuse Heatmap

Datasets and implementation. We evaluate on Beauty (Yu et al., 2026) and Amazon-book (He & McAuley, 2016), with dataset details provided in Appendix D. Interactions are split into training, validation, and test sets with an $8{:}1{:}1$ ratio. For all models, only the training split is used during training, including the construction of graphs, propagators, and other data-dependent model components. All experiments are implemented with RecBole (Zhao et al., 2021), using an embedding dimension of 64.

Training encoders and checkpoint selection. We consider the encoder family $\mathcal{H} = \{\boldsymbol{H}_0, \ldots, \boldsymbol{H}_4\}$ defined in equation 3 and the three training objectives described in Section 2.2: BPR, DirectAU, and SCCF. Each combination of dataset, training objective, and training encoder is trained with three random seeds. During each training run, validation MRR@10 is evaluated using the training encoder $\boldsymbol{H}_{\mathrm{tr}}$. Training stops when validation MRR@10 does not improve for 50 consecutive epochs, and the checkpoint with the highest validation MRR@10 is selected. The embedding table from this checkpoint is then frozen and denoted by $\boldsymbol{E}_0$.

Inference-encoder substitution and evaluation. For each selected checkpoint, we evaluate every inference encoder $\boldsymbol{H}_{\mathrm{inf}} \in \mathcal{H}$ on the test set, producing representations $\boldsymbol{X}_{\mathrm{inf}} = \boldsymbol{H}_{\mathrm{inf}}\boldsymbol{E}_0$ without further optimization. Thus, comparisons within each training run use exactly the same learned embedding table and differ only in the inference encoder. The diagonal setting $\boldsymbol{H}_{\mathrm{inf}} = \boldsymbol{H}_{\mathrm{tr}}$ reuses the training encoder, whereas off-diagonal settings substitute another encoder. All evaluations use the full-ranking protocol. Validation ranking excludes training interactions; test ranking excludes training and validation interactions, with padding IDs excluded throughout. When constructing the normalized interaction operator $\boldsymbol{S}$ in equation 2, the implementation adds 10^{-7} to node degrees before inverse-square-root normalization. All encoders otherwise follow the uniform aggregation in equation 3.

Aggregation and heatmap reporting. For a fixed dataset and training objective, let $M(\boldsymbol{H}_{\mathrm{tr}}, \boldsymbol{H}_{\mathrm{inf}})$ denote test MRR@10 averaged over the three random seeds. The heatmap in Figure 1 reports the percentage change relative to encoder reuse:

$$100\left(\frac{M(\boldsymbol{H}_{\mathrm{tr}}, \boldsymbol{H}_{\mathrm{inf}})}{M(\boldsymbol{H}_{\mathrm{tr}}, \boldsymbol{H}_{\mathrm{tr}})} - 1\right). \tag{15}$$

Positive off-diagonal values indicate that substitution improves mean test MRR@10 over reuse. All comparisons and row-wise maxima are determined using unrounded metrics. Stars mark retrospective row-wise maxima on the test set; they summarize the observed results rather than define an encoder-selection procedure. All reported comparisons between encoder reuse and substitution are statistically significant ($p < 0.05$, two-tailed t-test).

B.2 Proof and Scope of Proposition 1

Proof. Using the Frobenius inner product, the differential of the data term is

$$d\ell = \langle \boldsymbol{G}, \boldsymbol{H}\, d\boldsymbol{E} \rangle_F = \langle \boldsymbol{H}^\top \boldsymbol{G}, d\boldsymbol{E} \rangle_F. \tag{16}$$

Adding the differential of Ω establishes equation 4. A plain gradient-descent step is $\boldsymbol{E}^+ = \boldsymbol{E} - \eta(\boldsymbol{H}^\top \boldsymbol{G} + \nabla_{\boldsymbol{E}}\Omega)$. Left-multiplication by the fixed $\boldsymbol{H}$ yields equation 5.

For a frozen $\boldsymbol{E}_0$, define the data objective as a function of the readout, $F(\boldsymbol{H}) = \ell(\boldsymbol{H}\boldsymbol{E}_0)$. Its derivative in a continuous direction $\Delta\boldsymbol{H}$ is

$$DF(\boldsymbol{H})[\Delta\boldsymbol{H}] = \langle \boldsymbol{G}\boldsymbol{E}_0^\top, \Delta\boldsymbol{H} \rangle_F. \tag{17}$$

This differs from stationarity in the embedding variables, $\boldsymbol{H}^\top \boldsymbol{G} + \nabla_{\boldsymbol{E}}\Omega = 0$. For unrestricted or locally bidirectional continuous perturbations, readout stationarity requires the corresponding directional derivatives to vanish. For a finite candidate family, readout optimality is instead defined by comparing candidate objective values.

A two-node example establishes that embedding optimality does not imply readout optimality in general. Let

$$\boldsymbol{H} = \tfrac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \boldsymbol{E}_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \boldsymbol{a}, \quad \ell(\boldsymbol{X}) = \tfrac{1}{2}\|\boldsymbol{X} - \boldsymbol{a}\|_F^2, \quad \Omega = 0. \tag{18}$$

The range of $\boldsymbol{H}$ is spanned by $(1,1)^\top$ and is orthogonal to $\boldsymbol{a}$. Hence $\boldsymbol{H}\boldsymbol{E}_0 = 0$ minimizes $\ell(\boldsymbol{H}\boldsymbol{E})$ over all $\boldsymbol{E}$, with objective value 1 and embedding gradient $\boldsymbol{H}^\top(-\boldsymbol{a}) = 0$. Replacing the readout by $\widetilde{\boldsymbol{H}} = \boldsymbol{I}_2$ while keeping $\boldsymbol{E}_0$ fixed gives $\widetilde{\boldsymbol{H}}\boldsymbol{E}_0 = \boldsymbol{a}$, with objective value 0. The two encoders are the uniform one-step encoder and the identity encoder of a two-node, single-edge graph. □

The counterexample concerns a differentiable representation objective and establishes the non-implication in the proposition; it does not assert a CF ranking improvement or a fixed-item improvement. Reuse can also be optimal. For example, if $\boldsymbol{H}$ is invertible, $\Omega = 0$, and $\boldsymbol{E}_0$ globally minimizes $\ell(\boldsymbol{H}\boldsymbol{E})$ over unrestricted $\boldsymbol{E}$, then $\boldsymbol{H}\boldsymbol{E}_0$ globally minimizes ℓ over all representation matrices. No replacement readout can improve that same objective. The proposition distinguishes optimization variables and roles without requiring an improvement to exist.

The chain-rule identity applies to each differentiable minibatch objective as well. Equation 5 specifically assumes plain gradient descent and does not describe Adam's coordinate-wise gradient transformation. In the reported diagnostic, validation early stopping selects a trained embedding table rather than certifying an exact training stationary point.

B.3 LightGCN++: Training-Time Normalization and the Frozen-Checkpoint Readout

LightGCN++ (Lee et al., 2024a) requires a distinction between its training-time computation and the post-training readout used by PEAR. Unlike the fixed propagation operator in LightGCN, its forward pass normalizes the current node embeddings before each propagation layer. Let $\boldsymbol{S}_{++} \in \mathbb{R}^{N\times N}$ denote the degree-controlled graph propagation operator, which is fixed by the training interaction graph and the LightGCN++ propagation hyperparameters. Writing $\boldsymbol{E}^{(0)} = \boldsymbol{E}$, the layer-wise computation can be expressed as

$$\begin{aligned} \boldsymbol{D}_{\mathrm{emb}}^{(\ell)} &= \mathrm{diag}\Big(\|\boldsymbol{E}_{1:}^{(\ell)}\|_2 + \varepsilon, \ldots, \|\boldsymbol{E}_{N:}^{(\ell)}\|_2 + \varepsilon \Big), \\ \boldsymbol{E}^{(\ell+1)} &= \boldsymbol{S}_{++}\big(\boldsymbol{D}_{\mathrm{emb}}^{(\ell)}\big)^{-1}\boldsymbol{E}^{(\ell)}, \qquad \ell = 0, \ldots, K-1, \end{aligned} \tag{19}$$

where $\varepsilon > 0$ denotes the numerical stabilizer used in the implementation.

Because $\boldsymbol{D}_{\mathrm{emb}}^{(\ell)}$ depends on the current layer embeddings, the mapping from the trainable embedding table to the final LightGCN++ representations is generally *state-dependent* during training. It therefore cannot, throughout embedding optimization, be identified with a single matrix $\boldsymbol{H}$ that is independent of $\boldsymbol{E}$. Accordingly, Proposition 1, whose premise is a fixed training encoder, should not be interpreted as a characterization of the full LightGCN++ training dynamics.

