CASQRec: Collaborative-Adaptive Semantic Quantization for Multimodal Recommendation

Qingfeng Li
Sun Yat-sen University
liqf37@mail2.sysu.edu.cn

Wenhui Tu
Sun Yat-sen University
tuwh@mail2.sysu.edu.cn

Wei Liu
Sun Yat-sen University
liuw259@mail.sysu.edu.cn

Huaijie Zhu
Sun Yat-sen University
zhuhuaijie@mail.sysu.edu.cn

Jielong Tang
Sun Yat-sen University
tangjlong3@mail2.sysu.edu.cn

Jian Yin*
Sun Yat-sen University
issjyin@mail.sysu.edu.cn

Abstract

Multimodal recommendation leverages item content, such as text, images, and audio signals, to complement sparse user-item interactions. Existing methods mainly combine content and collaborative signals at the representation level through graph propagation, feature fusion, or auxiliary alignment. However, continuous item embeddings make semantic sharing implicit and provide limited support for transferring behavioral evidence through reusable semantic factors. Discrete semantic codes offer explicit shared variables, but existing code-based recommenders usually treat item-code assignments as fixed outputs of pretrained quantizers, clustering algorithms, or frozen encoders, making the assignment mechanism insensitive to user behavior. We propose **CASQRec**, a **C**ollaborative-**A**daptive **S**emantic **Q**uantization framework for multimodal recommendation. CASQRec learns item-code assignments as content-inferable routing decisions supervised by collaborative behavior during training. In the semantic-code graph, fixed residual-quantization (RQ) anchors serve as shared nodes, and a learnable prior router predicts item–code edges from multimodal item content. During training, a collaborative posterior teacher uses contexts from the user–item graph to refine the prior assignments, and the content router learns from the refined distributions through distillation. During both training and inference, the router selects item–code edges from content, allowing items connected to the same code node to exchange information through graph propagation. Experiments on three public datasets show that CASQRec consistently outperforms representative multimodal recommendation baselines.

1 Introduction

Recommender systems increasingly rely on multimodal item content, such as text, images, videos, and audio signals, to complement sparse user-item interactions. Multimodal content describes an item's characteristics, while patterns in user interactions provide evidence about preferences across items [14, 12, 11, 13]. Multimodal recommendation therefore seeks to connect item content with collaborative signals from user behavior [15, 31, 34, 18, 19, 9]. A central challenge is how to build this connection in a way that is both semantically meaningful and effective for preference modeling.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Existing multimodal recommendation methods mainly combine content and behavior at the representation level. Early methods incorporate visual or textual features into collaborative filtering [3, 1]. Graph-based methods propagate multimodal features over user-item graphs or modality-specific item-item graphs [28, 26, 38, 40, 29], while self-supervised methods improve multimodal representations through reconstruction, contrastive learning, perturbation, or cross-modal alignment [41, 27, 24, 32]. Although effective, these methods typically represent each item as one or several continuous embeddings. Such item-level fusion makes semantic sharing implicit: collaborative supervision may improve item representations, but it does not explicitly specify which reusable semantic factors are shared across items or how behavioral evidence should flow through them.

Discrete semantic codes provide explicit shared variables for transferring behavioral evidence across items. By representing an item with multiple code tokens, semantic quantization can organize multimodal content into reusable code-level factors. Recent code-based recommendation methods have shown the value of discrete item representations for recommendation, semantic indexing, information diffusion, and transferable recommendation [5, 8, 16, 30, 25, 6, 21]. However, most existing methods treat item-code assignments as fixed outputs of pretrained quantizers, clustering algorithms, or frozen encoders. Under this design, collaborative training may update code embeddings or downstream representations, but it does not directly adapt the assignment mechanism that determines how items share semantic factors. As a result, item-code relations tend to act as static content identifiers rather than routing variables adapted to user behavior.

This limitation suggests two requirements for semantic-code assignments in multimodal recommendation. First, assignments should be content-inferable and shareable: items should be routed to shared semantic-code nodes from multimodal content alone, so that items with limited interactions can exchange behavioral evidence with related items through common codes. Second, user interactions should guide the assignments learned during training, while the router should be able to select code neighbors for a new item using only its content at inference time. This combination is difficult for both continuous representation fusion, which lacks explicit shared semantic variables, and fixed content-side quantization, whose assignments are insensitive to collaborative preferences. The missing component is therefore a deployable assignment mechanism: semantic-code routing should be supervised by collaborative behavior during training while remaining content-based at inference.

In this paper, we propose **CASQRec**, a **C**ollaborative-**A**daptive **S**emantic **Q**uantization framework for multimodal recommendation. CASQRec first builds a fixed content-derived RQ code space and treats semantic codes as graph nodes. A learnable content router predicts the item–code edges that augment the observed user–item graph using multimodal item features. To make code assignments adaptive to user behavior, CASQRec introduces a collaborative posterior teacher only during training. The posterior uses collaborative contexts from the user-item graph to provide behavior-aware assignment targets, and these targets are distilled into the content prior through collaborative semantic agreement. At inference time, the posterior branch is removed, and item-code edges are constructed by the prior router from item content alone. Thus, CASQRec learns item-code assignments that are shaped by collaborative preferences during training while remaining inferable from content at inference. CASQRec further performs graph propagation over the prior-induced user-item-code graph. Since multiple items can connect to shared semantic-code nodes, item-code-item paths enable collaborative evidence to transfer through reusable code-level factors. During both training and inference, the content prior selects the item–code edges used for propagation. This shared routing rule also applies to new items, whose connections to code nodes are determined by their content. Experiments on three public multimodal recommendation datasets show that CASQRec consistently outperforms representative baselines. Further analyses verify the effectiveness of collaborative-adaptive semantic quantization in assignment learning, semantic-code usage, and prior-based graph construction. In summary, our contributions are as follows:

- We formulate multimodal recommendation with semantic codes as an assignment-learning problem, where item-code assignments serve as explicit shared variables for transferring collaborative evidence.
- We propose a content prior router supervised by a training-only collaborative posterior, enabling item-code assignments to be collaboration-adaptive yet content-inferable while avoiding train-test routing mismatch.
- Experiments on three public datasets show consistent improvements over representative baselines and validate collaborative-adaptive semantic quantization.

2 Related Work

Multimodal recommendation. Multimodal recommendation mitigates sparse user-item interactions by using heterogeneous item content, such as text, images, videos, and audio signals [18, 19, 9, 23]. Early methods extend collaborative filtering with modality features, such as VBPR [3] and ACF [1]. Recent studies improve multimodal recommendation through graph learning and self-supervised learning, including MMGCN [28], GRCN [29], DualGNN [26], LATTICE [38], FREEDOM [40], SLMRec [24], BM3 [41], MMSSL [27], MICRO [39], and MENTOR [32]. Further efforts explore local-global graph learning [2], diffusion-based modeling [10], dual-stage modality fusion [31], and semantic user initialization [33]. These methods mainly couple content and collaborative signals through continuous representations, where shared semantic factors across items are modeled only implicitly. CASQRec instead moves this interaction to the assignment level by learning item-code relations over explicit semantic-code nodes.

Discrete semantic codes for recommendation. Discrete semantic codes provide structured alternatives to continuous item embeddings for semantic organization, transfer, retrieval, and information sharing [30, 25, 6, 35, 36]. Representative methods include VQRec [5], MMSR [8], CCFRec [16], and DQRec [21]; recent semantic-ID-based generative recommenders, such as TIGER [22], HSTU [37], ActionPiece [7], RPG [6], and DiffGRM [20], further use discrete identifiers for item generation. However, these methods usually obtain item-code assignments from pretrained tokenizers, quantizers, clustering algorithms, or frozen encoders, so the assignment rule is largely fixed before or outside collaborative training. CASQRec focuses on the assignment mechanism itself: it keeps a fixed content-derived RQ code space for stable inference, while learning content-inferable item-code edges with a prior router supervised by a training-only collaborative posterior. This enables assignments to be content-inferable, collaboration-adaptive, and shareable across items.

3 The CASQRec Framework

We propose **CASQRec**, a collaborative-adaptive semantic quantization framework for multimodal recommendation. CASQRec learns item-code assignments over a fixed content-derived code space, so semantic routing is shaped by collaborative behavior during training while remaining content-inferable at inference. A content prior router constructs the semantic-code graph, and a training-only collaborative posterior provides behavior-aware assignment targets. Figure 1 shows the overall framework. Let $\mathcal{U}$ and $\mathcal{I}$ denote the user and item sets. CASQRec constructs the user-item graph $\mathcal{G}_{ui} = (\mathcal{U} \cup \mathcal{I}, \mathcal{E}_{ui})$ from training interactions, where $(u, i) \in \mathcal{E}_{ui}$ denotes an observed interaction. Each item i has multimodal features $\{\mathbf{f}_i^m\}_{m\in\mathcal{M}}$. For simplicity, we use textual and visual modalities, $\mathcal{M} = \{t, v\}$, while the formulation naturally extends to more modalities. Each item is represented by R semantic-code positions, each with C candidate codes: $\mathcal{C} = \{(r, c) \mid r = 1, \ldots, R,\ c = 1, \ldots, C\}$. The goal is to learn a ranking score $\hat{y}_{ui}$ for top-K recommendation. Here $\mathrm{LN}(\cdot)$ denotes layer normalization, $\mathrm{sg}(\cdot)$ denotes stop-gradient, $\sigma(\cdot)$ denotes the sigmoid function, ϵ is a small constant.

