

# ViT2CMH: Vision Transformer Cross-Modal Hashing for Fine-Grained Vision-Text Retrieval

Mingyong Li, Qiqi Li, Zheng Jiang and Yan Ma\*

College of Computer and Information Science, Chongqing Normal University, Chongqing, 401331, China

\*Corresponding Author: Yan Ma. Email: cqu\_mayan@163.com

Received: 26 July 2022; Accepted: 13 November 2022

**Abstract:** In recent years, the development of deep learning has further improved hash retrieval technology. Most of the existing hashing methods currently use Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to process image and text information, respectively. This makes images or texts subject to local constraints, and inherent label matching cannot capture fine-grained information, often leading to suboptimal results. Driven by the development of the transformer model, we propose a framework called ViT2CMH mainly based on the Vision Transformer to handle deep Cross-modal Hashing tasks rather than CNNs or RNNs. Specifically, we use a BERT network to extract text features and use the vision transformer as the image network of the model. Finally, the features are transformed into hash codes for efficient and fast retrieval. We conduct extensive experiments on Microsoft COCO (MS-COCO) and Flickr30K, comparing with baselines of some hashing methods and image-text matching methods, showing that our method has better performance.

**Keywords:** Hash learning; cross-modal retrieval; fine-grained matching; transformer

## 1 Introduction





In recent years, with the development of multimedia and big data, deep learning networks for processing images and texts are also developing rapidly. The typical ones are Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) [1–3]. Various CNNs have been studied to target different application scenarios, including object detection [4], classification [5], recognition [6], retrieval [7], etc. Similarly, different text networks are used to handle tasks such as emotion classification [8], machine translation [9], etc. In the field of deep hash retrieval, the early deep hash learning mostly used CNNs to extract image features, and used Long Short-Term Memory (LSTM) or RNNs [1,10] to process text. Although CNNs have achieved good development, CNNs are limited to extracting local features and cannot achieve a better global representation of images in feature space. On the other hand, RNNs can utilize the entire data represented by textual features, however, RNNs are sequence-constrained, and the output of the previous time step can be obtained to determine the input of the current time step. With the development of deep neural networks, the emergence of transformer [11] enables a high degree of parallelization and can model the global representation of features at each step. So the transformer



This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

architecture has achieved good results in some tasks of natural language processing, such as translation and sentence classification. Among them, BERT [12] demonstrates the powerful role of the attention mechanism in accurately perceiving textual context. The transformer encoder can be used to process not only texts but also images, and its attention mechanism can connect different image patches to capture important relationships between objects. Vision Transformer (ViT) [13] performs well on image recognition tasks using self-attention without convolution, achieving results comparable to current state-of-the-art convolutional neural networks such as AlexNet [5] and VGGNet [14]. Recently, some researchers tried to use ViT for image hash retrieval and achieved good results. Such as TransHash [15] and Vision Transformer based Hashing (VTS) [16]. These methods are single-modal image hashing proposed with the development of transformer. However, there are few transformer-based fine-grained deep hashing. We believe that fine-grained cross-modal hashing with transformer as the backbone is worth a try.

Most existing deep image text hashing utilizes the shared label information of the input image text, for example, MS-COCO [17] contains 80 labels. Fig. 1 shows several text queries with complex semantic information. We can see that these methods convert cross-modal matching into label matching, but these only use the rough label information, ignoring the fine-grained high-order semantic information, often less than expected results. Fine-grained semantic information includes higher-order relationships and numerical information between objects. Label-matching methods may often place too much importance on the objects appearing in the query and fail to capture some important fine-grained information, for example, the number of cats in the first example in Fig. 1 and the man standing next to horses in the second example instead of riding a horse. Our work is devoted to exploiting the reasoning ability and attention mechanism of transformer encoders to study efficient cross-modal retrieval of images and texts with fine-grained semantics. Both of these methods are designed to study concept correlation between different modalities and achieve effective cross-modal matching. But the two methods have different degrees of refinement for the correlation of different modal concepts. Most existing hashing methods simply rely on matching a pre-specified set of objects, resulting in missing semantic relationships between objects and attributes or numeral information. Different from these methods, to accomplish these tasks, we need to have a deep semantic understanding of images and texts and analyze high-level semantic relationships between images and texts.