This distinction disappears for the post-training readout required by PEAR. Let $\boldsymbol{E}_0$ be a selected and frozen LightGCN++ checkpoint, and run the original LightGCN++ propagation on this checkpoint. Denote the resulting layer embeddings by $\boldsymbol{E}_\star^{(0)}, \dots, \boldsymbol{E}_\star^{(K)}$, with $\boldsymbol{E}_\star^{(0)} = \boldsymbol{E}_0$, and define the checkpoint-specific diagonal normalizers

$$\boldsymbol{D}_{\mathrm{emb},\star}^{(\ell)} = \mathrm{diag}\Big(\|\boldsymbol{E}_{\star,1:}^{(\ell)}\|_2 + \varepsilon, \dots, \|\boldsymbol{E}_{\star,N:}^{(\ell)}\|_2 + \varepsilon \Big). \tag{20}$$

Once the checkpoint is fixed, every $\boldsymbol{D}_{\mathrm{emb},\star}^{(\ell)}$ is fixed as well. Define

$$\boldsymbol{T}_\star^{(0)} = \boldsymbol{I}_N, \qquad \boldsymbol{T}_\star^{(\ell+1)} = \boldsymbol{S}_{++}\big(\boldsymbol{D}_{\mathrm{emb},\star}^{(\ell)}\big)^{-1}\boldsymbol{T}_\star^{(\ell)}. \tag{21}$$

An induction over the propagation depth then gives the exact identity

$$\boldsymbol{E}_\star^{(\ell)} = \boldsymbol{T}_\star^{(\ell)}\boldsymbol{E}_0, \qquad \ell = 0, \dots, K. \tag{22}$$

Let γ_{pp} denote the LightGCN++ layer-pooling coefficient, which combines the initial embeddings with the mean of the K propagated layers. Its frozen-checkpoint output can therefore be written as

$$\begin{aligned} \boldsymbol{X}_\star &= \gamma_{\mathrm{pp}}\boldsymbol{E}_0 + \frac{1-\gamma_{\mathrm{pp}}}{K}\sum_{\ell=1}^{K}\boldsymbol{E}_\star^{(\ell)} \\ &= \underbrace{\left(\gamma_{\mathrm{pp}}\boldsymbol{I}_N + \frac{1-\gamma_{\mathrm{pp}}}{K}\sum_{\ell=1}^{K}\boldsymbol{T}_\star^{(\ell)}\right)}_{\boldsymbol{H}_{++}^\star}\boldsymbol{E}_0. \end{aligned} \tag{23}$$

Thus $\boldsymbol{H}_{++}^\star$ is a fixed linear readout *conditional on the frozen checkpoint*. Equation equation 23 is an exact representation of that checkpoint's forward pass, rather than a first-order linearization or an approximation. It does not claim that the same matrix would reproduce LightGCN++ after changing $\boldsymbol{E}_0$ and recomputing its layer-wise normalization factors.

For PEAR, this checkpoint-conditioned distinction is sufficient. For a LightGCN++ backbone, we instantiate the original user encoder and representations as

$$\boldsymbol{p}_u^{(0)\top} = (\boldsymbol{e}^{(u)})^\top\boldsymbol{H}_{++}^\star, \qquad [\boldsymbol{Z}^{(0)}; \boldsymbol{V}] = \boldsymbol{H}_{++}^\star\boldsymbol{E}_0. \tag{24}$$

PEAR subsequently freezes $\boldsymbol{E}_0$ and $\boldsymbol{V}$ and adapts only the user-side inference readout as in equation 11; the LightGCN++ normalization factors are not recomputed during this stage. Consequently, applying PEAR to LightGCN++ does not require treating LightGCN++ pretraining as optimization through a fixed linear encoder.

More generally, the LightGCN++ experiment separates two requirements that coincide for backbones such as MF and LightGCN but need not coincide in general. The fixed-encoder analysis in Proposition 1 concerns the role of an encoder *during embedding training*. PEAR itself operates *after training* and requires a fixed linear readout of the selected checkpoint. LightGCN++ satisfies the latter condition exactly after its checkpoint-dependent normalization factors are frozen. Its inclusion therefore illustrates that post-training encoder adaptation can remain well defined even when the backbone's training-time forward operator is state-dependent.

C PEAR: Derivations and Implementation Details

This appendix specifies the construction and optimization used in Section 3. All sums and dictionary sizes refer to real users and items; padding IDs are excluded from prediction distributions, neighborhoods, and losses.

C.1 Prediction Geometry and the Positive-Set Objective

The likelihood and conditional positive distribution are defined for users with $\mathcal{P}_u \neq \emptyset$. Users without training positives receive a zero construction signal and retain their original encoder. To account for score-scale differences among frozen checkpoints and users, we compute the baseline root-mean-square score once and keep it fixed:

$$\sigma_u = \sqrt{\frac{1}{m}\sum_{i\in\mathcal{I}}(\boldsymbol{z}_u^{(0)\top}\boldsymbol{v}_i)^2}, \qquad \pi_{ui}(\boldsymbol{z}) = \frac{\exp(\boldsymbol{z}^\top\boldsymbol{v}_i/\sigma_u)}{\sum_{j\in\mathcal{I}}\exp(\boldsymbol{z}^\top\boldsymbol{v}_j/\sigma_u)}. \tag{25}$$

The adaptation objective is the positive-set negative log-likelihood

$$\ell_u(\boldsymbol{z};\boldsymbol{V}) = -\log\sum_{i\in\mathcal{P}_u}\pi_{ui}(\boldsymbol{z}) = \log\sum_{i\in\mathcal{I}}e^{\boldsymbol{z}^\top\boldsymbol{v}_i/\sigma_u} - \log\sum_{i\in\mathcal{P}_u}e^{\boldsymbol{z}^\top\boldsymbol{v}_i/\sigma_u}. \tag{26}$$

The normalizer includes the full item catalog. Within the positive set, contributions are weighted by current predicted probabilities rather than uniformly.

For basis construction, evaluate $\pi_{ui} = \pi_{ui}(\boldsymbol{z}_u^{(0)})$ and condition on the training-positive set: $\pi_{ui}^+ = \mathbf{1}\{i\in\mathcal{P}_u\}\pi_{ui}/\sum_{j\in\mathcal{P}_u}\pi_{uj}$. The construction signal, predictive mean, and covariance are

$$\boldsymbol{g}_u = \sum_i(\pi_{ui}^+ - \pi_{ui})\boldsymbol{v}_i, \qquad \boldsymbol{\mu}_u = \sum_i\pi_{ui}\boldsymbol{v}_i, \qquad \boldsymbol{C}_u = \sum_i\pi_{ui}(\boldsymbol{v}_i-\boldsymbol{\mu}_u)(\boldsymbol{v}_i-\boldsymbol{\mu}_u)^\top. \tag{27}$$

Differentiating equation 26 with the baseline scale fixed gives

$$\nabla_{\boldsymbol{z}}\ell_u(\boldsymbol{z}_u^{(0)};\boldsymbol{V}) = \frac{1}{\sigma_u}\sum_i(\pi_{ui}-\pi_{ui}^+)\boldsymbol{v}_i = -\frac{\boldsymbol{g}_u}{\sigma_u}. \tag{28}$$

Thus $\boldsymbol{d}_u = \boldsymbol{g}_u/\sigma_u$ instantiates the first-order signal in equation 10.

The covariance $\boldsymbol{C}_u$ is the prediction-weighted counterpart of $\boldsymbol{Q}_{\boldsymbol{V}}$ in equation 8; a uniform predictive distribution recovers $\boldsymbol{Q}_{\boldsymbol{V}}$. For a perturbation $\Delta\boldsymbol{z}$, the predictive variance of the induced logit changes is $\Delta\boldsymbol{z}^\top\boldsymbol{C}_u\Delta\boldsymbol{z}/\sigma_u^2$. The categorical score function is $\nabla_{\boldsymbol{z}}\log\pi_{ui}(\boldsymbol{z}_u^{(0)}) = (\boldsymbol{v}_i-\boldsymbol{\mu}_u)/\sigma_u$. Taking its outer product and expectation under π_u yields the Fisher matrix $\boldsymbol{C}_u/\sigma_u^2$. Consequently, a local distributional change satisfies

$$D_{\mathrm{KL}}\Big(\pi_u(\boldsymbol{z}_u^{(0)})\,\|\,\pi_u(\boldsymbol{z}_u^{(0)}+\Delta\boldsymbol{z})\Big) = \frac{1}{2\sigma_u^2}\Delta\boldsymbol{z}^\top\boldsymbol{C}_u\Delta\boldsymbol{z} + O(\|\Delta\boldsymbol{z}\|_2^3). \tag{29}$$

In contrast, if $\boldsymbol{C}_u^+$ denotes the item covariance under π_u^+, then

$$\nabla_{\boldsymbol{z}}^2\ell_u(\boldsymbol{z}_u^{(0)};\boldsymbol{V}) = \frac{\boldsymbol{C}_u-\boldsymbol{C}_u^+}{\sigma_u^2}. \tag{30}$$

This Hessian can be indefinite. The positive-definite $\boldsymbol{K}_u$ in equation 12 therefore defines a damped predictive metric, rather than the Hessian of the fitting objective. The objective defining $\boldsymbol{t}_u$ can equivalently be written as $\sigma_u\boldsymbol{d}_u^\top\boldsymbol{t} - \frac{1}{2}\boldsymbol{t}^\top\boldsymbol{C}_u\boldsymbol{t} - \frac{\gamma_u}{2}\|\boldsymbol{t}\|_2^2$. The covariance term is σ_u^2 times the local KL quadratic, making the surrogate's scaling explicit. Damping penalizes changes in directions weakly constrained by the predictive covariance.

