DECOUPLED MULTI-CODEBOOK QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-codebook quantization commonly ties code selection to codebook geometry, although individually nearest codewords need not form the best joint representation. We propose Decoupled Multi-Codebook Quantization (DMQ), which separates what codewords represent from how they are selected. A learned router coordinates assignments without input–codeword distance matching, unifying parallel and residual quantization. Differentiable training jointly optimizes assignments and codebooks while accommodating downstream objectives. We establish the potential suboptimality of independent Euclidean assignment, prove strictly greater assignment expressivity under suitable capacity assumptions at unchanged code budgets, and derive a sufficient condition for initialization-independent global convergence of Iterated Conditional Modes. Experiments on four million-scale benchmarks demonstrate improvements over representative baselines in reconstruction and retrieval quality, while maintaining efficient inference. These results establish decoupled assignment as an expressive and effective design principle for learning compact discrete representations. Our implementation is anonymously provided at `https://anonymous.4open.science/r/DMQ`.

1 INTRODUCTION

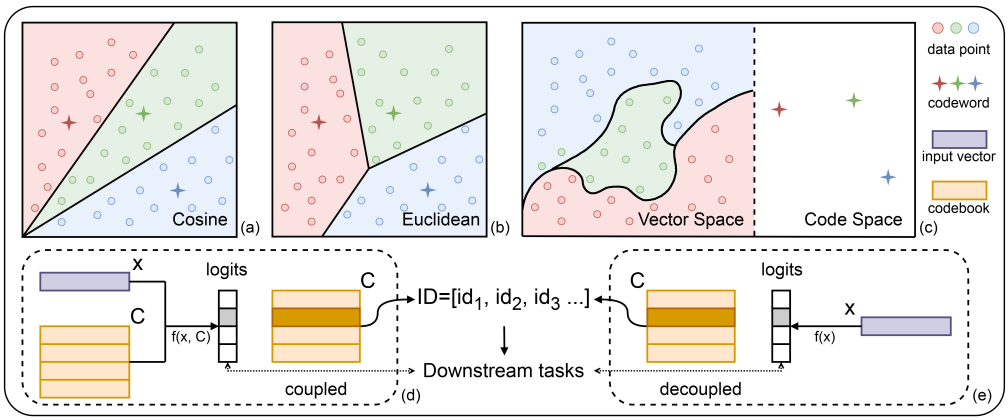

Figure 1: **Geometric vs. decoupled code assignment.** (a) Cosine similarity and (b) Euclidean distance assign codes by geometry. (c) DMQ separates vector space from code space. (d) Conventional MCQ couples assignment with codebooks via $f(x, C)$. (e) DMQ decouples them: $f(x)$ predicts IDs directly, and codebooks serve only as lookup tables.

Vector quantization (VQ) provides a compact representation of high-dimensional vectors and has been widely adopted across a broad range of applications, including large-scale vector retrieval (Vallaeys et al., 2025; Zhong et al., 2026), image compression and generation (Sun et al., 2024; Lee et al., 2022), and recommendation (Liu et al., 2024; Bin et al., 2025). Multi-codebook quantization (MCQ) further improves the expressive capacity of discrete representations by composing multiple codewords from a collection of codebooks as the vector representation. This principle underlies a broad family of quantization methods, including product quantization (PQ; Jégou et al., 2011a), residual quantization (RQ; Chen et al., 2010), and additive quantization (AQ; Babenko & Lempitsky, 2014), and has become an important paradigm for compact vector representation and large-scale search.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

A common design for MCQ is to determine code assignments according to the geometric relationship between the input vector and codewords. Depending on the quantization scheme, this relationship can be measured by Euclidean distance, inner product, or other geometry-specific metrics (Wang et al., 2015; Guo et al., 2016; Morozov & Babenko, 2019; Piekos et al., 2025), as depicted in Fig. 1(a) and (b). Hence, the geometry of the codebooks directly determines which codewords are selected, coupling code assignment with the representation space of the codebooks (Fig. 1(d)).

This *geometric coupling*, however, is not necessarily required for effective discrete representation learning. In MCQ, the utility of an individual codeword depends on the codewords selected from other codebooks, reflecting dependencies among different parts of the discrete representation (Chen et al., 2010; Martinez et al., 2018; Huijben et al., 2024; Zhong et al., 2026). A codeword that is geometrically preferred in isolation may therefore not yield the best overall composition (Sec. 4.6 provides the theoretical analysis). More importantly, the reconstruction error, frequently used in MCQ, is often only an intermediate surrogate for downstream applications. Some works have shown that improving reconstruction fidelity does not necessarily translate into better downstream generation quality (Gu et al., 2024). For retrieval and recommendation, a desirable discrete representation should instead preserve neighborhood structure or task-specific relevance (Guo et al., 2020; Zhan et al., 2021). These observations suggest that code assignment and codebook geometry serve fundamentally different roles. Code assignment determines which codewords should be composed, whereas codebook geometry defines the continuous representation embodied by those codewords. **Coupling code assignment and codebook geometry through the same geometric criterion therefore unnecessarily constrains the learning of discrete representations.**

Neural approaches have also been explored for code assignment, replacing explicit distance matching with learned prediction (Chen et al., 2020; Zhu et al., 2023). However, these methods remain tied to specific quantization structures or assignment objectives. This motivates treating code assignment as an independent component that can be learned separately from codebook geometry.

To this end, we propose *Decoupled Multi-Codebook Quantization* (DMQ), a multi-codebook quantization framework that decouples code assignment from codebook geometry. DMQ learns code assignments through a dedicated routing mechanism rather than distances or similarities to codewords, while codebooks are independently optimized to represent the underlying vectors. As shown in Fig. 1(e), this separation allows the assignment process to focus on the composition of discrete codes without being constrained by the geometry of individual codewords. The same principle naturally applies to both parallel and residual multi-codebook quantization, providing a flexible framework for learning discrete representations.

DMQ is agnostic to the underlying composition scheme. In the parallel setting, assignments can be jointly determined by allowing information to flow across codebooks, while in the residual setting, subsequent assignments can recursively condition on the residuals produced by preceding stages. Thus, DMQ naturally extends to different multi-codebook structures while preserving the same decoupled assignment principle.