Query Text		Result	Ground Truth
Two cats sitting on a green towel inside of a bathroom.	label cat towel bathroom		
Two horses standing in a field and a man standing next to them.	label horse field man		

**Figure 1:** Examples of text queries for most existing hashing methods

For these reasons, we decide to try to exploit the processing power of transformer encoder and propose a fine-grained cross-modal hashing method based on transformer. The method first extracts image and text features separately, converts them into binary codes after hashing, and then matches them to a common Hamming space for efficient retrieval. The two branches of the model utilize the self-attention mechanism

of transformer to process images and texts separately. After obtaining compact image and text features, they are jointly input to a modal interaction layer with shared weights, and finally, the hash codes are output. Our contributions can be summarized as follows:

1. We point out the problem that most existing hashing methods focus on label matching, and such coarse-grained methods often lead to suboptimal results. We are dedicated to exploring fine-grained matching methods in the current field and propose a fine-grained semantic hashing method, which is an early attempt in the field of cross-modal hashing.
2. We propose an end-to-end cross-modal hashing method entirely based on transformer encoder, which is one of the pioneering works to solve the fine-grained deep cross-modal hashing problem without using convolutional neural networks at all. We use evaluation metrics commonly used in the field of image-text matching. Experiments show on MS-COCO and Flickr30K that our model can achieve advanced results.

## 2 Related Work

### 2.1 Coarse-Grained Deep Hashing

Hash learning has the advantages of high efficiency and low storage and is widely used in large-scale data retrieval. Cross-modal hashing of images and text maps the data in the original feature space of the modality to a common Hamming space, which is stored in the form of binary codes. The similarity is judged by sorting the Hamming distance between the binary code of the query data and the binary code of the database data. The smaller the Hamming distance, the higher the similarity. We can summarize hash learning methods into two types: shallow hashing method and deep learning based hashing method. Shallow hashing methods extracted hand-made features through some shallow structures. The Collective Matrix Factorization Hashing (CMFH) proposed by Ding et al. [18] assumed that the same sample in different models generates the same hash code, and learns different hash codes in the shared potential semantic space. The Semantic Preserving Hashing (SePH) method proposed by Lin et al. [19] converts the semantic affinity of training data into a probability distribution, and it is similar to the hash code to be learned in Hamming space by minimizing KL divergence (Kullback-Leibler Divergence).

In recent years, deep learning-based deep hashing methods have performed well in retrieval performance and efficiency. Deep Supervised Hashing (DSH) [20] was one of the early attempts to use CNNs to quantify image features into binary hash codes. Deep Cross-Modal Hashing (DCMH) [7] was the first to combine deep learning and hash learning and propose an end-to-end learning framework for the cross-modal hashing of images and texts. Self-Supervised Adversarial Hashing Networks for Cross-Modal Retrieval (SSAH) [21] was an early attempt to incorporate adversarial learning into self-supervised methods for deep cross-modal hashing. The fusion of early and late features of Self-constraining and attention-based hashing network for bit-scalable cross-modal retrieval (SCAHN) [22] was based on attention mechanism, which is integrated into hash representation and hash function learning together with early and late label constraints. Unsupervised hashing methods achieve retrieval tasks by mining structural associations between two modalities of data. Joint-modal Distribution-based Similarity Hashing for Large-scale Unsupervised (DJSRH) [23] proposed a joint semantic affinity matrix for input multi-modal instances that carefully integrates raw neighborhood relations from different modalities, thus being able to capture potential inter-instance intrinsic semantic affinity. Deep Graph-neighbor Coherence Preserving Network for Unsupervised Cross-modal Hashing (DGCPN) [24] is derived from a graph model and solves the problem of insufficient accuracy by exploring the fixed nature of the data in the graph.