The construction distributions, covariance, and target are all evaluated at $\boldsymbol{z}_u^{(0)}$ and remain fixed during coefficient fitting. The likelihood in equation 14, however, is evaluated at the current adapted vector on every update.

C.2 Dictionary Construction, Scaling, and Reduced Item Solve

Write $\boldsymbol{u}_v^0$ and $\boldsymbol{i}_i^0$ for the transposed rows of $\boldsymbol{U}_0$ and $\boldsymbol{I}_0$. For nonzero atoms, normalization uses $\widehat{\boldsymbol{x}} = \boldsymbol{x}/\|\boldsymbol{x}\|_2$; zero atoms remain zero. The item dictionary contains every original item embedding: $\boldsymbol{B}_u^I = \boldsymbol{B}^I = m^{-1/2}[\widehat{\boldsymbol{i}}_1^0,\ldots,\widehat{\boldsymbol{i}}_m^0]^\top$.

The user dictionary is constructed from training-interaction overlap. For another user v, define

$$o_{uv} = \sum_{i\in\mathcal{P}_u\cap\mathcal{P}_v}\frac{1}{\deg_I(i)}, \qquad \deg_I(i) = \sum_w R_{wi}. \tag{31}$$

We retain up to k_{nbr} users with the largest positive scores, excluding u, and denote this set by $\mathcal{N}_u$. The normalized overlap weights are $\omega_{uv} = o_{uv} / \sum_{w \in \mathcal{N}_u} o_{uw}$. The rows of $\boldsymbol{B}_u^U$ are $\sqrt{\omega_{uv}}\, \widehat{\boldsymbol{u}}_v^{0\top}$ for $v \in \mathcal{N}_u$. Inverse item degree reduces the contribution of widely shared interactions to neighbor selection, while k_{nbr} limits the collaborative dictionary size. Users with no positive-overlap neighbors receive a zero user-source basis element. Both dictionaries admit $\boldsymbol{B}_u^a = \boldsymbol{W}_u^a \boldsymbol{E}_0$, with the source-specific selection and rescaling matrices expanded in Appendix C.3.

Let $\boldsymbol{A}_u^a = \boldsymbol{B}_u^a \boldsymbol{K}_u \boldsymbol{B}_u^{a\top}$ and $k_u = |\mathcal{N}_u|$. The construction damping and ridge coefficients scale with the corresponding curvature traces:

$$\gamma_u = \rho_C \frac{\mathrm{tr}(\boldsymbol{C}_u)}{d}, \qquad \lambda_u^I = \rho \frac{\mathrm{tr}(\boldsymbol{A}_u^I)}{d}, \qquad \lambda_u^U = \rho \frac{\mathrm{tr}(\boldsymbol{A}_u^U)}{\min(d, k_u)}. \tag{32}$$

The user-source expression applies when $k_u > 0$; an empty dictionary requires no user-source solve. The damping and ridge multipliers ρ_C and ρ are given in Table 2. The user solve is padded to k_{nbr} rows with zero atoms and weights. Its nonzero coordinates agree with the unpadded solve.

For the item dictionary, a dense $m \times m$ solve is unnecessary. Set $\boldsymbol{G} = \boldsymbol{B}^{I\top} \boldsymbol{B}^I$ and $\boldsymbol{L} = \boldsymbol{G}^{1/2}$. The implementation checks that $\boldsymbol{G}$ is positive definite, with $\lambda_{\min}(\boldsymbol{G}) > 10^{-10} \lambda_{\max}(\boldsymbol{G})$, before forming its square root and inverse square root. Under this condition the raw item-source representation component from equation 13 equals

$$\boldsymbol{y}_u = (\boldsymbol{L}\boldsymbol{K}_u\boldsymbol{L} + \lambda_u^I \boldsymbol{I}_d)^{-1} \boldsymbol{L}\boldsymbol{g}_u, \qquad \widetilde{\boldsymbol{b}}_u^I = \boldsymbol{L}\boldsymbol{y}_u. \tag{33}$$

To see this, fix a representation $\boldsymbol{b} = \boldsymbol{B}^{I\top}\boldsymbol{\eta}$. The minimum-norm coefficient vector is $\boldsymbol{\eta} = \boldsymbol{B}^I \boldsymbol{G}^{-1}\boldsymbol{b}$, whose squared norm is $\boldsymbol{b}^\top \boldsymbol{G}^{-1}\boldsymbol{b}$. Substituting $\boldsymbol{b} = \boldsymbol{L}\boldsymbol{y}$ into the ridge objective gives the normal equation in equation 33. Also $\mathrm{tr}(\boldsymbol{L}\boldsymbol{K}_u\boldsymbol{L}) = \mathrm{tr}(\boldsymbol{A}_u^I)$, so the ridge scaling is unchanged by this reduction. The implementation uses this full-rank reduction; it does not substitute a pseudoinverse when the rank check fails.

C.3 Reconstructing the Encoder Basis Elements

The two basis elements have disjoint supports in the original encoder space:

$$\boldsymbol{q}_u^I = \begin{bmatrix} \boldsymbol{0}_n \\ \boldsymbol{a}_u^I \end{bmatrix}, \qquad \boldsymbol{q}_u^U = \begin{bmatrix} \boldsymbol{a}_u^U \\ \boldsymbol{0}_m \end{bmatrix}, \qquad \boldsymbol{q}_u^I, \boldsymbol{q}_u^U \in \mathbb{R}^N, \tag{34}$$

where $\boldsymbol{a}_u^I \in \mathbb{R}^m$ and $\boldsymbol{a}_u^U \in \mathbb{R}^n$ weight the original item and user embeddings, respectively. Let $h_i = \|\boldsymbol{i}_i^0\|_2$ and $h_v = \|\boldsymbol{u}_v^0\|_2$ for nonzero atoms. The matrices $\boldsymbol{W}_u^a$ select item rows with weights $1/(\sqrt{m}\, h_i)$ or neighbor rows with weights $\sqrt{\omega_{uv}}/h_v$; zero atoms use zero weights.

Given the ridge minimizers in equation 13, define the raw components $\widetilde{\boldsymbol{b}}_u^a = \boldsymbol{B}_u^{a\top} \boldsymbol{\eta}_u^a$. For a nonzero raw component, we normalize its induced readout scale by

$$r_u^a = \frac{\|\boldsymbol{z}_u^{(0)}\|_2}{\|\widetilde{\boldsymbol{b}}_u^a\|_2}, \qquad \boldsymbol{q}_u^a = r_u^a \boldsymbol{W}_u^{a\top} \boldsymbol{\eta}_u^a, \qquad \boldsymbol{b}_u^a = \boldsymbol{E}_0^\top \boldsymbol{q}_u^a = r_u^a \widetilde{\boldsymbol{b}}_u^a. \tag{35}$$

A zero raw component and its encoder basis element remain zero without applying the ratio. For nonzero components, norm matching leaves their relative contribution to be determined by the fitted coefficients. The resulting source weights are

$$[\boldsymbol{a}_u^I]_i = \frac{r_u^I [\boldsymbol{\eta}_u^I]_i}{\sqrt{m}\, h_i}, \qquad [\boldsymbol{a}_u^U]_v = \begin{cases} r_u^U \sqrt{\omega_{uv}} [\boldsymbol{\eta}_u^U]_v / h_v, & v \in \mathcal{N}_u, \\ 0, & v \notin \mathcal{N}_u, \end{cases} \tag{36}$$

where user-dictionary coordinates are indexed by their neighbor IDs and zero atoms receive zero coefficients. Expanding the dictionary products gives $\boldsymbol{I}_0^\top \boldsymbol{a}_u^I = \boldsymbol{b}_u^I$ and $\boldsymbol{U}_0^\top \boldsymbol{a}_u^U = \boldsymbol{b}_u^U$. Substituting into equation 11 therefore yields

$$\boldsymbol{z}_u(\boldsymbol{c}_u) = \boldsymbol{E}_0^\top \boldsymbol{p}_u(\boldsymbol{c}_u) = \boldsymbol{z}_u^{(0)} + c_u^I \boldsymbol{b}_u^I + c_u^U \boldsymbol{b}_u^U, \qquad \boldsymbol{s}_u = \boldsymbol{V}\boldsymbol{z}_u(\boldsymbol{c}_u). \tag{37}$$

The encoder-space and representation-space computations produce exactly the same scores. Although the encoder basis elements have disjoint source supports, their representation components

Table 2: PEAR hyperparameters and coefficient-fitting settings.

Setting	Symbol	Value
Collaborative neighbor budget	k_{nbr}	32
Construction damping multiplier	ρ_C	0.1
Dictionary ridge multiplier	ρ	0.1
Item-source coefficient bound	κ_I	2
User-source coefficient bound	κ_U	2
Coefficient regularization	β	0.01
Coefficient Fisher damping multiplier	ρ_F	0.1
Preconditioned learning rate	α_{opt}	0.1
Metric step bound	τ_{step}	0.1
Coordinate step bound	δ_{step}	0.3
Users per fitting minibatch	B_{fit}	256
Maximum fitting epochs	$T_{\max}$	100
Early-stopping patience	P_{stop}	8
Minimum validation improvement	ϵ_{val}	10^{-8}

need not be distinct or linearly independent. Coefficients can be signed, and adapted encoder rows need not be stochastic aggregation weights.

