A Related Work

A.1 ANNs and MIPS

ANNS achieves highly efficient vector search by allowing a small number of errors. Generally, there are two kinds of ANNS algorithms: *non-exhaustive ANNS methods* [37, 34, 36, 28] and *vector compression methods* [12, 29, 3, 18]. Specifically, *Non-exhaustive ANNS methods* do not compress the index. They reduce the number of candidates for each query to speed up the retrieval process. Popular algorithms include tree search [37, 11] and graph search [34, 36, 28]. *Vector compression methods* mainly aim to compress the index to accelerate retrieval. Popular algorithms include hashing [12, 38, 41] and quantization [29, 3, 18, 17].

Under the constraints of storage, compressed methods are widely investigated by researchers. Product quantization [22, 6] decomposes the vector representation space into the Cartesian product of subspaces. Optimized product quantization (OPQ) [16] jointly learns space decomposition and subspace quantization. Multi-scale Quantization [46] includes a multi-scale framework so that it can learn a separate scalar quantizer. Composite Quantization [50] and Additive Quantization [3] do not decompose space but directly learn multiple codebooks. There are also some algorithms that take query information into account. NEQ [10] decomposes the quantization error into norm error and direction error and improves existing VQ techniques for MIPS. ScaNN [18] computes the weight for each pair of vectors. Different from NEQ and ScaNN, KDindex utilizes query and corresponding top-k candidates. BLISS [19] regards ground truth as labels. However, the ground truth is difficult to obtain in huge quantities of databases. Interested readers could refer to the surveys [43, 31].

A.2 Knowledge Distillation

Knowledge Distilling (KD) was first proposed in [20], in which a complex neural network was firstly trained and then transferred to a small model. Following this, DarkRank [7] proposed a method combining deep metric learning and *Learning to rank* technique with KD to solve image retrieval and image clustering tasks. In addition, a few recent methods [30, 39] have adopted knowledge distillation to RS. RD [42] firstly proposes a KD method that makes the student give high scores on the top-ranked items of the teacher's recommendation list. Similarly, CD [30] makes the student imitate the teacher's prediction scores with particular emphasis on the items ranked highly by the teacher. The most recent work, RRD [26], formulates the distillation process as a relaxed ranking matching problem between the ranking list of the teacher and that of the student. However, there are limited works focusing on index building under knowledge distillation.

In the context of quantization problems under distillation, the most relevant work is Distill-VQ [47], which uses knowledge distillation for ranking candidates in web search tasks. This method applies the sampling technique to rank a sample of the document from all data each time. But this technique is not applicable to training a ranking model when documents and queries are represented with no content information. In this case, the labeled model training cannot be easily generalized to all documents and queries. In contrast, KDindex relaxes the requirement on labeled data and can be trained purely with unlabeled data.

B More Details of Experimental Settings

B.1 Dataset Statistics

Table 5: Statistics of the datasets.

Datasets	#Database	#Train	#Test	Dim
SIFT1M	1,000,000	100,000	10,000	128
GIST1M	1,000,000	500,000	1,000	960
MS MARCO Doc	3,213,833	367,013	5,193	768
MS MARCO Passage	8,841,823	808,731	101,093	768

B.2 Baselines

The two groups of baseline ANNS models are compared to KDindex.

The first group is *Non-quantization-based ANNS methods*, which accelerate the search by reducing the number of candidates. **BLISS** [19] adopts the learning-to-index framework to learn the hashing-based compressed functions. **ScaNN** [18] quantizes with anisotropic quantization loss functions which greatly penalizes the parallel component of a datapoint's residual relative to its orthogonal component. **HNSW** [35] builds a hierarchical set of proximity graphs.

