Adaptive Centroid Guided Hashing for Cross-Modal Retrieval

Anonymous submission

Abstract

Deep cross-modal hashing (DCMH) technology is widely used in retrieval tasks due to its low storage costs and efficient computation speed. However, proxy or predefined hash center-based DCMH methods suffer from the following challenges: (1) lack of awareness of data distribution, (2) inability of proxy or predefined hash centers to effectively explore the category semantic information and the unequal scales of each object in a given sample. To address these challenges, we propose a novel adaptive centroid guided hashing (ACGH) method for cross-modal retrieval. Specifically, we extract global and local features using Transformer models, and then fuse them to obtain fine-grained feature representations of multimodal data. Subsequently, the hash centroid generation module leverages the category semantic embedding to construct class hash centers and combine them with learnable label-affinity coefficients (LAC) memory banks to learn adaptive hash centroids. Furthermore, we design a hash centroid guidance module, which uses the hash centroids to guide hash code learning and then updates the hash centers and LAC memory banks through the newly learned hash codes. Extensive experimental results on several benchmark multimodality datasets demonstrate that the proposed ACGH method significantly outperforms other state-of-the-art methods in cross-modal retrieval tasks.

1 Introduction

With the rapid expansion of multimedia data, cross-modal retrieval has become a prominent research area. Its primary goal is to retrieve relevant samples from another modality based on queries from one modality. Cross-modal hashing is an efficient similarity search technology that maps multimodality data into a shared hash space, preserving semantic similarity across heterogeneous data. Moreover, its computational efficiency and reduced storage requirements have made it a crucial technique for large-scale cross-modal retrieval applications.

Deep cross-modal hashing (DCMH) methods can generally be categorized into unsupervised and supervised methods. UnsupervisedDCMH [Luo *et al.*, 2021][Mikriukov *et al.*, 2022][Hu *et al.*, 2022] show promising retrieval performance in recent years due to the progress of deep neural networks (DNNs). However, these unsupervised methods suffer from the absence of explicit semantic supervision, which degrades retrieval performance. In contrast, supervised DCMH methods fully utilize label information to capture the semantic relationships between multimodal data more effectively, and thus learn more discriminative hash codes. Most supervision methods [Jiang and Li, 2017] [Xu *et al.*, 2019] [Tu *et al.*, 2022a] [Gu *et al.*, 2019] [Bai *et al.*, 2020] rely on pairwise and triplet strategies to leverage the label information. Specifically, pairwise-based DCMH methods utilize the similarity relationship between data pairs with known labels. The triplet-based method aims to solve local optimization problems by minimizing the distance between anchor points and positive samples, while maximizing the distance between anchor points and negative samples. Recently, several studies have proposed several proxy-based [Huo *et al.*, 2024], [Bai *et al.*, 2023] and hash center-based DCMH methods [Tu *et al.*, 2024] for cross-modal retrieval tasks. These approaches typically use proxies to ensure intra-class compactness and inter-class separation or leverage predefined hash centers to capture the global similarity among multimodal hash codes. However, they still have several limitations in real cross-modal retrieval applications. First, they cannot effectively perceive multimodal data distributions. Second, they fail to fully leverage category semantic information and their semantic relationships.

To overcome these shortcomings, we propose an adaptive centroid guided hashing (ACGH) method for cross-modal retrieval. This method asynchronously learns hash codes and adaptive hash centroids. Specifically, the feature extraction framework of the proposed model is built on the Transformer architecture, thus fully leveraging its powerful feature representation ability to capture more fine-grained features. Meanwhile, we learn adaptive hash centroids by combining the BERTmodel with the LAC memory banks . Additionally, we optimize the LAC memory banks using the newly learned hash codes, achieving forward collaborative optimization between hash centroids and hash codes. Notably, our framework employs an asynchronous learning mechanism to inde-

pendently update hash centroids and hash codes, thereby effectively enhancing the model's overall performance. Extensive experimental results demonstrate the superiority of the proposed method in cross-modal retrieval.

The contributions of the work can be summarized as follows:

- We propose a novel DCMH method called Adaptive Centroid Guided Hashing (ACGH). To the best of our knowledge, this is the first work to learn adaptive hash centroids by combining hash centers with their object scales within DCMH methods. These hash centroids effectively capture category semantic information and the unequal scales of each label in a given sample.
- We propose a novel centroid-guided hash code learning strategy, where the hash code learning can be guided by the hash centroids. Then the LAC memory banks and hash centers are updated using the newly generated hash codes. This asynchronous optimization mechanism can learn the optimal hash codes and hash centroids.
- Extensive experimental results on three benchmark datasets demonstrate the effectiveness of our proposed ACGH approach in cross-modal retrieval tasks.

2 Related work

2.1 Deep Cross-modal Hashing

In recent years, several unsupervised DCMH methods [Liu *et al.*, 2020] [Yu *et al.*, 2021][Tan *et al.*, 2022][Sun *et al.*, 2023][Cui *et al.*, 2024] have achieved excellent performance thanks to the powerful feature extraction ability of DNNs. To fully utilize supervisory information among multimodalities, various supervised DCMH methods [Jiang and Li, 2017][Gu *et al.*, 2019][Shu *et al.*, 2022][Tu *et al.*, 2022a][Qin *et al.*, 2024] [Hu *et al.*, 2024] have been proposed to enhance retrieval performance. Specifically, DCMHT [Tu *et al.*, 2022a] proposes a differentiable cross-modal hashing method that uses a multi-modal Transformer as the backbone. It generates binary codes through a selection mechanism, allowing for better optimization of the hash codes. DNpH [Qin *et al.*, 2024] employs quadratic spherical mutual information (QSMI) to minimize neighborhood ambiguity. SCH [Hu *et al.*, 2024] divides sample pairs into three categories based on semantic similarity and then applies different constraints to each category to efficiently utilize the entire hash space. The pairwise or triplet-based strategy is commonly used among these supervised DCMH methods. Therefore, they promote inter-class separation by calculating the relative similarity between samples. However, the intra-class compactness of data across modalities is also weakened, resulting in fuzzy neighborhoods.