Dense encoder basis elements need not be stored. For the item-source basis element, define $\boldsymbol{h}_u = r_u^I \boldsymbol{L}^{-1}\boldsymbol{y}_u$ using equation 33. For nonzero item atoms, $[\boldsymbol{a}_u^I]_i = \widehat{\boldsymbol{i}}_i^{0\top}\boldsymbol{h}_u/(mh_i)$, matching the compact item-dual reconstruction in the implementation. The user-source basis element can be reconstructed from the neighbor IDs and their rescaled coefficients. All these are fixed quantities, rather than additional parameters optimized in equation 14.

C.4 First-Order Utility of the Encoder Bases

For either nonempty dictionary, write $\boldsymbol{B} = \boldsymbol{B}_u^a$, $\lambda = \lambda_u^a > 0$, and $\boldsymbol{\eta} = \boldsymbol{\eta}_u^a$. The ridge normal equation is $(\boldsymbol{B}\boldsymbol{K}_u\boldsymbol{B}^\top + \lambda \boldsymbol{I})\boldsymbol{\eta} = \boldsymbol{B}\boldsymbol{g}_u$. Multiplying by $\boldsymbol{\eta}^\top$ and using $\widetilde{\boldsymbol{b}} = \boldsymbol{B}^\top\boldsymbol{\eta}$ yields

$$\boldsymbol{g}_u^\top \widetilde{\boldsymbol{b}} = \widetilde{\boldsymbol{b}}^\top \boldsymbol{K}_u \widetilde{\boldsymbol{b}} + \lambda \|\boldsymbol{\eta}\|_2^2 \geq 0. \tag{38}$$

If $\boldsymbol{B}\boldsymbol{g}_u \neq 0$, the solution is nonzero and the inequality is strict. When also $\|\boldsymbol{z}_u^{(0)}\|_2 > 0$, norm matching multiplies the raw component by a positive scalar, so $\nabla \ell_u(\boldsymbol{z}_u^{(0)}; \boldsymbol{V})^\top \boldsymbol{b}_u^a < 0$. Since $\boldsymbol{b}_u^a = \boldsymbol{E}_0^\top \boldsymbol{q}_u^a$, this is also the derivative of $\ell_u(\boldsymbol{E}_0^\top(\boldsymbol{p}_u^{(0)} + c_u^a \boldsymbol{q}_u^a); \boldsymbol{V})$ with respect to c_u^a at zero. Smoothness then ensures a decrease in this user's training negative log-likelihood for a sufficiently small positive coefficient on that encoder basis element. The coefficient penalty has zero derivative at $\boldsymbol{c}_u = 0$, and the bounded parameterization permits such small steps. This establishes local descent at initialization under the stated nondegeneracy conditions. It does not establish descent for arbitrary fitted coefficients or improvement in held-out ranking metrics.

C.5 Hyperparameters and Coefficient Optimization

Table 2 collects the numerical settings used for basis construction and coefficient fitting. We fit $\mathcal{U}_{\mathrm{fit}} = \{u \in \mathcal{U} : |\mathcal{P}_u| > 1\}$ and retain zero coefficients for all other users. The learned variables are $\boldsymbol{\theta}_u \in \mathbb{R}^2$, mapped to bounded coefficients by

$$c_u^a = \kappa_a \tanh(\theta_u^a/\kappa_a), \qquad a \in \{I, U\}, \quad \kappa_a > 0. \tag{39}$$

All coefficients start at zero, so the initial model reproduces the pretrained encoder. Only these coefficients are fitted; the basis elements, frozen representations, and baseline scales do not change.

Let $\boldsymbol{T}_u = [\boldsymbol{b}_u^I, \boldsymbol{b}_u^U] \in \mathbb{R}^{d\times 2}$. At initialization, the Fisher matrix in coefficient coordinates and its damped version are

$$\boldsymbol{F}_u = \frac{1}{\sigma_u^2}\boldsymbol{T}_u^\top \boldsymbol{C}_u \boldsymbol{T}_u, \qquad \boldsymbol{M}_u = \boldsymbol{F}_u + \rho_F \frac{\mathrm{tr}(\boldsymbol{F}_u)}{2}\boldsymbol{I}_2. \tag{40}$$

The implementation symmetrizes $\boldsymbol{F}_u$ numerically and keeps $\boldsymbol{M}_u$ fixed. For the individual regularized loss $L_u = \ell_u(\boldsymbol{z}_u(\boldsymbol{c}_u); \boldsymbol{V}) + \beta\|\boldsymbol{c}_u\|_2^2$, an update uses

$$\begin{aligned} \boldsymbol{a}_u &= \alpha_{\mathrm{opt}}\, \boldsymbol{M}_u^{-1} \nabla_{\boldsymbol{\theta}_u} L_u, \\ \overline{\boldsymbol{a}}_u &= \boldsymbol{a}_u \min\left\{1, \frac{\tau_{\mathrm{step}}}{\sqrt{\boldsymbol{a}_u^\top \boldsymbol{M}_u \boldsymbol{a}_u}}\right\}, \\ \boldsymbol{\theta}_u &\leftarrow \boldsymbol{\theta}_u - \mathrm{clip}(\overline{\boldsymbol{a}}_u, -\delta_{\mathrm{step}}, \delta_{\mathrm{step}}), \end{aligned} \tag{41}$$

where clipping is coordinate-wise. The gradient includes the derivative of the tanh parameterization. Because a fixed coefficient-space metric is applied to gradients in $\boldsymbol{\theta}$ coordinates, this is a Fisher-preconditioned gradient procedure, not an exact natural-gradient update under the nonlinear parameterization. The gradient of the batch mean is multiplied by the actual batch size to recover each user's gradient in equation 41. User order is shuffled each epoch using the fitting seed.

C.6 Model Selection and Evaluation Protocol

We run at most $T_{\max}$ epochs, retain the coefficients with the best validation MRR@10, and stop after P_{stop} consecutive epochs without an increase exceeding ϵ_{val}. The initial zero-coefficient state is included in this comparison.

No negative sampling is used in equation 26. Validation ranking masks training positives; test ranking masks training and validation positives. These evaluation masks do not change the catalog-wide normalizer in the training objective. Basis construction and coefficient gradients use training interactions only. Validation interactions are used for early stopping and coefficient selection; test data enter the evaluation pipeline after model selection.

C.7 Parameter Count and Computational Costs

The trainable capacity is $2n$ scalars, of which only $2|\mathcal{U}_{\mathrm{fit}}|$ can change; the implementation additionally allocates an unused padding row. The cache of fixed representation components stores $2nd$ floating-point values, separately from the frozen representations and compact encoder reconstruction data. Materializing all user–item scores or dense encoder rows is unnecessary. The implementation batches users for catalog-wide probabilities and Fisher covariances, but stores all damped $d \times d$ user metrics during construction, requiring $O(nd^2)$ additional memory. It solves the target system $\boldsymbol{K}_u \boldsymbol{t}_u = \boldsymbol{g}_u$ before the dictionary systems, reuses the item Gram factorization, and then solves one $d \times d$ item system and one padded $k_{\mathrm{nbr}} \times k_{\mathrm{nbr}}$ user system per user. These construction operations, neighbor retrieval, and temporary catalog-sized score batches must be counted when measuring total adaptation time and peak memory. The coefficient fitting timer alone excludes basis construction. For deployment, $\boldsymbol{Z}^{\mathrm{PEAR}}$ can replace $\boldsymbol{Z}^{(0)}$ in the original scoring interface while preserving $\boldsymbol{V}$; the basis-related and optimizer caches are no longer needed for scoring.

C.8 Complete Adaptation Procedure

Algorithm 1 summarizes the construction, fitting, and selection stages. The encoder basis elements are represented implicitly by their reconstruction data and induced representation components; dense N-dimensional vectors need not be materialized. All construction quantities remain fixed once coefficient fitting begins. For users with no training positives or no collaborative neighbors, the algorithm follows the zero-signal and zero-component conventions in section C.1 and section C.2.