Our main contributions are summarized as follows:

- **Beyond Geometric Code Assignment.** We introduce DMQ, separating what codewords represent from how they are selected. A dedicated routing network learns code selection from the input and cross-codebook context, freeing assignment from direct input–codeword distance matching.
- **One Principle, Two Quantization Structures.** We unify parallel and residual quantization under the same decoupled assignment principle, supporting both joint code selection and progressive residual refinement. The differentiable formulation allows both reconstruction and downstream objectives to guide assignment and codebook learning.
- **Theoretical Foundations of Decoupled Assignment.** We show that independent Euclidean assignment can be suboptimal and prove that decoupling strictly expands assignment expressivity at the same code budget under stated capacity assumptions. We further establish a margin-based sufficient condition for Iterated Conditional Modes (ICM) to reach the unique global optimum within one sweep from any initialization.
- **Performance Gains.** We evaluate DMQ on million-scale benchmarks and demonstrate improvements over representative baselines in both reconstruction and retrieval quality. Both parallel and residual variants achieve competitive performance while maintaining efficient inference.

2 Related Work

Multi-Codebook Quantization (MCQ). MCQ can be broadly categorized into parallel and residual quantization according to how codewords are composed. Parallel methods select codewords from multiple codebooks to represent the input directly, whereas residual methods sequentially select codewords based on the residual produced by preceding stages. PQ (Jégou et al., 2011a) partitions the feature space into subspaces and quantizes each subvector independently, while AQ (Babenko & Lempitsky, 2014) represents a vector as the sum of codewords from multiple full-dimensional codebooks. RQ (Chen et al., 2010) follows the residual paradigm, where each stage quantizes the residual left by previous stages. Subsequent methods such as LSQ (Martinez et al., 2018) improve the optimization and encoding of additive multi-codebook representations through local search.

Learned Assignment and Neural Quantization. Neural networks have been introduced to learn code assignments in MCQ. UNQ (Morozov & Babenko, 2019) adopts an encoder-decoder architecture and uses inner product for assignment. DPQ (Chen et al., 2020) directly predicts code assignments for independently partitioned subspaces and decouples the key and value representations, but the assignment is still determined by geometric similarity between queries and product keys. DeepQ (Zhu et al., 2023) learns assignments for multiple codebooks with a neural network. Note that DeepQ still couples assignment learning with reconstruction through an EM procedure, where the codebook update is derived from the MSE objective. Other methods, such as QINCo (Huijben et al., 2024) and RQ-MoE (Zhong et al., 2026), primarily introduce neural networks to construct input-dependent codebooks, while code assignment remains based on geometric matching.

Compared to related work, DMQ learns assignments jointly across codebooks without requiring a predefined geometric matching rule or subspace decomposition. This decoupling allows the assignment mechanism to capture cross-codebook dependencies and provides greater flexibility in optimizing discrete representations toward different objectives.

3 Preliminaries

Let $x \in \mathbb{R}^d$ denote an input vector. MCQ represents x using M codebooks:

$$\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_M\}, \qquad \mathcal{C}_m = \{e_{m,1}, \ldots, e_{m,K}\}, \tag{1}$$

where $e_{m,k} \in \mathbb{R}^d$ denotes the k-th codeword of the m-th codebook. A discrete assignment selects one codeword from each codebook, indexed by $k_m \in \{1, \ldots, K\}$. The quantized representation is obtained by additive composition:

$$\hat{x} = \sum_{m=1}^{M} e_{m,k_m}. \tag{2}$$

The resulting representation requires $M \log_2 K$ bits per vector and provides K^M possible code combinations.

Different MCQ methods impose different structural constraints on the codebooks. PQ partitions the vector into subspaces and quantizes each subvector independently, while RQ successively quantizes the residual left by previous stages. AQ uses full-dimensional codebooks and reconstructs the vector by summing the selected codewords.

In conventional MCQ, code assignment is typically determined by the geometric relationship between an assignment input z_m and candidate codewords, where $z_m = x$ for parallel quantization and $z_m = r_{m-1}$ (with $r_0 = x$) for residual quantization. The selected codeword is given by

$$k_m = \arg\min_k d(z_m, e_{m,k}), \tag{3}$$

where $d(\cdot, \cdot)$ denotes a predefined distance or similarity measure. Thus, code assignment is directly tied to the geometry of the codebook.

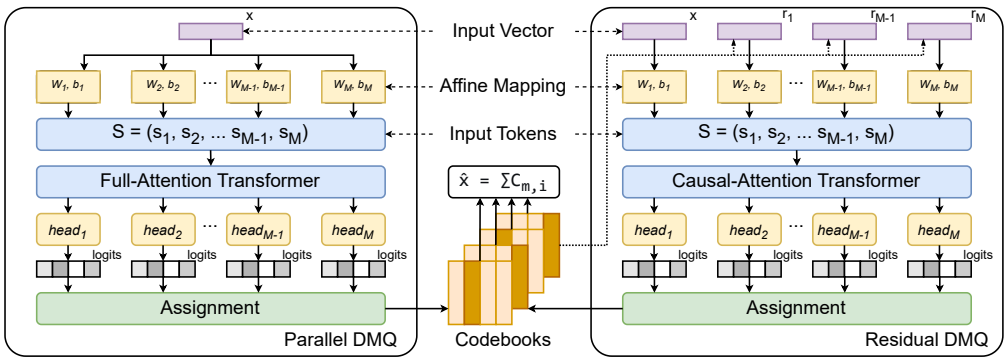

Figure 2: **Architecture of DMQ:** (1) Left: Parallel DMQ assigns all code indices simultaneously from the input x using full attention. (2) Right: Residual DMQ generates code indices autoregressively, where each stage takes the current residual r_{m-1} as input and uses causal attention over previous assignments. Both modes share the same Transformer backbone, and differ only in attention masking and input state. The predicted assignments are used to retrieve codewords from the codebooks, whose additive combination yields the reconstruction $\hat{x}$.