## 2.2 Transformer-Based Fine-Grained Retrieval Method

Transformer encoder consists of  $L$  identical blocks stacked. Each transformer block contains two parts, one is a multi-head self-attention layer, and the other is a feed-forward neural network. Transformer block uses a self-attention layer to non-linearly transform the input features to capture the internal correlation of features and reduce the dependence on external data. The attention function is described as a mapping from a query to a series of Key-Value pairs. Attention filters and focuses important information from large-scale information. The weight indicates the importance of the information, and the Value indicates the information corresponding to the weight.

The self-attention mechanism is a variant of the attention mechanism, which reduces the dependence on external information and is better at capturing the internal correlation of data or features. The attention mechanism can be more formally expressed as:

$$Att(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

Among  $Q \in R^{t \times d_k}$ ,  $K \in R^{s \times d_k}$ ,  $V \in R^{s \times d_v}$ ,  $s$  and  $t$  are the length of the input sequence and adjustment sequence, respectively.  $\sqrt{d_k}$  is used to ease the disappearance of the softmax gradient. When  $Q$ ,  $K$ ,  $V$  are all calculated from the same input set, self-attention is derived from the general attention mechanism. Under the circumstances,  $t = s$  and  $QK^T$  is a square matrix, which encodes the correlation between each element in the set and all other elements in the set. After the multi-head self-attention, the feed-forward neural network is essentially two fully connected layers, which can be expressed as follows:

$$FFN(X) = \text{Relu}(xW_1 + b_1)W_2 + b_2 \quad (2)$$

It is used to further process the vector generated by the self-attention mechanism, where  $x \in R^{1 \times d_x}$ ,  $W_1 \in R^{d_x \times d_x}$ ,  $W_2 \in R^{d_x \times d_x}$ ,  $b_1 \in R^{1 \times d_x}$ ,  $b_2 \in R^{1 \times d_x}$ . Whether it is visual or textual, the transformer encoder self-attention mechanism can discover the hidden relationship between vector entities.

In transformer-based fine-grained deep hashing, TransHash [15] builds a Siamese model based on vision transformer for image hash learning. Vision Transformer Hashing (VTS) [16] uses the ViT pretrained model to extract real-valued features, and adds a hashing module to improve the objective function by using recently studied image hashing methods. HashFormer [25] utilized the Vision Transformer as the backbone and binary code as an intermediate representation for the surrogate task. There are also some fine-grained but non-hashing methods. Learning Fragment Self-Attention Embeddings for Image-Text Matching (SAEM) [26] extracts salient image regions by bottom-up attention and labels the headwords as sentence fragments. Build self-attention layers to model fine-grained and fine-grained segment relationships in images and text, respectively. Transformer Reasoning Network for Image-Text Matching and Retrieval (TERN) [27] infers two different modalities separately and enforces the final common abstract concept space by sharing the weights of deeper transformer layers.

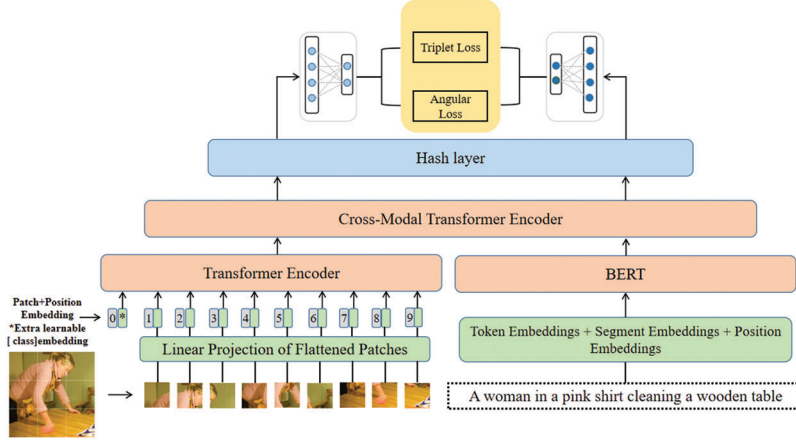
## 3 Proposed Method

### 3.1 Problem Formulation

Our purpose is to encode input images and text instances into hash codes and preserve the original similarity information. Suppose we have a dataset  $X = \{x_i\}_{i=1}^N$ . It contains a set of images  $I$  and description text  $T$ , and image-text pair information provided by annotations to supervise the learning process. The Hamming distance of the converted binary code is used to judge whether they are similar, and the Hamming distance of the same pair of image-text pairs should be smaller.