2.2 Proxy or Predefined Center-based Deep Cross-modal Hashing

Since several DCMH methods seek to fully explore the global similarity across multimodalities using the proxies and predefined hash centers, they generally outperform those based on pairwise and triplet-based approaches. Specifically, DCPH [Tu *et al.*, 2022b] employs a proxy hashing network to generate proxy hash codes containing category information, and introduces a new edge dynamic softmax loss function that directly uses proxy hash codes as supervisory signals during training. However, classic proxy-based DCMH methods are prone to causing neighborhood ambiguity, leading to suboptimal performance. To solve this issue, PGCH [Bai *et al.*, 2023] constructs a Hadamard matrix to learn the proxies, furthermore generating more precise hash codes. TWDH [Tu *et al.*, 2024] designs a novel quantization method to mitigate information loss during the quantization process. Moreover, it proposes a compression approach that better preserves information when reducing feature dimensions, thus effectively bridging the gap between high-dimensional and low-dimensional spaces. DNPH [Huo *et al.*, 2024] introduces a uniform distribution constraint, ensuring that each hash bit independently follows a discrete uniform distribution. Although the aforementioned methods achieve satisfactory performance in some tasks, they are limited by their lack of data distribution awareness and failure to fully leverage category semantics and label relationships.

3 Methodology

3.1 Notations

This work focuses on cross-modal retrieval tasks of image-text. Suppose that we have N training image-text pairs $\mathcal{S} = \{s_i\}_{i=1}^{N}$, where $s_i = (x_i^v, x_i^t, l_i)$ denotes a image-text pair. Here, $x_i^v \in \mathbb{R}^{d^v}$ and $x_i^t \in \mathbb{R}^{d^t}$ denote the i-th sample from image and text modalities, respectively. $l_i \in \{0,1\}^{1\times C}$ represents their corresponding multi-label annotation, where C is the category number.

3.2 Fine-grained Feature Extraction Module

To effectively extract the feature information of multi-modality data, as illustrated in Figure 1, we employ two Transformer encoders, ViT [Dosovitskiy, 2020] and GPT-2 [Radford *et al.*, 2019], to extract the features of the image and text modalities, respectively. Specifically, for the i-th image-text pair, the feature representations of the image and text modalities from encoders are given by $f_i^v = \{G_i^v, P_i^v\}$ and $f_i^t = \{G_i^t, P_i^t\}$, respectively. Here, $G_i^v \in \mathbb{R}^{1\times d}$ and $G_i^t \in \mathbb{R}^{1\times d}$ denote global embeddings of image and text modalities, respectively, while $P_i^v \in \mathbb{R}^{T_v\times d}$ and $P_i^t \in \mathbb{R}^{T_t\times d}$ represent their local embedding sequences, respectively, where d is the feature dimension, and T_v and T_t are the number of tokens. The feature extraction process can be represented as follows:

$$f_i^{(*)} = E_{\text{trans}}^{(*)}(x_i^{(*)}; \theta^{(*)}), * \in \{v, t\}. \tag{1}$$

where $E_{\text{trans}}^{(*)}(\cdot;\cdot)$ is the encoder of Transformer model and $\theta^{(*)}$ represents its parameters. Note that the parameters $\theta^{(*)}$ are frozen in this work.

After extracting the global and local features of the multimodalities using the encoders, we fuse them to obtain the representations of the image and text modalities, respectively. Specifically, we first combine the conceptual token aggregation (CTA) strategy with a single-layer Transformer to capture fine-grained semantic representations from a conceptual perspective. Thus, we have