The second is *Quantization-based ANNS methods*, which compress the embeddings by hashing or quantization functions. **PQ** [22] decomposes the vector representation space into the Cartesian product of subspaces. **OPQ** [16] jointly learns space decomposition and subspace quantization. **AQ** [3] represents each vector as a sum of several components each coming from a separate codebook. The baselines are implemented based on the Faiss ANNS library [25]. The parameters B and W are set to be the same as KDindex. **DiffPQ** [5], differentiable product quantization, a generic and end-to-end learnable compression framework. **DeepPQ** [15], deep progressive quantization, end-to-end learns the quantization codes sequentially. **PQ-VAE** [45], an unsupervised model for learning discrete representations by combining product quantization and auto-encoders. The CNN blocks are replaced with MLP because the image datasets have been extracted as 512-dimension features. **GCD** [23] learns rotation matrix via a family of block Givens coordinate descent algorithms. **RepCONC** [49] requires data points to be uniformly clustered around the quantization centroids.

B.3 Implemental details

Table 6: Details of teacher model (HNSW).

Teacher (HNSW)	SIFT1M	GIST1M	MS MARCO Doc	MS MARCO Passage
M	32	32	32	32
efConstruction	40	40	100	100
efSearch	100	512	1024	1024
Search Time (s)	0.5862	1.3082	1.4805	4.7689
Building Time (s)	20.1s	2m25.4s	17m52s	98m17.2s
Recall@10	0.9865	0.9859	0.9292	0.9182

Teacher model is instantiated as HNSW. The details are described as Tab. 6, where M denotes the number of neighbors each node, efConstruction denotes expansion factor at construction time and efSearch denotes expansion factor at search time. To obtain good recall performance, M, efConstruction and efSearch are tuned.

B.4 Complexity Analysis

For simplicity, we discuss the complexity of each codebook with W centroids. Posting List Balance requires $\mathcal{O}(MWD)$ to calculate the similarities between the M document vectors and the centroids and the space complexity is $\mathcal{O}(D^2)$. Besides, the query encoder brings an extra time cost of $\mathcal{O}(D^2)$ and space cost of $\mathcal{O}(WD)$. Overall, the time complexity and space complexity of KDindex is $\mathcal{O}(D^2 + MWD)$ and $\mathcal{O}(D^2 + WD)$, respectively, which is acceptable since W and D are small constants. As for the iterative initialization, the index assignment of documents only needs to be updated after several epochs of centroids optimization. For the differentiable training, both the index assignment and centroids are updated **every mini batch**.

C Varying Distillation Loss

C.1 Distillation Losses

Knowledge distillation was first proposed for classification tasks, where the probabilities of each class attained from the large-scale teacher network are considered as soft labels to supervise the learning of the small-size student network. The cross-entropy loss is commonly used as the distillation loss to minimize the difference between the teacher and student networks. Here, the teacher search model provides the top-k relevant candidates rather than the continuous value of probabilities. Thus, three ranking-oriented losses are designed to distill knowledge from the more accurate indexes to guide the student indexes to return the same nearest results.

Lambdarank loss: The pair-wise ranking-based loss is widely used to learn the ranking list by leading the high-ranked candidate to have higher similarity scores. Lambdarank [4] further introduces the change of the indicators, e.g., NDCG, to put more attention on more important candidates that have not been well ranked. The loss follows as:

$$\mathcal{L}(\boldsymbol{q}, \mathcal{D}_K^T; \boldsymbol{C}) = \sum_{i,j \in \mathcal{D}_K^T} \log\left(1 + \exp(p_i - p_j)\right) |\Delta NDCG@10_{ij}| \tag{4}$$

where $\mathcal{D}_K^T$ denotes the top-k results retrieved from the teacher model and $p_i = S(\boldsymbol{q}, Q(\boldsymbol{d}_i))$ is the similarity score between the query vector and the quantized vector of the candidate i. Q is the quantizer function related to the codebooks $\boldsymbol{C}$. $\Delta NDCG@10_{ij}$ denotes the change with respect to NDCG@10 if changing the i-th ranked and j-th ranked candidate.