3.1 Content-Derived Code Space Construction

CASQRec first constructs a fixed content-derived code space on which item-code assignments are learned. Raw modality features differ in dimensionality and scale, so direct quantization over their concatenation may be dominated by high-dimensional or high-variance modalities. We use a fixed descriptor transform $\mathcal{T}_{\mathrm{pca}}$ that applies modality-wise normalization, availability-aware weighted concatenation, and unsupervised PCA:

$$\bar{\mathbf{f}}_i = \mathcal{T}_{\mathrm{pca}}\left(\{\mathbf{f}_i^m, \delta_i^m\}_{m\in\mathcal{M}}\right) \in \mathbb{R}^d. \tag{1}$$

All statistics and PCA parameters in $\mathcal{T}_{\mathrm{pca}}$ are estimated from training items $\mathcal{I}_{\mathrm{tr}}$ and kept fixed afterwards. Residual quantization (RQ) is then fitted on $\{\bar{\mathbf{f}}_i \mid i \in \mathcal{I}_{\mathrm{tr}}\}$. Starting from $\mathbf{r}_{i,0} = \bar{\mathbf{f}}_i$, the stage-$r$ anchors and temporary hard assignments are

$$\begin{aligned} \{\mathbf{e}_{r,c}\}_{c=1}^{C} &= \text{K-means}\left(\{\mathbf{r}_{i,r-1} \mid i \in \mathcal{I}_{\mathrm{tr}}\}, C\right), \\ c_{i,r}^0 &= \arg\min_c \|\mathbf{r}_{i,r-1} - \mathbf{e}_{r,c}\|_2^2, \qquad \mathbf{r}_{i,r} = \mathbf{r}_{i,r-1} - \mathbf{e}_{r,c_{i,r}^0}. \end{aligned} \tag{2}$$

The resulting anchors $\{\mathbf{e}_{r,c}\}_{r=1,c=1}^{R,C}$ form a fixed content-derived code coordinate system and are excluded from trainable parameters. The temporary hard codes $\{c_{i,r}^0\}$ are used only to construct

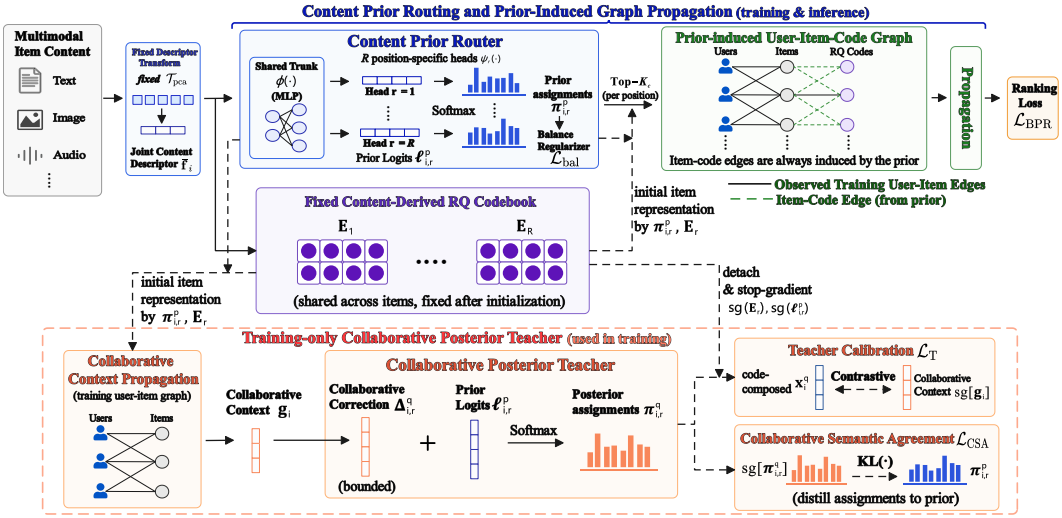

Figure 1: Overview of CASQRec with posterior teaching and prior-induced graph construction.

this coordinate system; they are not treated as supervision for recommendation. Instead, CASQRec learns item-code assignment distributions with a content prior router. This decouples code-space construction from assignment learning: RQ provides a fixed content-derived coordinate system, while the prior router learns collaboration-adaptive item-code assignments over it. Implementation details of $\mathcal{T}_{\text{pca}}$, RQ initialization, prior routing, and optional edge caching are provided in Appendix A.1.

3.2 Content Prior Routing

The content prior router maps the fixed joint content descriptor to item-code assignment distributions. It consists of a shared routing trunk $\phi(\cdot)$ and R position-specific heads $\{\psi_r(\cdot)\}_{r=1}^{R}$:

$$\mathbf{a}_i = \phi(\bar{\mathbf{f}}_i), \qquad \boldsymbol{\ell}^p_{i,r} = \psi_r(\mathbf{a}_i), \qquad \boldsymbol{\pi}^p_{i,r} = \mathrm{Softmax}(\boldsymbol{\ell}^p_{i,r}), \tag{3}$$

where $\mathbf{a}_i \in \mathbb{R}^d$ is the routing representation, ϕ is an MLP, and $\psi_r : \mathbb{R}^d \to \mathbb{R}^C$ maps $\mathbf{a}_i$ to logits over the C codes at position r. Here $\boldsymbol{\ell}^p_{i,r} \in \mathbb{R}^C$ and $\boldsymbol{\pi}^p_{i,r} \in \mathbb{R}^C$ are the prior logits and assignment distribution. The entry $\ell^p_{i,r,c}$ scores assigning item i to code anchor $\mathbf{e}_{r,c}$.

Let $\mathbf{E}_r = [\mathbf{e}_{r,1}; \ldots; \mathbf{e}_{r,C}] \in \mathbb{R}^{C\times d}$. The semantic token and initial item representation are

$$\mathbf{z}^p_{i,r} = \mathbf{E}_r^\top \boldsymbol{\pi}^p_{i,r}, \qquad \mathbf{x}^0_i = \mathrm{LN}\left(\frac{1}{R}\sum_{r=1}^{R} \mathbf{z}^p_{i,r}\right). \tag{4}$$

Since $\boldsymbol{\pi}^p_{i,r}$ depends only on the fixed descriptor computed from item content, the prior router is the routing mechanism used in both training and inference.

3.3 Collaborative Posterior Teaching

The content router predicts which code nodes an item should connect to from its multimodal features. During training, a collaborative posterior teacher refines these assignments using information from the item's neighborhood in the user–item graph. The refined distributions serve as targets for the content router. To obtain the teacher's input, we first compute a collaborative context for each item from the training interaction graph. Let $\tilde{\mathbf{A}}_{ui}$ be the normalized adjacency matrix of $\mathcal{G}_{ui}$. With initial node embeddings $\mathbf{H}^{(0)}_{ui} = [\{\mathbf{p}_u\}_{u\in\mathcal{U}}; \{\mathbf{x}^0_i\}_{i\in\mathcal{I}}]$, we compute

$$\mathbf{H}^c_{ui} = \sum_{\ell=0}^{L_c} \tilde{\mathbf{A}}^{\ell}_{ui} \mathbf{H}^{(0)}_{ui}, \qquad \mathbf{g}_i = (\mathbf{H}^c_{ui})_i, \tag{5}$$

where L_c is the number of context propagation layers and $\mathbf{g}_i$ is the collaborative context of item i. The context $\mathbf{g}_i$ is computed from training interactions only and is used only for the training-time

posterior, not for inference routing. The posterior applies a bounded collaborative correction to the prior logits:

$$\begin{aligned} \boldsymbol{\ell}^q_{i,r} &= \boldsymbol{\ell}^p_{i,r} + \boldsymbol{\Delta}^q_{i,r}, \qquad \boldsymbol{\pi}^q_{i,r} = \mathrm{Softmax}(\boldsymbol{\ell}^q_{i,r}), \\ \boldsymbol{\Delta}^q_{i,r} &= h^p_{i,r} \log C \, \sigma(\nu_r) \tanh\left(\mathbf{W}^g_r \, \mathrm{sg}(\mathbf{g}_i) + \mathbf{b}^g_r\right), \end{aligned} \tag{6}$$

where $\mathbf{W}^g_r \in \mathbb{R}^{C\times d}$, $\mathbf{b}^g_r \in \mathbb{R}^C$, and $\nu_r \in \mathbb{R}$ are learnable posterior parameters. $\boldsymbol{\Delta}^q_{i,r} \in \mathbb{R}^C$ denotes the bounded collaborative correction to the prior logits. The scalar $h^p_{i,r} = \mathrm{sg}(H(\pi^p_{i,r})/\log C)$ is the detached normalized prior entropy, where $H(\pi) = -\sum_c \pi_c \log \pi_c$. It allows larger collaborative corrections when the prior is uncertain and smaller corrections when the prior is confident. Since $\|\tanh(\cdot)\|_\infty \le 1$, we have $\|\boldsymbol{\ell}^q_{i,r} - \boldsymbol{\ell}^p_{i,r}\|_\infty \le h^p_{i,r} \log C \le \log C$. Thus, the posterior is a bounded refinement rather than an unconstrained replacement of the content prior. Appendix B.1 gives the corresponding distribution-ratio and KL bounds.

We calibrate the posterior teacher by aligning its code-composed representations with collaborative contexts. For teacher calibration, we detach the prior logits and codebook:

$$\begin{aligned} \boldsymbol{\pi}^{q,\mathrm{T}}_{i,r} &= \mathrm{Softmax}\left(\mathrm{sg}(\boldsymbol{\ell}^p_{i,r}) + \boldsymbol{\Delta}^q_{i,r}\right), \\ \mathbf{x}^q_i &= \mathrm{LN}\left(\frac{1}{R}\sum_{r=1}^{R} \mathrm{sg}(\mathbf{E}_r)^\top \boldsymbol{\pi}^{q,\mathrm{T}}_{i,r}\right). \end{aligned} \tag{7}$$

Here $\boldsymbol{\pi}^{q,\mathrm{T}}_{i,r}$ is used only in the calibration path; it has the same posterior correction form but detaches the prior logits. Given an item batch $\mathcal{B}_I$, the teacher calibration loss is

$$\mathcal{L}_{\mathrm{T}} = -\frac{1}{|\mathcal{B}_I|}\sum_{i\in\mathcal{B}_I} \log \frac{\exp(\mathrm{sim}(\mathbf{x}^q_i, \mathrm{sg}(\mathbf{g}_i))/\tau)}{\sum_{j\in\mathcal{B}_I}\exp(\mathrm{sim}(\mathbf{x}^q_i, \mathrm{sg}(\mathbf{g}_j))/\tau)}, \tag{8}$$

where $\mathrm{sim}(\cdot,\cdot)$ is cosine similarity and τ is a temperature. Since $\mathbf{x}^q_i$ is composed from posterior code distributions and the fixed codebook, the calibration loss cannot be satisfied by directly moving code anchors or prior logits. The calibration objective therefore updates the posterior correction to select code distributions whose composed representations are consistent with collaborative contexts. Appendix B.2 provides the gradient intuition.

3.4 Collaborative Semantic Agreement

The posterior teacher is behavior-aware, but its collaborative contexts are unavailable for content-only routing at inference. CASQRec therefore transfers posterior assignments into the content prior with a KL-based collaborative semantic agreement objective:

$$\mathcal{L}_{\mathrm{CSA}} = \frac{1}{|\mathcal{B}_I|R}\sum_{i\in\mathcal{B}_I}\sum_{r=1}^{R} \mathrm{KL}\left(\mathrm{sg}[\boldsymbol{\pi}^q_{i,r}] \,\middle\|\, \boldsymbol{\pi}^p_{i,r}\right). \tag{9}$$

The posterior distribution is detached as the teacher, while the prior acts as the content-inferable student. For each KL term, the gradient with respect to the prior logits is proportional to $\boldsymbol{\pi}^p_{i,r} - \mathrm{sg}[\boldsymbol{\pi}^q_{i,r}]$. Thus, CSA transfers collaborative knowledge at the assignment level: the prior is encouraged to match behavior-aware code assignments while remaining inferable from item content.

3.5 Prior-Induced Sparse Code Graph Propagation

CASQRec treats semantic codes as explicit graph nodes and builds $\mathcal{G}_{uic} = (\mathcal{U}\cup\mathcal{I}\cup\mathcal{C}, \mathcal{E}_{ui}\cup\mathcal{E}_{ic})$, where $\mathcal{E}_{ic}$ contains learned item-code edges. By construction, item-code edges are always induced by the content prior, while the posterior only shapes them through assignment supervision. This keeps the graph construction rule consistent between training and inference.

