LightLT: a Lightweight Representation Quantization Framework for Long-tail Data

Haoyu Wang
School of Electrical and Computer Engineering
Purdue University
West Lafayette, United States
wang5346@purdue.edu

Ruirui Li
Amazon
Palo Alto, United States
ruirul@amazon.com

Zhengyang Wang
Amazon
Palo Alto, United States
zhengywa@amazon.com

Xianfeng Tang
Amazon
Palo Alto, United States
xianft@amazon.com

Danqing Zhang
Amazon
Palo Alto, United States
danqinz@amazon.com

Monica Cheng
Amazon
Palo Alto, United States
chengxc@amazon.com

Bing Yin
Amazon
Palo Alto, United States
alexbyin@amazon.com

Jasha Droppo
Amazon
Sunnyvale, United States
drojasha@amazon.com

Suhang Wang
College of Information Sciences and Technology
Pennsylvania State University
University Park, United States
szw494@psu.edu

Jing Gao
School of Electrical and Computer Engineering
Purdue University
West Lafayette, United States
jinggao@purdue.edu

***Abstract*—Search tasks require finding items similar to a given query, making it a crucial aspect of various applications. However, storing and computing similarity for millions or billions of item representations can be computationally expensive. To address this, quantization-based hash methods present memory and inference-efficient solutions by converting continuous representations into non-negative integer codes. Despite their advantages, these methods often encounter difficulties in handling long-tail datasets due to imbalanced class distributions.**

To address this, we propose LightLT, a lightweight representation quantization framework tailored for long-tail datasets. LightLT produces compact codebooks and discrete IDs, enabling efficient inference by computing distances between query and codewords. Our framework includes innovative designs: 1) Quantization Step: We select the most similar codeword for continuous inputs using the differentiable argmax operation. 2) Double Skip Quantization Connection Module: This module promotes codebook diversity and stability during training. 3) Training Loss: Our comprehensive loss includes class-weighted cross-entropy, center loss, and ranking loss. 4) Model Ensemble: We incorporate a model ensemble step to improve generalization. Theoretical analysis confirms LightLT's low space and inference complexity. Experimental results demonstrate superior performance compared to state-of-the-art baselines in terms of search accuracy, efficiency, and memory usage.

***Index Terms*—long-tail, compression, lightweight representation**

I. Introduction

Pre-trained models (e.g., ResNet [1], BERT [2], RoBERTA [3]) are renowned for their remarkable ability to learn high-quality representations from complex image or text data. These learned representations can then be utilized in downstream search and matching tasks, where the goal is to find items that are most similar to a given query among a large pool of candidate items. Examples of such tasks include image retrieval [4]–[7], document retrieval [8]–[11], and question answering [12]–[15], where it has been demonstrated that leveraging representations from pre-trained models leads to outstanding performance. However, in many real-world scenarios, such as E-commerce, the number of candidates (i.e., images or text documents) for retrieval or matching can reach billions or even trillions. In such cases, storing the representations generated by pre-trained models for such a vast pool of candidates can be resource-intensive, requiring significant storage space, and computing the similarity between the query and all the candidates can be extremely challenging and time-consuming.

To overcome this bottleneck, a popular and promising approach is learning to hash [16], [17], which aims to compress item representations. Learning to hash involves mapping continuous representations to compact (discrete) representations, resulting in reduced storage requirements and improved inference efficiency while ensuring retrieval quality. Learning to hash methods can be categorized into two types: binarized hash and quantization-based hash. Binarized hash converts continuous representations into binary codes, while quantization-based hash maps them into non-negative integer codes. In this paper, we refer to binarized hash and quantization-based hash as *hash* and *quantization* respectively, for brevity. Both hash and quantization methods can be trained using two different paradigms: unsupervised [18]–[21] and supervised training [22]–[26], depending on the inclusion of semantic labels. In most cases, supervised quantization methods outperform other hashing-based methods [16]. Hence, this paper primarily focuses on *supervised quantization*.

Existing supervised quantization methods have primarily been developed and evaluated on balanced datasets, where items are approximately evenly distributed across different classes. However, real-world datasets often exhibit imbalanced distributions. Studies such as [27]–[29] have shown datasets in various domains, including images and text collections, tend to follow long-tail distributions. In a long-tail distribution, a small number of dominant classes (denoted as *head classes*) contain the majority of the data, while the remaining classes (known as *tail classes*) contribute only a small portion of the data. Traditional hash and quantization methods typically struggle to achieve satisfactory performance on long-tail datasets, particularly for data originating from tail classes. Addressing the long-tail issue in learning to hash has received limited attention until recently. One approach, LTHNet [30], was proposed to generate multiple prototypes for each class using the determinantal point process (DPP) [31]. This method aimed to transfer knowledge from head classes to tail classes. However, LTHNet has two limitations. Firstly, it is a hash-based method, lacking the ability to leverage quantization approaches known for superior performance. Secondly, if a tail class lacks similarity to any head class, it remains challenging for the tail class to benefit from the knowledge transfer.

To address the aforementioned challenges, we present a novel supervised quantization method called LightLT, specifically tailored for learning lightweight representations on long-tail datasets. LightLT incorporates several innovative designs within its framework to achieve a balance between accuracy and efficiency. At the core of the proposed framework lies a module responsible for quantizing continuous representations into discrete codes (referred to as an encoder). To evaluate the compression quality, a corresponding decoder is required to reconstruct the original data from the discrete codes. We introduce a novel quantization module that comprises a series of encoder-decoder pairs parameterized by codebooks. This design promotes codebook diversity and ensures expressive output quantized representations. It allows for exceptional accuracy while optimizing memory usage. The compression loss is measured using a novel loss function, which combines three distinct loss terms to collectively minimize information loss within limited space. First, the class-weighted cross-entropy loss enables the quantized representations to preserve original information and account for class diversity. By assigning different weights to different classes, we address the challenges posed by long-tailed distributions, ensuring that minority classes are not overlooked. Second, the center loss and ranking loss ensure that quantized representations for items with the same class label are closer to each other compared to those with different labels. To improve generalization and mitigate the risk of overfitting, we introduce an ensemble model into the framework. In contrast to existing bagging ensembles [32], we propose a technique inspired by [33] that involves averaging the weights of multiple trained backbones and quantization modules with different initializations. Overall, LightLT offers a comprehensive and innovative approach to supervised quantization, effectively tackling the challenges posed by long-tail datasets.

Training the proposed framework poses non-trivial challenges, necessitating effective solutions and adjustments to the model design. The quantization module within the framework comprises M encoder-decoder pairs, all sharing the same parameter matrix, which represents a codebook consisting of K codewords. The encoder encodes the continuous representation by selecting the codeword ID from the codebook that most closely resembles the input, while the decoder retrieves the corresponding codeword based on the ID. The decoded codewords are then summed to approximate the continuous representation. However, the non-differentiability of the codeword selection process presents a challenge for end-to-end training of the quantization module, as it cannot be directly optimized through gradient descent. To address this issue, we propose the use of tempered softmax [34], [35] as an approximation of the argmax operation and apply the Straight-Through Estimator [36] to estimate the gradient. This enables effective training of the quantization module despite the non-differentiability of the codeword selection. Another challenge arises when computing gradients across a stack of encoder-decoder pairs, as the gradients can shrink to very small values, particularly when there is a large number of such pairs. This limits our ability to add more encoder-decoder pairs to reduce information loss. To overcome this issue, we introduce an additional skip connection among codewords to stabilize the gradients. This design leads to the adoption of the ”double” skip quantization (DSQ) module within the proposed LightLT framework. The model ensemble step also presents challenges. Since the codewords in different codebooks of base models may not be aligned, a simple averaging of the codebooks would not yield meaningful results. To address this, we propose a solution that involves fixing the backbone model and fine-tuning the DSQ module for codeword alignment after averaging the weights of the models. By addressing these challenges and incorporating the proposed solutions, we enable effective training of the LightLT framework and overcome issues related to non-differentiability, gradient stability, and model ensemble alignment.

We provide theoretical evidence of LightLT's remarkable capabilities in saving storage space and reducing search time. Within the proposed LightLT framework, a single continuous representation can be represented by the IDs of its most similar codewords. This encoding requires only $\frac{1}{8}M\log K$ bytes, where M represents the number of encoder-decoder pairs/codebooks and K denotes the number of codewords. In contrast, continuous representations typically require 4d bytes, where d is the dimensionality. LightLT achieves a substantial reduction in memory consumption. Furthermore, when conducting k-nearest neighbor (kNN) search on the quantized representations, we only need to compute the relevance score between a query and codewords through a simple look-up table, eliminating the need for exhaustive search.

The contributions of this paper can be summarized as follows: 1) LightLT is the first quantization model specifically designed for long-tail data, addressing an important research gap in this area. 2) The proposed model design, which includes a stack of encoder-decoder pairs, a novel loss function, and model ensemble, aims to improve compression quality on long-tail data and enhance the generalization ability of the model. 3) We introduce strategies to overcome the challenges encountered during the training of LightLT, such as approximating the argmax operation, implementing the double skip quantization, and aligning codewords. 4) Extensive experiments are conducted on four datasets, demonstrating the effectiveness of LightLT. The results indicate significant improvements in retrieval performance compared to state-of-the-art baselines. On average, LightLT achieved an impressive 17.6% improvement across the four datasets. Moreover, LightLT offers substantial efficiency improvements in terms of both storage and inference. When applied to a real-world Amazon query dataset, LightLT achieved an impressive 62x speedup ratio and an extraordinary 240x compression ratio.

II. Related Works

In this section, we discuss related works including learning to hash, learning from long-tail data, ensemble learning, and briefly summarize their limitations.