### 3.2 Model Overview

Our model architecture is shown in Fig. 2. Our work is based on the transformer encoder structure, we take the patches of the images as the input of the visual pipeline and the words in the text as the input of the language pipeline. Transformer encoder can reason about these entities without considering their essence.



**Figure 2:** Our model framework (ViT2CMH). Pictures are processed by ViT. Words are extracted using BERT.  $I - CLS$  and  $T - CLS$  are used to carry global information along the pipeline respectively. Global approximate hash codes for images and text are generated by a hash layer after a cross-modal transformer encoder with shared weights, finally, the hash layer outputs to calculate the loss

**Image And Text Embeddings:** We take the original image of size  $H \times W \times 3$  as input, and then divide each image into  $N$  patches of fixed size, each patch is  $P \times P \times 3$ . The number of patches  $N = H \times W / P^2$ . The patches of the image are represented as the set  $I = \{i_1, i_2, \dots, i_n\}$  as the input of the image pipeline. The sliced patches are then linearly mapped into a 1D vector and input to the embedding layer, which then converts the input image format to the standard transformer input format. The words of the text are represented as the sequence  $C = \{w_1, w_2, \dots, w_m\}$  as the input of the text pipeline. The embedding process can be written as:

$$Z = [x_{cls}; x_1 W; x_2 W; \dots; x_n W] \quad (3)$$

$W$  represents the linear transformation of the input, then add the location information of each token:

$$X = pos + Z \quad (4)$$

**Transformer:** The backbone of ViT and BERT is  $L$  transformer encoders. Each transformer encoder consists of a multi-head self-attention layer and a Multi layer Perceptron (MLP). Layer Norm (LN) before each layer, residual connections after each layer. The computation of each transformer encode block can be expressed as:

$$\begin{aligned} z_l &= F_a(F_{ln}(z_{l-1})) + z_{l-1} \\ z_l &= F_m(F_{ln}(z_l)) + z_l \end{aligned} \quad (5)$$

where  $l = 1, 2, \dots, L$ ,  $F_a$  stands for multi-headed self-attention layer and  $F_m$  stands for MLP layer.

**Global Fine-grained Feature:** We set a special token at the beginning of the image collection and text sequence of the model, which are called  $I - CLS$  and  $T - CLS$  respectively.  $T - CLS$  is in BERT and  $I - CLS$  is initially a zero vector. During training and learning, the transformer encoder's self-attention

mechanism updates this information. After passing through the last layer of transformer encoder,  $T - CLS$  and  $I - CLS$  carry the global information of text and images and are output by the hash layer, and then calculate the loss. These two tokens carry global information and are transmitted along two pipelines. Therefore, the number of image patches set becomes  $n + 1$ , and the number of text word sequences becomes  $m + 1$ . The model shares the weight before calculating the loss, which can enhance the comparability of the image and the visual abstract representation.

**Hash:** The  $I - CLS$  and  $T - CLS$  carrying the global fine-grained information go through the last layer of transformer encoders and then input to the hash layer. During the training phase, the hash layer is processed using the  $\tanh()$  function, and we obtain a hash vector of length  $b$  as follows:

$$B = \tanh(out - \text{median}(\text{softmax}(out))) \quad (6)$$

### 3.3 Loss Function

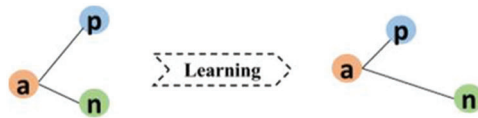
We use a hinge-based triplet ranking loss [1]. In order to match images and texts in the same space, we define a scoring function with the inner product  $s$ , scoring function  $s(i, j)$  equivalent to the cosine similarity between image and text features. Therefore, our triplet loss function is expressed as:

$$L_{\text{triplet}}(i, c) = \max[0, m - s(i, c) + s(i, c')] + \max[0, m - s(i, c) + s(i', c)] \quad (7)$$