For each item i and code position r, we select the top-K_c codes $\mathcal{N}^K_{i,r} = \mathrm{TopK}(\boldsymbol{\pi}^p_{i,r}, K_c)$. The edge weight from item i to code node (r,c) is

$$S_{i,(r,c)} = \frac{1}{R}\cdot\frac{\pi^p_{i,r,c}}{\sum_{c'\in\mathcal{N}^K_{i,r}}\pi^p_{i,r,c'} + \epsilon}, \quad c\in\mathcal{N}^K_{i,r}, \tag{10}$$

and $S_{i,(r,c)} = 0$ otherwise. Let $\mathbf{S}$ collect all sparse item-code edge weights. The top-K_c selection is treated as a sparse graph-construction step, and the selected indices are not differentiated through within each forward pass or cached graph refresh. The prior router receives BPR gradients through the differentiable soft code composition in Eq. (4) and assignment-level signals from Eqs. (9) and (11); the sparse item-code graph is constructed or refreshed from the current prior. This prior-induced sparse routing keeps the graph efficient, and the factor $1/R$ balances the contribution of different code positions. To avoid degenerate code usage, we use a batch-level balance regularizer:

$$\mathcal{L}_{\mathrm{bal}} = \sum_{r=1}^{R} \sum_{c=1}^{C} \left(\frac{1}{|\mathcal{B}_I|} \sum_{i \in \mathcal{B}_I} \pi_{i,r,c}^{p} - \frac{1}{C} \right)^2 . \tag{11}$$

Let $\tilde{\mathbf{A}}_{uic}(\mathbf{S})$ be the normalized adjacency matrix of $\mathcal{G}_{uic}$. We initialize user, item, and code nodes as $\mathbf{h}_u^{(0)} = \mathbf{p}_u$, $\mathbf{h}_i^{(0)} = \mathbf{x}_i^0$, and $\mathbf{h}_{(r,c)}^{(0)} = \mathbf{e}_{r,c}$. Final representations are obtained by LightGCN [4] propagation:

$$\mathbf{H} = \sum_{\ell=0}^{L} \tilde{\mathbf{A}}_{uic}^{\ell}(\mathbf{S}) \mathbf{H}^{(0)}, \qquad \hat{y}_{ui} = \mathbf{h}_u^{\top} \mathbf{h}_i. \tag{12}$$

Graph propagation along $i \to (r, c) \to j$ transfers collaborative evidence through shared content-derived code nodes; because these edges are prior-induced, the learned routing rule remains deployable at inference.

3.6 Training and Inference

For recommendation, CASQRec optimizes the Bayesian Personalized Ranking loss: $\mathcal{L}_{\mathrm{BPR}} = -\sum_{(u,i,j)\in\mathcal{D}} \log \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right)$, where $\mathcal{D}$ is the set of sampled training triples, i is a positive item, and j is a sampled negative item for user u. The overall objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{BPR}} + \lambda_{\mathrm{T}} \mathcal{L}_{\mathrm{T}} + \lambda_{\mathrm{CSA}} \mathcal{L}_{\mathrm{CSA}} + \lambda_{\mathrm{bal}} \mathcal{L}_{\mathrm{bal}} + \eta \|\Theta\|_2^2, \tag{13}$$

where Θ denotes all trainable parameters.

During training, CASQRec computes prior assignments from the fixed joint content descriptor and obtains item representations through the fixed codebook. These representations are used to compute collaborative contexts on the training user-item graph. The posterior branch refines the prior assignments with collaborative contexts and is calibrated by $\mathcal{L}_{\mathrm{T}}$. The detached posterior assignments supervise the content prior through $\mathcal{L}_{\mathrm{CSA}}$, while recommendation is optimized on the prior-induced user-item-code graph by $\mathcal{L}_{\mathrm{BPR}}$. Thus, the posterior provides assignment-level collaborative targets, while the prior learns a content-only routing rule.

During inference, all posterior-related computations are removed. CASQRec computes the fixed joint descriptor from item content, obtains prior assignments, constructs sparse item-code edges by top-K_c routing, performs propagation on the prior-induced user-item-code graph, and scores user-item pairs by Eq. (12). Item-code routing therefore requires no item-side collaborative context at inference. Because both training and inference construct item-code edges from the same content prior, CASQRec avoids train-test routing mismatch caused by posterior-induced graphs. The complete training procedure and complexity analysis are provided in Appendix C.

4 Experimental Results

4.1 Experimental Settings

We evaluate CASQRec on three public multimodal recommendation datasets: Baby, Sports, and TikTok. Baby and Sports provide visual and textual features, while TikTok provides visual, acoustic, and textual features. We follow the popular evaluation setting [40, 2] with a random data splitting 8:1:1 for training, validation, and testing. We use Recall (R@K) and Normalized Discounted Cumulative Gain (NDCG, N@K) with $K = \{10, 20\}$ as evaluation metrics. CASQRec is compared with ten representative baselines, including LightGCN [4], CCFRec [16], DQRec [21], MMGCN [28], LATTICE [38], FREEDOM [40], BM3 [41], MENTOR [32], SG-URInit [33] and MoToRec [17]. We also evaluate MoToRec-SGU, which applies the user initialization of SG-URInit to MoToRec.

Table 1: Performance comparison on three datasets. Bold and underline denote the best and second-best results.

Methods	Baby				Sports				TikTok			
	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20
LightGCN [4]	0.0479	0.0754	0.0257	0.0328	0.0569	0.0864	0.0311	0.0387	0.0538	0.0832	0.0275	0.0352
DQRec [21]	0.0518	0.0812	0.0276	0.0354	0.0608	0.0928	0.0331	0.0419	0.0550	0.0850	0.0284	0.0362
CCFRec [16]	0.0542	0.0844	0.0292	0.0371	0.0616	0.0936	0.0330	0.0412	0.0605	0.0928	0.0318	0.0400
MMGCN [28]	0.0378	0.0615	0.0200	0.0261	0.0370	0.0605	0.0193	0.0254	0.0485	0.0764	0.0200	0.0268
LATTICE [38]	0.0547	0.0850	0.0292	0.0370	0.0620	0.0953	0.0335	0.0421	0.0586	0.0900	0.0309	0.0389
FREEDOM [40]	0.0627	0.0992	0.0330	0.0424	0.0717	0.1089	0.0385	0.0481	0.0575	0.0879	0.0316	0.0400
BM3 [41]	0.0564	0.0883	0.0301	0.0383	0.0656	0.0980	0.0355	0.0438	0.0578	0.0881	0.0300	0.0378
MENTOR [32]	0.0658	0.1026	0.0351	0.0446	0.0735	0.1121	0.0398	0.0498	0.0634	0.0975	0.0334	0.0423
SG-URInit [33]	0.0692	0.1079	0.0370	0.0471	0.0773	0.1178	0.0419	0.0525	0.0719	0.1107	0.0374	0.0474
MoToRec [17]	0.0711	0.1104	0.0382	0.0483	0.0784	0.1187	0.0433	0.0540	0.0745	0.1138	0.0389	0.0489
MoToRec-SGU	0.0762	0.1169	0.0404	0.0509	0.0820	0.1239	0.0452	0.0564	0.0791	0.1205	0.0415	0.0522
CASQRec	**0.0864**	**0.1359**	**0.0460**	**0.0594**	**0.0908**	**0.1398**	**0.0493**	**0.0625**	**0.0849**	**0.1317**	**0.0441**	**0.0566**

Table 2: Ablation study. Bold/underline denote best/second-best results.

Variants	**Baby**		**Sports**		**TikTok**	
	R@20	**N@20**	**R@20**	**N@20**	**R@20**	**N@20**
Effect of semantic-code graph propagation						
NoGraph	0.1132	0.0495	0.1105	0.0490	0.1067	0.0458
Effect of learnable prior routing						
HardRQ	0.1191	0.0521	0.1168	0.0518	0.1128	0.0485
Prior	0.1250	0.0544	0.1242	0.0551	0.1191	0.0512
Effect of collaborative posterior teacher						
NoCal	0.1226	0.0534	0.1217	0.0540	0.1175	0.0505
NoDistill	0.1246	0.0542	0.1240	0.0549	0.1188	0.0510
Effect of prior-based graph consistency						
PostGraph	0.1228	0.0535	0.1212	0.0538	0.1168	0.0502
CASQRec	**0.1359**	**0.0594**	**0.1398**	**0.0625**	**0.1317**	**0.0566**

4.2 Comparative Experiments

As shown in Table 1, CASQRec consistently achieves the best performance on Baby, Sports, and TikTok. These consistent gains across product and short-video recommendation scenarios demonstrate the effectiveness of the proposed framework.

The gains suggest that CASQRec improves multimodal recommendation by moving content-behavior interaction from representation learning to assignment learning. SG-URInit enhances MENTOR with semantic user initialization, but it still mainly improves continuous user/item representations. In contrast, CASQRec learns item-code assignments over explicit semantic-code nodes, so collaborative evidence can be shared through reusable code-level factors. CASQRec also outperforms DQRec and CCFRec, which use discrete representations for recommendation. To examine the role of assignment learning within CASQRec, we compare fixed and learned assignments and evaluate the contribution of posterior supervision in the ablation study.

The improvements over LATTICE, FREEDOM, BM3, and MENTOR further support the benefit of assignment-level semantic quantization over representation-level multimodal fusion. Through prior-induced item-code edges, CASQRec builds item-code-item paths that propagate collaborative evidence through shared content-derived code nodes. This is useful in sparse multimodal recommendation, where direct user-item signals are limited but content-inferable routing can still connect related items through common semantic codes.

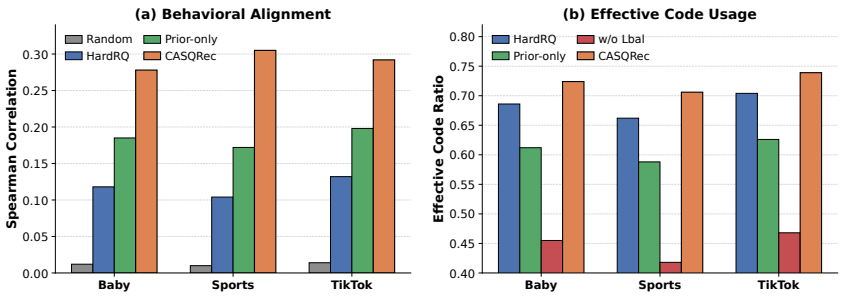

Figure 2: Behavioral alignment and usage of semantic codes.

4.3 Ablation Study

To examine the contribution of each component, we construct six variants of CASQRec. **NoGraph** removes semantic-code nodes and propagates only on the user-item graph. **HardRQ** replaces the learnable prior router with fixed hard RQ assignments. **Prior** removes the collaborative posterior teacher by setting $\lambda_{\mathrm{T}} = \lambda_{\mathrm{CSA}} = 0$. **NoCal** removes the posterior calibration loss $\mathcal{L}_{\mathrm{T}}$. **NoDistill** removes the collaborative semantic agreement loss $\mathcal{L}_{\mathrm{CSA}}$. **PostGraph** constructs item-code edges from posterior assignments during training but still uses prior assignments at inference.