A. Learning to Hash

Learning to hash [16], [17] is a task to learn a hash function $h(\cdot)$, i.e. $\boldsymbol{y} = h(\boldsymbol{x})$, which maps the continuous input representation $\boldsymbol{x}$ into a compact representation $\boldsymbol{y}$ (i.e., a discrete representation). According to [16], learning to hash methods can be categorized as binarized hash (i.e., hash) [18], [19], [22]–[24], [30], [37]–[40] and quantization-based hash (i.e, quantization) [20], [21], [25], [26], [41]–[45]. The hash methods map input into binary codes while quantization methods map input into discrete codes. Both hash and quantization can improve inference efficiency and save memory space. However, it is pointed out that usually quantization methods achieve higher accuracy than hash methods [16], [46]. Existing quantization methods can be classified as shallow [20], [21], [41], [47] and deep quantization methods. As better performance is demonstrated with deep models, we focus on deep quantization methods in this paper. Existing methods in this category [25], [26], [42], [44], [45], [48]–[50] extract the feature representation and learn the quantization representation in an end-to-end deep model. The loss functions are defined based on learning to rank [51]. Some example functions include pointwise loss function [26], [49], pairwise loss function [42], [48], [50], and listwise loss function [52]. However, existing deep quantization methods are not designed for long-tail scenarios.

B. Learning from Long-tail Data

Long-tail distributions are commonly observed in real-world scenarios, garnering significant attention in this research area. We provide a comprehensive review of techniques falling into the following categories. Sampling-based approaches aim to achieve a balanced data representation by either oversampling the tail data [53]–[56] or undersampling the head data [57]–[59]. However, these methods carry a potential risk of overfitting on the tail data or underfitting on the head data [53], [57]. *Re-weighting based methods* assign varying weights to classes in their loss functions. Typically, these methods assign higher weights to tail classes and lower weights to head classes [60]–[62]. Re-weighting methods can be considered similar to data resampling methods [30], with the advantage of being more computationally efficient. *Data augmentation based methods* aim to generate additional data specifically for the tail classes. Various generation mechanisms have been proposed, such as generating data based on neighboring points [53], [54], [63], head data [64], [65], or even employing meta learning techniques [66]. Recent research has explored the utilization of *external memory* to store knowledge across head and tail data, enabling knowledge sharing between them [28], [30], [67]–[70].

It is crucial to acknowledge that most of these approaches were originally designed to address the imbalanced classification task, which is different from the searching and matching task studied in our research. Although some of these approaches might be relevant to our task, they could still encounter limitations like overfitting or underfitting. Additionally, the existing literature does not sufficiently address the challenges related to memory and inference efficiency.

C. Model Weight Ensemble

Model weight ensemble is a specific approach within ensemble learning. Recent studies [33], [71], [72] have investigated aggregating model weights to enhance the model's generalization ability. [71] proposes a method that approximates the maximization of the joint likelihood of the posteriors of the models' weights by averaging parameters. [33] suggests averaging models with the same task and architecture while employing different hyper-parameter configurations. [72] focuses on averaging newly-added model weights. However, these methods have primarily been developed for classification tasks, where the correspondence between class labels is clear.

In ensemble quantization models, the correspondence between codewords is unknown, making these existing methods inapplicable. Therefore, alternative approaches must be explored to address the unique challenges posed by ensemble quantization models.

III. Methodology

A. Problem Formulation

In the context of a long-tail dataset, defined as per Definition 1 and denoted by $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^N$, where $\boldsymbol{x}i$ represents the training data (such as images or text) and $y_i \in 1, 2, ..., C$ corresponds to the label of $\boldsymbol{x}i$, the objective of long-tail quantization is to obtain a set of M discretized functions $q_j(\cdot)_{j=1}^M$. The use of quantization enables more efficient semantic search.

Formally, an instance $\boldsymbol{x}_i$ can be represented by $\boldsymbol{b}_i$ using the M discretized functions:

$$\boldsymbol{b}_i \triangleq [q_1(f(\boldsymbol{x}_i)), ..., q_M(f(\boldsymbol{x}_i))], \tag{1}$$

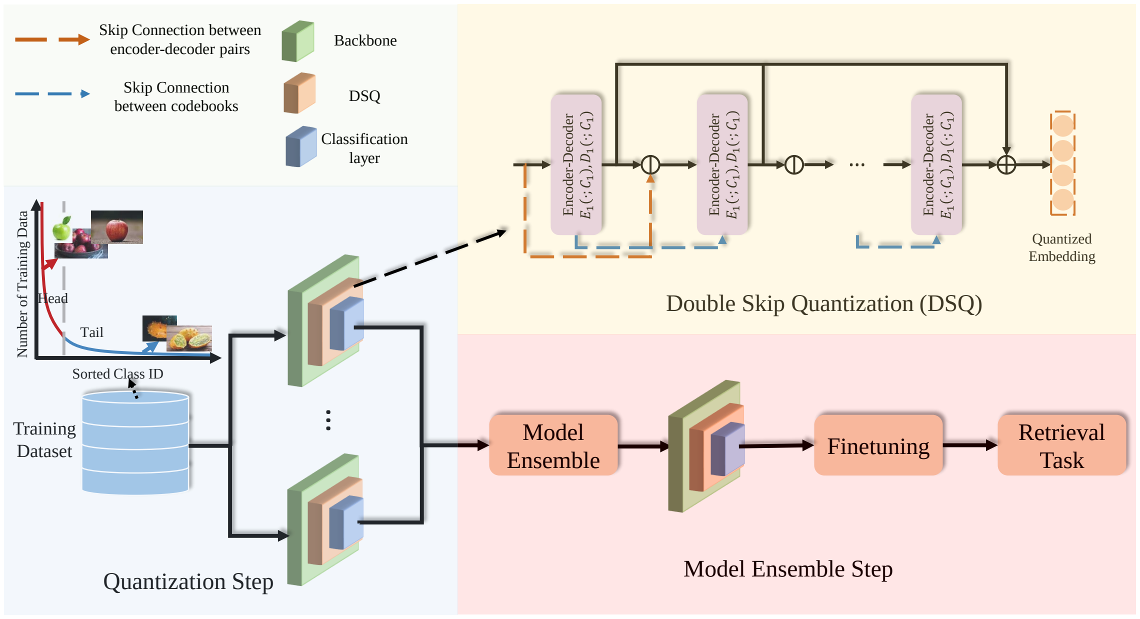

Skip Connection between encoder-decoder pairs
Skip Connection between codebooks
Backbone
DSQ
Classification layer
Number of Training Data
Head
Tail
Sorted Class ID
Training Dataset
Quantization Step
Encoder-Decoder $E_1(\cdot;C_1), D_1(\cdot;C_1)$
Encoder-Decoder $E_1(\cdot;C_1), D_1(\cdot;C_1)$
Encoder-Decoder $E_1(\cdot;C_1), D_1(\cdot;C_1)$
Quantized Embedding
Double Skip Quantization (DSQ)
Model Ensemble
Finetuning
Retrieval Task
Model Ensemble Step

Fig. 1. The framework of LightLT.

where $f(\cdot)$ is the backbone model, $f(\boldsymbol{x}_i)$ is the continuous representation with a d-dimension, and $q_j(\cdot) \in 1, 2, ..., K$. The discrete representation $\boldsymbol{b}_i$ should fulfill the following requirements: 1) It should be capable of approximating the input continuous representation $f(\boldsymbol{x}_i)$. 2) It should preserve the semantic information of the data. 3) It should be efficient in terms of both storage and inference.

Definition 1 (Long-tail Dataset)**.** *For a dataset $\mathcal{D}$, we denote the number of data points with the i-th class as π_i. Without loss of generality, we assume $\pi_1 \geq \pi_2 \geq ... \geq \pi_C$. $\mathcal{D}$ is a long-tail dataset if the class size approximately satisfies that $\pi_i = \pi_1 \cdot i^{-\mu}$, where μ is a positive value. It is known as Zipf's law as well. The imbalance factor (**IF**) is measured by π_1/π_C.*

B. Overview

The proposed LightLT framework comprises two key stages, as illustrated in Fig. 1: 1) The quantization step, and 2) the model ensemble step. In the quantization step, encoders process the continuous representation obtained from the backbone model and generate discrete codes for the input. Subsequently, decoders reconstruct an approximate representation using these discrete codes. To ensure stable gradients and expressive representations, the architecture of encoders and decoders incorporates double skip quantization (DSQ), as discussed in Section III-C. A novel loss function is introduced to make the quantized representation both discriminative and informative, as detailed in Section III-D. This loss function aids in obtaining meaningful and useful representations. Additionally, to mitigate overfitting on tail data and enhance generalization, a novel ensemble module is proposed, as described in Section III-E.

C. Double Skip Quantization

The proposed Double Skip Quantization (DSQ) is a composite mapping function that operates between continuous spaces in $\mathbb{R}^d$. To align these two continuous spaces, we leverage multiple encoders $E_i(\cdot)$ and decoders $D_i(\cdot)$ (where $i = 1, 2, ..., M$), and the DSQ function is represented as the composition of these encoders and decoders. The encoder maps the d-dimensional continuous representation into a discrete code ranging from 1 to K, which serves as the quantized representation (compression output), i.e., $E_i(\cdot) : \mathbb{R}^d \to 1, 2, ..., K$. Correspondingly, the decoder $D_i(\cdot)$ maps the discrete input back into continuous space, i.e., $D_i(\cdot) : \{1, 2, ..., K\} \to \mathbb{R}^d$. The sum of decoders' output approximates the input continuous representation $f(\cdot)$. To minimize memory usage, we ensure that the encoder and decoder in each encoder-decoder pair (i.e., $E_i(\cdot)$ and $D_i(\cdot)$) share the same parameters and architecture, following common practices adopted by quantization methods [41], [44]. In the upcoming sections, we will provide a detailed description of the DSQ's topology architecture and the design of the encoders and decoders.

1) Topology Architecture of DSQ: Given the input vector of DSQ, $f(\boldsymbol{x}_i)$, the architecture generates M discrete codes. More specifically, the encoded discrete representation $\boldsymbol{b}_i$ (i.e., encoder output) and reconstructed representation $\boldsymbol{o}_i$ (i.e., decoder output) can be formulated as

$$\boldsymbol{b}_i = [E_1(f(\boldsymbol{x}_i)), ..., E_M(f(\boldsymbol{x}_i))], \boldsymbol{o}_i = \sum_{j=1}^{M} D_j(E_j(f(\boldsymbol{x}_i))),$$

respectively. However, one limitation of this design is the lack of diversity among the encoder-decoder pairs. Without any constraints, the M encoders and decoders tend to become highly similar, essentially memorizing the same primary signals in the input. As a result, this redundancy in the quantized representations hampers the full utilization of the multiple encoder-decoder pairs. It is crucial, therefore, to encourage diversity among these encoder-decoder pairs. By encouraging diversity, the discrete codes can capture complementary information from the input, enabling a more effective utilization of the entire DSQ architecture.