Weighted KL loss: Similar to the class distribution in classification tasks, the similarity distribution over the top-k retrieved candidates can also be obtained. One is based on the ground-truth vectors and the other one is based on the quantized vectors. In order to ensure the ranking orders correspond to the top-k list, the rank information is also considered where the high-ranked items are more concerned. Finally, the loss function follows as:

$$\mathcal{L}(\boldsymbol{q}, \mathcal{D}_K^T; \boldsymbol{C}) = -\sum_{i \in \mathcal{D}_K^T} \tilde{p}_i^g \log \frac{\tilde{p}_i^g}{\tilde{p}_i} \tag{5}$$

where $\tilde{p}_i$ denotes the normalized ranked similarity score with the quantized vector and $\tilde{p}_i^g$ with the ground-truth vector. Specifically,

$$p_i = w_i \cdot S\left(\boldsymbol{q}, Q(\boldsymbol{d}_i)\right), \quad p_i^g = w_i \cdot S(\boldsymbol{q}, \boldsymbol{d}_i),$$

$\tilde{p}_i$ and $\tilde{p}_i^g$ are the normalized values over the top-k retrieved candidates depending on the softmax function. $w_i = \frac{1}{rank(i)}$ denotes the ranking weight according to the ranking orders among the top-k results from the teacher model. The weighted KL loss attempts to minimize the distance between the ground-truth vector and the quantized vector for the top-k relevant candidates to learn better centroids. The introduced rank-oriented weight further guides the student index to return the same ranking list.

Distributed-based loss: Instead of being oriented by the score between query and candidates as above, we attempt to minimize the distance between the queries and top-k neighbors by calculating the similarity scores with all the centroids. Thus, we could obtain more information from centroids and focus on the top-K nearest neighbors. The distributed-based loss function follows as:

$$\mathcal{L}(\boldsymbol{q}, \mathcal{D}_K^T; \boldsymbol{C}) = -\sum_{i \in \mathcal{D}_K^T} \sum_{b=1}^{B} \sum_{k=1}^{W} \tilde{p}_{bk}^q \log(\tilde{p}_{bk}^{d_i} \cdot w_i) \tag{6}$$

where B denotes the number of codebooks and W is the number of codewords in each codebook. $w_i = \frac{1}{rank(i)}$ corresponds to the top-k list given from the teacher model. p_{bk}^q denotes the similarity score between the query $\boldsymbol{q}$ and the codeword $\boldsymbol{c}_k^b$, i.e., $p_{bk}^q = S(\boldsymbol{q}, \boldsymbol{c}_k^b)$, and $p_{bk}^{d_i}$ denotes the similarity score between the candidate $\boldsymbol{d}_i$ and the codeword $\boldsymbol{c}_k^b$, i.e., $p_{bk}^{d_i} = S(\boldsymbol{d}_i, \boldsymbol{c}_k^b)$. The normalized value $\tilde{p}_{bk}^q$ and $\tilde{p}_{bk}^{d_i}$ are calculated over the W codewords for each codebook through the softmax function.

This loss requires the enumeration of all the centroids, while the Weighted KL loss only includes parts of the centroids corresponding to the quantized function. It also aligns with the goal of nearest searching for the query with the learnable centroids as the bridge.

C.2 Experimental Performances

We compare the effectiveness of the three different distillation losses, i.e., Weighted KL loss, Distributed-based loss, and Lambdarank loss as reported in Table 7.

Findings. Overall, the Distributed-based loss leads to comparatively better performances than Weighted KL loss and Lambdarank loss. Compared with Weighted KL Loss and Lambdarank

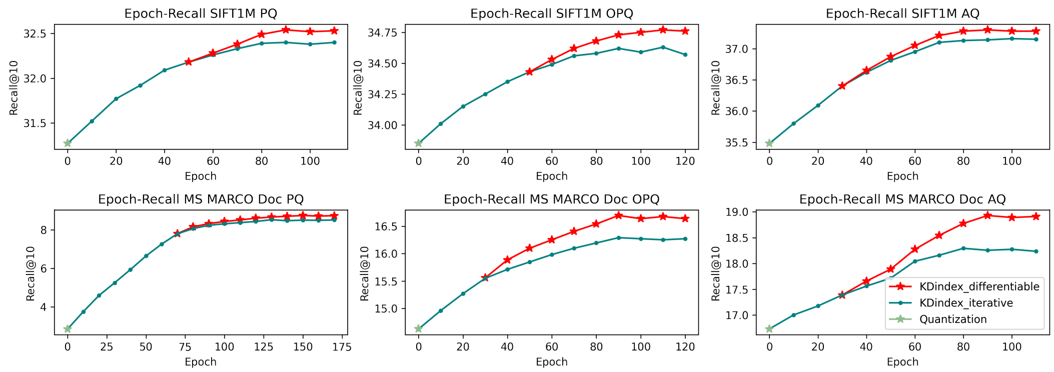

Figure 4: Curves for recall during training warmed up by initialization.