Among  $m$  represents the margin parameter in triplet loss,  $(i, c)$  represents a positive pair,  $c'$  represents a negative sentence of image  $I$ ,  $i'$  represents a negative image of text  $C$ .  $c'$  and  $i'$  calculated by the following:

$$\begin{aligned} c' &= \underset{d \neq c}{\operatorname{argmax}} s(i, d) \\ i' &= \underset{j \neq i}{\operatorname{argmax}} s(j, c) \end{aligned} \quad (8)$$

The relationship of anchor, positive and negative can be expressed as Fig. 3:



**Figure 3:** The relationship change of anchor, positive and negative after learning

The effect of the margin parameter is that it widens the gap between Anchor and Positive pair and Anchor and Negative pair. For example: The distance between Anchor and Positive can be expressed as  $d(f(A), f(P))$ , The distance between Anchor and Negative can be expressed as  $d(f(A), f(N))$ , after learning,  $d(f(A), f(P))$  should be less than or equal to  $d(f(A), f(N))$  as:

$$d(f(A), f(P)) - d(f(A), f(N)) \leq 0 \quad (9)$$

But one factor to consider is if

$$d(f(A), f(P)) = d(f(A), f(N)) \quad (10)$$

it is very likely that after the learning is over, all  $f$  will be learned as 0 vectors, so although the equation can always be satisfied, it is not the result we want. So in order to ensure that the network does not always output 0 for all codes, and to ensure that it does not set all codes to be equal to each other, we need to modify this goal, that is, this cannot be exactly less than or equal to 0, it should be less than 0, so this should be less than a value of  $-m$ , which is the margin parameter.



Here, considering the performance, we only select hard negatives from one mini-batch, which can obtain better retrieval performance and computational efficiency than all.

In addition, in order to improve the robustness, we also introduce an angular loss [28]. Angular loss adds the triplet geometric constraint and captures the additional local structure of triplet triangles. Angular loss encodes the triplet relationship within triplet triangles by constraining the angle at the negative point of triplet triangles. Given the same triples, it provides additional constraints to ensure that dissimilar points can be separated. Angle is a measure of invariant rotation and scaling, which makes the target more robust to the large changes of feature mapping in real data. In the original paper on angular loss, the author integrated angular loss and N-pair loss [29]. N-pair loss allows only one positive sample pair for each class. N-pair loss uses all negative samples in batch to guide gradient update, thereby accelerating convergence. We optimize bi-directional angular loss, using the following formula:

$$F(a, p, n) = 4 \tan^2 \alpha (a + p)n^T - 2(1 + \tan^2 \alpha) ap^T \quad (11)$$

$$L_{angular}(i, c) = \log[1 + \exp(F(i, c, c'))] + \log[1 + \exp(F(c, i, i'))] \quad (12)$$

In which  $a, p, n$  respectively represent images or texts,  $\alpha$  represents the angle boundary parameter. The angle parameter limits the angle of triplet triangles in the angular loss.  $c'$  and  $i'$  are hard negatives selected from a mini-batch, which are calculated by the following formulas:

$$\begin{aligned} c' &= \operatorname{argmax}_{c \neq d} F(i, c, d) \\ i' &= \operatorname{argmax}_{i \neq j} F(c, i, j) \end{aligned} \quad (13)$$

Finally, the angular loss is combined with hard-negative-based triple loss, which is our formula:

$$L(i, c) = L_{triplet}(i, c) + \eta L_{angular}(i, c) \quad (14)$$

In the following part, we will introduce our experiments and compare the influence of different value settings of hyperparameter  $\eta$  on the model.

## 4 Experiments

**Datasets:** After training the model, we made a comparison with some previous work on the two public datasets, Microsoft COCO [17] and the Flickr30K [30], and we also did some comparative experiments on our model.

**Microsoft COCO:** The 2014 edition of Microsoft COCO contains 82,783 training images, 40,504 verification images and 40,775 test images (about 1/2 training, 1/4 verification and 1/4 test). Each picture has 5 captions. We followed the allocation method in [10], using 113,287 pictures for training, 5,000 pictures for validation and 5,000 pictures for testing. The test result is reported by the average of 1,000 test images converted 5 times.