One approach to enhance diversity in models involves incorporating diversity regularizers, such as the Frobenius norm, von Neumann divergence [73], and log-determinant divergence citekulis2009low, into the loss function. These regularizers aim to minimize the similarity among the parameters of encoders and decoders. However, a drawback of these methods is their significant computational expense. With each encoder-decoder pair, we must compute their diversity regularizers, resulting in $M(M-1)/2$ terms, which leads to quadratic complexity relative to the number of encoder-decoder pairs. For instance, if $M=8$, this requires computing 28 regularizers, making the computation cost prohibitively high. Moreover, due to the large number of regularizers, it becomes challenging to ensure that each regularizer term attains its minimal value.

The limitations of diversity regularizers motivate us to incorporate a new module for encoder-decoder diversity. Specifically, we propose a skip connection to learn diverse encoders and decoders. The skip connection stacks the encoder-decoder pairs in a sequential way so that the input of each encoder is the residual of previous encoder-decoder pairs. This sequential stacking ensures that each encoder's input is distinct from the others, enabling them to extract diverse information more effectively. Formally, the k-th pair of encoder and decoder can be represented as

$$\boldsymbol{b}_i[k] = E_k(f(\boldsymbol{x}_i) - \sum_{j=1}^{k-1} \boldsymbol{o}_i^j), \boldsymbol{o}_i^k = D_k(\boldsymbol{b}_i[k]), \tag{2}$$

and the final reconstructed representation is $\boldsymbol{o}_i = \sum_{j=1}^{M} \boldsymbol{o}_i^j$, where $\boldsymbol{b}_i[k]$ is the k-th entry of discrete representation $\boldsymbol{b}_i$ and $\boldsymbol{o}_i^k$ is the k-th reconstructed representation with respect to input representation $f(\boldsymbol{x}_i)$. According to Eqn. 2, the output of the k-th decoder has a skip connection with the input of the k-th encoder, and the residual between them serves as the input to the next encoder. Therefore, the input to the encoder $E_k(\cdot)$ is different from each other, and this design could greatly improve the diversity among the pairs of encoders and decoders. This is the first "skip" connection in the DSQ module.

2) Encoder and Decoder Design: In pursuit of enhancing both storage and inference efficiency, we adopt a strategy where every encoder-decoder pair in the architecture shares identical parameters. In other words, we utilize a common codebook $\boldsymbol{C}_k \in \mathbb{R}^{K\times d}$ across all encoder-decoder pairs.

For convenience, we name the rows in a codebook as codewords. The encoder $E_k(\cdot)$ conducts the encoding by codeword selection: It computes the similarity between input representation $\boldsymbol{e}_i^k = f(\boldsymbol{x}_i) - \sum_{j=1}^{k-1} \boldsymbol{o}_i^j$ and each codeword in a codebook $\boldsymbol{C}_k$, and the index of the codeword that is the most similar to the input is the encoder output. Formally, the encoded discrete representation is

$$E_k(\boldsymbol{e}_i^k; \boldsymbol{C}_k[j]) = \arg\max_j s(\boldsymbol{e}_i^k, \boldsymbol{C}_k[j]) \triangleq \boldsymbol{b}_i[k], \tag{3}$$

where $s(\cdot,\cdot)$ is the similarity function, e.g. negative Euclidean distance or cosine similarity, and $\boldsymbol{C}_k[j]$ is a codeword stored in the j-th row of $\boldsymbol{C}_k$.

Then the decoder retrieves the closest codeword based on the index to reconstruct the input representation, i.e.

$$D_k(\boldsymbol{b}_i[k]; \boldsymbol{C}_k) = \boldsymbol{C}_k[\boldsymbol{b}_i[k]] \triangleq \boldsymbol{o}_i^k. \tag{4}$$

Although this architecture effectively achieves the compression objective, its training poses challenges. The main difficulty lies in the non-differentiability of the $\arg\max$ function, which hinders end-to-end model training. To address this issue, we draw inspiration from [35] and introduce the softmax with temperature to smoothen the $\arg\max$ operation. Additionally, we employ the Straight-Through Estimator [36] to estimate the gradient, leading to the following formulation:

$$\hat{\boldsymbol{b}}_i[k] = \text{Softmax}(s(\boldsymbol{e}_i^k, \boldsymbol{C}_k[j])/t), \tag{5}$$

$$\tilde{\boldsymbol{b}}_i[k] = \hat{\boldsymbol{b}}_i[k] + \text{Sg}(\text{One-Hot}(\hat{\boldsymbol{b}}_i[k]) - \hat{\boldsymbol{b}}_i[k]), \tag{6}$$

$$\boldsymbol{o}_i^k = \boldsymbol{C}_k^T \tilde{\boldsymbol{b}}_i[k], \tag{7}$$

where Sg means stop gradient operation (i.e., the optimizer does not compute the gradient for the stop gradient term when backpropagation). In this way, the model uses one-hot index $\boldsymbol{b}_i[k]$ in the forward propagation while the smoothed $\hat{\boldsymbol{b}}_i[k]$ is used in backward propagation to ensure model gradient exists.

By utilizing the strategy mentioned above, both the encoder and decoder can be effectively trained in an end-to-end manner. However, the application of the softmax function can pose challenges in stably calculating the gradient of the reconstructed representation to the codebooks, particularly when more encoder-decoder pairs are integrated. Consequently, the addition of more encoder-decoder pairs only offers minimal performance improvements due to these limitations in gradient calculation. More specifically, consider the gradient of $\boldsymbol{o}_i^M$ to $\boldsymbol{C}_1$:

$$\frac{\partial \boldsymbol{o}_i^M}{\partial \boldsymbol{C}_1} = \frac{\partial \boldsymbol{o}_i^M}{\partial \boldsymbol{o}_i^{M-1}} \frac{\partial \boldsymbol{o}_i^{M-1}}{\partial \boldsymbol{C}_1} + ... + \frac{\partial \boldsymbol{o}_i^M}{\partial \boldsymbol{o}_i^1} \frac{\partial \boldsymbol{o}_i^1}{\partial \boldsymbol{C}_1}. \tag{8}$$

To make it easier to analyze, let's assume $M=4$ and unfold it as follows:

$$\begin{aligned}\frac{\partial \boldsymbol{o}_i^4}{\partial \boldsymbol{C}_1} &= \frac{\partial \boldsymbol{o}_i^4}{\partial \boldsymbol{o}_i^3}\frac{\partial \boldsymbol{o}_i^3}{\partial \boldsymbol{o}_i^2}\frac{\partial \boldsymbol{o}_i^2}{\partial \boldsymbol{o}_i^1}\frac{\partial \boldsymbol{o}_i^1}{\partial \boldsymbol{C}_1} + \frac{\partial \boldsymbol{o}_i^4}{\partial \boldsymbol{o}_i^2}\frac{\partial \boldsymbol{o}_i^2}{\partial \boldsymbol{o}_i^1}\frac{\partial \boldsymbol{o}_i^1}{\partial \boldsymbol{C}_1} \\ &+ \frac{\partial \boldsymbol{o}_i^4}{\partial \boldsymbol{o}_i^3}\frac{\partial \boldsymbol{o}_i^3}{\partial \boldsymbol{o}_i^1}\frac{\partial \boldsymbol{o}_i^1}{\partial \boldsymbol{C}_1} + \frac{\partial \boldsymbol{o}_i^4}{\partial \boldsymbol{o}_i^1}\frac{\partial \boldsymbol{o}_i^1}{\partial \boldsymbol{C}_1}.\end{aligned} \tag{9}$$

Based on Eqn. 9, the first term corresponds to the gradient crossing all previous layers. Because $\boldsymbol{o}_i^k = D_k(E_k(f(\boldsymbol{x}_i - \sum_{j=1}^{k-1} \boldsymbol{o}_i^j)))$, $\frac{\partial \boldsymbol{o}_i^4}{\partial \boldsymbol{o}_i^j}$ $(j=1,2,3)$ includes the gradient with respect to softmax function. Because the output of the softmax function is similar to a one-hot vector, its gradient tends to approach zero. Consequently, when multiple small gradients are multiplied together, the resulting value of the gradient may be further diminished, leading to a loss of information. Similarly, as the gradient passes through multiple layers, it can be progressively reduced in magnitude. As a result, many terms in the gradient of the reconstructed representation in the last few layers with respect to the codebooks in the first few layers become too small to propagate effectively. This limitation hampers significant performance improvement when employing additional encoders and decoders.

To mitigate the issue of gradient information loss, we propose a novel skip connection among codebooks, and it is the second "skip" in the proposed double skip quantization. Formally, the codebook $\boldsymbol{C}_k$ has a skip connection with $\boldsymbol{C}_{k-1}$, which can be represented as

$$\boldsymbol{C}_k = \text{FFN}(\boldsymbol{C}_{k-1}) \cdot g_k + \boldsymbol{P}_k, \tag{10}$$

where $\text{FFN}(\cdot)$ is a feedforward neural network with one hidden layer and ReLU activation, g_k is a learnable weight, and $\boldsymbol{P}_k \in \mathbb{R}^{K\times d}$. According to Eqn. 10, the codebook $\boldsymbol{C}_k$ is a combination of two parts: 1) the transformation of codebook based on previous step $\boldsymbol{C}_{k-1}$, and 2) a main codebook $\boldsymbol{P}_k$ specific to the current encoder and decoder. If the gate g_k is close to zero, $\boldsymbol{C}_k$ takes less information from $\boldsymbol{C}_{k-1}$, and vice versa. The advantage of the skip connection is that the gradient crossing multiple layers will not vanish. The gradient of $\boldsymbol{o}_i^M$ to $\boldsymbol{C}_1$ converts to:

$$\begin{aligned}\frac{\partial \boldsymbol{o}_i^M}{\partial \boldsymbol{C}_1} = \frac{\partial \boldsymbol{o}_i^M}{\partial \boldsymbol{o}_i^{M-1}}\frac{\partial \boldsymbol{o}_i^{M-1}}{\partial \boldsymbol{C}_1} + \dots + \frac{\partial \boldsymbol{o}_i^M}{\partial \boldsymbol{o}_i^1}\frac{\partial \boldsymbol{o}_i^1}{\partial \boldsymbol{C}_1} \\ + \frac{\partial \boldsymbol{o}_i^M}{\partial \boldsymbol{C}_M}\frac{\partial \boldsymbol{C}_M}{\partial \boldsymbol{C}_{M-1}}\dots\frac{\partial \boldsymbol{C}_2}{\partial \boldsymbol{C}_1}.\end{aligned} \tag{11}$$

Because $\frac{\partial \boldsymbol{C}_k}{\partial \boldsymbol{C}_{k-1}} = g_k \frac{\partial \text{FFN}(\boldsymbol{C}_{k-1})}{\partial \boldsymbol{C}_{k-1}}$, it is more likely to make the gradient backpropagation stable (e.g., ReLU can prevent gradient vanishing). As the gradient can pass multiple layers without vanishing, we can use more encoders and decoders to make the residual as small as possible and reduce the compression loss.