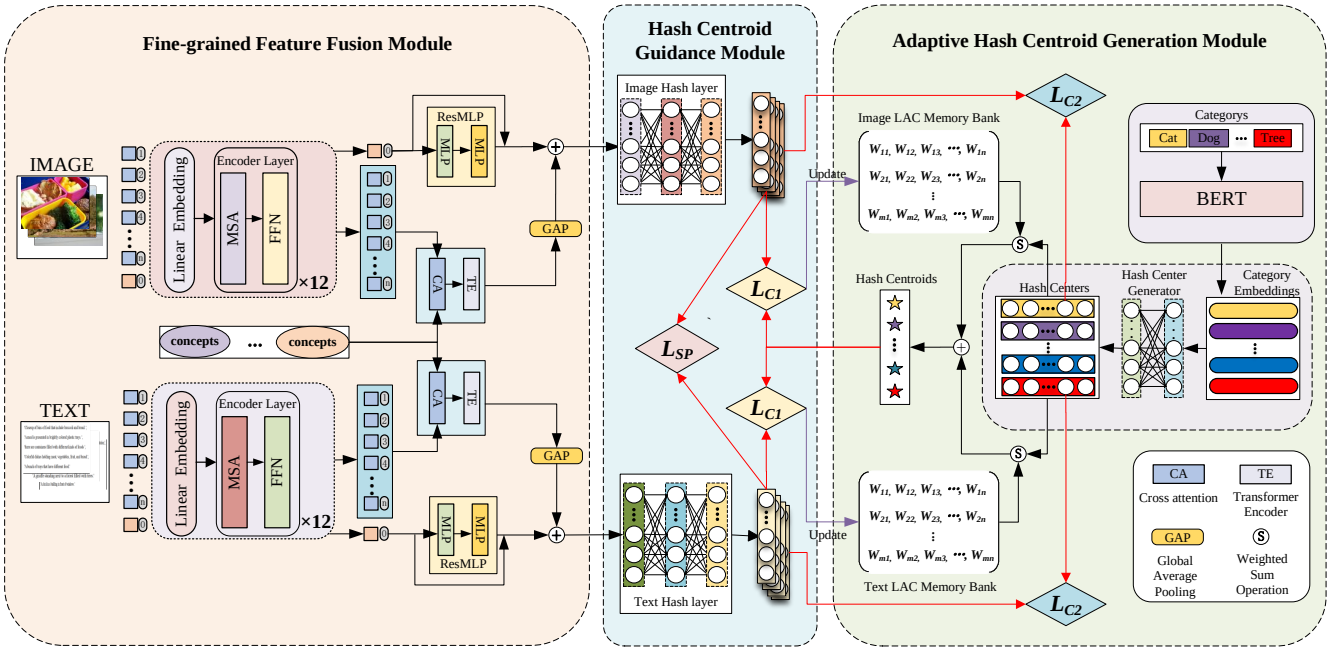

Figure 1: The overall framework of our proposed ACGH approach. It consists of three main modules: the fine-grained feature extraction module, the hash centroid generation module, and the hash centroid guidance module. Specifically, the fine-grained feature extraction module aims to capture intrinsic feature information of multimodalities. The hash centroid generation module combines hash centers with the LAC strategy to generate accurate hash centroids. In the hash centroid guidance module, these centroids guide the learning of hash codes, and the LAC memory bank across multimodalities is updated using the newly learned hash codes and hash centers.

$$A_i^{(*)} = \text{Softmax}\left(\frac{(QW_q^{(*)})(P_i^{(*)}W_k^{(*)})^T}{\sqrt{d}}\right), \quad (2)$$

$$\tilde{C}_i^{(*)} = A_i^{(*)}(P_i^{(*)}W_v^{(*)}), \quad (3)$$

where $A_i^{(*)} \in \mathbb{R}^{P\times L^{(*)}}$ represents the attention map, and $\tilde{C}_i^{(*)} \in \mathbb{R}^{P\times d}$ is the coarse semantic tokens of multimodalities. $W_q^{(*)}, W_k^{(*)}, W_v^{(*)} \in \mathbb{R}^{d\times d}$ are learnable matrices, and $Q \in \mathbb{R}^{P\times d}$ represents the concept embeddings shared with different modalities.

Afterward, a single-layer Transformer encoder is applied to refine the coarse semantic tokens by capturing their relationships. Therefore, the fine-grained semantic representations of multimodalities are expressed as follows:

$$C_i^{(*)} = E_{\text{st}}(\tilde{C}_i^{(*)}; \theta_{\text{est}}^{(*)}), \quad (4)$$

where $\theta_{\text{est}}^{(*)}$ represents the encoder parameters and $E_{\text{st}}(\cdot;\cdot)$ denotes a single-layer Transformer encoder. In this way, CTA transforms variable-length, low-level local embeddings into fine-grained semantic representations of fixed length.

Since global and local information are complementary, we use $ResMLP$ to align their dimensions. The specific process is given as follows:

$$r_i^{(*)} = ResMLP(G_i^{(*)}; \theta_{\text{res}}) \in \mathbb{R}^d, \quad (5)$$

where $ResMLP(\cdot;\cdot)$ consists of two identical blocks, each containing one multi-layer perceptron (MLP) with residual connections. θ_{res} is the trainable weight-shared parameters for image and text modalities.

Then the local and global features with the same dimensions from Eq.(2) and Eq.(5) are fused using an addition operation. Therefore, we can obtain the final representation $X_i^{(*)}$ of the i-th sample as follows:

$$X_i^{(*)} = r_i^{(*)} + GAP(C_i^{(*)}) \in \mathbb{R}^d. \quad (6)$$

where $GAP(\cdot)$ refers to the global average pooling.

3.3 Hash Centroid Guidance Module

The hash centroid guidance module constructs the hashing layer to learn discriminative hash codes. It consists of fully connected layers and a nonlinear activation function, which maps the extracted fused features to continuous hash codes $FU_i^{(*)}$. Thus, we have

$$FU_i^{(*)} = \tanh\left(Hash\left(X_i^{(*)}; \theta_h^{(*)}\right)\right). \quad (7)$$

where $Hash(\cdot)$ represents the hash layer, $\theta_h^{(*)}$ denotes the parameters of the hash layer, and $\tanh(\cdot)$ is the activation function. During the inference phase, we map the continuous hash codes $FU_i^{(*)}$ to binary values of -1 and 1 using the function $H_i^{(*)} = \text{sign}(FU_i^{(*)})$.

3.4 Adaptive Hash Centroid Generation Module

In this module, we first leverage the semantic extraction capability of the BERT [Kenton and Toutanova, 2019] model

to embed the category semantic information into a category embedding matrix $CA = \{ca_i\}_{i=1}^{C} \in \mathbb{R}^{C\times D}$, where ca_i is expressed as the embedding vector of the i-th category, and C represents the category number. D is the dimensionality of the word embeddings, and v_i is the embedding vector of the i-th category.