As shown in Table 2, all variants underperform CASQRec. These variants isolate three factors: semantic-code graph propagation, content-inferable assignment learning, and collaborative posterior supervision. **NoGraph** shows the largest degradation, especially on Sports, the sparsest dataset, indicating that semantic-code nodes are crucial for transferring collaborative evidence through item-code-item paths. **HardRQ** uses fixed content-derived assignments, while **Prior** learns soft content-based assignments only from the recommendation objective. Their gaps to CASQRec show that discrete codes and soft routing are not sufficient by themselves; the assignment rule must also be shaped by collaborative behavior.

The posterior-related variants further clarify how collaborative supervision is transferred. Without $\mathcal{L}_{\mathrm{T}}$, **NoCal** produces weaker posterior targets because the code-composed posterior representations are not calibrated with collaborative contexts. Without $\mathcal{L}_{\mathrm{CSA}}$, **NoDistill** cannot transfer these behavior-aware assignments to the inference-time prior, so its performance remains close to **Prior**. Thus, calibration makes the posterior behavior-aware, while distillation makes its assignments usable by the content prior. Finally, **PostGraph** performs worse than CASQRec because posterior-induced training edges are unavailable at inference, creating a routing mismatch. CASQRec avoids this mismatch by always constructing the recommendation graph from the content prior and using the posterior only as a training-time assignment teacher.

4.4 Further Analysis

Analysis of semantic-code alignment and usage. We further examine whether CASQRec learns useful item-code assignments, rather than only improving ranking scores. We analyze the learned assignments from two perspectives: behavioral alignment and code usage. The compared variants follow the same settings as in the ablation study. For behavioral alignment, we collect each item's top-K_c prior-induced code neighbors, compute the Jaccard similarity between item-pair code sets, and report its Spearman correlation with collaborative similarity measured from item collaborative contexts. For code usage, we report the effective code ratio to assess routing collapse.

As shown in Figure 2(a), CASQRec achieves the strongest behavioral alignment across all datasets. HardRQ shows limited alignment, indicating that fixed content-derived quantization does not fully capture collaborative preference patterns. Prior improves alignment through recommendation supervision, but still lacks explicit behavior-aware assignment guidance. CASQRec further improves alignment by using the posterior teacher to provide behavior-aware assignment targets for the content prior. Figure 2(b) shows that CASQRec also maintains a high effective code ratio, while removing $\mathcal{L}_{\mathrm{bal}}$ substantially reduces code usage diversity. This shows that the balance regularizer helps avoid routing collapse and keeps semantic-code nodes effectively used. Notably, HardRQ also uses codes reasonably well, but its behavioral alignment is much weaker than that of CASQRec. Thus, CASQRec

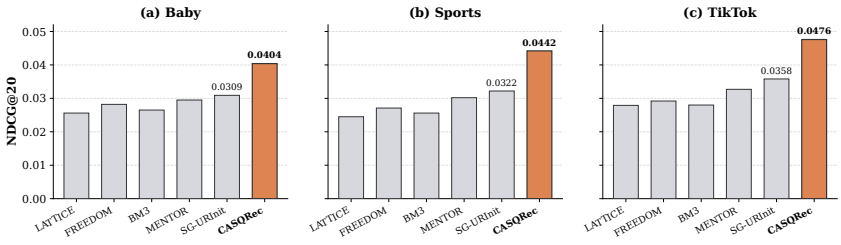

Figure 3: Item cold-start performance.

Table 3: Effect of code-space design on recommendation performance.

Variants	Baby		Sports		TikTok	
	R@20	N@20	R@20	N@20	R@20	N@20
Random Codebook	0.1218	0.0529	0.1207	0.0534	0.1165	0.0500
K-means Codebook	0.1296	0.0564	0.1312	0.0584	0.1263	0.0542
Trainable Codebook	0.1310	0.0570	0.1324	0.0587	0.1276	0.0547
HardRQ	0.1191	0.0521	0.1168	0.0518	0.1128	0.0485
CASQRec	**0.1359**	**0.0594**	**0.1398**	**0.0625**	**0.1317**	**0.0566**

does not merely use more codes; it learns code assignments that are both diverse and better aligned with collaborative behavior.

Item Cold-Start Recommendation. We evaluate item cold-start recommendation by randomly selecting 5% of items for validation and 10% for testing. All training interactions of these cold items are removed, while their multimodal content features are retained for inference. The fixed descriptor transform and RQ anchors are fitted only on warm training items and then applied to cold validation and test items. During evaluation, ranking is performed over the full item set, and metrics are computed only on test positives from cold-test items. This setting directly tests the deployability of the prior router, since cold items can enter the graph only through content-inferable item-code edges. As shown in Figure 3, CASQRec consistently performs best on all datasets. Although multimodal baselines can encode cold items from content, their representations are weakly connected to collaborative signals learned from warm items. CASQRec instead routes cold items, using content alone, to content-derived code nodes that have accumulated collaborative evidence from warm items. Thus, unseen items can enter the prior-induced semantic-code graph without item-side interaction history. The stronger gains on Sports and TikTok suggest that content-based routing is especially useful under sparse interactions or richer multimodal evidence. These results show that the prior router supports inductive recommendation by connecting cold items to reusable semantic-code structures.

Effect of Code-Space Design. We further examine the role of code-space design in CASQRec. For **Random Codebook**, we replace the content-derived RQ anchors with fixed random anchors while keeping the prior router, posterior teacher, and distillation objective unchanged. For **K-means Codebook**, we build a single-stage K-means codebook from the fixed joint content descriptors. For **Trainable Codebook**, we initialize the codebook with RQ anchors and update code embeddings during recommendation training. **HardRQ** uses fixed hard RQ assignments to build item-code edges, removing learnable soft routing. CASQRec keeps the content-derived RQ codebook fixed and adapts item-code assignments through prior routing and posterior distillation.

As shown in Table 3, CASQRec performs best on all datasets. Random Codebook only slightly improves over HardRQ, showing that learnable routing needs a meaningful content-derived code space. K-means Codebook performs better than Random Codebook because content clustering provides more structured anchors, but it still falls behind CASQRec. This suggests that residual quantization gives a more expressive multi-stage code space. Trainable Codebook achieves the second-best results, indicating that updating code embeddings improves flexibility. However, its gap to CASQRec suggests that freely changing the codebook may weaken the stable content coordinate

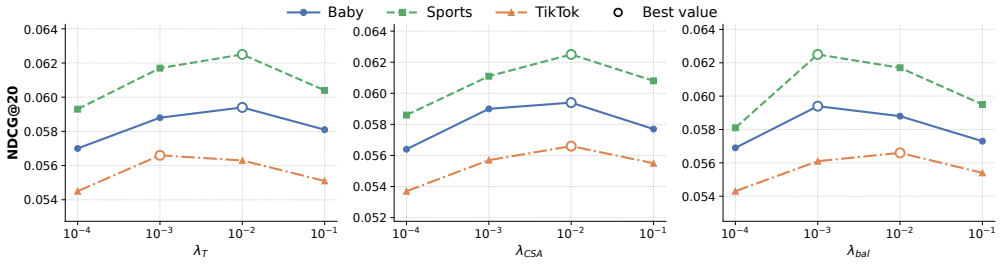

Figure 4: Sensitivity analysis of the training loss weights in terms of NDCG@20.

system needed for content-inferable routing. These results support our design choice: keep the code space fixed and learn collaborative adaptation mainly through item-code assignments. A fixed content-derived RQ codebook provides stable semantic anchors, while the prior router and posterior teacher adapt assignments to collaborative behavior.

Sensitivity to Loss Weights. Figure 4 shows the sensitivity of CASQRec to λ_T, λ_{CSA}, and λ_{bal}. CASQRec is stable under moderate weights, while very small or very large values hurt performance.

For λ_T, a small value gives weak supervision to the posterior teacher, so the teacher may not capture collaborative semantics well. A large value can overemphasize teacher calibration and weaken the main recommendation objective. For λ_{CSA}, a small value limits the transfer of behavior-aware assignments from the posterior to the content prior. A large value makes distillation dominate training and may reduce ranking quality. For λ_{bal}, weak regularization may cause routing to concentrate on a few codes, while excessive regularization forces assignments to be overly uniform and weakens semantic selectivity.

The trends also reflect dataset differences. Sports is more sensitive to λ_{CSA} and λ_{bal} because its interactions are sparser, making collaborative distillation and balanced code usage more important. TikTok changes more smoothly, as richer multimodal content provides more stable routing evidence. These results show that the three loss terms play complementary roles and that CASQRec does not depend on a fragile weight setting.

5 Conclusion

We present CASQRec, a collaborative-adaptive semantic quantization framework for multimodal recommendation. CASQRec represents residual-quantization anchors as shared graph nodes and learns which nodes each item connects to. During training, a collaborative posterior uses interaction-derived item contexts to refine the assignments predicted from content. Distilling these assignments into the content router allows user behavior to shape the paths along which information propagates between items. At inference, the router selects code neighbors from multimodal content alone, allowing new items to receive collaborative information through codes they share with previously observed items. Across three public datasets, CASQRec outperforms the evaluated baselines in full-ranking recommendation, and ablations support the contributions of learned routing and posterior supervision. The fixed codebook may, however, provide limited coverage as new types of items enter the catalog. A natural next step is to expand the codebook as the catalog evolves and evaluate whether new anchors improve recommendations for emerging items while preserving ranking quality for existing ones.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. U22B2060), the Key-Area Research and Development Program of Guangdong Province (Grant No. 2024B0101050005), the Research Foundation of Science and Technology Plan Project of Guangzhou City (Grant No. 2024B01W0004), and the Hengqin Guangdong-Macao Deep Cooperation Zone Industrial Core and Key Technology Breakthrough Project (Grant No. 2530004025501).

References

[1] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 335–344, 2017.

[2] Zhiqiang Guo, Jianjun Li, Guohui Li, Chaoyang Wang, Si Shi, and Bin Ruan. Lgmrec: Local and global graph learning for multimodal recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8454–8462, 2024.

[3] Ruining He and Julian McAuley. Vbpr: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[4] 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*, pages 639–648, 2020.

[5] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning vector-quantized item representation for transferable sequential recommenders. In *Proceedings of the ACM Web Conference 2023*, pages 1162–1171, 2023.

[6] Yupeng Hou, Jiacheng Li, Ashley Shin, Jinsung Jeon, Abhishek Santhanam, Wei Shao, Kaveh Hassani, Ning Yao, and Julian McAuley. Generating long semantic ids in parallel for recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 956–966, 2025.

[7] Yupeng Hou, Jianmo Ni, Zhankui He, Noveen Sachdeva, Wang-Cheng Kang, Ed H Chi, Julian Mcauley, and Derek Zhiyuan Cheng. Actionpiece: Contextually tokenizing action sequences for generative recommendation. In *International Conference on Machine Learning*, pages 24004–24024. PMLR, 2025.

[8] Hengchang Hu, Wei Guo, Yong Liu, and Min-Yen Kan. Adaptive multi-modalities fusion in sequential recommendation systems. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 843–853, 2023.