D. Loss Function

To improve downstream searching and matching results, it is crucial to ensure the high quality of the quantized representations. To achieve this, we establish three fundamental criteria: 1) The quantized representations must be informative and discriminative concerning the class labels. 2) The quantized representations for items within the same class should exhibit similarity, facilitating better grouping of similar items. 3) The quantized representations should possess robustness, ensuring that the distance between quantized representations for items within the same class is smaller than the distance between those belonging to different classes. To meet these criteria, we introduce three corresponding loss functions: the classification loss, center loss, and ranking loss. These loss functions will be explained in detail in the following sections.

1) Classification Loss: To achieve discriminative quantized representations, we employ the cross-entropy classification loss. This loss function has demonstrated effectiveness in distinguishing representations belonging to distinct classes and accurately defining class boundaries. Nevertheless, when dealing with a training set that adheres to a long-tail distribution, a direct application of cross-entropy loss might cause the model to prioritize head data over tail data, impeding the accurate determination of decision boundaries. To address this challenge posed by the significant class imbalance, we choose the class-weighted cross-entropy loss [60]:

$$\mathcal{L}_{ce} = -\sum_{i=1}^{N} \frac{1-\gamma}{1-\gamma^{\pi_{y_i}}} \sum_{j=1}^{C} \mathbb{I}(j = y_i) \log(\hat{\boldsymbol{y}}_i[j]), \tag{12}$$

where $\hat{\boldsymbol{y}}_i = \text{Softmax}(\text{FC}(\boldsymbol{o}_i))$ is the prediction of quantized representation $\boldsymbol{o}_i$ ($\text{FC}(\cdot)$ is the fully-connected layer), $\hat{\boldsymbol{y}}_i[j]$ is the j-th entry of $\hat{\boldsymbol{y}}_i$, $\mathbb{I}(\cdot)$ is the indicator function, and $\gamma \in [0, 1)$ is a hyper-parameter. When $\gamma = 0$, Eqn. 12 degrades into the standard cross-entropy loss, while $\gamma \to 1$, $\frac{1-\gamma}{1-\gamma^{\pi_{y_i}}}$ approaches the reciprocal of π_{y_i}. In subsequent sections, we refer to the class-weighted cross-entropy loss as the "cross-entropy loss" for brevity.

2) Center Loss: The cross-entropy loss primarily focuses on the separation between classes at the boundary, neglecting the promotion of compact intra-class representations. However, compact intra-class representations hold significant importance for downstream searching tasks such as nearest neighbor search [26], [74], which are commonly employed. To enhance the centralization of intra-class representations, we incorporate the center loss, as introduced in [74]. This entails assuming that each class possesses a prototype, and quantized representations within a given class should closely align with the prototype. To express this concept more precisely, the center loss can be formulated as follows:

$$\mathcal{L}_c = \sum_{i=1}^{N} \|\boldsymbol{z}_{y_i} - \boldsymbol{o}_i\|_p, \tag{13}$$

where $\boldsymbol{z}_{y_i} \in \mathbb{R}^d$ is the prototype of class y_i, and $\|\cdot\|_p$ represents the ℓ_p-norm.

3) Ranking Loss: While the center loss enhances the compactness of intra-class representations, it solely considers the absolute distance between data representations and their respective prototypes, overlooking their relative distances. Neglecting this relative order can lead to inaccuracies in the output. For instance, a representation positioned near the boundary might be equidistant from two distinct class prototypes, but it should be closer to the prototype of the class it truly belongs to. To address this issue, we introduce the ranking loss, denoted as $\mathcal{L}_r$, which ensures that each representation remains closer to its corresponding prototype than to the prototypes of other classes. This ranking loss aims to minimize the following loss function:

$$\mathcal{L}_r = -\sum_{i=1}^{N} \log \frac{\exp(-\|\boldsymbol{o}_i - \boldsymbol{z}_{y_i}\|_p/\tau)}{\sum_{j=1}^{C}\exp(-\|\boldsymbol{o}_i - \boldsymbol{z}_j\|_p/\tau)}, \tag{14}$$

where τ is positive hyper-parameter.

4) Final Loss Function: The final loss is the combination of the proposed three loss functions, i.e.

$$\min_{\Theta} \mathcal{L} = \mathcal{L}_{ce} + \alpha(\mathcal{L}_c + \mathcal{L}_r), \tag{15}$$

where Θ is the set of model parameters, and α is the hyper-parameter to control the weight of center loss and ranking loss, respectively.

Interestingly, we find the proposed loss function term $\mathcal{L}_c + \mathcal{L}_r$ is an upper bound of widely used triplet loss [51], i.e.

$$L_t = \sum_i \sum_{j\in\{y_i\}} \sum_{k\notin\{y_i\}} \max(\|\boldsymbol{o}_i - \boldsymbol{o}_j\| - \|\boldsymbol{o}_i - \boldsymbol{o}_k\| + m, 0),$$

where m is the margin and $\{y_i\}$ represents the set of data with the label y_i, according to Proposition 1.

The triplet loss incurs significant computational complexity, reaching $\mathcal{O}(N^3)$. This high complexity becomes impractical when dealing with large-scale datasets. However, the proposed loss function offers an approximation of the triplet loss with linear time complexity, specifically $\mathcal{O}(N)$. As a result, the proposed loss function is both feasible and efficient for large-scale datasets, leading to improved downstream searching results.

Proposition 1. *$\mathcal{L}_c + \mathcal{L}_r$ is an upper bound of triplet loss approximately.*

Proof. The triplet loss can be represented as

$$\mathcal{L}_t = \sum_i \sum_{j\in\{y_i\}} \sum_{k\notin\{y_i\}} \max(\|\boldsymbol{o}_i - \boldsymbol{o}_j\| - \|\boldsymbol{o}_i - \boldsymbol{o}_k\| + m, 0),$$

where m is the margin, and $\{y_i\}$ represents the set of data with the label y_i. For the sake of analysis, we consider the simplified triplet loss $\mathcal{L}_t = \sum_i \sum_{j\in\{y_i\}} \sum_{k\notin\{y_i\}} \|\boldsymbol{o}_i - \boldsymbol{o}_j\| - \|\boldsymbol{o}_i - \boldsymbol{o}_k\|$. Considering

$$\|\boldsymbol{o}_i - \boldsymbol{o}_j\| \leq \|\boldsymbol{o}_i - \boldsymbol{z}_{y_i}\| + \|\boldsymbol{o}_j - \boldsymbol{z}_{y_i}\| \tag{16}$$

and

$$\|\boldsymbol{o}_i - \boldsymbol{o}_k\| \geq \|\boldsymbol{o}_i - \boldsymbol{z}_{y_k}\| - \|\boldsymbol{o}_k - \boldsymbol{z}_{y_k}\|, \tag{17}$$

we have

$$\begin{aligned}\|\boldsymbol{o}_i - \boldsymbol{o}_j\| - \|\boldsymbol{o}_i - \boldsymbol{o}_k\| \leq& \|\boldsymbol{o}_i - \boldsymbol{z}_{y_i}\| + \|\boldsymbol{o}_j - \boldsymbol{z}_{y_i}\| \\ &-(\|\boldsymbol{o}_i - \boldsymbol{z}_{y_k}\| - \|\boldsymbol{o}_k - \boldsymbol{z}_{y_k}\|).\end{aligned} \tag{18}$$

Therefore, we have

$$\begin{aligned}&\sum_i \sum_{j\in\{y_i\}} \sum_{k\notin\{y_i\}} \|\boldsymbol{o}_i - \boldsymbol{o}_j\| - \|\boldsymbol{o}_i - \boldsymbol{o}_k\| \\ \leq& \sum_i \sum_{j\in\{y_i\}} \sum_{k\notin\{y_i\}} \|\boldsymbol{o}_i - \boldsymbol{z}_{y_i}\| + \|\boldsymbol{o}_j - \boldsymbol{z}_{y_i}\| \\ &-(\|\boldsymbol{o}_i - \boldsymbol{z}_{y_k}\| - \|\boldsymbol{o}_k - \boldsymbol{z}_{y_k}\|).\end{aligned} \tag{19}$$

And we can find that

$$\sum_i \sum_{j\in\{y_i\}} \sum_{k\notin\{y_i\}} \|\boldsymbol{o}_i - \boldsymbol{z}_{y_i}\| + \|\boldsymbol{o}_j - \boldsymbol{z}_{y_i}\| = \mathcal{L}_c. \tag{20}$$

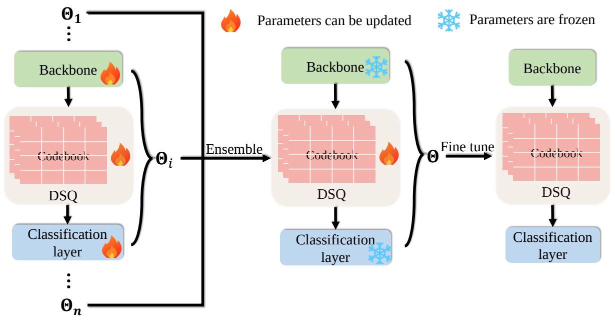


Fig. 2. The framework of model ensemble and fine-tuning.