Next, we construct a hash center generator, which consists of two fully connected layers. This generator maps the category embedding vectors V into learnable hash centers HC. This process can be formalized as follows:

$$hc_i = Hash_c(ca_i; \theta_c), \tag{8}$$

$$HC = \{hc_j\}_{j=1}^{C} \in \mathbb{R}^{C\times K}, \tag{9}$$

where θ_c represents the parameters of the hash center generator $Hash_c(\cdot;\cdot)$, K is the dimensionality of the hash centers, and hc_i is the hash center for the i-th category.

After obtaining the hash centers, we can combine them with the LAC memory bank to construct the hash centroid p_i. The calculation process is given as follows:

$$p_i^{(*)} = \sum_{j=1}^{C} w_{ij}^{(*)} hc_j. \tag{10}$$

where $w_{ij}^{(*)}$ denotes the weight of the j-th label for the i-th sample. Here, $w_i^{(*)} = \{w_{ij}^{(*)}\}_{j=1}^{C}$ denotes the weight of the i-th sample in the LAC memory bank, where $\sum_{j=1}^{C} w_{ij}^{(*)} = 1$. Note that the initialization of $w_{ij}^{(*)}$ is $w_{ij}^{(*)} = \frac{1}{sum(l_i)}$, where $sum(l_i)$ is the sum of the one-hot encoded labels for the i-th sample. The updating process for w_i is given in Section 3.5.

3.5 Loss Functions

To effectively optimize the hash codes and hash centroids, we employ an asynchronous learning mechanism that allows both to be updated independently while adapting to each other. Once the hash codes $FU_i^{(*)}$ and its corresponding hash centroid $p_i^{(*)}$ are obtained, we utilize the weighted-softmax loss L_{WS} to aggregate the samples towards their respective hash centroids and push them away from irrelevant hash centers. Thus, we have

$$L_{WS} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{ES\_P(i)}{ES\_N(i)}, \tag{11}$$

$$ES\_P(i) = \exp\left(\frac{cos(FU_i^{(*)}, p_i^{(*)})}{\tau}\right), \tag{12}$$

$$ES\_N(i) = \sum_{hc_j \in S_{neg}} \exp\left(\frac{cos(FU_i^{(*)}, hc_j)}{\tau}\right), \tag{13}$$

where S_{neg} represents a subset of hash centers, with each element corresponding to a hash center hc_j, and j denoting the index of the 0 element in the one-hot encoded label. τ is a scale parameter. It is worth noting that this loss is applied in both the hash code optimization and hash centroid optimization processes described below.

Hash Code Learning: Firstly, to better maintain the pairwise semantic similarity between instances, we construct similarity-preserving loss functions both within and across modalities, based on the negative log-likelihood loss function.

(1) To effectively preserve the intra-modal similarity, two asymmetric pairwise negative log-likelihood losses for two modalities are given as follows:

$$L_{i,j}^{(*)} = -S_{ij}\Omega_{i,j}^{(*)} + \log(1 + e^{\Omega_{i,j}^{(*)}}), \tag{14}$$

$$L_{intra}^{(*)} = \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M} \mathcal{L}_{i,j}^{(*)}, \tag{15}$$

where $\Omega_{i,j}^{(*)} = \frac{(FU_i^{(*)})^T (FU_j^{(*)})}{2}$ denotes the inner product of hash codes within each modality, and S_{ij} is the similarity matrix derived from the known labels.

(2) Inter-modal similarity preservation ensures that semantic similarities are maintained across different modalities. Therefore, we employ the following asymmetric pairwise negative log-likelihood loss:

$$L_{i,j}^{v2t} = -S_{ij}\Theta_{ij} + \log(1 + e^{\Theta_{ij}}), \tag{16}$$

$$L_{i,j}^{t2v} = -S_{ij}\Phi_{ij} + \log(1 + e^{\Phi_{ij}}), \tag{17}$$

$$L_{inter} = \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(L_{i,j}^{v2t} + L_{i,j}^{t2v}\right), \tag{18}$$

where $\Theta_{ij} = \frac{(FU_i^{(t)})^T (FU_j^{(v)})}{2}$ and $\Phi_{ij} = \frac{(FU_i^{(v)})^T (FU_j^{(t)})}{2}$ indicate the inner products of hash codes across modalities. Therefore, the overall loss for the semantic similarity perseveration of the hash codes can be given as follows:

$$L_{SP} = \beta_1 L_{inter} + \beta_2 L_{intra}^{(*)}, \tag{19}$$

where β_1 and β_2 are the weight hyperparameters.

Then we use the L_{WS} loss in Eq.(11) to ensure intra-class compactness and inter-class separation of the hash codes. In addition, to ensure the quality of the hash codes and prevent any label weight from dominating the loss, we construct the quantization loss L_q. Furthermore, the maximum entropy regularization $R(w)$ is applied to optimize the label weights. Thus, we have

$$L_q = \frac{1}{N}\sum_{i=1}^{N} \|H_i^{(*)} - FU_i^{(*)}\|_2^2, \tag{20}$$

$$R(w) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} w_{ij}^{(*)} \log(w_{ij}^{(*)}), \tag{21}$$

where $H_i^{(*)}$ denote the binary hash codes corresponding to the i-th sample and w_{ij} represents the weight of the j-th label for the i-th sample. Here, the total loss function L_{C1} is formulated by integrating L_{WS}, L_q, and $R(w)$ as follows:

$$L_{C1} = L_{WS} + \beta_3 L_q + \beta_4 R(w). \tag{22}$$

where β_3 and β_4 denote the weight hyperparameters.

Finally, the overall objective function L_H for optimizing the hash codes can be written as follows:

$$L_H = \mu L_{C1} + \lambda L_{SP}, \tag{23}$$

where μ and λ are hyperparameters that control the relative weight of the respective loss terms L_{C1} and L_{SP}.