D Experimental Protocol and Additional Results

D.1 Datasets and Experimental Settings

Datasets and splitting. We evaluate on four public implicit-feedback benchmarks: Douban-book, Amazon-kindle, Amazon-book, and Beauty. To ensure data quality, we apply 10-core filtering to Douban-book, Amazon-kindle, and Amazon-book following He & McAuley (2016). For Beauty, we adopt the 5-core setting of Yu et al. (2026) to accommodate its sparsity. We use the built-in data preprocessing and splitting pipeline of RecBole (Zhao et al., 2021). Each user's interaction

Algorithm 1 PEAR: post-training user-encoder adaptation

Require: Frozen $\boldsymbol{E}_0$, $\boldsymbol{Z}^{(0)}$, $\boldsymbol{V}$; training interactions $\boldsymbol{R}$; validation interactions; settings in Table 2
Ensure: Adapted user representations $\boldsymbol{Z}^{\mathrm{PEAR}}$ and unchanged $\boldsymbol{V}$

Compute baseline scales σ_u and distributions π_u using equation 25
Compute $\boldsymbol{g}_u$, $\boldsymbol{C}_u$, $\boldsymbol{K}_u$, and $\boldsymbol{t}_u$ using equations equation 27 and equation 12
Form $\boldsymbol{B}^I$ and factor its Gram matrix as in Appendix C.2
for each user u **do**
    Select $\mathcal{N}_u$ from training interactions and form $\boldsymbol{B}_u^U$
    Solve the item and user ridge problems in equation 13
    Normalize and cache $\boldsymbol{b}_u^I, \boldsymbol{b}_u^U$ and the encoder reconstruction data
    Form the fixed coefficient metric $\boldsymbol{M}_u$ using equation 40
end for
Set $\boldsymbol{\theta} \leftarrow \mathbf{0}$, $\boldsymbol{\theta}^\star \leftarrow \mathbf{0}$, and $w \leftarrow 0$
Set $q^\star$ to the validation MRR@10 of the pretrained readout
for $t = 1, \ldots, T_{\max}$ **do**
    for each shuffled minibatch $\mathcal{B} \subseteq \mathcal{U}_{\mathrm{fit}}$ **do**
        Form $\boldsymbol{c}_u$ and $\boldsymbol{z}_u$ using equations equation 39 and equation 37
        Evaluate the full-catalog positive-set objective and its coefficient penalty
        Update only $\{\boldsymbol{\theta}_u : u \in \mathcal{B}\}$ using equation 41
    end for
    Evaluate the current validation MRR@10, denoted by q
    if $q > q^\star + \epsilon_{\mathrm{val}}$ **then**
        $q^\star \leftarrow q$, $\boldsymbol{\theta}^\star \leftarrow \boldsymbol{\theta}$, $w \leftarrow 0$
    else
        $w \leftarrow w + 1$
    end if
    if $w \geq P_{\mathrm{stop}}$ **then**
        break
    end if
end for
Restore $\boldsymbol{\theta}^\star$ and materialize $\boldsymbol{Z}^{\mathrm{PEAR}}$ using equation 37
return $\boldsymbol{Z}^{\mathrm{PEAR}}, \boldsymbol{V}$

history is divided into training, validation, and test sets with an 8:1:1 ratio. All methods share identical data splits and user/item token mappings within each dataset. Only the training split is used to construct interaction graphs, propagators, sampling distributions, and all other data-dependent model components. Table 3 summarizes the dataset statistics.

Table 3: Statistics of the experimental datasets. The number of interactions is summed over the training, validation, and test splits. Sparsity denotes the percentage of unobserved user–item pairs.

Dataset	#Users	#Items	#Interactions	Sparsity
Douban-book	9,308	15,471	723,484	99.4976%
Amazon-kindle	14,356	15,885	367,478	99.8389%
Amazon-book	52,643	91,599	2,984,108	99.9381%
Beauty	22,363	12,101	198,502	99.9266%

Evaluation. We adopt the full-ranking evaluation protocol implemented in RecBole. For each test user, candidate items from the full item catalog are ranked using the model's predicted scores, following RecBole's built-in filtering and metric computation procedures. We report MRR@10 and NDCG@10, averaged over users with at least one target interaction. Both test metrics are computed at the checkpoint selected using validation MRR@10, as described below.

Training and model selection. The embedding dimension is 64 and the pretraining batch size is 8,192. To ensure a fair comparison, method-specific hyperparameters are chosen with reference to the corresponding original papers, while all methods share the same data preprocessing, splitting,

Table 4: Full results for LightGCN with 1, 3, and 4 layers. Results are mean ± sample standard deviation over five random seeds ($n-1$ denominator), with both statistics rounded to four decimal places. Pairwise bolding and average improvements follow the protocol in Table 1.