4 Our Method DMQ

4.1 Overview

DMQ decouples code assignment from codebook geometry by replacing geometric matching with a learned assignment mechanism. Given M independently parameterized codebooks $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_M\}$, DMQ predicts one code index for each codebook *without accessing codeword values or computing input–codeword distances*.

The assignment network uses a shared Transformer and FFN backbone with position-specific input projections and prediction heads. As illustrated in Fig. 2, this design supports both parallel and residual quantization by varying the input state and attention structure, while keeping the assignment mechanism independent of codebook geometry.

4.2 Decoupled Code Assignment

The key idea of DMQ is to separate the decision of which codewords to select from the geometry of the codebooks. To achieve this, we learn the code assignment directly from the assignment inputs rather than measuring their geometric relationships with candidate codewords. Specifically, let s_m denote the assignment token at position m, constructed from the corresponding assignment input as

$$s_m = W_m z_m + b_m, \tag{4}$$

where W_m is a position-specific projection and b_m is a learnable position embedding. The position-specific parameters preserve the correspondence between each token and its codebook, while allowing all positions to share the same subsequent Transformer and FFN backbone.

Given the token sequence $S = (s_1, \ldots, s_M)$, the shared Transformer produces

$$H = \text{Transformer}(S), \qquad H = (h_1, \ldots, h_M). \tag{5}$$

A shared FFN followed by a position-specific prediction head then produces the assignment logits

$$Z_m = \text{Head}_m\left(\text{FFN}(h_m)\right), \qquad Z_m \in \mathbb{R}^K. \tag{6}$$

The shared backbone allows information to be exchanged across codebooks, enabling the assignment mechanism to model dependencies among different code selections, while the position-specific heads retain the distinct roles of individual codebooks.

Since discrete code selection is non-differentiable, applying argmax during training would prevent gradients from flowing through the process. We therefore use a soft assignment distribution

$$p_m = \text{softmax}\left(Z_m/\tau\right), \tag{7}$$

and represent the selected codeword by its probability-weighted combination

$$\tilde{e}_m = \sum_{k=1}^{K} p_m(k) e_{m,k}, \qquad \hat{x} = \sum_{m=1}^{M} \tilde{e}_m. \tag{8}$$

This formulation makes both the assignment network and codebooks jointly differentiable through the reconstruction objective. We use cosine annealing to gradually decrease the temperature τ throughout training, encouraging progressively sharper assignment distributions while maintaining differentiability in the early stages. Hard codeword selection is applied only at inference time.

Importantly, the codeword values are excluded from the assignment process. DMQ predicts Z_m solely from the assignment inputs and does not compute distances or similarities between z_m and candidate codewords. This separation allows the assignment network to learn which codewords should be composed based on the input and cross-codebook dependencies, while the codebooks independently learn the representation primitives needed for reconstruction or downstream objectives.

4.3 Assignment Structures

MCQ methods can be broadly categorized into parallel and residual quantization according to their assignment dependencies. Parallel quantization does not impose an explicit ordering among codebooks, allowing all assignments to be jointly determined. This flexible composition is desirable when no particular ordering is required. However, in some domains, the sequential structure of residual quantization can provide a hierarchical organization of discrete representations that is beneficial for downstream tasks. **DMQ supports both structures while preserving the same decoupled assignment principle.**

4.3.1 Parallel DMQ

In parallel DMQ, all codebooks are assigned directly from the input without imposing an explicit ordering among them:

$$s_m = W_m x + b_m, \qquad m = 1, \ldots, M. \tag{9}$$

The resulting tokens are processed jointly by the shared Transformer:

$$H = \mathrm{Transformer}_{\mathrm{full}}(s_1, \ldots, s_M). \tag{10}$$

Full attention allows each position to access the representations of all other codebooks before making its assignment. This enables the routing network to capture dependencies among code selections while avoiding a predefined sequential dependency among codebooks.

4.3.2 Residual DMQ

While parallel quantization provides flexible joint composition, it does not explicitly encode a sequential or hierarchical relationship among codebooks. Residual quantization provides such a structure by progressively refining the representation through successive residuals. To retain this useful structure while preserving the decoupling between assignment and codebook geometry, DMQ can instead perform assignments autoregressively over residual states.

Starting from $r_0 = x$, each stage takes the current residual as its assignment input:

$$s_m = W_m r_{m-1} + b_m. \tag{11}$$

The current token is appended to the previously generated tokens, forming $S_m = (s_1, \ldots, s_m)$. A causal Transformer then processes the sequence:

$$H_m = \mathrm{Transformer}_{\mathrm{causal}}(S_m), \qquad h_m = H_m[-1]. \tag{12}$$

Only the representation of the current token is used to predict the assignment at stage m:

$$Z_m = \mathrm{Head}_m\left(\mathrm{FFN}(h_m)\right), \qquad p_m = \mathrm{softmax}\left(Z_m/\tau\right). \tag{13}$$

The soft assignment is then used to update the residual:

$$r_m = r_{m-1} - \sum_{k=1}^{K} p_m(k) e_{m,k}. \tag{14}$$

The updated residual is used to construct the next assignment token, yielding the recursive process

$$r_{m-1} \to s_m \to p_m \to r_m \to s_{m+1}.$$

Thus, residual DMQ preserves the sequential and hierarchical structure of residual quantization while decoupling code assignment from codebook geometry.

4.4 TRAINING

To prevent codeword collapse and encourage effective codebook utilization, we apply a load-balancing loss throughout training. Given raw routing logits $\mathbf{Z} \in \mathbb{R}^{B\times M\times K}$, we define the hard assignment frequency

$$\hat{k}_{b,m} = \arg\max_k Z_{b,m,k}, \qquad f_{m,k} = \frac{1}{B}\sum_b \mathbb{I}[\hat{k}_{b,m} = k], \tag{15}$$

where $f_{m,k}$ is detached. The mean soft routing probability is

$$\bar{p}_{m,k} = \frac{1}{B}\sum_b \mathrm{softmax}(\mathbf{Z}_{b,m})_k. \tag{16}$$

The load-balancing loss is

$$\mathcal{L}_{\mathrm{bal}} = \frac{K}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} f_{m,k}\bar{p}_{m,k}. \tag{17}$$

Uniform routing yields $\mathcal{L}_{\mathrm{bal}} = 1$, whereas concentrated routing results in larger values. The loss is computed from the raw routing logits and is independent of Gumbel noise and the temperature.

