OneRec Technical Report (OneRec-V1) (2025)

arXiv:2506.13695v4 (cs.IR), 16 Sep 2025.

Challenges

- **Fragmented compute in cascaded pipelines:** significant serving resources go to communication/storage (not compute), and ranker GPU utilization is low (the paper cites training MFU 4.6% and inference MFU 11.2% for a cascaded system).
- **Objective collision:** hundreds of competing objectives (users/creators/platform) introduce inconsistent optimizations and "patching" across stages.
- **Lag behind mainstream AI:** cascaded architectures make it hard to adopt scaling laws + RL techniques that have worked well for LLM/VLM systems.

Key Initiatives

- **End-to-end generative recommendation:** unify retrieve+rank into a single generative model outputting item semantic identifiers.
- **Scaling up recommendation compute:** increase model FLOPs (paper claims 10×) and empirically study scaling behaviors across parameters, features, codebook size, and inference sampling (Pass@K).
- **Preference alignment via RL:** learn a personalized reward (P-Score) and optimize generation with a stabilized RL variant (ECPO), plus format reward to maintain legality.
- **Systems/infra optimization:** push training+inference MFU to LLM-like levels and reduce overall operating expense.

Methodology

- **Tokenizer + semantic IDs:** build collaborative multimodal item representations from item pairs, then quantize into discrete semantic identifiers using RQ-Kmeans (coarse-to-fine; the paper uses $L_t = 3$ layers).
- **Multimodal inputs:** caption/tag/ASR/OCR + cover + frames; processed with miniCPM-V-8B, compressed by a QFormer; trained with an item-to-item contrastive loss plus a caption generation loss (to "prevent hallucination").
- **Encoder–decoder model (MoE):** encoder ingests multi-scale user behavior features (static, short-term, positive feedback, lifelong history with hierarchical compression up to 100k items); decoder autoregressively generates semantic IDs. Figure 1 shows the high-level encoder–decoder layout.
- **Reward system:** combines (1) Preference Reward (P-Score, learned fusion over many engagement objectives), (2) Format Reward (legality of semantic-ID sequences), and (3) Industrial rewards (ecosystem/business constraints).
- **ECPO (Early-Clipped GRPO):** modifies GRPO by early clipping large policy ratios for negative advantages to improve stability; the paper removes KL loss because RL and SFT are trained jointly.

Experiments

- **Online deployment impact:** "Deployed in Kuaishou/Kuaishou Lite APP, it handles 25% of total queries per second (QPS), enhancing overall App Stay Time by 0.54% and 1.24%, respectively."
- **Efficiency claims:** the paper reports "23.7% and 28.8% Model FLOPs Utilization (MFU)" for training and inference, and "operating expense (OPEX) that is only 10.6% of traditional recommendation pipelines."
- **Parameter scaling:** OneRec models from 0.015B to 2.633B show lower NTP loss with larger size; the paper notes the first ~10B samples yield rapid convergence, with slower gains beyond.
- **Feature scaling:** adding the full feature set substantially improves multiple reward metrics (e.g., P-Score +28.88% in their Table 2).
- **Codebook + inference scaling:** larger codebooks (8k $\rightarrow$ 32k) and larger Pass@K both improve reward metrics; they select Pass@512 for production as a cost/perf trade-off.
- **Semantic-ID input:** using semantic identifiers for history input achieves comparable performance to sparse vid embeddings at 2.6B scale, with benefits in parameter/communication efficiency.

References (selected; cited/used by the paper)

- SIM (Search-based Interest Model): Pi et al., 2020.
- QFormer: Li et al., 2023.
- miniCPM-V-8B: Hu et al., 2024.
- LLaMA3 (caption decoding objective): Dubey et al., 2024.
- RQ-Kmeans (tokenization): Luo et al., 2024. (Compared with RQ-VAE: Lee et al., 2022.)
- GRPO / loss-free MoE load balancing: Liu et al., 2024 (also referenced via DeepSeek).

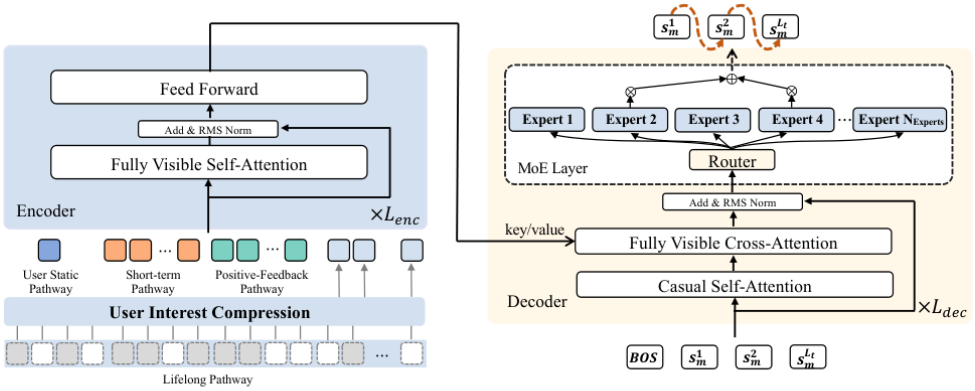

Figure 4 | Illustration of our encoder-decoder architecture.

2.1.2. Tokenization

We utilize RQ-Kmeans (Luo et al., 2024) for tokenization, which employs residual quantization to generate semantic IDs in a coarse-to-fine manner. This method constructs codebooks by applying K-means clustering directly on the residuals. An illustration of the RQ-Kmeans process is provided in Figure 3 (right).

Formally, the initial residual at layer $l = 1$ is defined as:

$$\mathcal{R}^{(1)} = \left\{\tilde{\mathbf{M}}_i \in \mathbb{R}^{N_{\tilde{M}} \times d_t} \mid \forall \text{ video } i\right\}. \tag{5}$$

For each layer l, the codebook $C^{(l)}$ is derived from K-means centroids of $\mathcal{R}^{(l)}$:

$$C^{(l)} = \text{K-means}\left(\mathcal{R}^{(l)}, N_t\right), \tag{6}$$

where $C^{(l)} = \left\{\boldsymbol{c}_k^{(l)} \in \mathbb{R}^{N_{\tilde{M}} \times d_t} \mid k = 1, \ldots, N_t\right\}$ and N_t is the codebook size. The nearest centroid index for item i is computed as:

$$s_i^l = \arg\min_k \left\|\mathcal{R}_i^{(l)} - \boldsymbol{c}_k^{(l)}\right\|, \tag{7}$$

where $\|\cdot\|$ denotes the Euclidean norm. The residual of video i for layer $l+1$ is then updated:

$$\mathcal{R}_i^{(l+1)} = \mathcal{R}_i^{(l)} - \boldsymbol{c}_{s_i^l}^{(l)}. \tag{8}$$

This quantization iterates across $L_t = 3$ layers.

As demonstrated in Section 4.4, RQ-Kmeans offers enhanced reconstruction quality, better codebook utilization, and improved balance compared to the widely used RQ-VAE (Lee et al., 2022; Rajput et al., 2024). At this stage, each video m can be represented by L_t coarse-to-fine semantic identifiers: $\{s_m^1, s_m^2, \ldots, s_m^{L_t}\}$, which will serve as the output of the OneRec recommendation system, enabling progressive item generation.

2.2. Encoder

2.2.1. Multi-Scale Feature Engineering

This section presents the feature engineering component of OneRec. We process user behavior data through four specialized embedding pathways, each designed to capture distinct scales of user