[9] Jun Hu, Bryan Hooi, Bingsheng He, and Yinwei Wei. Modality-independent graph neural networks with global transformers for multimodal recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 11790–11798, 2025.

[10] Yangqin Jiang, Lianghao Xia, Wei Wei, Da Luo, Kangyi Lin, and Chao Huang. Diffmm: Multimodal diffusion model for recommendation. In *Proceedings of the 32nd ACM international conference on multimedia*, pages 7591–7599, 2024.

[11] Qingfeng Li, Huifang Ma, Ruoyi Zhang, Wangyu Jin, and Zhixin Li. Intra-and inter-behavior contrastive learning for multi-behavior recommendation. In *International Conference on Database Systems for Advanced Applications*, pages 147–162. Springer, 2023.

[12] Qingfeng Li, Huifang Ma, Wangyu Jin, Yugang Ji, and Zhixin Li. Multi-interest network with simple diffusion for multi-behavior sequential recommendation. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, pages 734–742. SIAM, 2024.

[13] Qingfeng Li, Huifang Ma, Wangyu Jin, Yugang Ji, and Zhixin Li. Hypergraph-enhanced multi-interest learning for multi-behavior sequential recommendation. *Expert Systems with Applications*, 255:124497, 2024.

[14] Qingfeng Li, Wei Liu, Zaiqiao Meng, and Jian Yin. Decision-aware preference modeling for multi-behavior recommendation. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pages 3063–3071. ijcai.org, 2025.

[15] Xixun Lin, Rui Liu, Yanan Cao, Lixin Zou, Qian Li, Yongxuan Wu, Yang Liu, Dawei Yin, and Guandong Xu. Contrastive modality-disentangled learning for multimodal recommendation. *ACM Transactions on Information Systems*, 43(3):1–31, 2025.

[16] Enze Liu, Bowen Zheng, Wayne Xin Zhao, and Ji-Rong Wen. Bridging textual-collaborative gap through semantic codes for sequential recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 1788–1798, 2025.

[17] Jialin Liu, Zhaorui Zhang, and Ray CC Cheung. Motorec: Sparse-regularized multimodal tokenization for cold-start recommender. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 15324–15332, 2026.

[18] Qijiong Liu, Jieming Zhu, Yanting Yang, Quanyu Dai, Zhaocheng Du, Xiao-Ming Wu, Zhou Zhao, Rui Zhang, and Zhenhua Dong. Multimodal pretraining, adaptation, and generation for recommendation: A survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6566–6576, 2024.

[19] Yifan Liu, Kangning Zhang, Xiangyuan Ren, Yanhua Huang, Jiarui Jin, Yingjie Qin, Ruilong Su, Ruiwen Xu, Yong Yu, and Weinan Zhang. Alignrec: Aligning and training in multimodal recommendations. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 1503–1512, 2024.

[20] Zhao Liu, Yichen Zhu, Yiqing Yang, Xiao Lv, Guoping Tang, Rui Huang, Qiang Luo, Ruiming Tang, and Guorui Zhou. Diffgrm: Diffusion-based generative recommendation model. In *Proceedings of the ACM Web Conference 2026*, pages 5853–5864, 2026.

[21] Yunze Luo, Yinjie Jiang, Gaode Chen, Jingchi Wang, Shicheng Wang, Ruina Sun, Jiang Yuezihan, Jun Zhang, Jian Liang, Han Li, et al. Representation quantization for collaborative filtering augmentation. *arXiv preprint arXiv:2508.11194*, 2025.

[22] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

[23] Jielong Tang, Xujie Yuan, Jiayang Liu, Jianxing Yu, Xiao Dong, Lin Chen, Yunlai Teng, Shimin Di, and Jian Yin. SAKE: Self-aware knowledge exploitation-exploration for grounded multimodal named entity recognition. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '26, pages 4592–4603. Association for Computing Machinery, 2026.

[24] Zhulin Tao, Xiaohao Liu, Yewei Xia, Xiang Wang, Lifang Yang, Xianglin Huang, and Tat-Seng Chua. Self-supervised learning for multimedia recommendation. *IEEE Transactions on Multimedia*, 25:5107–5116, 2022.

[25] Dongsheng Wang, Yuxi Huang, Shen Gao, Yifan Wang, Chengrui Huang, and Shuo Shang. Generative next poi recommendation with semantic id. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 2904–2914, 2025.

[26] Qifan Wang, Yinwei Wei, Jianhua Yin, Jianlong Wu, Xuemeng Song, and Liqiang Nie. Dualgnn: Dual graph neural network for multimedia recommendation. *IEEE Transactions on Multimedia*, 25:1074–1084, 2021.

[27] Wei Wei, Chao Huang, Lianghao Xia, and Chuxu Zhang. Multi-modal self-supervised learning for recommendation. In *Proceedings of the ACM web conference 2023*, pages 790–800, 2023.

[28] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. Mmgcn: Multi-modal graph convolution network for personalized recommendation of micro-video. In *Proceedings of the 27th ACM international conference on multimedia*, pages 1437–1445, 2019.

[29] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, and Tat-Seng Chua. Graph-refined convolutional network for multimedia recommendation with implicit feedback. In *Proceedings of the 28th ACM international conference on multimedia*, pages 3541–3549, 2020.

[30] Longtao Xiao, Haozhao Wang, Cheng Wang, Linfei Ji, Yifan Wang, Jieming Zhu, Zhenhua Dong, Rui Zhang, and Ruixuan Li. Unger: Generative recommendation with a unified code via semantic and collaborative integration. *ACM Transactions on Information Systems*, 44(2):1–31, 2025.

[31] Jinfeng Xu, Zheyu Chen, Wei Wang, Xiping Hu, Sang-Wook Kim, and Edith CH Ngai. Cohesion: Composite graph convolutional network with dual-stage fusion for multimodal recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1830–1839, 2025.

[32] Jinfeng Xu, Zheyu Chen, Shuo Yang, Jinze Li, Hewei Wang, and Edith CH Ngai. Mentor: multi-level self-supervised learning for multimodal recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 12908–12917, 2025.

[33] Jinfeng Xu, Zheyu Chen, Shuo Yang, Jinze Li, Hewei Wang, Jianheng Tang, Wei Wang, Xiping Hu, and Edith CH Ngai. Well begun is half done: Training-free and model-agnostic semantically guaranteed user representation initialization for multimodal recommendation. *arXiv preprint arXiv:2604.14839*, 2026.

[34] Yimeng Yang, Haokai Ma, Lei Meng, Shuo Xu, Ruobing Xie, and Xiangxu Meng. Curriculum conditioned diffusion for multimodal recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 13035–13043, 2025.

[35] Xujie Yuan, Shimin Di, Jielong Tang, Libin Zheng, and Jian Yin. Towards self-cognitive exploration: Metacognitive knowledge graph retrieval augmented generation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '26, pages 1844–1855. Association for Computing Machinery, 2026.

[36] Xujie Yuan, Yongxu Liu, Shimin Di, Shiwen Wu, Libin Zheng, Rui Meng, Lei Chen, Xiaofang Zhou, and Jian Yin. A pilot empirical study on when and how to use knowledge graphs as retrieval augmented generation. In *Database Systems for Advanced Applications*, volume 16538 of *Lecture Notes in Computer Science*, pages 471–488, Singapore, 2026. Springer Nature Singapore.

[37] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, et al. Actions speak louder than words: trillion-parameter sequential transducers for generative recommendations. In *Proceedings of the 41st International Conference on Machine Learning*, pages 58484–58509, 2024.

[38] Jinghao Zhang, Yanqiao Zhu, Qiang Liu, Shu Wu, Shuhui Wang, and Liang Wang. Mining latent structures for multimedia recommendation. In *Proceedings of the 29th ACM international conference on multimedia*, pages 3872–3880, 2021.

[39] Jinghao Zhang, Yanqiao Zhu, Qiang Liu, Mengqi Zhang, Shu Wu, and Liang Wang. Latent structure mining with contrastive modality fusion for multimedia recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):9154–9167, 2022.

[40] Xin Zhou and Zhiqi Shen. A tale of two graphs: Freezing and denoising graph structures for multimodal recommendation. In *Proceedings of the 31st ACM international conference on multimedia*, pages 935–943, 2023.

[41] Xin Zhou, Hongyu Zhou, Yong Liu, Zhiwei Zeng, Chunyan Miao, Pengwei Wang, Yuan You, and Feijun Jiang. Bootstrap latent representations for multi-modal recommendation. In *Proceedings of the ACM web conference 2023*, pages 845–854, 2023.

A Additional Details of CASQRec

This appendix provides implementation details and supporting analysis for CASQRec. We focus on three mechanisms that support the main claims of the paper: item-code routing remains inferable from item content at inference time; collaborative behavior shapes routing through training-time assignment supervision; and the recommendation graph is constructed from the same content prior during training and inference.

A.1 Fixed Descriptor Transform, RQ Coordinates, and Prior Routing

CASQRec constructs the code space from a fixed joint content descriptor rather than from trainable modality projection modules. The descriptor transform provides a reproducible input space for RQ codebook construction and reduces the effect of modality imbalance caused by heterogeneous feature dimensions and scales. For each modality m, we estimate the feature-wise mean and standard deviation, $\boldsymbol{\mu}_m$ and $\boldsymbol{\sigma}_m$, using training items for which modality m is available. For an available modality, the raw feature is standardized and normalized as

$$\tilde{\mathbf{f}}_i^m = \mathrm{Norm}\left(\frac{\mathbf{f}_i^m - \boldsymbol{\mu}_m}{\boldsymbol{\sigma}_m + \epsilon}\right), \tag{14}$$

where $\mathrm{Norm}(\cdot)$ denotes ℓ_2 normalization. Missing modalities are masked by the availability indicator δ_i^m. We assign each available modality a normalized weight

$$\alpha_i^m = \frac{\delta_i^m}{\sum_{n\in\mathcal{M}} \delta_i^n + \epsilon}, \tag{15}$$

so that items with different numbers of available modalities have comparable descriptor magnitudes.

The pre-PCA content descriptor concatenates the weighted modality blocks and the modality-availability indicators:

$$\mathbf{d}_i^{\mathrm{pre}} = \left[\sqrt{\alpha_i^m}\delta_i^m\tilde{\mathbf{f}}_i^m\right]_{m\in\mathcal{M}} \oplus \left[\delta_i^m\right]_{m\in\mathcal{M}}. \tag{16}$$

The factor $\sqrt{\alpha_i^m}$ makes the modality-feature part of the squared Euclidean distance correspond to a weighted sum of modality-wise distances, while the availability indicators preserve missing-modality information after concatenation.