4.4.1 SUPERVISED ASSIGNMENT INITIALIZATION

Random initialization of the assignment network and codebooks can lead to poor codebook utilization. We initialize DMQ from conventional quantizers to provide a favorable starting point.

For parallel DMQ, we first train a PQ model and use its discrete code indices to supervise the assignment network, while initializing the codebooks with K-means centroids scaled by $1/M$. For residual DMQ, we similarly train an RQ model and use its discrete code indices to supervise the assignment network, while initializing the codebooks with the corresponding RQ codebooks. The initialization stage optimizes

$$\mathcal{L}_{\mathrm{init}} = \mathcal{L}_{\mathrm{CE}} + \lambda_{\mathrm{bal}}\mathcal{L}_{\mathrm{bal}}, \tag{18}$$

where

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{M}\sum_{m=1}^{M}\log p_m(k_m^*). \tag{19}$$

For residual DMQ, subsequent residual states are generated from the model's own soft assignments rather than target assignments, preserving the autoregressive self-generated dynamics.

4.4.2 TRAINING OBJECTIVE

After initialization, the assignment network and codebooks are jointly optimized with

$$\mathcal{L} = \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{bal}}\mathcal{L}_{\mathrm{bal}}, \tag{20}$$

where

$$\mathcal{L}_{\mathrm{rec}} = \|x - \hat{x}\|_2^2. \tag{21}$$

Since code assignment is independent of codebook geometry, the same framework can also incorporate downstream objectives, such as retrieval or ranking losses, to directly optimize the resulting discrete representations.

4.5 Discrete Inference

At inference time, we first obtain discrete assignments through a direct forward pass:

$$k_m = \arg\max_k p_m(k). \tag{22}$$

For parallel DMQ, Iterated Conditional Modes (ICM; Besag, 1986) can optionally be applied as a post-hoc refinement step to further enhance reconstruction quality. ICM iteratively optimizes one codebook at a time while fixing the assignments of all other codebooks:

$$k_m \leftarrow \arg\min_k \left\| x - \left(e_{m,k} + \sum_{j \neq m} e_{j,k_j} \right) \right\|_2^2. \tag{23}$$

Interestingly, we observe that ICM achieves comparable performance regardless of whether it is initialized with random assignments or the assignments predicted by DMQ. Moreover, introducing random code replacements during ICM, as adopted by LSQ (Martinez et al., 2018), provides no further performance gains. We provide a theoretical explanation for this observation in Sec. 4.6.

For residual DMQ, we use beam search to explore multiple autoregressive code sequences. Each candidate path maintains its own residual and updates it using the selected hard codeword:

$$r_m = r_{m-1} - e_{m,k_m}. \tag{24}$$

After the final stage, we select the candidate sequence yielding the lowest reconstruction error.

4.6 Theoretical Analysis

Under this framework, we establish three theoretical results that characterize the advantage of decoupled assignment and explain the behavior of ICM observed in Sec. 4.5.

Theorem 4.1 (Suboptimality of Independent Euclidean Assignment)**.** *Independent Euclidean assignment is not, in general, globally optimal for additive reconstruction, since individually nearest codewords need not form the best joint codeword combination.*

Independent nearest-code selection ignores cross-codebook interactions and can yield suboptimal reconstruction. This motivates coordinated assignment in DMQ to reduce errors arising from individually favorable but jointly inferior codeword choices.

Theorem 4.2 (Expressive Capacity Expansion through Decoupled Assignment)**.** *For fixed finite codebook cardinalities, decoupling assignment from Euclidean codeword geometry strictly enlarges the class of representable assignment functions, while retaining all assignments realizable by Euclidean nearest-code rules.*

Under the stated capacity assumptions, decoupled assignment strictly expands expressivity at the same code budget. It enables decision regions beyond Euclidean nearest-code rules, providing greater flexibility for reconstruction and task-oriented learning.

Theorem 4.3 (Margin-Dominated Global Convergence of ICM)**.** *If the coordinate margin of each globally optimal codeword dominates the worst-case perturbation induced by cross-codebook interactions, then the optimal coordinate update is invariant to the assignments of the other codebooks. Consequently, the global optimum is unique, and cyclic ICM converges to it from any initialization after at most one complete sweep.*

Under the margin-dominance condition, ICM reaches the unique global optimum within one sweep from any initialization. This provides a conditional explanation for the observed initialization robustness and lack of gains from random code perturbations.

The proofs of all Theorems are provided in Appendix A.

5 Experiments

5.1 Experimental Setup

Datasets and Baselines. We evaluate DMQ on four large-scale vector retrieval benchmarks, including BigANN1M (Jégou et al., 2011b), Deep1M (Babenko & Lempitsky, 2016),

Table 1: Comparison of reconstruction error (MSE) and retrieval recall (R@1 and R@10) on four benchmark datasets. We evaluate quantization fidelity and retrieval accuracy under codebook budgets of 8 and 16 bytes. The best results are highlighted in bold.