**Flickr30K:** Flickr30K collected 31,783 pictures on the Flickr website, among which each picture has 5 descriptions corresponding to it. Following the segmentation method disclosed in [1,10], we used 1,000 images for validating, 1,000 for testing and the rest for training.

**Evaluation Metrics:** For the evaluation metrics, we believe that the mean of Average Precision (MAP) used in the hash retrieval field is based on a multi-label matching scheme. As a fine-grained matching method, we believe that the Recall@topK method in the image-text matching field may be more suitable.

Recall@topK refers to the ratio of the number of relevant results retrieved in the topK results to the number of all relevant results in the database. Recall@topK is calculated as follows:

$$R@K = \frac{TP@K}{N} \quad (15)$$

where TP@K represents the number of retrieved relevant results appearing in the top K positions, and N represents the number of positive samples. K is set to 1, 5 and 10, indicating the percentage of matching items appearing in the first, top 5, and top 10 positions respectively. The larger the R@K, the greater the probability that the retrieved matching results are ranked in the front.

#### 4.1 Implementation Details

In the image reasoning part, our model is a reimplementation of Google’s repository for the ViT model. We use the version of the ViT-B-16 pre-trained model, which uses the patch sizes of 16. For the text, we use the BERT model pre-trained in the covering language task of English sentences implemented by HuggingFace. After the image and text pipes, we use two transformer encoder layers with shared weights to connect.

Because the image and text pipelines have to pass through the transformer encoder layer with shared weight, their output is set to 1024 dimensions, and the output dimension of the transformer encoder layer with shared weight is 1024 dimensions. We set the batch size to 30 and trained 30 epochs using the Adam optimizer with a learning rate of 0.00001. The values of  $m$  and  $\alpha$  are based on some experience and experiments. From experience, most researchers like to set  $m$  to 0.2. For  $\alpha$ , in its original paper, the model performs better when  $\alpha = 45^\circ$ .

#### 4.2 Performance Comparison

We first compare with some state-of-the-art and open source cross-modal hashing methods on the two datasets in Table 1. Some data in Table 1 come from [31]. Due to the poor performance of previous methods, we only report the test results on 100 samples. For a fair comparison, we conduct the experiments with the hash code set to 128 bits compared to the previous hashing method. It is obvious that our model can achieve much better results than previous hashing methods. Compared to the State-Of-The-Art (SOTA), our method outperforms in most cases. Then we make a comparison with [31], which is also a fine-grained cross-modal hashing method, but their method is a two-step strategy, the first step is coarse-grained primary screening, and the second is fine-grained reranking. For a fair comparison, we only compare with their proposed fine-grained Full-Reranking method. We do the same settings as their original paper, and the comparison results are as Table 2. As can be seen from the table, the performance of our model is somewhat higher than the current baseline performance.

**Table 1:** Comparison with other classical methods under two datasets (MS-COCO, Flickr30K)

Task	Method	MS-COCO			Flickr30K		
		R@1	R@5	R@10	R@1	R@5	R@10
I->T	<b>ViT2CMH(128)</b>	<b>53.3</b>	84.6	<b>93.7</b>	<b>54.9</b>	<b>81.5</b>	<b>90.3</b>
	DCMH [7]	4.0	17.0	29.0	2.0	6.0	12.0
	CMHH [32]	5.0	8.0	24.0	2.0	6.0	12.0
	SCAHN [22]	19.0	47.0	67.0	3.0	13.0	24.0
	DJSRH [23]	51.0	86.0	93.0	25.0	57.0	68.0
	<b>ViT2CMH(128)</b>	<b>44.5</b>	80.3	89.9	<b>40.8</b>	<b>72.7</b>	<b>81.2</b>

(Continued)