Adaptive Hash Centroid Learning: As described above, the hash centroids are fundamentally derived by combining the hash centers with the LAC memory banks. Thus, hash centroid learning consists of two stages: hash code optimization and LAC memory bank optimization.

(1) Optimizing hash centers: To optimize hash centers and enhance the quality of hash centroids, we first use BERT to learn category embeddings and then apply a hash center generator to obtain semantically rich hash centers. To ensure that the distribution of the hash center m_{ij} approximates the distribution of the label word vector s_{ij}, we utilize the Kullback-Leibler (KL) divergence to construct the label center loss L_{CS}. Thus, we have

$$L_{CS} = \sum_{i=1}^{C}\sum_{j=1}^{C} s_{ij} \log \frac{s_{ij}}{m_{ij}}, \tag{24}$$

$$s_{ij} = \frac{1}{2}\left(\cos(ca_i, ca_j) + 1\right), \tag{25}$$

$$m_{ij} = \frac{1}{2}\left(\cos(hc_i, hc_j) + 1\right), \tag{26}$$

where ca_i represents the i-th category embedding vector, and hc_i represents the i-th hash center.

Similar to the hash function optimization, we employ L_{WS} to enhance the similarity (or dissimilarity) of generated hash centers. Therefore, the total optimization loss for the adaptive hash centers is formulated as follows:

$$L_{C2} = \alpha_1 L_{WS} + \alpha_2 L_{CS}. \tag{27}$$

where α_1 and α_2 denote hyperparameters that control the contribution of each loss function.

(2) Optimizing LAC memory banks: To optimize the LAC memory banks, we compute the gradient of L_{C1} to update the LAC memory bank for both modalities. The optimization process is given as follows:

$$\hat{w}_{ij}^{(*)t} = w_{ij}^{(*)t-1} - \eta \nabla w_{ij}^{(*)}, \nabla w_{ij}^{(*)} = \frac{\partial L_{C1}}{\partial w_{ij}^{(*)t-1}}, \tag{28}$$

where $w_{ij}^{(*)t}$ denotes the value of $w_{ij}^{(*)}$ at the t-th iteration.

To effectively optimize $w_{ij}^{(*)}$, we employ the Euclidean projection method [Wang and Carreira-Perpinán, 2013] to map $\hat{w}_i^{(*)t} = \{w_{ij}^{(*)t}\}_{j=1}^{C}$ to $w_i^{(*)t} = \{w_{ij}^{(*)t}\}_{j=1}^{C}$, thereby obtaining the optimal value for each $w_{ij}^{(*)}$. The process is expressed as follows:

$$w_i^{(*)t} = \min \frac{1}{2}\left\| w_i^{(*)t} - \hat{w}_i^{(*)t} \right\|^2, \sum_{j=1}^{C} w_{ij}^{(*)} = 1, w_{ij}^{(*)} \geq 0. \tag{29}$$

where, $w_i^{(*)t}$ represents the label weight of x_i in the t-th iteration.

Finally, after optimizing the hash centers and the LAC memory banks, we can compute the more precise hash centroid $p_i^t = \frac{1}{2}(p_i^{(v)t} + p_i^{(t)t})$ based on Eq.(10). Therefore, we adopt an asynchronous mechanism in our model to optimize both the hash centroids and the hash codes.

4 Experiments

4.1 Experimental Settings

Datasets. For our experiments, we selected three widely used datasets for cross-modal hashing retrieval: MIRFLICKR-25K, MS-COCO, and NUS-WIDE. Their division methods followed those of previous works.

Baselines. In this experiment, we conducted a comprehensive comparison with other state-of-the-art DCMH methods, including both CNN-based and Transformer-based methods. Specifically, the CNN-based DCMH methods include AGAH [Gu *et al.*, 2019], PGCH [Bai *et al.*, 2023], DADH [Bai *et al.*, 2020], and HMAH [Tan *et al.*, 2022]. The Transformer-based DCMH methods include DCMHT [Tu *et al.*, 2022a], SCH [Hu *et al.*, 2024], DNpH [Qin *et al.*, 2024], DNPH [Huo *et al.*, 2024], and TWDH [Tu *et al.*, 2024].

Evaluation Metrics. To better demonstrate the advantages of our method, we adopted the mean average precision (mAP) of the top 5000 retrieval results to evaluate the retrieval performances of all methods.

4.2 Experimental Results

To evaluate the effectiveness of our proposed ACGH method, we conducted experiments on the MIRFLICKR-25K, MS-COCO, and NUS-WIDE datasets. We compared our method with its competitors in two tasks: Image-to-Text (I2T) and Text-to-Image (T2I). Table 1 shows the detailed comparison results of different DCMH methods on three datasets. It can be seen that our approach achieves the best retrieval performances across all bit sizes among all methods. Compared to the CNN-based DCMH method, such as AGAH, DADH, HMAH, and PGCH, the Transformer-based DCMH approaches achieve significant improvement across different bit settings. This is because the feature representation ability of Transformers is significantly better than that of CNNs. However, our method still achieves the best retrieval performance among Transformer-based DCMH approaches. Specifically, compared to the latest predefined hash center-based method, TWDH, our method achieves an average improvement of 2.03% at 16 bits, 2.94% at 32 bits, and 2.39% at 64 bits for the I2T task across three datasets. For the T2I task, the average improvements are 1.88% at 16 bits, 1.97% at 32 bits, and 1.15% at 64 bits. These results validate that