And we have

$$\begin{aligned}&\sum_i \sum_{j\in\{y_i\}} \sum_{k\notin\{y_i\}} \|\boldsymbol{o}_i - \boldsymbol{z}_{y_k}\| - \|\boldsymbol{o}_k - \boldsymbol{z}_{y_k}\| \\ =& n_{y_i} \sum_i \sum_{k\notin\{y_i\}} \|\boldsymbol{o}_i - \boldsymbol{z}_{y_k}\| - \|\boldsymbol{o}_k - \boldsymbol{z}_{y_k}\|,\end{aligned} \tag{21}$$

where n_{y_i} is the number of training data with label y_i. Then let's consider the loss $\mathcal{L}_r$.

$$\begin{aligned}\mathcal{L}_r &= -\sum_{i=1}^{N} \log \frac{\exp(-\|\boldsymbol{o}_i - \boldsymbol{z}_{y_i}\|_p/\tau)}{\sum_{j=1}^{C} \exp(-\|\boldsymbol{o}_i - \boldsymbol{z}_j\|_p/\tau)} \\ &= \sum_{i=1}^{N} \log(1 + \sum_{j\neq y_i} \exp(\|\boldsymbol{o}_i - \boldsymbol{z}_{y_i}\|/\tau - \|\boldsymbol{o}_i - \boldsymbol{z}_j\|/\tau)) \\ &\approx \sum_{i=1}^{N} \sum_{j\neq y_i} \exp(\|\boldsymbol{o}_i - \boldsymbol{z}_{y_i}\|/\tau - \|\boldsymbol{o}_i - \boldsymbol{z}_j\|/\tau) (\text{Taylor expansion}) \\ &\approx \sum_{i=1}^{N} \sum_{j\neq y_i} \|\boldsymbol{o}_i - \boldsymbol{z}_{y_i}\|/\tau - \|\boldsymbol{o}_i - \boldsymbol{z}_j\|/\tau.\end{aligned} \tag{22}$$

Therefore,

$$\sum_i \sum_{j\in\{y_i\}} \sum_{k\notin\{y_i\}} \|\boldsymbol{o}_i - \boldsymbol{z}_{y_k}\| - \|\boldsymbol{o}_k - \boldsymbol{z}_{y_k}\| \approx \mathcal{L}_r$$

when $\tau = 1$. Therefore, $\mathcal{L}_c + \mathcal{L}_r$ can be considered as an upper bound of triplet loss approximately. □

E. Model Ensemble and Fine-tuning

To address the issue of the long-tail distribution, we have adopted the class-weighted cross-entropy loss $\mathcal{L}_{ce}$. However, this approach has a drawback - it can lead to overfitting on tail data due to the assignment of much higher weights to tail classes. Consequently, the quality of the learned representations is compromised, resulting in reduced retrieval accuracy. One potential solution to mitigate overfitting is to utilize ensemble methods like bagging, as demonstrated in prior research [75]. Bagging has proven effective in preventing overfitting. Nevertheless, a significant drawback of bagging is its increased space and time requirements. This stems from the need to average multiple model predictions, resulting in a higher storage space cost. For instance, if there are n models, bagging would require n times the storage space compared to a single model.

To solve this problem, inspired by [33], we adopt the model ensemble with respect to parameters. Formally, we train the proposed LightLT with different initialization for n times, and $\{\Theta_i\}_{i=1}^n$ are their corresponding parameters. Then the final model parameter is the average of these parameters, i.e.

$$\Theta = \frac{1}{n}\sum_{i=1}^{n}\Theta_i. \tag{23}$$

But before applying this model ensemble to quantization, we need to first solve the codeword alignment problem. For each encoder, we use the index of codeword closest to the input as the encoding result. Thus, if we multiply the codebook a permutation matrix, the encoding results will not change. Specifically, the selected codeword is represented as $\boldsymbol{C}_k^T\tilde{\boldsymbol{b}}_i[k]$. Consider a permutation matrix $\boldsymbol{P}$. If we permute the codebook $\boldsymbol{C}_k$, i.e. $\boldsymbol{C}_k\boldsymbol{P}$, the index vector $\tilde{\boldsymbol{b}}_i[k]$ will also be permuted as $\boldsymbol{P}\tilde{\boldsymbol{b}}_i[k]$ based on Eqn. 5. Then the output is still $\boldsymbol{C}_k^T\tilde{\boldsymbol{b}}_i[k]$ because $\boldsymbol{C}_k^T\boldsymbol{P}^T\boldsymbol{P}\tilde{\boldsymbol{b}}_i[k] = \boldsymbol{C}_k^T\tilde{\boldsymbol{b}}_i[k]$. Therefore, the index of each codeword is not unique. During ensemble, codewords in each Θ_i may not have one-to-one correspondence. As a result, the averaged codewords do not make sense. We provide a concrete example in Example. 1.

example 1. *Assume codebooks* $\boldsymbol{C}_1^1 = \begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}$ *and* $\boldsymbol{C}_1^2 = \begin{bmatrix}2.9 & 4\\ 5.1 & 6\\ 1 & 2.1\end{bmatrix}$ *are the first codebook of* Θ_1 *and* Θ_2 *respectively. The two codebooks have the relation* $\boldsymbol{C}_1^1 \approx \boldsymbol{P}\boldsymbol{C}_1^2$, *where* $\boldsymbol{P} = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 0 & 0\end{bmatrix}$ *is a permutation matrix. Therefore, each codeword of the two codebooks does not correspond to the same ID. The mean of* $\boldsymbol{C}_1^1$ *and* $\boldsymbol{C}_1^2$ *is* $\begin{bmatrix}1.95 & 3\\ 4.05 & 5\\ 3 & 4.05\end{bmatrix}$, *which has lost the information of codewords in codebooks.*

In order to solve this problem, we propose an effective method which is shown in Fig. 2. We fix the Θ except the parameters of DSQ, and fine-tune DSQ parameters for several epochs, which can be formulated as $\min_{\boldsymbol{\Theta}_{DSQ}}\mathcal{L}$, where $\boldsymbol{\Theta}_{DSQ}$ represents parameters of DSQ module. In this way, backbone module and classification layer can enjoy the benefit of the ensemble to generate better representations, and the DSQ module can re-learn high-quality codebooks via the fine-tuning step.

We summarize the whole training steps in Algorithm 1.

Algorithm 1: LightLT

```
Input: Training set $\{\mathcal{D}\}$; the number of codebooks $M$;
       the number of codewords $K$; hyper-parameter
       $\alpha$ and $\beta$;the number of ensemble models $n$.
1  Initialize model parameters $\{\Theta_i\}_{i=1}^n$.
   // Train backbone and DSQ.
2  for $i \leftarrow 1$ to $n$ do
3      while converge do
4          for batch in $\mathcal{D}$ do
5              Forward propagation based on Eqn. 15 with
                 respect to $\Theta_i$;
6              Update model weight $\Theta_i$;
   // Model ensemble and fine-tuning.
7  Average $n$ model weights to get ensemble model $\Theta$;
8  Fix backbone and classifier; while converge do
9      for batch in $\mathcal{D}$ do
10         Forward propagation based on Eqn. 15 with
             respect to the averaged model;
11         Update DSQ weight $\boldsymbol{\Theta}_{DSQ}$;
12 return $\Theta$
```

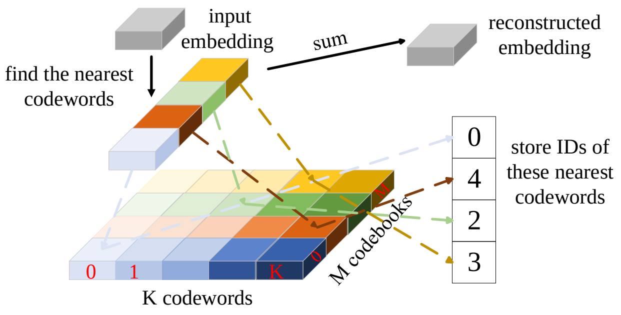


Fig. 3. The workflow of the indexing codewords.

IV. Complexity Analysis

In this section, we provide an overview of the indexing process for codewords. Subsequently, we conduct an analysis of the space complexity and inference complexity of LightLT. The indexing process is illustrated in Fig. 3. Given an input embedding (representing database data) denoted as $\boldsymbol{o}_i$, the algorithm identifies the nearest codeword from M codebooks using Eqn. 3. The IDs of these selected codewords are then stored. The reconstructed embedding is obtained by summing up these selected codewords.

A. Space Complexity

In this section, we analyze the space complexity of the proposed LightLT. We aim to determine the storage requirements for computing the distance between a query $\boldsymbol{q}$ and the encoded database data representation $\boldsymbol{o}_i$. This distance can be represented as

$$\|\boldsymbol{q} - \boldsymbol{o}_i\|^2 = \|\boldsymbol{q} - \sum_{j=1}^{M}\boldsymbol{o}_i^j\|^2 = \|\boldsymbol{q}\|^2 + \|\sum_{j=1}^{M}\boldsymbol{o}_i^j\|^2 - 2\sum_{j=1}^{M}\langle\boldsymbol{q}, \boldsymbol{o}_i^j\rangle. \tag{24}$$

Let us consider the information that needs to be stored for distance computation. First, the codebooks need to be stored, which comes at a cost of $4KMd$ Bytes. Then, only codeword

TABLE I
STATISTICS OF DATASETS.