	Deep1M			**BigANN1M**			**FB-ssnpp1M**			**Contriever1M**		
Method	MSE	R@1	R@10	MSE	R@1	R@10	MSE	R@1	R@10	MSE	R@1	R@10
8 bytes												
OPQ	0.26	16.0	51.3	2.95	21.7	64.3	9.52	2.5	5.1	1.87	8.1	24.5
RQ	0.20	21.3	63.6	2.49	27.5	75.4	9.20	2.7	6.1	1.82	10.4	27.0
LSQ	**0.17**	24.5	69.4	1.91	31.7	79.4	8.87	3.3	7.5	1.65	13.1	33.9
DPQ	0.23	19.8	58.4	2.71	24.6	70.1	9.32	2.6	5.7	1.83	9.8	26.3
DeepQ	0.19	21.8	64.9	2.36	28.3	76.7	9.12	2.8	6.4	1.78	10.6	28.2
Parallel DMQ	**0.17**	24.7	**69.8**	**1.77**	**32.3**	**80.0**	**8.77**	**3.4**	**8.7**	**1.61**	**14.9**	**36.8**
Residual DMQ	0.18	**24.9**	68.9	1.92	31.9	79.3	8.88	3.3	8.0	1.64	15.1	36.2
16 bytes												
OPQ	0.14	34.9	82.2	1.79	40.5	89.9	7.25	5.0	11.8	1.71	18.3	40.9
RQ	0.10	43.0	90.8	1.30	49.0	95.0	7.01	5.4	13.0	1.65	20.2	43.5
LSQ	**0.09**	42.3	89.7	0.98	51.1	95.4	**6.63**	**6.2**	14.8	1.35	25.6	53.8
DPQ	0.12	38.5	86.1	1.56	45.3	92.8	7.14	5.1	12.3	1.69	19.6	41.7
DeepQ	0.10	41.5	88.3	1.24	49.7	95.2	7.07	5.3	12.6	1.58	21.6	45.1
Parallel DMQ	0.10	41.9	89.1	**0.75**	**55.9**	**97.3**	6.79	**6.2**	**15.6**	**1.31**	**26.5**	**54.4**
Residual DMQ	**0.09**	**45.0**	**92.2**	1.08	51.7	95.9	6.80	5.7	15.3	1.36	25.3	53.3

FB_ssnpp1M (Simhadri et al., 2021), and Contrieval1M (Huijben et al., 2024). Each benchmark uses 500K vectors for training, 10K queries, and 1M database vectors. We compare DMQ with OPQ (Ge et al., 2013), RQ (Chen et al., 2010), LSQ (Martinez et al., 2018), DPQ (Chen et al., 2020), and DeepQ (Zhu et al., 2023), using FAISS implementations for OPQ, RQ, and LSQ, and official implementations for DPQ and DeepQ.

Metrics and Implementation. We evaluate reconstruction and retrieval quality using mean squared error (MSE) and Recall@R (R@R). Unless otherwise specified, all methods use the same code budget, with $M \in \{8, 16\}$ codebooks and $K = 256$ codewords per codebook, corresponding to 64-bit and 128-bit representations. For DMQ, we use a lightweight Transformer-based assignment network with shared backbone parameters and position-specific input projections and prediction heads, with full and causal attention for parallel and residual variants, respectively. More implementation details are provided in Appendix B.

5.2 Overall Performance

Tab. 1 presents the reconstruction and retrieval results under 64-bit and 128-bit code budgets. Among the baselines, LSQ consistently provides the strongest overall performance, outperforming both conventional residual quantization and neural assignment methods such as DPQ and DeepQ across most datasets. This indicates that effective code assignment remains important even when neural networks are introduced into the quantization process.

DMQ further improves upon this strong baseline. Parallel DMQ achieves the best overall performance on BigANN1M and Contriever1M, while also obtaining competitive results on Deep1M and FB-ssnpp1M. Residual DMQ also achieves strong performance, obtaining the best retrieval results on Deep1M at 128 bits.

5.3 Ablation Study

We conduct ablations on FB_ssnpp1M with $M = 8$ codebooks. "w/o PQ/RQ pretrain" replaces supervised PQ/RQ pretraining with load-balancing pretraining; "w/o pretrain" removes pretraining entirely; "w/o Attention" removes self-attention; "w/o Soft Assignment" uses argmax with straight-through estimation (STE); and "w/o Projection" removes position-specific input projections.

As shown in Tab. 2, load-balancing pretraining achieves nearly identical final performance to PQ/RQ pretraining, although it converges more slowly. Removing pretraining causes a substantial degradation, while removing attention, soft assignment, or projection also consistently hurts performance.

Table 2: Ablation study of different design choices in DMQ.

	Parallel DMQ			**Residual DMQ**		
Variant	MSE	R@1	R@10	MSE	R@1	R@10
Full DMQ	8.77	3.4	8.7	8.88	3.3	8.0
w/o PQ/RQ pretrain	8.79	3.4	8.6	8.90	3.3	8.0
w/o pretrain	11.2	1.5	4.3	14.3	1.2	3.5
w/o Attention	9.15	2.8	6.2	9.21	2.7	6.0
w/o Soft Assignment	11.6	1.5	4.1	13.6	1.3	3.8
w/o Projection	9.03	3.1	6.7	9.11	2.9	6.4

Table 3: Comparison of different inference strategies for parallel and residual DMQ.

Structure	Search Strategy	MSE	R@1	R@10	Time (s)
Parallel DMQ	Argmax	1.83	32.0	79.6	18.0
	Argmax + ICM	1.77	32.3	80.0	18.4
	Random + ICM	1.78	32.2	80.1	17.0
	Beam Search ($B = 5$)	2.02	26.8	73.8	18.3
	Beam Search ($B = 10$)	1.94	28.4	78.2	18.6
Residual DMQ	Argmax	1.96	30.9	78.6	16.3
	Argmax + ICM	2.02	29.9	77.0	18.6
	Random + ICM	2.44	23.7	67.7	19.1
	Beam Search ($B = 5$)	1.92	31.9	79.3	18.0
	Beam Search ($B = 10$)	1.87	32.5	80.6	18.5

5.4 Inference Analysis

We compare different inference strategies for parallel and residual DMQ in Tab. 3. "Argmax" directly selects the most probable codeword, "Argmax + ICM" applies ICM initialized by argmax assignments, and "Random + ICM" starts ICM from random assignments. Beam Search uses beam sizes of 5 and 10. As shown in Tab. 3, for parallel DMQ, ICM provides only marginal gains over direct argmax, while random initialization followed by ICM achieves comparable performance, indicating that ICM is relatively insensitive to initialization. For residual DMQ, ICM does not consistently improve the autoregressive assignments, whereas beam search progressively improves performance as the beam size increases and achieves the best results with $B = 10$.

We also report the total inference time, including both encoding and retrieval, in Tab. 3. All experiments are conducted on a single NVIDIA H200 GPU with 143 GB memory and an Intel Xeon Platinum 8558 CPU with 192 cores. Under the same setting, LSQ takes 61.3 seconds in total, while DMQ completes inference in less than 20 seconds. For residual DMQ, we further employ KV caching to accelerate autoregressive inference.