Loss, Distributed-based Loss gains the 2.03% and 3.54% improvements of Recall@10, 0.38%, and 0.70% of NDCG@10, respectively. The Lambdarank Loss concerns more about the relationships between the pair of items, while the other two care about the whole ranking order of the list. The weighted KL loss actually optimizes parts of the centroids, depending on which query vectors and candidate vectors are quantized, to match the ranking list, while all of the centroids are updated in the Distributed-based loss since the probabilities are calculated over all the centroids. Furthermore, the Distributed-based loss requires the similarity calculation between the original input vectors and the centroids, which eliminates the error caused by the compressed functions. The last observation is that Distributed-based Loss works better on inner product metric datasets, since it obtains the average improvements of 2.24% and 3.33% of Recall@10 on L2 distance and inner product, respectively, wherein the overall improvements for inner-product similarities.

Table 7: The results of KDindex under different distillation loss functions.

Loss Function	SIFT1M		GIST1M		MS MARCO Doc		MS MARCO Passage	
	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	MRR@10	Recall@10	MRR@10
Lambdarank Loss	36.32	79.18	20.69	62.06	17.60	40.01	11.10	34.86
Weighted KL Loss	36.68	79.33	21.02	62.75	18.24	40.93	11.06	34.63
Distributed-based Loss	37.30	80.01	21.33	63.17	18.93	41.69	11.19	35.23

D Performance of Differentiable Training

It is extremely difficult for the model to learn the codebooks as well as the index at the same time during the initialization phase in a differentiable training manner. Thus, we perform experiments by warming up the codebooks by *Initialization* and we get the following results in terms of Recall@10 on four datasets as Fig. 4. We adopt the early stop strategy to get the best model.

Initialization. We obtain the pre-trained codebooks by iterative training manners and continue differentiable training when the index assignment is approaching being balanced $(max|\mathcal{P}_i| - min|\mathcal{P}_j| < \frac{N}{W}, i, j \in W)$ where $|\mathcal{P}_i|$ denotes the length of the i-th posting list. To accelerate the iterative training, codebooks are warmed by original quantization methods such as PQ, OPQ, and AQ.

Findings. KDIndex converges to a better solution through the differential training manner. Within the dozens of epochs, the index assignment of iterative training becomes balanced, which warms up the centroids for later easier learning and thus relatively reduces the learning difficulty for both codebooks and indexes. Starting from this point, KDIndex with differentiable training consecutively outperforms that with iterative training, which achieves a relative improvement of 1.63% in terms of Recall@10 on both datasets, demonstrating the effectiveness of synchronizing updates for codebooks and indexes. As for the different quantization functions, the improvements of Recall@10 among different student models (PQ, OPQ, and AQ) are 1.49%, 1.46%, and 1.94%, respectively. The better performance of KDindex(AQ) may be attributed to its better expressiveness with more parameters. Finally, the improvement of Recall@10 on MS MARCO Doc by KDindex(PQ) is 0.40%, which is smaller than the other model since the express ability of PQ is limited. The improvement of

Recall@10 on SIFT1M by KDindex(AQ) is 0.39% since the express ability of AQ is strong on the L2 distance dataset and no more improvements can be obtained easily.

E Limitations and Future Works

In this paper, we propose a novel knowledge distillation framework for high dimension index, which reduce storage obviously and can learn neighbor information from the teacher model. Especially, KDindex is independent to label (such as interaction information in the recommendation system or ground-truth neighbors in ANNS), which makes it flexible to be applied in more label-free scenarios. In the future, we will try more student models such as lattice quantization, whose codes already imply neighbors relationship. And we will take labels into account to improve retrieval performance progressively. We will further improve our work to benefit the broad community.