	IF=50						IF=100					
	C	π_1	π_C	n_{train}	n_{query}	n_{db}	C	π_1	π_C	n_{train}	n_{query}	n_{db}
Cifar100	100	500	10	3,732	10k	50k	100	500	5	2,598	10k	50k
ImageNet100	100	1.3k	26	9,437	5k	130k	100	1.3k	13	6,834	5k	130k
NC	10	29k	584	52,027	2k	65k	10	29k	292	45,300	2k	72k
QBA	25	10k	199	29,236	5k	636k	25	10k	99	23,527	5k	642k

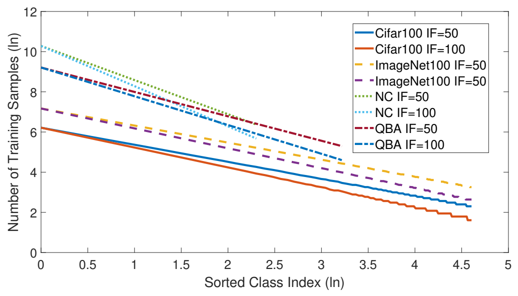


Fig. 4. Label distributions of datasets with different IF values.

indices and $\|\sum_{j=1}^{M}\boldsymbol{o}_i^j\|^2$ should be stored, which entail costs of $\frac{1}{8}n_d M\log K$ and $4n_d$ Bytes respectively, where n_e is the number of database data. It is worth mentioning that the term $\|\sum_{j=1}^{M}\boldsymbol{o}_i^j\|^2$ can also be computed through table look-up operations to further reduce space consumption. However, the dimension of pre-trained model representation is usually large (e.g., 768 for BERT-based, 1,024 for BERT-large). Therefore, the additional cost of one Byte to store norm of representation is negligible compared to the storage requirements of high-dimensional representations. As a result, the total space complexity is given by $4KMd + \frac{1}{8}n_d M\log K + 4n_d$ Bytes, and the compression ratio can be approximated as $\frac{32d}{M\log K+32}$ if $n_d \gg KMd$.

B. Inference Complexity

In this section, we analyze the inference complexity of LightLT. According to Eqn. 24, the process invovles pre-computing the distances between the query $\boldsymbol{q}$ and codewords in each codebook. Subsequently, we perform a simple look-up of these pre-computed distances during inference. As a result, the distance computation in LightLT requires $\mathcal{O}(dMK)$ operations. In comparison, the exhaustive search method incurs a complexity of $\mathcal{O}(n_e d)$. However, by employing LightLT, the inference complexity is significantly improved, presenting a substantial advantage over exhaustive search.

V. EXPERIMENT

In this section, we evaluate the proposed LightLT with the goal of answering the following research questions: 1) How does LightLT perform when compared with state-of-the-art baselines? 2) Does the proposed loss functions and DSQ module effectively contribute to learning better embeddings? 3) Can LightLT enhance inference and memory efficiency? 4) How does the performance vary concerning different hyperparameters?

A. Datasets and Experiment Settings

1) Datasets: We construct eight long-tail datasets using three well-known public benchmarks including Cifar100 [76], ImageNet100 [24], Amazon News (NC) [77] and 1 Amazon query dataset (QBA). Following [30], we split these datasets based on Zipf's law with IF set to 50 and 100. We summarize the statistics of datasets in Table I and visualize the class label distributions of each dataset in Fig. 4. In Fig. 4, C represents the number of classes, π_1 represents the number of data of the class with the largest amount of data, π_C represents the number of data of the class with the least amount of data, and n_{train}, n_{query}, n_{db} represent the size of the training set, the query set and the database, respectively.

2) Baselines: For image data, following [30], we compare the proposed LightLT with the representative and state-of-the-art baselines: LSH [78], PCAH [18], ITQ [18], KNNH [79], SDH [23], COSDISH [80], FastHash [22], FSSH [81], SCDH [45], DPSH [82], HashNet [24], DSDH [83], CSQ [38], and LTHNet [30]. The first nine methods are shallow models and the last five methods are deep models. Regarding text data, we compare LightLT with five state-of-the-art baselines, including LSH [78], PQ [20], DPQ [45], KDE [44], LTHNet [30]. Among these, the first two methods are shallow models, and the last three are deep models.

3) Evaluation Metrics: To evaluate the proposed model, following [30], we use Mean Average Precision (MAP). It is widely used in hashing and quantization literature such as [24], [30], [83] to measure ranking results. Specifically, for a given query set, the Average Precision (AP) of each query can be computed by AP@$n_{db} = \frac{\sum_{i=1}^{n_{db}} P(i)\delta(i)}{\sum_{i=1}^{n_{db}}\delta(i)}$, where $P(i)$ represents the precision of the i-th retrieved results, $\delta(i) = 1$ if the i-th retrieved result is relative to the query, otherwise $\delta(i) = 0$, and n_{db} denotes the number of data in the database. MAP is the mean of each query AP value, which can be formulated as MAP $= \frac{\sum_{i=1}^{n_{query}} \text{AP@}n_{db}(i)}{n_{query}}$, where AP@$n_{db}(i)$ represents the AP value of the i-th query. For MAP, the higher the better.

4) Implementation Details: To make a fair comparison, for the shallow model, we take the output of pre-trained model ResNet34 [1] and BERT [2] as the input of the shallow model following [30]. We set the encoded representation of both hashing-based models and quantization-based models as 32 bits in experiments. The number of codebooks is four and the number of codeword in each codebook is 256. The LightLT is trained by AdamW [2] optimizer with learning rate $5e-5$ on Cifar100 and ImageNet100, and learning rate $1e-5$ on NC and QBA. On Cifar100 and ImageNet100, we use cosine annealing strategy according to [30], and linear schedule with warm up. The number of model ensemble is set to four on all datasets. We tune the hyper-parameter α with grid search on the validation set over the set $\{1e-5, 1e-4, ..., 1e0\}$.

TABLE II
COMPARISON WITH BASELINES ON CIFAR100 AND IMAGENET100. RRSULT OF METHOD WITH "*" IS REPORTED FROM [30] DIRECTLY. THE HIGHEST SCORES PER CATEGORY ARE BOLD.

	Cifar100		ImageNet100	
	IF=50	IF=100	IF=50	IF=100
LSH*	0.0333	0.0307	0.0606	0.0556
PCAH*	0.0532	0.0519	0.1306	0.1280
ITQ*	0.0709	0.0677	0.1803	0.1719
KNNH*	0.0703	0.0689	0.1830	0.1766
SDH*	0.1115	0.1006	0.3553	0.3126
COSDISH*	0.0695	0.0583	0.2072	0.1763
FastHash*	0.0787	0.0714	0.2462	0.1932
FSSH*	0.1101	0.0957	0.3681	0.3312
SCDH*	0.1282	0.1138	0.3937	0.3601
DPSH*	0.1069	0.0978	0.2186	0.1788
HashNet*	0.1726	0.1444	0.3465	0.3101
DSDH*	0.1119	0.0940	0.2568	0.1841
CSQ*	0.2221	0.1716	0.6629	0.5989
LTHNet*	0.2687	0.1819	0.7612	0.7146
LightLT w/o ensemble	0.3464	0.2499	0.7532	0.7148
LightLT	**0.3801**	**0.2740**	**0.7804**	**0.7398**

TABLE III
COMPARISON WITH BASELINES ON NC AND QBA. THE HIGHEST SCORES PER CATEGORY ARE BOLD.

	Amazon News (NC)		QBA	
	IF=50	IF=100	IF=50	IF=100
LSH	0.1093	0.1092	0.0417	0.0416
PQ	0.2546	0.2543	0.0955	0.0939
DPQ	0.5809	0.5408	0.3707	0.3346
KDE	0.6042	0.5454	0.3815	0.3410
LTHNet	0.5990	0.5372	0.3703	0.3403
LightLT w/o ensemble	0.6200	0.5750	0.3899	0.3594
LightLT	**0.6560**	**0.6131**	**0.4097**	**0.3824**

B. Comparison with Baselines

In this section, we present the performance results of both baselines and the proposed LightLT for retrieving items from the database using queries from the query set. The results are summarized in Table II and Table III for image and text data, respectively, which provide insights to address our first research question and showcase our findings.

And we can also find that LightLT has more improvement on Cifar100 than on ImageNet100 compared to LTHNet. One potential reason is that the ResNet34 is pre-trained on ImageNet, so it can learn higher quality representation and can achieve higher MAP value on ImageNet100 (a subset of ImageNet) than on Cifar100.

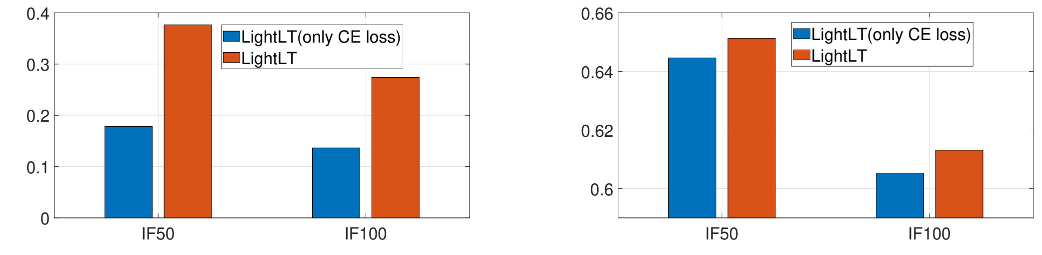
First, LightLT demonstrates a remarkable superiority over other baseline methods. On four different datasets, the proposed LightLT achieves an average improvement of approximately 17.6%. It is worth noting that the other baselines, except for LTHNet, were not specifically designed for handling long-tail scenarios, which is why they struggle to attain high accuracy in such situations. In comparison to LTHNet, a hashing-based model tailored for long-tail data, LightLT performs significantly better. This could be attributed to two potential reasons: The DSQ method employed in LightLT helps minimize information loss when compared to conventional hashing techniques. Additionally, the proposed ensemble module aids in enhancing the model's generalization capability, making it more effective in long-tail scenarios. We also observe that LightLT exhibits more substantial improvements on the Cifar100 dataset when compared to LTHNet, in contrast to the results on ImageNet100. One plausible explanation for this observation is that the ResNet34 utilized in LightLT is pre-trained on ImageNet, enabling it to learn higher quality representations and consequently achieve a higher MAP value on ImageNet100 (a subset of ImageNet) compared to Cifar100.

Second, the proposed ensemble module is effective to improve model performance. Compared to LightLT w/o ensemble, LightLT consistently demonstrates improvements on four datasets: Cifar100, ImageNet100, NC, and QBA. These improvements amount to over 9.64%, 3.50%, 5.81%, and 7.00% respectively. To achieve these enhancements, LightLT incorporates the parameter ensemble and fine-tuning techniques. These measures serve two key purposes: preventing over-fitting for the tail data and aiding the model in discovering a flat local optimum. By doing so, the model's generalization ability is significantly enhanced, resulting in improved performance overall.