6 Conclusion

We presented DMQ, a framework that separates code assignment from codebook geometry and unifies parallel and residual quantization. By learning assignments from input states and cross-codebook context, DMQ enables coordinated code selection and joint optimization of assignments and representations. Our theoretical analysis establishes the potential suboptimality of independent Euclidean assignment, greater assignment expressivity under suitable capacity assumptions, and a sufficient condition for initialization-independent global convergence of ICM. Extensive experiments demonstrate the effectiveness of this design for reconstruction and retrieval, while ablation and inference analyses clarify the contributions of its components and refinement strategies.

Our evaluation focuses on vector retrieval with fixed code budgets; applications with different objectives and resource constraints remain to be explored. Future work will investigate task-specific training for recommendation and generation, as well as adaptive routing and search strategies that balance representation quality and computational cost.

AI USE STATEMENT

AI tools were employed solely to support the preparation of the manuscript, primarily for improving readability, refining wording and organization, and checking the text for language-related issues. The authors independently developed all scientific and technical contributions presented in this work, including the experimental results and the conclusions drawn from them. LLMs were not involved in generating data, formulating the methodology or experimental setup, implementing the proposed approach, or interpreting and analyzing the experimental findings.

All content produced with AI assistance was carefully reviewed and verified by the authors. The authors assume full responsibility for the accuracy and integrity of the final manuscript.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, our implemenation is anonymously provided at `https://anonymous.4open.science/r/DMQ`. Sec. 4 describes the DMQ architecture, assignment mechanisms, training objectives, and inference procedures for both parallel and residual quantization. For the theoretical results in Sec. 4.6, Appendix A presents the assumptions and complete proofs of all three theorems. Sec. 5.1 specifies the benchmark datasets, training and evaluation set sizes, baseline implementation sources, evaluation metrics, and code budgets. Appendix B provides additional implementation details, including initialization procedures, optimization settings, temperature annealing, and inference-time search configurations. Sec. 5.3 further documents the ablation settings used to assess individual design choices.

REFERENCES

Artem Babenko and Victor S. Lempitsky. Additive quantization for extreme vector compression. In *CVPR*, pp. 931–938, 2014.

Artem Babenko and Victor S. Lempitsky. Efficient indexing of billion-scale datasets of deep descriptors. In *CVPR*, pp. 2055–2063, 2016.

Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society: Series B (Methodological)*, 48(3):259–302, 1986.

Xingyan Bin, Jianfei Cui, Wujie Yan, Zhichen Zhao, Xintian Han, Chongyang Yan, Feng Zhang, Xun Zhou, Xiao Yang, and Zuotao Liu. Real-time indexing for large-scale recommendation by streaming vector quantization retriever. In *KDD*, pp. 4273–4283, 2025.

Ting Chen, Lala Li, and Yizhou Sun. Differentiable product quantization for end-to-end embedding compression. In *ICML*, volume 119, pp. 1617–1626, 2020.

Yongjian Chen, Tao Guan, and Cheng Wang. Approximate nearest neighbor search by residual vector quantization. *Sensors*, 10(12):11259–11273, 2010.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, pp. 2946–2953, 2013.

Yuchao Gu, Xintao Wang, Yixiao Ge, Ying Shan, and Mike Zheng Shou. Rethinking the objectives of vector-quantized tokenizers for image synthesis. In *CVPR*, pp. 7631–7640, 2024.

Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. Quantization based fast inner product search. In *AISTATS*, volume 51, pp. 482–490, 2016.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *ICML*, volume 119, pp. 3887–3896, 2020.

Iris A. M. Huijben, Matthijs Douze, Matthew J. Muckley, Ruud van Sloun, and Jakob Verbeek. Residual quantization with implicit neural codebooks. In *ICML*, volume 235, pp. 20682–20699, 2024.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011a.

Hervé Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. Searching in one billion vectors: Re-rank with source coding. In *ICASSP*, pp. 861–864, 2011b.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, pp. 11513–11522, 2022.

Qijiong Liu, Xiaoyu Dong, Jiaren Xiao, Nuo Chen, Hengchang Hu, Jieming Zhu, Chenxu Zhu, Tetsuya Sakai, and Xiao-Ming Wu. Vector quantization for recommender systems: A review and outlook. *arXiv Preprint*, 2024. https://arxiv.org/abs/2405.03110.

Julieta Martinez, Shobhit Zakhmi, Holger H. Hoos, and James J. Little. LSQ++: lower running time and higher recall in multi-codebook quantization. In *ECCV*, volume 11220, pp. 508–523, 2018.

Stanislav Morozov and Artem Babenko. Unsupervised neural quantization for compressed-domain similarity search. In *ICCV*, pp. 3036–3045, 2019.

Piotr Piekos, Subhradeep Kayal, and Alexandros Karatzoglou. Hyperbolic residual quantization: Discrete representations for data with latent hierarchies. *arXiv Preprint*, 2025. https://arxiv.org/abs/2505.12404.

Harsha Vardhan Simhadri, George Williams, Martin Aumüller, Matthijs Douze, Artem Babenko, Dmitry Baranchuk, Qi Chen, Lucas Hosseini, Ravishankar Krishnaswamy, Gopal Srinivasa, Suhas Jayaram Subramanya, and Jingdong Wang. Results of the neurips'21 challenge on billion-scale approximate nearest neighbor search. In *NeurIPS*, pp. 177–189, 2021.

Mingzhen Sun, Weining Wang, Xinxin Zhu, and Jing Liu. Reparameterizing and dynamically quantizing image features for image generation. *Pattern Recognit.*, 146:109962, 2024.

Théophane Vallaeys, Matthew J. Muckley, Jakob Verbeek, and Matthijs Douze. Qinco2: Vector compression and search with improved implicit neural codebooks. In *ICLR*, 2025.

Jianfeng Wang, Jingdong Wang, Jingkuan Song, Xin-Shun Xu, Heng Tao Shen, and Shipeng Li. Optimized cartesian k-means. *IEEE Trans. Knowl. Data Eng.*, 27(1):180–192, 2015.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Jointly optimizing query encoder and product quantization to improve retrieval performance. In *CIKM*, pp. 2487–2496, 2021.