Methods	Reference	MIRFLICKR-25K			NUS-WIDE			MS-COCO		
		16bits	32bits	64bits	16bits	32bits	64bits	16bits	32bits	64bits
I → T										
AGAH	ICMR'2019	0.7558	0.7815	0.7763	0.7421	0.7821	0.7786	-	-	-
DADH	ICMR'2020	0.8080	0.8280	0.8346	0.6635	0.6700	0.6681	-	-	-
HMAH	TMM'2022	0.8496	0.8771	0.8734	0.7950	0.8002	0.8367	0.6629	0.6693	0.7307
DCMHT	ACMMM'2022	0.8512	0.8715	0.8715	0.7926	0.8021	0.8171	0.7025	0.7477	0.7693
PGCH	TBD'2023	0.7870	0.8151	0.8388	0.7819	0.8190	0.8134	0.5829	0.7389	0.7925
SCH	TPAMI'2024	0.8493	0.8735	0.8913	0.7419	0.7911	0.8113	0.6662	0.7261	0.7754
DNpH	TMM'2024	0.9053	0.9072	0.9111	0.8121	0.8209	0.8246	0.7350	0.7446	0.7454
DNPH	TCSVT'2024	0.8542	0.8788	0.8911	0.7927	0.8121	0.8253	0.7361	0.8089	0.8326
TWDH	TMM'2024	0.8770	0.8682	0.8911	0.8123	0.8287	0.8326	0.7378	0.7992	0.8391
ACGH	Ours	**0.9095**	**0.9195**	**0.9235**	**0.8245**	**0.8450**	**0.8605**	**0.7542**	**0.8199**	**0.8505**
T → I										
AGAH	ICMR'2019	0.7555	0.7664	0.7606	0.7416	0.7672	0.7622	-	-	-
DADH	ICMR'2020	0.7971	0.8169	0.8254	0.6522	0.6547	0.7081	-	-	-
HMAH	TMM'2022	0.8042	0.8248	0.8363	0.7438	0.7724	0.7857	0.7367	0.7780	0.8172
DCMHT	ACMMM'2022	0.8503	0.8647	0.8731	0.7570	0.7608	0.7733	0.6962	0.7478	0.7587
PGCH	TBD'2023	0.7548	0.7900	0.8106	0.7195	0.7762	0.7577	0.5703	0.7595	0.8181
SCH	TPAMI'2024	0.8511	0.8685	0.8802	0.7577	0.7893	0.7996	0.6797	0.7339	0.7770
DNpH	TMM'2024	0.8693	0.8731	0.8759	0.7879	0.7934	0.8010	0.7158	0.7396	0.7489
DNPH	TCSVT'2024	0.8560	0.8682	0.8779	0.7872	0.8097	0.8184	0.7227	0.8159	0.8364
TWDH	TMM'2024	0.8538	0.8638	0.8783	0.7872	0.8039	0.8175	0.7321	0.7980	0.8417
ACGH	Ours	**0.8732**	**0.8812**	**0.8879**	**0.8007**	**0.8258**	**0.8316**	**0.7557**	**0.8170**	**0.8526**

Table 1: Performance comparison of different DCMH methods on three multimodality datasets. The best result is highlighted in bold, and the second-best result is underlined.

the hash centroid guidance strategy, combined with LAC, is more effective than methods relying on predefined hash centers. Furthermore, the top-N precision curve in Figure 2 provides further evidence of the effectiveness of the proposed ACGH method, consistent with the mAP values in Table 1.

4.3 Ablation Study

To assess the contribution of each component in our ACGH framework, we designed four variants to evaluate their impact on the NUS-WIDE dataset. The results of the ablation study are presented in Table 2.

1. ACGH w/o L_{C1}: It was created by removing the loss function L_{c1}. We can observe that its performance is inferior to that of the proposed ACGH method on this dataset. Since it aims to update the LAC memory banks, the experimental results further demonstrate that this loss plays an indispensable role in our model.

2. w/o L_{SP}: This variant was constructed by eliminating the loss function L_{SP}. Its performance is significantly worse than that of our ACGH method. Since L_{SP} is used to preserve the semantic similarity of multimodalities, this suggests that it is crucial for the model's performance.

3. w/o L_{C2}: We constructed this variant of the proposed method by removing the loss function L_{C2}. Similarly, its performance is inferior to that of the ACGH method. It is clear that L_{C2} is designed to optimize the hash centers. Therefore, this ablation experiment further validates the effectiveness of continuously updating the hash center in enhancing the retrieval performance.

4. ACGH (PHC): This variant used predefined hash centers (PHC) to replace the adaptive hash centroids. The experimental results show that the proposed ACGH model significantly outperforms the variant in most cases, espe-

Methods	16bits	32bits	64bits
I → T			
ACGH w/o L_{C1}	0.8227	0.8424	0.8524
ACGH w/o L_{SP}	0.8113	0.8280	0.8359
ACGH w/o L_{C2}	0.8141	0.8373	0.8554
ACGH(PHC)	0.8208	**0.8473**	0.8377
ACGH	**0.8245**	0.8450	**0.8605**
T → I			
ACGH w/o L_{C1}	0.7899	0.8179	0.8260
ACGH w/o L_{SP}	0.7992	0.8138	0.8274
ACGH w/o L_{C2}	0.7950	0.8173	0.8310
ACGH(PHC)	0.8000	0.8086	0.8083
ACGH	**0.8007**	**0.8258**	**0.8316**

Table 2: The ablation study of the proposed method on the NUS-WIDE dataset.