Third, the comparison presented in Table II and Table III clearly demonstrates the superiority of deep learning-based methods over shallow models. Across both image and text datasets, deep models exhibit a significant advantage when compared to their shallow counterparts. The limitations of shallow models become apparent, as they struggle to effectively capture complex features from images or text. Due to this difficulty in capturing intricate information, the compressed representation obtained from shallow models lacks the necessary informativeness to achieve accurate retrieval results. To address this limitation, the decision to design deep models is justified. By utilizing deep models, the learning process can create quantized representations that better encapsulate the essential characteristics of the data, resulting in improved retrieval performance.

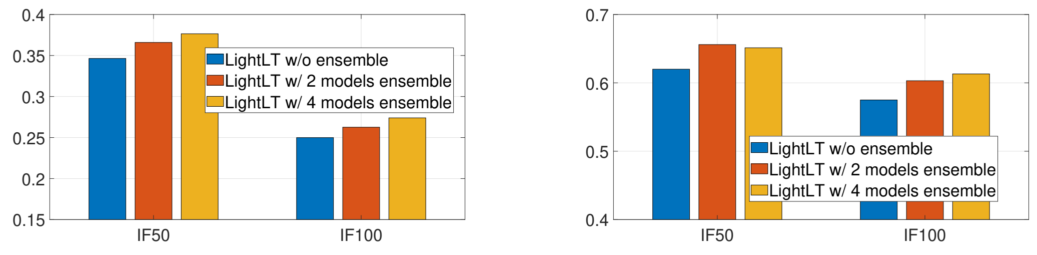
C. Effectiveness of The Proposed Loss Function

In this section, we demonstrate an ablation study with respect to the proposed loss function to answer the second



(a) Cifar100 (b) NC

Fig. 5. The results of LightLT w/ and w/o the proposed loss function on Cifar100 and NC.



(a) Cifar100 (b) NC

Fig. 6. The results of LightLT adopting different numbers of ensemble models on Cifar100 and NC.

TABLE IV
THE RESULTS OF USING DSQ AND USING VANILLA RESIDUAL MECHANISM ON CIFAR100 AND NC. IMP REPRESENTS THE IMPROVEMENT COMPARED TO VANILLA RESIDUAL MECHANISM.

	Cifar100				NC			
	IF=50	IMP(%)	IF=100	IMP(%)	IF=50	IMP(%)	IF=100	IMP(%)
Residual	0.3385	2.33	0.2478	0.85	0.5970	3.85	0.5606	2.57
DSQ	0.3464	-	0.2499	-	0.6200	-	0.5750	-

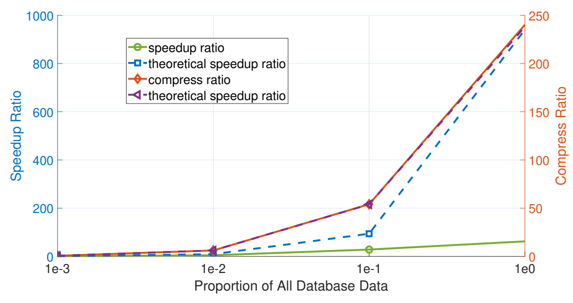


Fig. 7. The efficiency comparison on QBA with varying database scale. The x-axis is the proportion of all database data. The left y-axis is the speedup ratio and the right y-axis is the compress ratio.

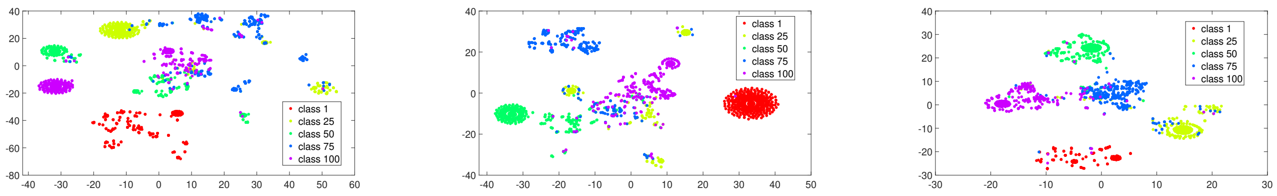
research question. We compare LightLT only with cross-entropy loss $\mathcal{L}_{ce}$ and with proposed loss $\mathcal{L}$. We conduct the experiments on one image dataset Cifar100 and one text dataset NC. The results are shown in Fig. 5.

According to Fig. 5, LightLT using the proposed loss function achieves better retrieval performance than LightLT using only cross-entropy loss. The cross-entropy loss is widely used in existing works. However, it just leverages label information of each training data while ignoring the target is about ranking. The proposed loss function facilitates the closeness of representations associated with the same label and promotes a greater distance between representations linked to different labels. Therefore, the proposed loss functions achieve better performance. Besides, we can also find the proposed loss function has more improvements on Cifar100 than on NC. For images with same label on Cifar100, such as apple images, although each image is different, they share a lot of common characteristics (e.g. round outline, having a pedicel). Different from Cifar100, for text with same label on NC, they still contain totally different words, sentences and paragraphs. As a result, the variance within the NC label is greater than that within the Cifar100 label. This characteristic may enhance the effectiveness of the proposed loss function on the Cifar100 dataset.

To further highlight the distinctions among representations learned with different loss functions, we present visualizations of the representations on the Cifar100 dataset in Fig. 8. For this visualization, we have selected five classes from the total of 100 classes. When using the cross-entropy loss, the representations of one label appear scattered, making it unsuitable for effectively searching for similar items. With the combination of cross-entropy loss and center loss, we observe that the representations of the same label form clusters. However, since it does not explicitly enforce a separation between representations of different labels, some classes end up being mixed in Fig. 8. In contrast, when employing the combination of cross-entropy loss, center loss, and ranking loss, the representations of each class are noticeably clustered, and different classes are well separated. This demonstrates the superior performance of this combined loss function in creating distinct and discriminative representations for the given dataset.

D. *Effectiveness of DSQ*

In this section, we conduct an ablation study with respect to the proposed DSQ module to answer the second research question. We compare the model with DSQ and the model only



(a) CE (b) CE+center loss (c) CE+center loss+ranking loss

Fig. 8. The visualization of different loss functions on Cifar100.

with vanilla residual mechanism on Cifar100 and NC. To filter the influence of other modules, we remove the ensemble module. We show the results in Table IV. According to Table IV, we can find the DSQ achieves consistent improvements on Cifar100 and NC compared to vanilla residual mechanism. The DSQ brings 2.33% and 0.85% improvement with respect to $IF = 50$ and $IF = 100$ on Cifar100 respectively, and brings 3.85% and 2.57% improvement with respect to $IF = 50$ and $IF = 100$ on NC respectively. The proposed DSQ inherits the advantage of vanilla residual mechanism, i.e. ensuring the diversity of codebooks, and DSQ can enable the gradient back propagation to be more stable. This experiment verifies the superiority of DSQ.

E. Efficiency Comparison

In this section, we study how LightLT improves the inference efficiency and compression efficiency. To filter the influence of hardwares used in experiments, e.g. GPUs and CPUs, we report the value of speedup ratio and compress ratio instead of the absolute inference time. We conduct the experiment on QBA with $IF = 100$ dataset and show the speedup ratio and compress ratio with various scale of database in Fig. 7. Based on Fig. 7, we have following findings.

First, we can find when the database is large, LightLT can improve both inference efficiency and storage efficiency a lot. When the number of database data is 1/10 of the whole database data, the speedup ratio is 28.36 and compress ratio is 54.04. And when we use the whole database data, the speedup ratio is 62.36 and compression ratio is 240.20. It shows when the amount of database data is large, the LightLT can improve inference efficiency and save storage cost dramatically.

Second, the proposed LightLT can not provide efficiency improvement when the database data is very limited. In Fig. 7, when the number of database data is only 1/1000 of whole database data (around 642 data), the inference efficiency and storage efficiency are not improved. Because there are four codebooks and each codebook contains 256 codewords, the 1,024 codewords totally costs more space than original continuous database data. And for each query, it needs to compute the distance between these 1,024 codewords, which also costs more than computing distance between database data directly. Therefore, when the data scale is too small, the proposed LightLT can not show its superiority.

F. Effect of The Number of Ensemble Models

In this section, we investigate the impact of the number of ensemble models on model performance, addressing the fourth research question. Our experimentation is conducted on both the Cifar100 and NC datasets, and the results are presented in Fig. 6. Upon analyzing Fig. 6, it becomes evident that as the number of ensemble models increases, the MAP value consistently rises. Whether utilizing an ensemble of 2 models or 4 models, both configurations yield substantial improvements compared to LightLT without ensemble. This observation highlights that even with just 2 models in the ensemble, there is a significant enhancement in the model's generalization ability. For instance, on the NC dataset with the IF50 setting, the MAP value increases from 0.62 to 0.65. Thus, this experiment further reinforces the effectiveness of the ensemble module in improving overall performance.

VI. Conclusion

In this paper, we propose a lightweight representation quantization framework for long-tail data, named LightLT. This framework uses the sum of codewords within multiple codebooks to represent continuous representations approximately, which saves the storage space tremendously as only the IDs of codewords need to be stored. To learn the quantized representations, the proposed LightLT framework includes several novel designs: The proposed double skip quantization module uses skip connection between encoder-decoder pairs and among codebooks to ensure the diversity of codebooks and the stability of gradients. A novel loss function is proposed to improve model ranking performance on long-tail data, ensuring that the distance among quantized representations with the same label is close while the quantized representations with the different labels are pushed away. To overcome possible over-fitting on tail data, we propose the ensemble module, which averages the weights of multiple models for better generalization. During model averaging, the codeword alignment issue is resolved by fixing the other layers of LightLT and only fine-tune the codebooks. Experiments on four datasets show the significant improvement of LightLT with respect to storage efficiency, inference efficiency, and search accuracy.

Acknowledgement

This work is supported in part by the US National Science Foundation under grant NSF IIS-1747614 and NSF IIS-2226108. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[4] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Computing Surveys (Csur)*, vol. 40, no. 2, pp. 1–60, 2008.

[5] I. M. Hameed, S. H. Abdulhussain, and B. M. Mahmmod, "Content-based image retrieval: A review of recent trends," *Cogent Engineering*, vol. 8, no. 1, p. 1927469, 2021.

[6] J. Revaud, J. Almazán, R. S. Rezende, and C. R. d. Souza, "Learning with average precision: Training image retrieval with a listwise loss," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5107–5116.