		Douban-book		Amazon-kindle		Amazon-book		Beauty	
Backbone	Objective	MRR@10	NDCG@10	MRR@10	NDCG@10	MRR@10	NDCG@10	MRR@10	NDCG@10
LightGCN (1 layer)	BPR	$0.1973_{\pm 0.0031}$	$0.1021_{\pm 0.0016}$	$0.1050_{\pm 0.0012}$	$0.0812_{\pm 0.0011}$	$0.0814_{\pm 0.0011}$	$0.0456_{\pm 0.0007}$	$0.0485_{\pm 0.0009}$	$0.0494_{\pm 0.0009}$
	+PEAR	$\mathbf{0.2536}_{\pm 0.0046}$	$\mathbf{0.1328}_{\pm 0.0025}$	$\mathbf{0.1412}_{\pm 0.0014}$	$\mathbf{0.1056}_{\pm 0.0011}$	$\mathbf{0.1131}_{\pm 0.0019}$	$\mathbf{0.0634}_{\pm 0.0010}$	$\mathbf{0.0554}_{\pm 0.0010}$	$\mathbf{0.0554}_{\pm 0.0010}$
	DirectAU	$0.2269_{\pm 0.0020}$	$0.1201_{\pm 0.0009}$	$0.1249_{\pm 0.0019}$	$0.0959_{\pm 0.0012}$	$0.0738_{\pm 0.0011}$	$0.0411_{\pm 0.0006}$	$0.0661_{\pm 0.0013}$	$0.0664_{\pm 0.0011}$
	+PEAR	$\mathbf{0.2566}_{\pm 0.0026}$	$\mathbf{0.1344}_{\pm 0.0015}$	$\mathbf{0.1658}_{\pm 0.0022}$	$\mathbf{0.1239}_{\pm 0.0019}$	$\mathbf{0.1037}_{\pm 0.0021}$	$\mathbf{0.0570}_{\pm 0.0008}$	$\mathbf{0.0696}_{\pm 0.0017}$	$\mathbf{0.0686}_{\pm 0.0017}$
	LightCCF	$0.2720_{\pm 0.0021}$	$0.1464_{\pm 0.0009}$	$0.1628_{\pm 0.0026}$	$0.1250_{\pm 0.0018}$	$0.1141_{\pm 0.0025}$	$0.0653_{\pm 0.0015}$	$0.0549_{\pm 0.0006}$	$0.0555_{\pm 0.0004}$
	+PEAR	$\mathbf{0.3001}_{\pm 0.0026}$	$\mathbf{0.1600}_{\pm 0.0011}$	$\mathbf{0.1851}_{\pm 0.0030}$	$\mathbf{0.1383}_{\pm 0.0022}$	$\mathbf{0.1382}_{\pm 0.0030}$	$\mathbf{0.0771}_{\pm 0.0017}$	$\mathbf{0.0617}_{\pm 0.0006}$	$\mathbf{0.0607}_{\pm 0.0006}$
	SCCF	$0.2545_{\pm 0.0049}$	$0.1372_{\pm 0.0027}$	$0.1434_{\pm 0.0029}$	$0.1113_{\pm 0.0023}$	$0.1023_{\pm 0.0018}$	$0.0579_{\pm 0.0009}$	$0.0553_{\pm 0.0005}$	$0.0557_{\pm 0.0005}$
	+PEAR	$\mathbf{0.2875}_{\pm 0.0037}$	$\mathbf{0.1522}_{\pm 0.0019}$	$\mathbf{0.1794}_{\pm 0.0041}$	$\mathbf{0.1332}_{\pm 0.0027}$	$\mathbf{0.1347}_{\pm 0.0023}$	$\mathbf{0.0743}_{\pm 0.0011}$	$\mathbf{0.0592}_{\pm 0.0005}$	$\mathbf{0.0575}_{\pm 0.0005}$
	Avg. Improv.	+16.23%	+15.55%	+26.51%	+22.39%	+33.06%	+31.03%	+9.74%	+7.01%
LightGCN (3 layers)	BPR	$0.1917_{\pm 0.0012}$	$0.0977_{\pm 0.0008}$	$0.1031_{\pm 0.0017}$	$0.0802_{\pm 0.0011}$	$0.0828_{\pm 0.0011}$	$0.0466_{\pm 0.0006}$	$0.0478_{\pm 0.0010}$	$0.0494_{\pm 0.0009}$
	+PEAR	$\mathbf{0.2466}_{\pm 0.0027}$	$\mathbf{0.1302}_{\pm 0.0012}$	$\mathbf{0.1338}_{\pm 0.0023}$	$\mathbf{0.1019}_{\pm 0.0019}$	$\mathbf{0.1103}_{\pm 0.0018}$	$\mathbf{0.0615}_{\pm 0.0009}$	$\mathbf{0.0546}_{\pm 0.0011}$	$\mathbf{0.0562}_{\pm 0.0010}$
	DirectAU	$0.2256_{\pm 0.0028}$	$0.1197_{\pm 0.0016}$	$0.1169_{\pm 0.0024}$	$0.0895_{\pm 0.0019}$	$0.0693_{\pm 0.0010}$	$0.0384_{\pm 0.0006}$	$0.0667_{\pm 0.0016}$	$0.0667_{\pm 0.0016}$
	+PEAR	$\mathbf{0.2590}_{\pm 0.0029}$	$\mathbf{0.1365}_{\pm 0.0012}$	$\mathbf{0.1602}_{\pm 0.0036}$	$\mathbf{0.1184}_{\pm 0.0025}$	$\mathbf{0.0983}_{\pm 0.0014}$	$\mathbf{0.0533}_{\pm 0.0007}$	$\mathbf{0.0689}_{\pm 0.0017}$	$\mathbf{0.0682}_{\pm 0.0018}$
	LightCCF	$0.2675_{\pm 0.0052}$	$0.1435_{\pm 0.0030}$	$0.1622_{\pm 0.0027}$	$0.1247_{\pm 0.0021}$	$0.1126_{\pm 0.0011}$	$0.0642_{\pm 0.0006}$	$0.0511_{\pm 0.0005}$	$0.0524_{\pm 0.0006}$
	+PEAR	$\mathbf{0.3008}_{\pm 0.0064}$	$\mathbf{0.1594}_{\pm 0.0029}$	$\mathbf{0.1868}_{\pm 0.0033}$	$\mathbf{0.1389}_{\pm 0.0024}$	$\mathbf{0.1404}_{\pm 0.0018}$	$\mathbf{0.0781}_{\pm 0.0010}$	$\mathbf{0.0594}_{\pm 0.0010}$	$\mathbf{0.0586}_{\pm 0.0011}$
	SCCF	$0.2579_{\pm 0.0055}$	$0.1386_{\pm 0.0028}$	$0.1445_{\pm 0.0035}$	$0.1119_{\pm 0.0028}$	$0.1049_{\pm 0.0009}$	$0.0591_{\pm 0.0006}$	$0.0564_{\pm 0.0010}$	$0.0570_{\pm 0.0013}$
	+PEAR	$\mathbf{0.2874}_{\pm 0.0062}$	$\mathbf{0.1525}_{\pm 0.0030}$	$\mathbf{0.1801}_{\pm 0.0053}$	$\mathbf{0.1333}_{\pm 0.0041}$	$\mathbf{0.1377}_{\pm 0.0011}$	$\mathbf{0.0756}_{\pm 0.0007}$	$\mathbf{0.0602}_{\pm 0.0011}$	$\mathbf{0.0587}_{\pm 0.0012}$
	Avg. Improv.	+16.83%	+17.10%	+26.66%	+22.46%	+32.75%	+30.09%	+10.13%	+7.71%
LightGCN (4 layers)	BPR	$0.1991_{\pm 0.0015}$	$0.1022_{\pm 0.0013}$	$0.1103_{\pm 0.0016}$	$0.0851_{\pm 0.0012}$	$0.0901_{\pm 0.0011}$	$0.0508_{\pm 0.0006}$	$0.0500_{\pm 0.0010}$	$0.0524_{\pm 0.0010}$
	+PEAR	$\mathbf{0.2536}_{\pm 0.0027}$	$\mathbf{0.1323}_{\pm 0.0014}$	$\mathbf{0.1388}_{\pm 0.0024}$	$\mathbf{0.1051}_{\pm 0.0021}$	$\mathbf{0.1215}_{\pm 0.0015}$	$\mathbf{0.0679}_{\pm 0.0007}$	$\mathbf{0.0583}_{\pm 0.0013}$	$\mathbf{0.0593}_{\pm 0.0014}$
	DirectAU	$0.2212_{\pm 0.0021}$	$0.1172_{\pm 0.0009}$	$0.1166_{\pm 0.0019}$	$0.0893_{\pm 0.0016}$	$0.0695_{\pm 0.0011}$	$0.0382_{\pm 0.0006}$	$0.0667_{\pm 0.0011}$	$0.0667_{\pm 0.0012}$
	+PEAR	$\mathbf{0.2721}_{\pm 0.0028}$	$\mathbf{0.1418}_{\pm 0.0010}$	$\mathbf{0.1606}_{\pm 0.0026}$	$\mathbf{0.1187}_{\pm 0.0020}$	$\mathbf{0.0998}_{\pm 0.0014}$	$\mathbf{0.0540}_{\pm 0.0007}$	$\mathbf{0.0694}_{\pm 0.0012}$	$\mathbf{0.0679}_{\pm 0.0013}$
	LightCCF	$0.2615_{\pm 0.0044}$	$0.1399_{\pm 0.0025}$	$0.1636_{\pm 0.0027}$	$0.1267_{\pm 0.0022}$	$0.1156_{\pm 0.0018}$	$0.0656_{\pm 0.0010}$	$0.0547_{\pm 0.0006}$	$0.0548_{\pm 0.0007}$
	+PEAR	$\mathbf{0.3015}_{\pm 0.0050}$	$\mathbf{0.1601}_{\pm 0.0028}$	$\mathbf{0.1856}_{\pm 0.0020}$	$\mathbf{0.1390}_{\pm 0.0014}$	$\mathbf{0.1430}_{\pm 0.0020}$	$\mathbf{0.0797}_{\pm 0.0011}$	$\mathbf{0.0610}_{\pm 0.0008}$	$\mathbf{0.0596}_{\pm 0.0007}$
	SCCF	$0.2567_{\pm 0.0056}$	$0.1386_{\pm 0.0029}$	$0.1403_{\pm 0.0022}$	$0.1088_{\pm 0.0017}$	$0.1011_{\pm 0.0015}$	$0.0570_{\pm 0.0008}$	$0.0574_{\pm 0.0009}$	$0.0570_{\pm 0.0008}$
	+PEAR	$\mathbf{0.2894}_{\pm 0.0067}$	$\mathbf{0.1537}_{\pm 0.0037}$	$\mathbf{0.1766}_{\pm 0.0025}$	$\mathbf{0.1302}_{\pm 0.0022}$	$\mathbf{0.1329}_{\pm 0.0023}$	$\mathbf{0.0729}_{\pm 0.0014}$	$\mathbf{0.0606}_{\pm 0.0008}$	$\mathbf{0.0589}_{\pm 0.0008}$
	Avg. Improv.	+19.60%	+18.94%	+25.72%	+21.45%	+33.40%	+31.10%	+9.44%	+6.76%

and evaluation protocols. For the experiments reported in Tables 1 and 4, each configuration is run with five random seeds. For each seed, training is stopped when validation MRR@10 does not improve for 50 consecutive epochs, and the checkpoint with the highest validation MRR@10 is retained. Test MRR@10 and NDCG@10 are then evaluated at the selected checkpoint for each seed. We report their mean and sample standard deviation across the five seeds, using $n-1$ in the denominator for the standard deviation and rounding both statistics to four decimal places.

Computing environment. All experiments are conducted on a single NVIDIA GeForce RTX 4090 GPU.

D.2 Additional LightGCN Depths

The following table reports every archived one-, three-, and four-layer condition using the same five-seed selection, evaluation, and aggregation protocol as Table 1.

D.3 Controlled Comparisons with Generic Post-training User Adaptation

The component study in Section 4.4 tests PEAR's internal design choices and compares it with further embedding training. Here we ask whether its gains can instead be recovered by generic user-representation adaptation with the item space fixed. The controls separate full-dimensional fitting, adaptation along a single training-informed direction, and adaptation in a generic two-dimensional subspace.

Shared protocol. We use Douban-book with MF and the same four fixed pretrained checkpoints as in Section 4.4, one for each objective. Their baseline test MRR@10 values are 0.1794, 0.1499, 0.2693, and 0.2525 for BPR, DirectAU, LightCCF, and SCCF, respectively. All generic controls and PEAR freeze $\boldsymbol{V}$ and use the same training-positive sets $\mathcal{P}_u$, eligible users $\mathcal{U}_{\text{fit}} = \{u : |\mathcal{P}_u| > 1\}$, and baseline scales σ_u from equation 25. They optimize the full-catalog positive-set loss in equation 26, with no negative sampling. For context, we also reproduce **FT-Orig** and **FT-PEAR** from the main-text study; these references update both embedding tables and thus differ from the fixed-item controls.

Model selection uses aggregate validation MRR@10 after complete epochs, retaining one model state for all users and including the initial zero-adaptation state. Validation labels never select a separate coefficient or stopping epoch for each user. We use the fitting budget and stopping rule in Appendix C.6: at most 100 epochs, patience 8, and minimum validation improvement 10^{-8}. All methods use the same full-ranking evaluation filters: training interactions are masked for validation, and training plus validation interactions are masked for testing; padding IDs are always excluded. These masks do not alter the full-catalog training loss.

Fixed directions and common scaling. With $r_u = \|\boldsymbol{z}_u^{(0)}\|_2$, define

$$\boldsymbol{g}_u = -\sigma_u \nabla_{\boldsymbol{z}} \ell_u(\boldsymbol{z}_u^{(0)}; \boldsymbol{V}), \qquad \boldsymbol{K}_u \boldsymbol{t}_u = \boldsymbol{g}_u, \qquad \mathrm{Norm}_u(\boldsymbol{q}) = \begin{cases} r_u \boldsymbol{q}/\|\boldsymbol{q}\|_2, & \|\boldsymbol{q}\|_2 > 0, \\ \boldsymbol{0}, & \|\boldsymbol{q}\|_2 = 0. \end{cases} \tag{42}$$

Here $\boldsymbol{K}_u = \boldsymbol{C}_u + \gamma_u \boldsymbol{I}_d$ uses PEAR's predictive covariance and construction damping, including $\rho_C = 0.1$ in equation 32. Every nonzero direction therefore has the same norm as the original user representation, as in PEAR. We construct all directions once at $\boldsymbol{z}_u^{(0)}$ and hold them fixed throughout fitting; the likelihood is evaluated at the current adapted representation. Users outside $\mathcal{U}_{\mathrm{fit}}$, or with $r_u = 0$ or $\sigma_u = 0$, retain their original representation and are skipped before any normalization or loss evaluation. Zero directions remain zero and contribute no update. When all directions are zero, the coefficients remain zero and preconditioned fitting is skipped.