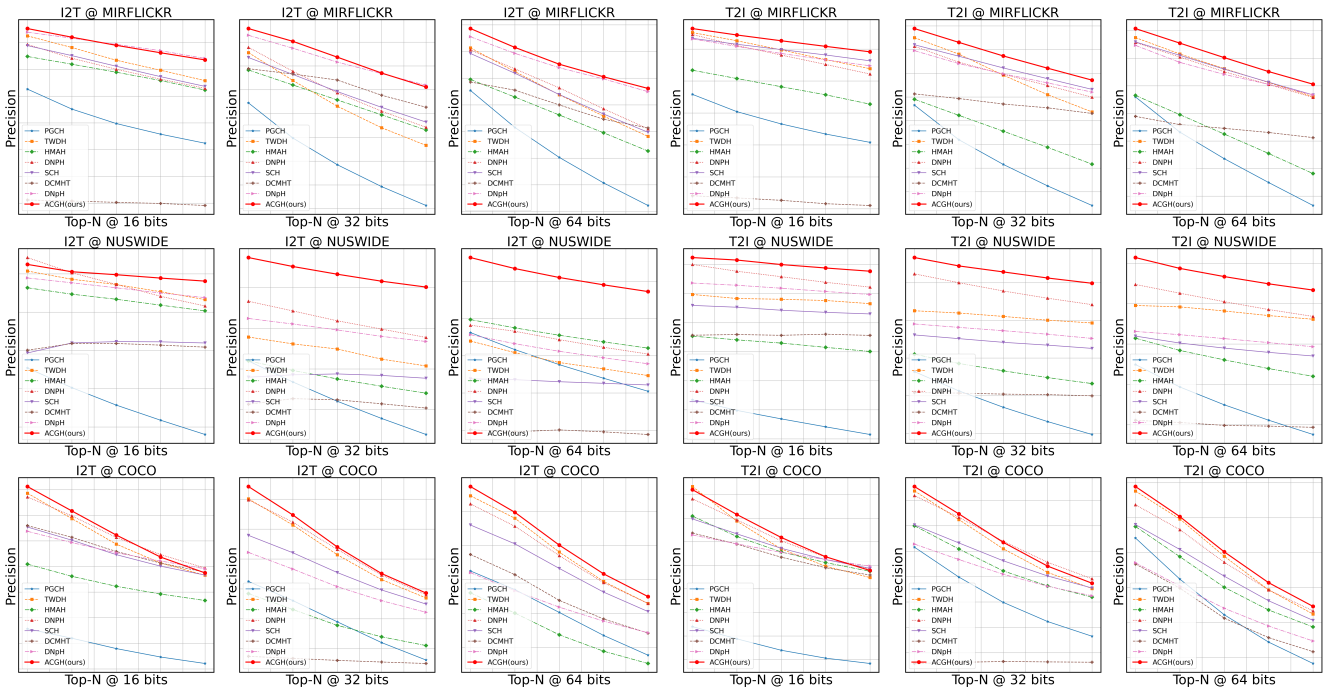

Figure 2: The P@N curves w.r.t. different code lengths on the MIRFLICKR-25K, NUS-WIDE, and MS-COCO datasets.

cially at higher bit sizes. Therefore, it demonstrates that the proposed adaptive hash centroid strategy is superior to the predefined hash center strategy in this framework.

5. ACGH: It can be seen that the full version of our proposed ACGH method outperforms all the aforementioned variants, further demonstrating the effectiveness of each module in our model.

4.4 Parameter Sensitivity

To demonstrate the parameter sensitivity of the proposed method, we conducted experiments on the NUS-WIDE dataset using a 16-bit setting. Specifically, we focused on four parameters in the model: μ, λ, α_1, and α_2. Specifically, the parameters μ and λ were set between 0.001 and 10, while the parameters α_1 and α_2 were analyzed within the range of 0.005 to 10. Figure 3 shows the retrieval performance of our ACGH method under different parameter settings. We can see from the results that the proposed ACGH method exhibits excellent stability across a wide range of parameter values on the NUS-WIDE dataset. Therefore, our proposed ACGH method can be easily applied to various retrieval tasks.

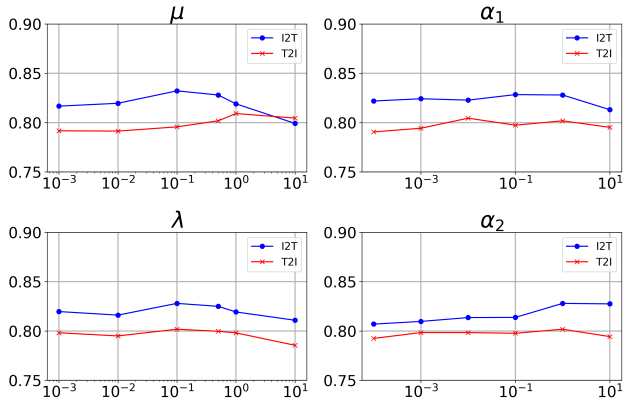

Figure 3: The mAP values of the proposed method under different parameter settings on the NUS-WIDE dataset.

5 Conclusion

In this paper, we propose a novel DCMH method termed adaptive centroid guided hashing (ACGH). Initially, we utilize Transformer models to extract global and local features for each modality, which are then fused into a comprehensive fine-grained feature representation. Next, we design two modules: the hash centroid generation module and the hash centroid guidance module. These modules employ an asynchronous optimization mechanism, enabling mutual promotion and optimization. Specifically, the hash centroid generation module maps the category embeddings obtained from the BERT model to the corresponding hash centers and constructs hash centroids using the LAC strategy. These hash centroids are then used to guide hash code learning. Simultaneously, the newly generated hash codes are used to update the LAC memory banks, thereby improving the accuracy of hash centroids. Therefore, this optimization mechanism enhances the accuracy of both hash codes and hash centroids. Extensive experimental results on three well-known multimodal datasets demonstrate the superiority of the proposed ACGH method in cross-modal retrieval applications.