Zhengjia Zhong, Shuyan Ke, Zaizhou Lin, Jiaqi Song, Hongyi Lan, and Hui Li. Rq-moe: Residual quantization via mixture of experts for efficient input-dependent vector compression. In *ICML*, 2026.

Xiaosu Zhu, Jingkuan Song, Lianli Gao, Xiaoyan Gu, and Heng Tao Shen. Revisiting multi-codebook quantization. *IEEE Trans. Image Process.*, 32:2399–2412, 2023.

A Proofs of Theorems

A.1 Proof for Theorem 4.1

Proof. For a joint assignment $\mathbf{k} = (k_1, \ldots, k_M)$, independent Euclidean assignment minimizes

$$J_G(\mathbf{k}; x) = \sum_{m=1}^{M} \|x - e_{m,k_m}\|_2^2, \tag{25}$$

whereas the globally optimal compositional assignment minimizes

$$J_C(\mathbf{k}; x) = \left\| x - \sum_{m=1}^{M} e_{m,k_m} \right\|_2^2. \tag{26}$$

Expanding the two objectives gives

$$J_C(\mathbf{k}; x) = J_G(\mathbf{k}; x) - (M-1)\|x\|_2^2 + 2\sum_{m<n} e_{m,k_m}^\top e_{n,k_n}. \tag{27}$$

For fixed x, the second term is independent of $\mathbf{k}$, whereas the last term depends jointly on codewords selected from different codebooks. Hence the minimizers of J_G and J_C need not coincide.

A counterexample suffices. Consider

$$\mathcal{C}_1 = \left\{0, \frac{4}{5}\right\}, \qquad \mathcal{C}_2 = \left\{0, \frac{7}{10}\right\}, \qquad x = 1. \tag{28}$$

Independent Euclidean assignment selects $(2, 2)$, since $4/5$ and $7/10$ are individually closer to 1 than 0. The reconstruction errors of the four possible assignments are

$$\begin{aligned} E(1,1) &= 1, \qquad E(1,2) = \frac{9}{100}, \\ E(2,1) &= \frac{1}{25}, \quad E(2,2) = \frac{1}{4}. \end{aligned} \tag{29}$$

Thus $(2, 1)$ is the unique global minimizer, whereas independent Euclidean assignment selects $(2, 2)$. Therefore,

$$a_G(x, \mathcal{C}) \neq a^*(x, \mathcal{C}). \tag{30}$$

Hence independent Euclidean assignment does not, in general, minimize the global additive reconstruction error. This proves Theorem 4.1. □

A.2 Proof for Theorem 4.2

Proof. Fix finite codebook cardinalities $2 \leq K_m < \infty$ for $m = 1, \ldots, M$. Let $\mathcal{A}_{\text{geo}}$ denote the family of Euclidean nearest-code assignments with these cardinalities.

Let $\mathcal{H}$ denote the scalar scoring-function class used by the decoupled assignment. We assume that $\mathcal{H}$ contains all affine functions and, at minimum, the functions realizable by a one-hidden-layer ReLU network of width two with an affine output. Let $\mathcal{A}_{\text{dec}}$ denote the corresponding family of decoupled assignments

$$a_\theta(x) = \left(\arg\max_k h_{\theta,1,k}(x), \ldots, \arg\max_k h_{\theta,M,k}(x)\right), \qquad h_{\theta,m,k} \in \mathcal{H}. \tag{31}$$

Both assignment classes take values in $[K_1] \times \cdots \times [K_M]$, and ties are resolved using the same fixed, input-independent priority rule.

We first show that every Euclidean nearest-code assignment belongs to $\mathcal{A}_{\text{dec}}$. For each codebook,

$$\arg\min_k \|x - e_{m,k}\|_2^2 = \arg\max_k \left(2e_{m,k}^\top x - \|e_{m,k}\|_2^2\right). \tag{32}$$

The score on the right is affine in x and therefore belongs to $\mathcal{H}$. Choosing

$$h_{\theta,m,k}(x) = 2e_{m,k}^\top x - \|e_{m,k}\|_2^2 \tag{33}$$

thus reproduces exactly the Euclidean nearest-code rule under the same tie-breaking convention. Consequently,

$$\mathcal{A}_{\text{geo}} \subseteq \mathcal{A}_{\text{dec}}. \tag{34}$$

It remains to show that the inclusion is strict. Consider one codebook and a codeword e_k. For any competing codeword e_j,

$$\|x - e_k\|_2^2 \leq \|x - e_j\|_2^2 \iff 2(e_j - e_k)^\top x \leq \|e_j\|_2^2 - \|e_k\|_2^2. \tag{35}$$

Hence the Euclidean nearest-code region associated with e_k is the intersection, over all $j \neq k$, of affine half-spaces of the form in Eq. 35. Ignoring boundary ownership, this is precisely a Euclidean Voronoi cell and is therefore a convex polyhedral region whenever nonempty. Under the fixed tie-breaking rule, some boundary inequalities may become strict; since both open and closed affine half-spaces are convex, the resulting decision region remains convex.

We now construct a decision region representable by $\mathcal{A}_{\mathrm{dec}}$ that violates this necessary convexity property. For the first codebook, define

$$h_{\theta,1,1}(x) = \mathrm{ReLU}(x_1) + \mathrm{ReLU}(-x_1) = |x_1|, \qquad h_{\theta,1,2}(x) = 1. \tag{36}$$

If $K_1 > 2$, set every additional score to the constant -1. These functions belong to $\mathcal{H}$ by assumption. Thus code 1 is uniquely selected whenever $|x_1| > 1$, whereas code 2 is uniquely selected whenever $|x_1| < 1$.

Consider

$$x^{(-)} = (-2, 0, \ldots, 0), \qquad x^{(+)} = (2, 0, \ldots, 0). \tag{37}$$

Both points are assigned to code 1. Their midpoint is

$$\frac{x^{(-)} + x^{(+)}}{2} = (0, \ldots, 0), \tag{38}$$

which is assigned to code 2. Therefore, if R_1 denotes the decision region of code 1, then

$$x^{(-)}, x^{(+)} \in R_1, \qquad \frac{x^{(-)} + x^{(+)}}{2} \notin R_1. \tag{39}$$

Hence R_1 is not convex. Since every Euclidean nearest-code region must be convex, no assignment in $\mathcal{A}_{\mathrm{geo}}$ can realize this decision rule. All three comparisons above are strict, so the argument is independent of the tie-breaking behavior on the boundary $|x_1| = 1$.