We fit PCA on the training descriptors $\{\mathbf{d}_i^{\mathrm{pre}} \mid i \in \mathcal{I}_{\mathrm{tr}}\}$. Let $\boldsymbol{\mu}_{\mathrm{pre}}$ be the mean of the training descriptors and let $\mathbf{W}_{\mathrm{pca}} \in \mathbb{R}^{D_{\mathrm{pre}}\times d}$ contain the top-d principal directions, where D_{pre} is the dimension of $\mathbf{d}_i^{\mathrm{pre}}$. The final fixed joint content descriptor is

$$\bar{\mathbf{f}}_i = \mathrm{LN}\left(\mathbf{W}_{\mathrm{pca}}^\top\left(\mathbf{d}_i^{\mathrm{pre}} - \boldsymbol{\mu}_{\mathrm{pre}}\right)\right). \tag{17}$$

All statistics $\{\boldsymbol{\mu}_m, \boldsymbol{\sigma}_m\}$, $\boldsymbol{\mu}_{\mathrm{pre}}$, and $\mathbf{W}_{\mathrm{pca}}$ are estimated only from training items and kept fixed during recommendation training and inference. For validation, test, and cold-start items, CASQRec applies the same fixed transform to their available content features. Thus, the descriptor construction introduces no trainable modality projection module, and no validation or test items are used to estimate the transform. It only defines a compact content-derived descriptor space for RQ initialization; collaborative semantics are introduced later through posterior-guided assignment learning.

Residual quantization is then applied to the fixed descriptors to initialize a content-derived code coordinate system. The construction is performed on the training item set $\mathcal{I}_{\mathrm{tr}}$ and does not use validation or test items. Given $\bar{\mathbf{f}}_i$ in Eq. (17), the residual is initialized as $\mathbf{r}_{i,0} = \bar{\mathbf{f}}_i$. At residual stage r, CASQRec fits C anchors on the current residuals of training items and updates the residual by

$$\begin{aligned}
\{\mathbf{e}_{r,c}\}_{c=1}^{C} &= \text{K-means}\left(\{\mathbf{r}_{i,r-1} \mid i \in \mathcal{I}_{\mathrm{tr}}\}, C\right), \\
c_{i,r}^0 &= \arg\min_c \|\mathbf{r}_{i,r-1} - \mathbf{e}_{r,c}\|_2^2, \\
\mathbf{r}_{i,r} &= \mathbf{r}_{i,r-1} - \mathbf{e}_{r,c_{i,r}^0}.
\end{aligned} \tag{18}$$

After initialization, the anchors $\{\mathbf{e}_{r,c}\}_{r=1,c=1}^{R,C}$ are kept fixed during recommendation training. The temporary hard assignments $\{c_{i,r}^0\}$ are not used as fixed item-code labels and are not used as

supervised targets for the router. Instead, CASQRec learns a content prior router to infer soft item-code distributions from item content.

The prior router consists of a shared trunk $\phi : \mathbb{R}^d \to \mathbb{R}^d$ and position-specific heads $\psi_r : \mathbb{R}^d \to \mathbb{R}^C$. The trunk extracts a routing representation $\mathbf{a}_i = \phi(\bar{\mathbf{f}}_i)$, and each head predicts logits over the C code anchors at position r. The use of position-specific heads follows the RQ structure: different residual stages define distinct assignment spaces and therefore need not share the same classifier.

This design decouples coordinate construction from assignment learning. RQ provides a stable content-derived coordinate system, while collaborative semantics are introduced through assignment learning. Since the code anchors remain tied to the fixed descriptor space used for content-based routing, CASQRec does not rely on updating code embeddings to absorb collaborative signals.

The item-code graph is always constructed from the content prior distribution $\boldsymbol{\pi}^p_{i,r}$. For each item i and code position r, CASQRec selects

$$\mathcal{T}_{i,r} = \mathrm{TopK}(\boldsymbol{\pi}^p_{i,r}, K_c) \tag{19}$$

and assigns sparse edge weights according to Eq. (10). The resulting item-code edge set is

$$\mathcal{E}_{ic} = \{(i, (r, c)) \mid c \in \mathcal{T}_{i,r},\ r = 1, \ldots, R\}. \tag{20}$$

Each item connects to at most RK_c semantic-code nodes. Because $\boldsymbol{\pi}^p_{i,r}$ is computed only from item content, the same routing rule applies to any item with available content features, without requiring item-side interaction history.

For scalable full-graph training, the prior-induced item-code edges can be refreshed once per epoch and cached for graph propagation. Let $\mathbf{S}^{(e)}$ denote the cached item-code assignment matrix generated by the current prior router at epoch e. Within the epoch, $\mathbf{S}^{(e)}$ is treated as a fixed sparse adjacency for LightGCN-style propagation. At the next epoch, the cache is regenerated from the updated prior router. This changes only the computation schedule; every cached edge is still produced by the content prior rather than by the collaborative posterior.

A.2 Training-Only Collaborative Posterior Teacher

The collaborative posterior is used only during training to provide behavior-aware assignment targets for the content prior. It receives the collaborative context $\mathbf{g}_i$ computed from the training user-item graph and produces a bounded correction to the prior logits as in Eq. (6). The posterior assignment $\boldsymbol{\pi}^q_{i,r}$ is never used to construct item-code graph edges.

The posterior branch has two training roles. First, the teacher calibration loss $\mathcal{L}_{\mathrm{T}}$ calibrates the posterior by aligning code-composed representations with collaborative contexts. Second, the collaborative semantic agreement loss $\mathcal{L}_{\mathrm{CSA}}$, implemented as a KL-based distillation objective, distills the calibrated posterior assignments into the content prior. At inference time, the collaborative context $\mathbf{g}_i$, the posterior correction $\boldsymbol{\Delta}^q_{i,r}$, the posterior assignment $\boldsymbol{\pi}^q_{i,r}$, and the training-only losses $\mathcal{L}_{\mathrm{T}}$ and $\mathcal{L}_{\mathrm{CSA}}$ are removed. Recommendation is performed on the prior-induced user-item-code graph constructed from item content.

A.3 Gradient Flow and Parameter Updates

CASQRec uses different gradient paths for the prior router, posterior teacher, and graph propagation.

Teacher calibration. In Eq. (7), the prior logits $\boldsymbol{\ell}^p_{i,r}$ and codebook $\mathbf{E}_r$ are detached. The collaborative contexts $\mathbf{g}_i$ in Eq. (8) are also detached. Therefore, the teacher calibration loss $\mathcal{L}_{\mathrm{T}}$ updates the posterior correction parameters $\{\mathbf{W}^g_r, \mathbf{b}^g_r, \nu_r\}$ through this branch. It cannot reduce the loss by directly moving the code anchors, changing the prior logits, or modifying the collaborative contexts. This restricted gradient path allows collaborative information to enter through posterior assignment corrections.

Collaborative semantic agreement. In Eq. (9), the posterior distribution $\boldsymbol{\pi}^q_{i,r}$ is detached and used as the teacher target. Thus, $\mathcal{L}_{\mathrm{CSA}}$ updates the content prior router, i.e., the shared trunk ϕ and the assignment heads $\{\psi_r\}$, but does not update the posterior branch through the target. This makes the prior learn to approximate behavior-aware assignments from item content.

Recommendation loss. The BPR loss is optimized on the prior-induced user-item-code graph. The sparse item-code graph is constructed from $\boldsymbol{\pi}^p_{i,r}$ by top-K_c routing, but the discrete selected indices are treated as a graph-construction result and are not differentiated through. When epoch-level caching is used, the cached sparse adjacency is kept fixed within the cache interval and refreshed from the updated prior router in the next refresh. Thus, recommendation training never uses posterior-induced edges and keeps the graph construction rule consistent with inference. The prior router can be updated through the differentiable soft code composition in Eq. (4) and is directly supervised at the assignment level by $\mathcal{L}_{\mathrm{CSA}}$ and $\mathcal{L}_{\mathrm{bal}}$. Therefore, assignment learning does not require gradients through the discrete top-K_c selection, while the sparse graph can still evolve as it is periodically reconstructed from the current prior.

B Analysis of the Posterior Teacher and Prior Graph

This section analyzes why the posterior branch serves as a training-time assignment teacher rather than an alternative inference router. The posterior is bounded around the content prior, trained through a detached code-composed representation path, and used only to supervise the prior assignment mechanism. We also analyze why recommendation graph edges are always constructed from the prior and how edge normalization supports stable code usage.

B.1 Bounded Collaborative Refinement

We formalize the effect of the collaborative posterior correction. For a fixed item i and code position r, let $\boldsymbol{\pi}^p_{i,r} = \mathrm{Softmax}(\boldsymbol{\ell}^p_{i,r})$ and $\boldsymbol{\pi}^q_{i,r} = \mathrm{Softmax}(\boldsymbol{\ell}^p_{i,r} + \boldsymbol{\Delta}^q_{i,r})$, where Eq. (6) defines

$$\boldsymbol{\Delta}^q_{i,r} = h^p_{i,r} \log C \, \sigma(\nu_r) \tanh\left(\mathbf{W}^g_r \, \mathrm{sg}(\mathbf{g}_i) + \mathbf{b}^g_r\right), \tag{21}$$

with $h^p_{i,r} = \mathrm{sg}(H(\boldsymbol{\pi}^p_{i,r})/\log C)$. For $C \geq 2$, the entropy bound $0 \leq H(\pi) \leq \log C$ gives $0 \leq h^p_{i,r} \leq 1$.

Proposition 1. For any item i, code position r, and code c, the posterior correction satisfies

$$\left\|\boldsymbol{\ell}^q_{i,r} - \boldsymbol{\ell}^p_{i,r}\right\|_\infty \leq b_{i,r}, \qquad b_{i,r} = h^p_{i,r} \log C \, \sigma(\nu_r) \leq \log C. \tag{22}$$

Moreover, the posterior and prior assignment probabilities obey

$$\exp(-2b_{i,r}) \leq \frac{\pi^q_{i,r,c}}{\pi^p_{i,r,c}} \leq \exp(2b_{i,r}), \tag{23}$$

and their KL divergence is bounded by

$$\mathrm{KL}\left(\boldsymbol{\pi}^q_{i,r} \middle\| \boldsymbol{\pi}^p_{i,r}\right) \leq 2b_{i,r} \leq 2 \log C. \tag{24}$$

Proof. Since $0 \leq h^p_{i,r} \leq 1$, $0 < \sigma(\nu_r) < 1$, and $\|\tanh(\cdot)\|_\infty \leq 1$, Eq. (21) gives

$$\left\|\boldsymbol{\ell}^q_{i,r} - \boldsymbol{\ell}^p_{i,r}\right\|_\infty = \left\|\boldsymbol{\Delta}^q_{i,r}\right\|_\infty \leq b_{i,r}, \qquad b_{i,r} = h^p_{i,r} \log C \, \sigma(\nu_r) \leq \log C. \tag{25}$$

Let $\Delta_{i,r,c} = \ell^q_{i,r,c} - \ell^p_{i,r,c}$. For any code c, the softmax ratio can be written as

$$\log \frac{\pi^q_{i,r,c}}{\pi^p_{i,r,c}} = \Delta_{i,r,c} - \log \sum_{c'=1}^{C} \pi^p_{i,r,c'} \exp(\Delta_{i,r,c'}). \tag{26}$$