References

[Bai *et al.*, 2020] Cong Bai, Chao Zeng, Qing Ma, Jinglin Zhang, and Shengyong Chen. Deep adversarial discrete hashing for cross-modal retrieval. In *Proceedings of the 2020 international conference on multimedia retrieval*, pages 525–531, 2020.

[Bai *et al.*, 2023] Yibing Bai, Zhenqiu Shu, Jun Yu, Zhengtao Yu, and Xiao-Jun Wu. Proxy-based graph convolutional hashing for cross-modal retrieval. *IEEE Transactions on Big Data*, 2023.

[Cui *et al.*, 2024] Hui Cui, Lihai Zhao, Fengling Li, Lei Zhu, Xiaohui Han, and Jingjing Li. Effective comparative prototype hashing for unsupervised domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8329–8337, 2024.

[Dosovitskiy, 2020] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[Gu *et al.*, 2019] Wen Gu, Xiaoyan Gu, Jingzi Gu, Bo Li, Zhi Xiong, and Weiping Wang. Adversary guided asymmetric hashing for cross-modal retrieval. In *Proceedings of the 2019 on international conference on multimedia retrieval*, pages 159–167, 2019.

[Hu *et al.*, 2022] Peng Hu, Hongyuan Zhu, Jie Lin, Dezhong Peng, Yin-Ping Zhao, and Xi Peng. Unsupervised contrastive cross-modal hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3877–3889, 2022.

[Hu *et al.*, 2024] Zhikai Hu, Yiu-ming Cheung, Mengke Li, and Weichao Lan. Cross-modal hashing method with properties of hamming space: A new perspective. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[Huo *et al.*, 2024] Yadong Huo, Qibing Qin, Wenfeng Zhang, Lei Huang, and Jie Nie. Deep hierarchy-aware proxy hashing with self-paced learning for cross-modal retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[Jiang and Li, 2017] Qing-Yuan Jiang and Wu-Jun Li. Deep cross-modal hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3232–3240, 2017.

[Kenton and Toutanova, 2019] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota, 2019.

[Liu *et al.*, 2020] Song Liu, Shengsheng Qian, Yang Guan, Jiawei Zhan, and Long Ying. Joint-modal distribution-based similarity hashing for large-scale unsupervised deep cross-modal retrieval. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1379–1388, 2020.

[Luo *et al.*, 2021] Xiao Luo, Daqing Wu, Zeyu Ma, Chong Chen, Minghua Deng, Jianqiang Huang, and Xian-Sheng Hua. A statistical approach to mining semantic similarity for deep unsupervised hashing. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4306–4314, 2021.

[Mikriukov *et al.*, 2022] Georgii Mikriukov, Mahdyar Ravanbakhsh, and Begüm Demir. Unsupervised contrastive hashing for cross-modal retrieval in remote sensing. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4463–4467. IEEE, 2022.

[Qin *et al.*, 2024] Qibing Qin, Yadong Huo, Lei Huang, Jiangyan Dai, Huihui Zhang, and Wenfeng Zhang. Deep neighborhood-preserving hashing with quadratic spherical mutual information for cross-modal retrieval. *IEEE Transactions on Multimedia*, 2024.

[Radford *et al.*, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[Shu *et al.*, 2022] Zhenqiu Shu, Yibing Bai, Donglin Zhang, Jun Yu, Zhengtao Yu, and Xiao-Jun Wu. Specific class center guided deep hashing for cross-modal retrieval. *Information sciences*, 609:304–318, 2022.

[Sun *et al.*, 2023] Lina Sun, Yewen Li, and Yumin Dong. Learning from expert: Vision-language knowledge distillation for unsupervised cross-modal hashing retrieval. In *Proceedings of the 2023 ACM International Conference on Multimedia Retrieval*, pages 499–507, 2023.

[Tan *et al.*, 2022] Wentao Tan, Lei Zhu, Jingjing Li, Huaxiang Zhang, and Junwei Han. Teacher-student learning: Efficient hierarchical message aggregation hashing for cross-modal retrieval. *IEEE Transactions on Multimedia*, 25:4520–4532, 2022.

[Tu *et al.*, 2022a] Junfeng Tu, Xueliang Liu, Zongxiang Lin, Richang Hong, and Meng Wang. Differentiable cross-modal hashing via multimodal transformers. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 453–461, 2022.

[Tu *et al.*, 2022b] Rong-Cheng Tu, Xian-Ling Mao, Rong-Xin Tu, Binbin Bian, Chengfei Cai, Hongfa Wang, Wei Wei, and Heyan Huang. Deep cross-modal proxy hashing. *IEEE Transactions on Knowledge and Data Engineering*, 35(7):6798–6810, 2022.

[Tu *et al.*, 2024] Junfeng Tu, Xueliang Liu, Yanbin Hao, Richang Hong, and Meng Wang. Two-step discrete hashing for cross-modal retrieval. *IEEE Transactions on Multimedia*, 2024.

[Wang and Carreira-Perpinán, 2013] Weiran Wang and Miguel A Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*, 2013.

[Xu *et al.*, 2019] Ruiqing Xu, Chao Li, Junchi Yan, Cheng Deng, and Xianglong Liu. Graph convolutional network hashing for cross-modal retrieval. In *Ijcai*, volume 2019, pages 982–988, 2019.

[Yu *et al.*, 2021] Jun Yu, Hao Zhou, Yibing Zhan, and Dacheng Tao. Deep graph-neighbor coherence preserving network for unsupervised cross-modal hashing. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4626–4634, 2021.