Generic adaptation controls. Direct-Z-full (full-dimensional direct user-representation fine-tuning) learns all $d = 64$ correction coordinates from $\boldsymbol{a}_u = \boldsymbol{0}$:

$$\boldsymbol{z}_u = \boldsymbol{z}_u^{(0)} + r_u \boldsymbol{a}_u, \qquad \min_{\|\boldsymbol{a}_u\|_2 \le 4} \ell_u(\boldsymbol{z}_u^{(0)} + r_u \boldsymbol{a}_u; \boldsymbol{V}) + \beta \|\boldsymbol{a}_u\|_2^2. \tag{43}$$

The penalty anchors the correction at the pretrained representation; penalizing $\|\boldsymbol{z}_u\|_2^2$ would instead pull the representation toward zero. The radius 4 matches an outer bound on relative displacement: PEAR's normalized bases and coefficient bounds imply $\|\boldsymbol{z}_u - \boldsymbol{z}_u^{(0)}\|_2 \le (2+2) r_u = 4 r_u$. The full ball contains PEAR's feasible updates, while allowing additional directions; the two parameterizations do not induce identical displacement penalties.

Grad-1D (one-dimensional gradient-direction adaptation) and **NatGrad-1D** (one-dimensional damped natural-gradient direction adaptation) use

$$\begin{aligned} \text{Grad-1D:} \quad & \boldsymbol{z}_u = \boldsymbol{z}_u^{(0)} + c_u \,\mathrm{Norm}_u(\boldsymbol{g}_u), \\ \text{NatGrad-1D:} \quad & \boldsymbol{z}_u = \boldsymbol{z}_u^{(0)} + c_u \,\mathrm{Norm}_u(\boldsymbol{t}_u). \end{aligned} \tag{44}$$

Both fit one signed coefficient with bound $\kappa = 4$. NatGrad-1D directly uses PEAR's target direction from the linear solve $\boldsymbol{K}_u \boldsymbol{t}_u = \boldsymbol{g}_u$, without the source dictionaries or their ridge regularization. Its name refers to the damped predictive Fisher direction, not a Hessian-based Newton direction (Appendix C.1).

Generic-2D (gradient–natural-gradient subspace adaptation) combines the two training-informed directions:

$$\boldsymbol{z}_u = \boldsymbol{z}_u^{(0)} + c_{u,1} \,\mathrm{Norm}_u(\boldsymbol{t}_u) + c_{u,2} \,\mathrm{Norm}_u(\boldsymbol{g}_u), \qquad (\kappa_1, \kappa_2) = (2, 2). \tag{45}$$

This control matches PEAR's two fitted coefficients and their bounds while replacing its source-specific bases with generic target-aware directions. The two directions need not be linearly independent, just as PEAR's two representation components need not be independent.

Random-2D (random two-dimensional subspace adaptation) draws a $d \times 2$ Gaussian matrix independently for each user and obtains $\boldsymbol{Q}_u$ by QR factorization, with $\boldsymbol{Q}_u^\top \boldsymbol{Q}_u = \boldsymbol{I}_2$. It uses $\boldsymbol{z}_u = \boldsymbol{z}_u^{(0)} + r_u \boldsymbol{Q}_u \boldsymbol{c}_u$ and bounds $(2, 2)$. The random directions are fixed within each run, with three predetermined direction seeds per checkpoint. This control tests whether a two-coefficient restriction alone yields useful adaptation.

Coefficient fitting and repeated runs. All one- and two-dimensional controls reuse PEAR's zero initialization, bounded parameterization $c_{u,j} = \kappa_j \tanh(\theta_{u,j}/\kappa_j)$, and regularized loss $\ell_u(\boldsymbol{z}_u; \boldsymbol{V}) +$

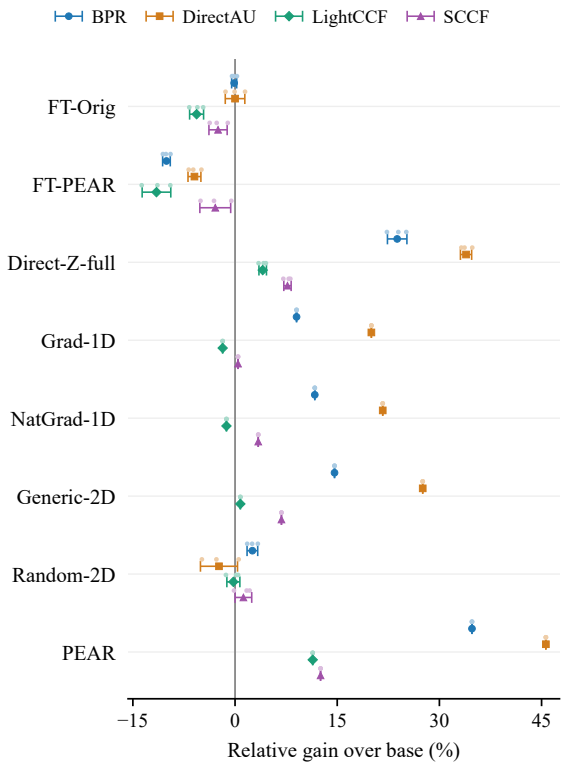


Figure 5: Controlled post-training comparisons on Douban-book with MF. Relative test MRR@10 gains are measured against the shared pretrained checkpoint for each objective. Large markers and whiskers show the mean and sample standard deviation over three post-training runs; small dots show individual runs. Coincident runs overlap and have zero dispersion at the reported precision. FT-Orig and FT-PEAR update both embedding tables; the remaining methods keep the item representations fixed.

$\beta\|\boldsymbol{c}_u\|_2^2$. We use $\beta = 0.01$ for these controls and Direct-Z-full. The low-dimensional controls also reuse the fixed Fisher-preconditioned fitting procedure in Appendix C.5, replacing $\boldsymbol{T}_u$ by their own direction matrix and using the corresponding coefficient dimension in the trace-averaged damping. The learning rate, step bounds, and fitting batch size follow Table 2. For each method, the three runs share the same pretrained checkpoint; their variation concerns post-training fitting and, for Random-2D, direction generation. Grad-1D, NatGrad-1D, Generic-2D, and PEAR return identical reported MRR@10 values across the three runs. This repeatability is conditional on the fixed checkpoints and does not measure variability across independently pretrained models.

User-only fitting recovers substantial gains, but leaves a gap to PEAR. Figure 5 shows that Direct-Z-full is the strongest generic control under every objective. In the order BPR, DirectAU, LightCCF, and SCCF, its mean MRR@10 is 0.2221, 0.2007, 0.2802, and 0.2719, improving on all four frozen checkpoints. Thus, the weak results of FT-Orig and FT-PEAR do not extend to all forms of post-training fitting: adapting users within a fixed item space is already effective. PEAR nevertheless reaches 0.2418, 0.2183, 0.3000, and 0.2842, exceeding Direct-Z-full by 8.9%, 8.8%, 7.1%, and 4.5%, respectively. The advantage persists despite Direct-Z-full's larger parameter count and its update ball containing PEAR's feasible displacements, supporting the effectiveness of PEAR's structured adaptation space under this protocol.

Training-informed directions help; source-specific construction adds further value. NatGrad-1D improves on Grad-1D under every objective, and Generic-2D further improves on both one-dimensional controls. However, Generic-2D reaches only 0.2056, 0.1912, 0.2714, and 0.2697; PEAR exceeds these values by 17.6%, 14.2%, 10.5%, and 5.4%, respectively. Because Generic-2D

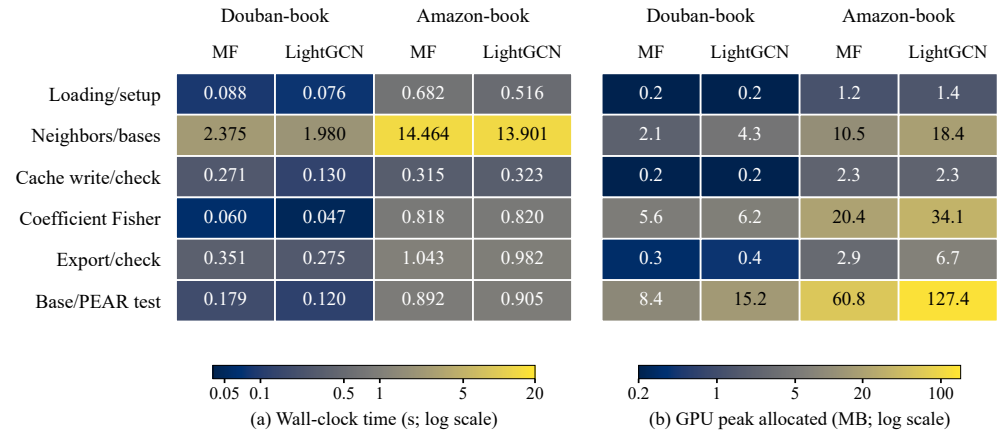


Figure 6: Six supplemental PEAR stages, excluding the pretraining and coefficient fitting reported in Figure 4. Left: wall-clock time in seconds. Right: stage peak GPU memory allocated by PyTorch, in MB. Cells show the reported measurements for each seed-2020 setting; annotations give untransformed values, with separate logarithmic color scales for time and memory. LightGCN uses two layers, and all four settings use BPR pretraining. Stage memory peaks include tensors still resident from earlier stages and must not be added.

shares the target signal, direction normalization, coefficient count, bounds, and fitting procedure, this comparison supports the additional value of constructing the two directions through PEAR's source dictionaries and ridge regularization. The target direction alone does not reproduce those gains, even though it optimizes the local surrogate in equation 12. Random-2D yields much smaller gains and falls below the baseline in mean MRR@10 for DirectAU and LightCCF, showing that restricting adaptation to two coefficients is insufficient by itself. Together with the source ablations in Section 4.4, these controls favor PEAR's combination of source-specific directions and bounded personalized fitting over the tested generic alternatives.