[7] S. R. Dubey, S. K. Singh, and W.-T. Chu, "Vision transformer hashing for image retrieval," in *2022 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2022, pp. 1–6.

[8] W. Yang, H. Zhang, and J. Lin, "Simple applications of bert for ad hoc document retrieval," *arXiv preprint arXiv:1903.10972*, 2019.

[9] Z. A. Yilmaz, S. Wang, W. Yang, H. Zhang, and J. Lin, "Applying bert to document retrieval with birch," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, 2019, pp. 19–24.

[10] Z. A. Yilmaz, W. Yang, H. Zhang, and J. Lin, "Cross-domain modeling of sentence-level evidence for document retrieval," in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019, pp. 3490–3496.

[11] Y. Mass and H. Roitman, "Ad-hoc document retrieval using weak-supervision with bert and gpt2," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4191–4197.

[12] F. Zhu, W. Lei, C. Wang, J. Zheng, S. Poria, and T.-S. Chua, "Retrieving and reading: A comprehensive survey on open-domain question answering," *arXiv preprint arXiv:2101.00774*, 2021.

[13] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer, "Quac: Question answering in context," *arXiv preprint arXiv:1808.07036*, 2018.

[14] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2004.04906*, 2020.

[15] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge graph embedding based question answering," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 105–113.

[16] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, "A survey on learning to hash," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 769–790, 2017.

[17] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, "Learning to hash for indexing big data—a survey," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57, 2015.

[18] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2916–2929, 2012.

[19] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," *Advances in neural information processing systems*, vol. 27, 2014.

[20] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

[21] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 4, pp. 744–755, 2013.

[22] G. Lin, C. Shen, Q. Shi, A. Van den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1963–1970.

[23] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 37–45.

[24] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5608–5617.

[25] T. Yu, J. Yuan, C. Fang, and H. Jin, "Product quantization network for fast image retrieval," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 186–201.

[26] B. Klein and L. Wolf, "End-to-end supervised product quantization for image search and retrieval," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5041–5050.

[27] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis, "Decoupling representation and classifier for long-tailed recognition," *arXiv preprint arXiv:1910.09217*, 2019.

[28] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu, "Large-scale long-tailed recognition in an open world," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2537–2546.

[29] L. Xiao, X. Zhang, L. Jing, C. Huang, and M. Song, "Does head label help for long-tailed multi-label text classification," *arXiv preprint arXiv:2101.09704*, 2021.

[30] Y. Chen, Y. Hou, S. Leng, Q. Zhang, Z. Lin, and D. Zhang, "Long-tail hashing," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1328–1338.

[31] L. Chen, G. Zhang, and E. Zhou, "Fast greedy map inference for determinantal point process to improve recommendation diversity," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[32] P. Bühlmann, "Bagging, boosting and ensemble methods," in *Handbook of computational statistics*. Springer, 2012, pp. 985–1022.

[33] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith *et al.*, "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time," in *International Conference on Machine Learning*. PMLR, 2022, pp. 23 965–23 998.

[34] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[35] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[36] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[37] X.-S. Wei, Y. Shen, X. Sun, H.-J. Ye, and J. Yang, "A^2-net: Learning attribute-aware hash codes for large-scale fine-grained image retrieval," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5720–5730, 2021.

[38] L. Yuan, T. Wang, X. Zhang, F. E. Tay, Z. Jie, W. Liu, and J. Feng, "Central similarity quantization for efficient image and video retrieval," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3083–3092.

[39] J. Wang, S. Xu, F. Zheng, K. Lu, J. Song, and L. Shao, "Learning efficient hash codes for fast graph-based data similarity retrieval," *IEEE Transactions on Image Processing*, vol. 30, pp. 6321–6334, 2021.

[40] G. Gu, J. Liu, Z. Li, W. Huo, and Y. Zhao, "Joint learning based deep supervised hashing for large-scale image retrieval," *Neurocomputing*, vol. 385, pp. 348–357, 2020.

[41] T. Zhang, C. Du, and J. Wang, "Composite quantization for approximate nearest neighbor search," in *International Conference on Machine Learning*. PMLR, 2014, pp. 838–846.

[42] C. Zhou, L.-M. Po, W.-F. Ou, P.-F. Xian, and K.-W. Cheung, "Deep triplet residual quantization," *Expert Systems with Applications*, vol. 184, p. 115467, 2021.

[43] Y. Chen, T. Guan, and C. Wang, "Approximate nearest neighbor search by residual vector quantization," *Sensors*, vol. 10, no. 12, pp. 11 259–11 273, 2010.

[44] T. Chen, M. R. Min, and Y. Sun, "Learning k-way d-dimensional discrete codes for compact embedding representations," in *International Conference on Machine Learning*. PMLR, 2018, pp. 854–863.

[45] T. Chen, L. Li, and Y. Sun, "Differentiable product quantization for end-to-end embedding compression," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1617–1626.

[46] D. Lian, H. Wang, Z. Liu, J. Lian, E. Chen, and X. Xie, "Lightrec: A memory and search-efficient recommender system," in *Proceedings of The Web Conference 2020*, 2020, pp. 695–705.

[47] A. Babenko and V. Lempitsky, "Additive quantization for extreme vector compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 931–938.

[48] L. Gao, X. Zhu, J. Song, Z. Zhao, and H. T. Shen, "Beyond product quantization: Deep progressive quantization for image retrieval," *arXiv preprint arXiv:1906.06698*, 2019.

[49] Q. Zhai and M. Jiang, "Deep product quantization for large-scale image retrieval," in *2019 IEEE 4th International Conference on Big Data Analytics (ICBDA)*. IEEE, 2019, pp. 198–202.

[50] B. Liu, Y. Cao, M. Long, J. Wang, and J. Wang, "Deep triplet quantization," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 755–763.

[51] T.-Y. Liu *et al.*, "Learning to rank for information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[52] S. Xiao, Z. Liu, W. Han, J. Zhang, D. Lian, Y. Gong, Q. Chen, F. Yang, H. Sun, Y. Shao *et al.*, "Distill-vq: Learning retrieval oriented vector quantization by distilling knowledge from dense embeddings," *arXiv preprint arXiv:2204.00185*, 2022.

[53] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[54] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*. Springer, 2005, pp. 878–887.

[55] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[56] S. Park, Y. Hong, B. Heo, S. Yun, and J. Y. Choi, "The majority can help the minority: Context-rich minority oversampling for long-tailed classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6887–6896.

[57] Q. Kang, X. Chen, S. Li, and M. Zhou, "A noise-filtered under-sampling scheme for imbalanced classification," *IEEE transactions on cybernetics*, vol. 47, no. 12, pp. 4263–4274, 2016.

[58] G. Liang and C. Zhang, "An efficient and simple under-sampling technique for imbalanced time series classification," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 2339–2342.

[59] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, 2009.

[60] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9268–9277.

[61] A. K. Menon, S. Jayasumana, A. S. Rawat, H. Jain, A. Veit, and S. Kumar, "Long-tail learning via logit adjustment," *arXiv preprint arXiv:2007.07314*, 2020.

[62] S. Zhang, Z. Li, S. Yan, X. He, and J. Sun, "Distribution alignment: A unified framework for long-tail visual recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2361–2370.

[63] S. S. Mullick, S. Datta, and S. Das, "Generative adversarial minority oversampling," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1695–1704.

[64] J. Kim, J. Jeong, and J. Shin, "M2m: Imbalanced classification via major-to-minor translation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 13 896–13 905.

[65] C. Wang, S. Gao, C. Gao, P. Wang, W. Pei, L. Pan, and Z. Xu, "Label-aware distribution calibration for long-tailed classification," *arXiv preprint arXiv:2111.04901*, 2021.

[66] S. Li, K. Gong, C. H. Liu, Y. Wang, F. Qiao, and X. Cheng, "Metasaug: Meta semantic augmentation for long-tailed visual recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5212–5221.

[67] Y.-X. Wang, D. Ramanan, and M. Hebert, "Learning to model the tail," *Advances in neural information processing systems*, vol. 30, 2017.

[68] Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie, "Large scale fine-grained categorization and domain-specific transfer learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4109–4118.

[69] X. Niu, B. Li, C. Li, R. Xiao, H. Sun, H. Deng, and Z. Chen, "A dual heterogeneous graph attention network to improve long-tail performance for shop search in e-commerce," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3405–3415.

[70] Y. Hu, Y. Liu, C. Miao, and Y. Miao, "Memory bank augmented long-tail sequential recommendation," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 791–801.

[71] M. Matena and C. Raffel, "Merging models with fisher-weighted averaging," *arXiv preprint arXiv:2111.09832*, 2021.

[72] Y. Wang, S. Mukherjee, X. Liu, J. Gao, A. H. Awadallah, and J. Gao, "Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models," *arXiv preprint arXiv:2205.12410*, 2022.

[73] P. Xie, A. Singh, and E. P. Xing, "Uncorrelation and evenness: a new diversity-promoting regularizer," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3811–3820.

[74] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European conference on computer vision*. Springer, 2016, pp. 499–515.

[75] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*. Springer, 2000, pp. 1–15.

[76] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[77] Y. Liang, N. Duan, Y. Gong, N. Wu, F. Guo, W. Qi, M. Gong, L. Shou, D. Jiang, G. Cao *et al.*, "Xglue: A new benchmark dataset for cross-lingual pre-training, understanding and generation," *arXiv preprint arXiv:2004.01401*, 2020.

[78] A. Gionis, P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing," in *Vldb*, vol. 99, no. 6, 1999, pp. 518–529.

[79] X. He, P. Wang, and J. Cheng, "K-nearest neighbors hashing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2839–2848.

[80] W.-C. Kang, W.-J. Li, and Z.-H. Zhou, "Column sampling based discrete supervised hashing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[81] X. Luo, L. Nie, X. He, Y. Wu, Z.-D. Chen, and X.-S. Xu, "Fast scalable supervised hashing," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 735–744.

[82] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," *arXiv preprint arXiv:1511.03855*, 2015.

[83] Q. Li, Z. Sun, R. He, and T. Tan, "Deep supervised discrete hashing," *Advances in neural information processing systems*, vol. 30, 2017.