Because $-b_{i,r} \leq \Delta_{i,r,c'} \leq b_{i,r}$ for all c', we have

$$\exp(-b_{i,r}) \leq \sum_{c'=1}^{C} \pi^p_{i,r,c'} \exp(\Delta_{i,r,c'}) \leq \exp(b_{i,r}), \tag{27}$$

where $\sum_{c'=1}^{C} \pi^p_{i,r,c'} = 1$. Thus, the log-normalizer in Eq. (26) lies in $[-b_{i,r}, b_{i,r}]$. Combining this with $-b_{i,r} \leq \Delta_{i,r,c} \leq b_{i,r}$ yields

$$-2b_{i,r} \leq \log \frac{\pi^q_{i,r,c}}{\pi^p_{i,r,c}} \leq 2b_{i,r}. \tag{28}$$

Exponentiating both sides gives the probability-ratio bound. Finally,

$$\mathrm{KL}\left(\boldsymbol{\pi}_{i,r}^{q}\middle\|\boldsymbol{\pi}_{i,r}^{p}\right)=\sum_{c=1}^{C}\pi_{i,r,c}^{q}\log\frac{\pi_{i,r,c}^{q}}{\pi_{i,r,c}^{p}}\leq\sum_{c=1}^{C}\pi_{i,r,c}^{q}\cdot 2b_{i,r}=2b_{i,r}\leq 2\log C. \tag{29}$$

This completes the proof. The bound is uncertainty-adaptive because $b_{i,r}=h_{i,r}^{p}\log C\,\sigma(\nu_r)$ depends on the normalized prior entropy. When the content prior is confident, $H(\boldsymbol{\pi}_{i,r}^{p})$ is small and the posterior can only make a small correction. When the prior is uncertain, the posterior can apply a larger behavior-aware correction. Therefore, the posterior acts as a bounded refinement of the content prior rather than an unconstrained replacement.

B.2 Why Calibration Trains Behavior-Aware Assignments

The teacher calibration loss in Eq. (8) calibrates the posterior branch as a behavior-aware assignment teacher. The calibration distribution $\boldsymbol{\pi}_{i,r}^{q,\mathrm{T}}$ and the posterior assignment $\boldsymbol{\pi}_{i,r}^{q}$ share the same correction form but serve different gradient paths. The former is used only for teacher calibration with detached prior logits and detached code anchors, while the latter is detached as the teacher target in collaborative semantic agreement.

For clarity, assume that $\mathbf{x}_i^q$ and $\mathbf{g}_j$ are ℓ_2-normalized so that $\mathrm{sim}(\mathbf{x}_i^q,\mathbf{g}_j)=\mathbf{x}_i^{q\top}\mathbf{g}_j$. For item i, define the in-batch matching probability

$$\rho_{ij}=\frac{\exp(\mathbf{x}_i^{q\top}\,\mathrm{sg}(\mathbf{g}_j)/\tau)}{\sum_{k\in\mathcal{B}_I}\exp(\mathbf{x}_i^{q\top}\,\mathrm{sg}(\mathbf{g}_k)/\tau)}. \tag{30}$$

The item-wise calibration loss is $\mathcal{L}_{\mathrm{T}}^{i}=-\log\rho_{ii}$. Ignoring the derivative of layer normalization for notational simplicity, its gradient with respect to the posterior code-composed representation is

$$\frac{\partial\mathcal{L}_{\mathrm{T}}^{i}}{\partial\mathbf{x}_i^q}=\frac{1}{\tau}\left(\sum_{j\in\mathcal{B}_I}\rho_{ij}\,\mathrm{sg}(\mathbf{g}_j)-\mathrm{sg}(\mathbf{g}_i)\right). \tag{31}$$

Thus, $\mathcal{L}_{\mathrm{T}}$ moves the code-composed posterior representation toward its own collaborative context and away from other in-batch contexts.

In Eq. (7), the posterior representation is constructed as

$$\mathbf{x}_i^q=\mathrm{LN}\left(\frac{1}{R}\sum_{r=1}^{R}\mathrm{sg}(\mathbf{E}_r)^{\top}\boldsymbol{\pi}_{i,r}^{q,\mathrm{T}}\right),\qquad\boldsymbol{\pi}_{i,r}^{q,\mathrm{T}}=\mathrm{Softmax}\left(\mathrm{sg}(\boldsymbol{\ell}_{i,r}^{p})+\boldsymbol{\Delta}_{i,r}^{q}\right). \tag{32}$$

Both the codebook and the prior logits are detached in this branch. Therefore, calibration cannot be satisfied by moving code embeddings or directly changing the prior logits. It updates the posterior correction $\boldsymbol{\Delta}_{i,r}^{q}$ to produce code distributions whose composed representations match collaborative contexts.

The calibrated posterior is then transferred to the content prior by Eq. (9). For a single item-position pair, the KL-softmax derivative gives

$$\frac{\partial}{\partial\ell_{i,r,c}^{p}}\mathrm{KL}\left(\mathrm{sg}[\boldsymbol{\pi}_{i,r}^{q}]\middle\|\boldsymbol{\pi}_{i,r}^{p}\right)=\pi_{i,r,c}^{p}-\mathrm{sg}[\pi_{i,r,c}^{q}]. \tag{33}$$

Thus, the content prior is directly pulled toward the behavior-aware posterior assignment at the code level, while remaining computable from item content. This mechanism makes CASQRec collaboration-adaptive during training and content-inferable during inference.

B.3 Why Recommendation Graph Edges Are Prior-Based

A key design choice of CASQRec is that item-code edges for recommendation are always constructed from the content prior. Although posterior assignments are available during training, using them to build the training graph would create a train-test routing mismatch under content-only deployment.

Let $\mathbf{S}^p = \Gamma(\{\boldsymbol{\pi}^p_{i,r}\})$ denote the sparse item-code edge matrix produced by top-K_c prior routing, where $\Gamma(\cdot)$ represents the top-K_c selection and normalization in Eq. (10). Similarly, let $\mathbf{S}^q = \Gamma(\{\boldsymbol{\pi}^q_{i,r}\})$ denote the posterior-induced edge matrix. Under the content-only inference constraint, $\mathbf{S}^p$ is deployable because $\boldsymbol{\pi}^p_{i,r}$ depends only on item content, whereas $\mathbf{S}^q$ is not deployable because $\boldsymbol{\pi}^q_{i,r}$ depends on the collaborative context $\mathbf{g}_i$.

Proposition 2. Under content-only deployment, training the recommendation objective on $\mathbf{S}^q$ is graph-inconsistent with inference unless $\mathbf{S}^q = \mathbf{S}^p$. Training on $\mathbf{S}^p$ keeps the recommendation graph construction rule consistent between training and inference.

Proof. If recommendation training used posterior-induced edges, the model would optimize $\mathcal{L}_{\mathrm{BPR}}(\Theta; \mathbf{S}^q)$ and learn representations through graph propagation on $\mathbf{S}^q$. However, at inference time, $\mathbf{g}_i$ and $\boldsymbol{\pi}^q_{i,r}$ are unavailable for content-only item routing, so the deployed scorer must construct item-code edges from $\mathbf{S}^p$. Therefore, the graph used to learn recommendation representations would differ from the graph used for deployment. Unless $\mathbf{S}^q = \mathbf{S}^p$, this creates a train-test routing mismatch.

CASQRec avoids this mismatch by always optimizing the recommendation objective on the prior-induced graph, $\mathcal{L}_{\mathrm{BPR}}(\Theta; \mathbf{S}^p)$. The posterior affects routing only through the agreement loss $\mathcal{L}_{\mathrm{CSA}}$, which pulls the deployable prior assignment toward the behavior-aware posterior assignment. Thus, collaborative behavior can shape the deployed prior router without requiring posterior-based graph edges during recommendation training. The posterior therefore serves as an assignment teacher rather than an alternative graph constructor.

B.4 Edge Normalization and Code Usage

Eq. (10) normalizes selected item-code edges within each code position. For a fixed item i and position r, the selected edge weights satisfy

$$\sum_{c \in \mathcal{N}^K_{i,r}} S_{i,(r,c)} = \frac{1}{R} \frac{\sum_{c \in \mathcal{N}^K_{i,r}} \pi^p_{i,r,c}}{\sum_{c' \in \mathcal{N}^K_{i,r}} \pi^p_{i,r,c'} + \epsilon} \approx \frac{1}{R}. \tag{34}$$

Therefore, the total item-to-code weight over all positions is approximately $\sum_{r=1}^{R} \sum_{c \in \mathcal{N}^K_{i,r}} S_{i,(r,c)} \approx 1$. The factor $1/R$ prevents items with more code positions from having larger total outgoing edge weights and keeps different positions on a comparable scale. The balance regularizer

$$\mathcal{L}_{\mathrm{bal}} = \sum_{r=1}^{R} \sum_{c=1}^{C} \left(\frac{1}{|\mathcal{B}_I|} \sum_{i \in \mathcal{B}_I} \pi^p_{i,r,c} - \frac{1}{C} \right)^2 \tag{35}$$

discourages routing collapse. Without this term, the prior router may assign most items to a small subset of high-probability codes, reducing the number of effective shared variables. The regularizer does not force each individual item to use codes uniformly; it only encourages the batch-level marginal distribution at each position to use the codebook broadly. Thus, individual assignments can remain selective while the overall semantic-code graph avoids degeneracy.

C Complexity and Training Procedure

Fixed descriptor transform. Let D be the dimension of the concatenated descriptor before PCA. Estimating modality statistics costs $O(|\mathcal{I}_{\mathrm{tr}}|D)$, and constructing pre-PCA descriptors for all items costs $O(|\mathcal{I}|D)$. Fitting PCA to output d dimensions is an offline preprocessing step; with truncated PCA, it costs $O(|\mathcal{I}_{\mathrm{tr}}|Dd)$. After fitting, computing fixed descriptors for all items costs $O(|\mathcal{I}|Dd)$. The PCA parameters are fixed during recommendation training and inference.

RQ initialization. Let T_k denote the number of K-means iterations. Fitting R residual stages with C anchors per stage costs $O(T_k|\mathcal{I}_{\mathrm{tr}}|RCd)$, where d is the descriptor dimension. This is an offline initialization step, and the resulting anchors are fixed during recommendation training.

Algorithm 1 Training CASQRec

Require: Training user-item graph $\mathcal{G}_{ui}$, item multimodal features $\{\mathbf{f}_i^m\}_{m\in\mathcal{M}}$, RQ depth R, codebook size C, top-K_c routing sparsity.