For the remaining codebooks, the scores may be chosen arbitrarily, for example as constants selecting a fixed index. Therefore the above construction defines a valid joint assignment in $\mathcal{A}_{\mathrm{dec}}$ that cannot belong to $\mathcal{A}_{\mathrm{geo}}$. Hence

$$\mathcal{A}_{\mathrm{dec}} \setminus \mathcal{A}_{\mathrm{geo}} \neq \varnothing. \tag{40}$$

Combining this fact with Eq. 34, we obtain

$$\mathcal{A}_{\mathrm{geo}} \subsetneq \mathcal{A}_{\mathrm{dec}}. \tag{41}$$

Therefore, under the stated capacity assumption on $\mathcal{H}$, decoupling assignment from Euclidean codeword geometry strictly enlarges the set of representable assignment functions. This proves Theorem 4.2. □

A.3 Proof for Theorem 4.3

Proof. Let $\mathbf{k}^* = (k_1^*, \ldots, k_M^*)$ be a globally optimal assignment. For a fixed codebook m, write

$$E(k, \mathbf{k}_{-m}) = \left\| x - e_{m,k} - \sum_{j \neq m} e_{j,k_j} \right\|_2^2 \tag{42}$$

for the reconstruction error obtained by assigning codeword k to codebook m while keeping the other assignments fixed to $\mathbf{k}_{-m}$.

For every alternative $k \neq k_m^*$, define the coordinate margin

$$\Delta_{m,k} = E(k, \mathbf{k}_{-m}^*) - E(k_m^*, \mathbf{k}_{-m}^*), \tag{43}$$

where $\mathbf{k}_{-m}^*$ denotes the globally optimal assignments of all codebooks except m.

Define the coordinate energy difference under an arbitrary configuration $\mathbf{k}_{-m}$ as

$$D_{m,k}(\mathbf{k}_{-m}) = E(k, \mathbf{k}_{-m}) - E(k_m^*, \mathbf{k}_{-m}). \tag{44}$$

The codeword k_m^* is the unique coordinate minimizer whenever $D_{m,k}(\mathbf{k}_{-m}) > 0$ for every $k \neq k_m^*$.

Expanding the two squared reconstruction errors and subtracting the corresponding difference at $\mathbf{k}_{-m}^*$ gives

$$\begin{aligned} D_{m,k}(\mathbf{k}_{-m}) = \Delta_{m,k} \\ + 2 \left(e_{m,k} - e_{m,k_m^*} \right)^\top \sum_{j \neq m} \left(e_{j,k_j} - e_{j,k_j^*} \right). \end{aligned} \tag{45}$$

Therefore,

$$D_{m,k}(\mathbf{k}_{-m}) \geq \Delta_{m,k} - 2\left|\left(e_{m,k} - e_{m,k_m^*}\right)^\top \sum_{j\neq m}\left(e_{j,k_j} - e_{j,k_j^*}\right)\right|. \tag{46}$$

The dominance assumption in Theorem 4.3 is precisely that, for every m and every $k \neq k_m^*$,

$$\Delta_{m,k} > 2\sup_{\mathbf{k}_{-m}}\left|\left(e_{m,k} - e_{m,k_m^*}\right)^\top \sum_{j\neq m}\left(e_{j,k_j} - e_{j,k_j^*}\right)\right|. \tag{47}$$

Hence the right-hand side of Eq. 46 is strictly positive for every possible $\mathbf{k}_{-m}$. Therefore,

$$E(k_m^*, \mathbf{k}_{-m}) < E(k, \mathbf{k}_{-m}), \qquad \forall\, k \neq k_m^*, \tag{48}$$

regardless of the assignments of the remaining codebooks. Thus k_m^* is the unique optimal update for coordinate m under every possible configuration of the other coordinates.

Since m was arbitrary, the same property holds for every codebook. Starting from any initialization, cyclic ICM therefore sets each coordinate to its corresponding k_m^* when that coordinate is first updated. By induction over the M coordinates, after at most one complete sweep the assignment is exactly $\mathbf{k}^*$.

Finally, consider any assignment $\widetilde{\mathbf{k}} \neq \mathbf{k}^*$. For every coordinate on which $\widetilde{\mathbf{k}}$ differs from $\mathbf{k}^*$, replacing that coordinate by the corresponding k_m^* strictly decreases the energy by the result above. Repeating this operation reaches $\mathbf{k}^*$ after finitely many strictly decreasing steps. Consequently,

$$E(\mathbf{k}^*) < E(\widetilde{\mathbf{k}}) \qquad \text{for all } \widetilde{\mathbf{k}} \neq \mathbf{k}^*, \tag{49}$$

so $\mathbf{k}^*$ is the unique global minimizer.

Thus, under condition Eq. 47, cyclic ICM converges to the unique global optimum from arbitrary initialization after at most one complete sweep. This proves Theorem 4.3. □

B Implementation Details

For parallel DMQ, we initialize the assignment network using discrete code indices obtained from a pretrained PQ model and initialize the codebooks with K-means centroids scaled by $1/M$. For residual DMQ, we use discrete code indices from a pretrained RQ model to initialize the assignment network and initialize the codebooks with the corresponding RQ codebooks.

During training, DMQ is jointly optimized with the reconstruction loss and the load-balancing loss. We set the load-balancing coefficient to $\lambda_{\text{bal}} = 0.01$. We use Adam with its default hyperparameters and an initial learning rate of 0.001. The learning rate is reduced by a factor of 0.1 if the training loss does not decrease for 10 consecutive epochs, and training is stopped when the learning rate falls below 10^{-5}. The assignment temperature is cosine-annealed to 0.3 during training.

At inference time, we use six ICM iterations for parallel DMQ and beam search with beam size 5 for residual DMQ. Unless otherwise specified, the results reported in the main experiments use these inference-time refinement procedures.