D.4 End-to-End Computational Cost and Resource Usage

Section 4.5 reports coefficient fitting separately from basis construction. Here we account for the additional stages needed to construct the bases, fit and export the adapted representations, and evaluate the resulting model. The four conditions are fixed in advance: BPR-trained MF and two-layer LightGCN on Douban-book and Amazon-book. They cover a smaller and a larger item catalog, and both identity and graph encoders.

Hardware and pretraining replay. Each resource run uses one dedicated NVIDIA GeForce RTX 4090 (24 GB), with Python 3.9.21 and PyTorch 2.1.0+cu121. We replay the original pretraining entry point with seed 2020, 64-dimensional embeddings, Xavier-uniform initialization, and a fresh Adam optimizer with learning rate 0.001 and regularization coefficient 10^{-5}; no previous optimizer state is restored. Training and RecBole evaluation batches are both 8,192, and BPR uniformly samples one negative item per positive example. The original BPR `EmbLoss` uses unsquared Frobenius norms of the sampled embedding matrices; the replay retains this regularizer. Validation MRR@10 is evaluated using the same pretraining model-selection protocol as in the main experiments: training stops after 50 consecutive epochs without improvement, and the checkpoint with the highest validation MRR@10 is retained. The epoch labels in Figure 4 identify the selected checkpoints, while all time comparisons below use the measured stage timers directly. This stopping rule is specific to the resource replay, rather than the general 50-epoch patience described in Appendix D. The epoch labels in Figure 4 identify selected checkpoints; all time comparisons below use the measured stage timers directly.

Table 5: Time accounting for PEAR, with pretraining as a reference (seconds). Basis includes neighbor retrieval and basis construction; other stages comprise loading/setup, cache writing/verification, coefficient Fisher construction, export/verification, and final Base/PEAR testing. The PEAR total sums basis, fitting, and other stages using the reported rounded measurements; it excludes the unreported unattributed remainder.

Setting	Pretrain	Basis	Fitting	Other stages	PEAR total (accounted)	Total / pretrain
Douban-book / MF	197.66	2.375	12.76	0.949	16.084	8.14%
Douban-book / LightGCN	1,721.22	1.980	124.24	0.648	126.868	7.37%
Amazon-book / MF	4,485.64	14.464	324.98	3.750	343.194	7.65%
Amazon-book / LightGCN	14,864.34	13.901	1,618.83	3.546	1,636.277	11.01%

Adaptation settings. PEAR freezes the original embedding table $\boldsymbol{E}_0$, final item representations $\boldsymbol{V}$, and both source-specific bases, and fits only two coefficients per eligible user. The settings follow Table 2: learning rate 0.1, user batch size 256, at most 100 epochs, patience 8, and minimum validation improvement 10^{-8}. The item- and user-source coefficient bounds are both 2, the coefficient penalty is $\beta = 0.01$, and the dictionary ridge multiplier is $\rho = 0.1$. Coefficients start at zero, and the unchanged pretrained readout is eligible for selection. The metric step bound is 0.1, followed by coordinate-wise clipping of the $\boldsymbol{\theta}$ step to ± 0.3, as in equation 41. We use the full-catalog positive-set objective in equation 26, with the fixed baseline score RMS floored at 10^{-4}. Both dictionaries use the original embedding tables; the user dictionary retains at most 32 training neighbors selected by inverse-item-degree-weighted overlap. Only non-padding users with training degree greater than one are fitted; the remaining users retain their original vectors. Representations, main computations, and linear solves use float32 without automatic mixed precision. The item Gram matrix is formed in float32, converted to float64 for eigendecomposition, and converted back afterwards.

Stage boundaries and accounting. GPU operations are synchronized at stage boundaries, and PyTorch peak statistics are reset separately for non-overlapping stages. We distinguish loading and setup (T_{load}), neighbor retrieval and basis construction (T_{basis}), basis-cache writing and verification (T_{cache}), coefficient-space Fisher construction (T_{Fisher}), coefficient fitting (T_{fit}), fitted-representation export and verification (T_{export}), and final Base/PEAR test evaluation (T_{test}). In particular, T_{basis} includes the catalog-wide predictive probabilities, predictive covariances, item Gram factorization, target and dictionary solves, and neighbor retrieval described in Appendix C.7. The separate T_{Fisher} stage constructs the fixed 2×2 coefficient-space preconditioners. We charge basis construction and cache writing once in the reported run, in addition to fitting. The final test stage evaluates both Base and PEAR and is retained in the accounting, although it is not required to produce the adapted serving vectors.

For these seven post-training stages, define

$$T_{\mathrm{accounted}} = T_{\mathrm{load}} + T_{\mathrm{basis}} + T_{\mathrm{cache}} + T_{\mathrm{Fisher}} + T_{\mathrm{fit}} + T_{\mathrm{export}} + T_{\mathrm{test}},$$
$$T_{\mathrm{complete}} = T_{\mathrm{accounted}} + T_{\mathrm{unattributed}}, \qquad M^{\mathrm{alloc}}_{\mathrm{peak,recorded}} = \max_{j \in \mathcal{J}_{\mathrm{alloc}}} M^{\mathrm{alloc}}_{\mathrm{peak},j}. \tag{46}$$

Pretraining is a separate reference cost and is not included in the post-training sum. An independent outer-run duration and its unattributed remainder are not available in this record, so the reported PEAR totals are sums of measured stages, rather than a separately measured T_{complete}; we do not assume $T_{\mathrm{unattributed}} = 0$. The accounting starts at loading and setup and does not quantify raw-data preprocessing or hyperparameter-search costs. Here $\mathcal{J}_{\mathrm{alloc}}$ contains stages measured with the same allocated-memory counter; their maximum is a complete-run peak only when instrumentation covers the entire run.

Cost beyond coefficient fitting. Figure 6 reports the construction and pipeline costs excluded from Figure 4, and Table 5 combines them with fitting. The six additional stages sum to 3.324, 2.628, 18.214, and 17.447 s for Douban-book MF, Douban-book LightGCN, Amazon-book MF, and Amazon-book LightGCN, respectively. Neighbor retrieval and basis construction account for most of this additional time: 1.980–2.375 s on Douban-book and 13.901–14.464 s on Amazon-book. Including all six stages raises the accounted PEAR time to 16.084, 126.868, 343.194, and 1,636.277 s,

respectively, or 7.4–11.0% of pretraining time. The added stages themselves cost 0.12–1.68% of pretraining time. Their contribution is largest relative to fitting for Douban-book MF, where they add 26.1% to the fitting timer and comprise 20.7% of the accounted PEAR total. Thus, construction is a material part of the fastest adaptation run, while the overall measured post-training cost remains a small fraction of pretraining in all four settings. Together with the BPR recommendation gains for these settings in Table 1, this supports PEAR's favorable accuracy–cost relationship; the resource replays are not additional accuracy repetitions.

Memory and storage scope. The supplemental memory measurements in Figure 6 are absolute stage peaks of PyTorch *allocated* GPU memory, not increases relative to the start of a stage. Allocated memory measures active tensor allocations; *reserved* memory also includes blocks retained by the PyTorch allocator. Neither is the device-wide usage reported by `nvidia-smi` or the host process's resident set size (RSS). Resetting a peak counter does not release resident tensors: a stage peak includes tensors retained from preceding stages, so peaks must be aggregated by a maximum across consistently measured stages, not by addition. Among the six supplemental stages, final Base/PEAR testing has the largest allocated peaks: 8.4, 15.2, 60.8, and 127.4 MB, respectively. These are numerically below the fitting-stage memory values of 11, 42, 118, and 232 MB in Figure 4. The original figure does not specify its allocator counter, and coverage of intervening intervals is unverified; this record therefore does not establish a complete-run allocated-memory peak. Host peak RSS, reserved-memory peaks, device-wide GPU peaks, and the sizes of temporary files and persistent caches are not recorded here. In particular, the $O(nd^2)$ construction storage in Appendix C.7 remains part of the resource requirement and cannot be inferred from the GPU counter alone. Cache writing and export are included in the timing results, but their on-disk sizes are not measured. After export, serving needs only the adapted user vectors and fixed item vectors, as described in Section 4.5.