Fit the fixed descriptor transform $\mathcal{T}_{\mathrm{pca}}$ on training items and compute $\bar{\mathbf{f}}_i$ by Eq. (1).
Fit RQ anchors $\{\mathbf{e}_{r,c}\}_{r=1,c=1}^{R,C}$ on training descriptors by Eq. (2); keep the anchors fixed.
for each training epoch **do**
  Compute prior logits and distributions $\boldsymbol{\ell}_{i,r}^p, \boldsymbol{\pi}_{i,r}^p$ by Eq. (3).
  Build prior item representations $\mathbf{x}_i^0$ by Eq. (4).
  Compute collaborative contexts $\mathbf{g}_i$ from the training user-item graph by Eq. (5).
  Compute posterior corrections and assignments $\boldsymbol{\Delta}_{i,r}^q, \boldsymbol{\pi}_{i,r}^q$ by Eq. (6).
  Calibrate the posterior teacher with $\mathcal{L}_{\mathrm{T}}$ by Eq. (8).
  Distill posterior assignments into the prior with $\mathcal{L}_{\mathrm{CSA}}$ by Eq. (9).
  Compute the balance regularizer $\mathcal{L}_{\mathrm{bal}}$ by Eq. (11).
  Construct prior-induced item-code edges by Eq. (10); optionally cache the resulting graph for the current epoch.
  Propagate on $\mathcal{G}_{uic}$ and compute scores by Eq. (12).
  Sample BPR triples (u, i, j) and compute $\mathcal{L}_{\mathrm{BPR}}$.
  Optimize the full objective in Eq. (13).
end for
During inference, remove the posterior branch and all training-only losses.
Compute fixed descriptors, route items with the content prior, construct top-K_c item-code edges, propagate on the prior-induced graph, and compute recommendation scores.

Prior routing and sparse edge construction. For all items and all code positions, computing prior logits with linear assignment heads costs $O(|\mathcal{I}|RCd)$, up to the cost of the shared routing trunk. Top-K_c selection costs $O(|\mathcal{I}|RC)$ with linear selection, and produces at most $|\mathcal{I}|RK_c$ item-code edges. The sparse item-code matrix therefore requires $O(|\mathcal{I}|RK_c)$ storage. If edge caching is adopted, routing and top-K_c selection are performed once per refresh, and the cached graph is used for propagation within the refresh interval.

Collaborative context and posterior teacher. Computing collaborative contexts on the training user-item graph costs $O(L_c|\mathcal{E}_{ui}|d)$, where L_c is the number of context propagation layers. For an item batch $\mathcal{B}_I$, computing posterior corrections and posterior code distributions costs $O(|\mathcal{B}_I|RCd)$ for linear correction heads. The in-batch teacher calibration loss costs $O(|\mathcal{B}_I|^2 d)$ for pairwise similarities. These posterior-related computations are used only during training and are removed at inference.

Graph propagation. The prior-induced user-item-code graph contains $|\mathcal{E}_{ui}| + |\mathcal{I}|RK_c$ edges. With L LightGCN-style propagation layers, graph propagation costs $O(L(|\mathcal{E}_{ui}| + |\mathcal{I}|RK_c)d)$.

Inference. At inference time, CASQRec removes the collaborative posterior, teacher calibration, and CSA loss. Item routing uses only the content prior, with cost $O(|\mathcal{I}|RCd)$ for prior logits and $O(|\mathcal{I}|RC)$ for top-K_c selection. Each item connects to at most RK_c semantic-code nodes, yielding $O(|\mathcal{I}|RK_c)$ sparse item-code edges. Inference graph propagation costs $O(L(|\mathcal{E}_{ui}| + |\mathcal{I}|RK_c)d)$ and does not require posterior collaborative contexts for item routing.

D Experimental Details

D.1 Datasets

We conduct experiments on three publicly available multimodal recommendation datasets: Baby, Sports, and TikTok. These datasets contain user-item interactions and item-side multimodal features, and are widely used for evaluating multimodal recommendation models. The Amazon-Baby and Amazon-Sports datasets are selected from the Amazon product repository, corresponding to the Baby and Sports and Outdoors product categories, respectively. Each Amazon item is associated with product images and textual descriptions. To ensure the reliability of collaborative signals, we apply the 5-core filtering strategy to filter out users and items with fewer than five interactions. For model

Table 4: Statistics of the experimental datasets with multimodal item Visual (V), Acoustic (A), and Textual (T) contents.

Dataset	TikTok			Amazon-Baby		Amazon-Sports	
Modality	V	A	T	V	T	V	T
Embed Dim	128	128	768	4096	1024	4096	1024
User	9,319			19,445		35,598	
Item	6,710			7,050		18,357	
Interactions	59,541			139,110		256,308	
Sparsity	99.904%			99.899%		99.961%	

training, each dataset is randomly partitioned into training, validation, and testing sets with a ratio of 8:1:1.

- **Baby** is an Amazon product recommendation dataset from the baby product category. Each item is associated with product images and textual descriptions. Following common multimodal recommendation settings, we use 4096-dimensional visual features extracted from product images and 1024-dimensional textual features encoded from product titles and descriptions by Sentence-BERT. After filtering, the dataset contains 19,445 users, 7,050 items, and 139,110 interactions, with a sparsity of 99.899%.
- **Sports** is an Amazon product recommendation dataset from the sports product category. Similar to Baby, each item contains visual and textual information. We use 4096-dimensional visual embeddings extracted from product images and 1024-dimensional Sentence-BERT textual embeddings generated from product titles and descriptions. After filtering, the dataset contains 35,598 users, 18,357 items, and 256,308 interactions, with a sparsity of 99.961%.
- **TikTok** is a short-video recommendation dataset that captures user interactions with short-form video content, encompassing rich visual, acoustic, and textual features. The visual and acoustic features are 128-dimensional, and the textual features are 768-dimensional. The textual embeddings are generated from video captions and comments using Sentence-BERT. The dataset contains 9,319 users, 6,710 items, and 59,541 interactions, with a sparsity of 99.904%.

The statistics of the three datasets are summarized in Table 4. These datasets are highly sparse, which makes them suitable for evaluating the ability of recommendation models to exploit multimodal content and shared semantic structures.

D.2 Baselines

We compare CASQRec with ten published methods and one combined variant spanning collaborative filtering, discrete-code-based recommendation, and multimodal recommendation. All baselines are evaluated under the same data split, candidate ranking protocol, and validation-based model selection.

- **LightGCN** [4] is a strong collaborative filtering baseline that performs graph propagation on the user-item interaction graph without feature transformation or nonlinear activation.
- **CCFRec** [16] is a semantic-code-based recommendation method that bridges content information and collaborative signals through discrete semantic codes.
- **DQRec** [21] learns quantized user and item representations to augment collaborative filtering.
- **MMGCN** [28] is a graph-based multimodal recommendation method that learns modality-specific user and item representations through graph convolution.
- **LATTICE** [38] constructs latent item-item structures from multimodal features and uses them to enhance user-item recommendation.
- **FREEDOM** [40] improves multimodal recommendation by freezing item-item graphs and denoising the user-item interaction graph.

- **BM3** [41] is a self-supervised multimodal recommendation method that learns robust user and item representations through bootstrap-style representation learning.
- **MENTOR** [32] is a recent multimodal recommendation method that performs multi-level cross-modal alignment for representation learning.
- **SG-URInit** [33] builds on MENTOR and uses semantically guaranteed user representation initialization to incorporate item-level and cluster-level information into modality-specific user representations.
- **MoToRec** [17] is a multimodal recommendation method designed for item cold start. It learns compositional tokens from multimodal item features through sparse-regularized tokenization and integrates these tokens with collaborative signals through a graph encoder.
- **MoToRec-SGU** is our variant of MoToRec [17] that applies the user initialization scheme of SG-URInit [33] to MoToRec's modality-specific user representations. It retains MoToRec's tokenizer, graph encoder, fusion mechanism, and training objectives.

The selected baselines include both discrete-code-based methods and strong multimodal graph or self-supervised recommenders, allowing us to examine whether CASQRec improves over fixed code assignments, continuous multimodal representation learning, and recent semantic user initialization. Recent semantic-ID-based generative recommenders, such as TIGER, HSTU, ActionPiece, RPG, and DiffGRM, mainly target sequential or generative retrieval settings with different input formats, objectives, and evaluation protocols. Therefore, they are discussed in related work but are not included in the main top-K multimodal recommendation comparison.

D.3 Evaluation Metrics and Compute Environment

Following common top-K recommendation settings, we evaluate all methods with Recall@K and NDCG@K. We report results at $K = 10$ and $K = 20$ and average the metric values over all test users. All experiments we conducted are performed with Intel(R) Xeon(R) Bronze 3204 CPU @ 1.90GHz, Tesla A100, and 256GB memory, running the Ubuntu 20.04.4 LTS operating system.

D.4 Implementation Details

We implement CASQRec with PyTorch and optimize it with Adam. The embedding size is fixed to 64, and trainable embeddings are initialized with Xavier initialization. For CASQRec, the main recommendation objective is optimized with the Bayesian Personalized Ranking loss. For each positive interaction, one negative item is sampled from items that the user has not interacted with in the training set. The batch size is selected from $\{1024, 2048, 4096\}$ according to validation performance. The maximum number of training epochs is set to 1000, and early stopping is adopted with a patience of 20 epochs. We select the best checkpoint according to NDCG@20 on the validation set. During evaluation, we rank all candidate items that the user has not interacted with in the training set. For baselines, we follow their original training objectives and tune key hyperparameters on the validation set under the same data split and evaluation protocol. All experiments are repeated with five random seeds, and the average results are reported.

E Limitations and Future Work

CASQRec establishes a content-inferable and collaboration-adaptive semantic-code framework for multimodal recommendation. We study offline top-K recommendation in a setting where item content is available at inference. Our evaluation covers both a random interaction split and an item-level holdout in which test items have no training interactions. In both settings, the router assigns items to a codebook fitted on training items. Its ability to represent new content therefore depends on the coverage of the learned codebook. Several directions remain for future work. First, CASQRec uses fixed content-derived RQ anchors to ensure stable and inference-consistent routing. Richer code-space constructions, such as hierarchical codes, expandable codebooks, or domain-adaptive initialization, may further improve semantic expressiveness while preserving deployable prior-based routing. Second, CASQRec relies on item-side multimodal features for content-inferable assignment. Future studies may improve robustness under noisy, incomplete, or weakly informative modalities through uncertainty-aware modality weighting or stronger missing-modality modeling.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are aligned with contributions and scope. All claims match theoretical and experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We discuss the limitations in the appendix.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper provides the assumptions and derivations for the theoretical analysis in the appendix, including the bounded posterior correction, posterior-prior probability ratio, and KL divergence bound. These results follow from the bounded tanh correction, the uncertainty-dependent scaling term, and standard softmax normalization.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

Justification: Yes, all experimental settings, codes, pseudo-code and datasets are provided in paper and supplementary materials.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes, all experimental settings, codes, and datasets are provided in paper and supplementary materials.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper reports the data splits, cold-start protocol, evaluation setting, optimizer and training configuration, hyperparameter search ranges, random-seed setting. It also specifies that the fixed descriptor transform and RQ anchors are fitted only on training items and provides the complete training and inference procedure in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We retrained CASQRec five times and computed p-values.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

Justification: The paper reports the compute environment used for experiments, including GPU/CPU resources, memory, and running-time information. It also provides complexity analysis and distinguishes offline descriptor/RQ initialization from training-time and inference-time computation.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We carefully reviewed the Code of Ethics and confirmed that we have adhered to each one.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss in the appendix. Our paper uses real-world, publicly available datasets that do not raise ethical impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: All models and datasets used in this study are publicly accessible.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the creators or original owners of assets are properly credited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.