**Table 1 (continued)**

Task	Method	MS-COCO			Flickr30K		
		R@1	R@5	R@10	R@1	R@5	R@10
T->I	DCMH [7]	4.0	14.0	27.0	2.0	8.0	9.0
	CMHH [32]	4.0	15.0	20.0	3.0	9.0	17.0
	SCAHN [22]	17.0	48.0	68.0	3.0	11.0	21.0
	DJSRH [23]	41.0	<b>85.0</b>	<b>93.0</b>	28.0	55.0	66.0

**Table 2:** Comparison with the current baseline method

Flickr30K	Image-to-Text			Text-to-Image		
Method	R@1	R@5	R@10	R@1	R@5	R@10
<b>ViT2CMH(1024)</b>	<b>66.4</b>	<b>90.2</b>	<b>94.0</b>	<b>52.8</b>	<b>82.7</b>	<b>89.8</b>
FullRerank(1024)	62.3	86.7	93.4	43.1	74.0	83.4

We also compare the performance of the models under different hash code lengths in Table 3. We set the hash code to different lengths commonly used in the hash field for comparison. Experimental results show that the performance of our fine-grained model is sensitive to hash code length. The hash code length of most of the current hashing methods does not have a great impact on performance, because they cannot capture more fine-grained semantic information, and shorter hash codes can already satisfy such methods as tag matching. For fine-grained methods, longer hash codes are required to represent more and finer semantic information.

**Table 3:** Comparison of different code lengths of our method under two datasets (MS-COCO, Flickr30K)

Task	Method	MS-COCO			Flickr30K		
		R@1	R@5	R@10	R@1	R@5	R@10
I->T	ViT2CMH(64)	45.9	79.7	90.4	45.5	75.4	82.6
	ViT2CMH(128)	53.3	84.6	93.7	54.9	81.5	90.3
	ViT2CMH(256)	58.4	86.5	94.2	59.6	86.1	92.0
	ViT2CMH(512)	61.3	88.4	95.3	62.5	87.9	93.8
	<b>ViT2CMH(1024)</b>	<b>67.9</b>	<b>91.7</b>	<b>96.9</b>	<b>66.4</b>	<b>90.2</b>	<b>94.0</b>
T->I	ViT2CMH(64)	38.9	74.7	84.9	33.5	63.4	73.2
	ViT2CMH(128)	44.5	80.3	89.9	40.8	72.7	81.2
	ViT2CMH(256)	48.0	83.2	91.9	46.3	77.9	85.8
	ViT2CMH(512)	50.7	85.4	93.3	49.7	80.8	87.6
	<b>ViT2CMH(1024)</b>	<b>59.6</b>	<b>88.2</b>	<b>94.9</b>	<b>52.8</b>	<b>82.7</b>	<b>89.8</b>

We compare not only with hashing methods but also with some image-text matching methods based on continuous embeddings on the 1 k test set in Table 4. Some results come from [33]. The results tell that our model is in no way inferior to these powerful image-text matching methods.

**Table 4:** Comparison with image-text matching methods under two datasets (MS-COCO, Flickr30K)

Task	Method	MS-COCO			Flickr30K		
		R@1	R@5	R@10	R@1	R@5	R@10
I->T	<b>ViT2CMH</b>	67.9	<b>91.7</b>	<b>96.9</b>	<b>66.4</b>	<b>90.2</b>	<b>94.0</b>
	TERN [27]	63.7	90.5	96.2	53.2	79.4	86.0
	VSE++ [1]	64.6	90.0	95.7	52.9	80.5	87.2
	SMAN [34]	68.4	91.3	96.6	57.3	85.3	92.2
	M3A-Net [35]	<b>70.4</b>	<b>91.7</b>	96.8	58.1	82.8	90.1
T->I	<b>ViT2CMH</b>	<b>59.6</b>	<b>88.2</b>	<b>94.9</b>	<b>52.8</b>	<b>82.7</b>	<b>89.8</b>
	TERN [27]	51.9	84.9	93.6	41.1	71.9	81.2
	VSE++ [1]	52.0	84.3	92.0	39.6	70.1	79.5
	SMAN [34]	58.8	87.4	92.0	43.4	73.7	83.4
	M3A-Net [35]	58.4	87.1	94.0	44.7	72.4	81.1

### 4.3 Ablation Experiment

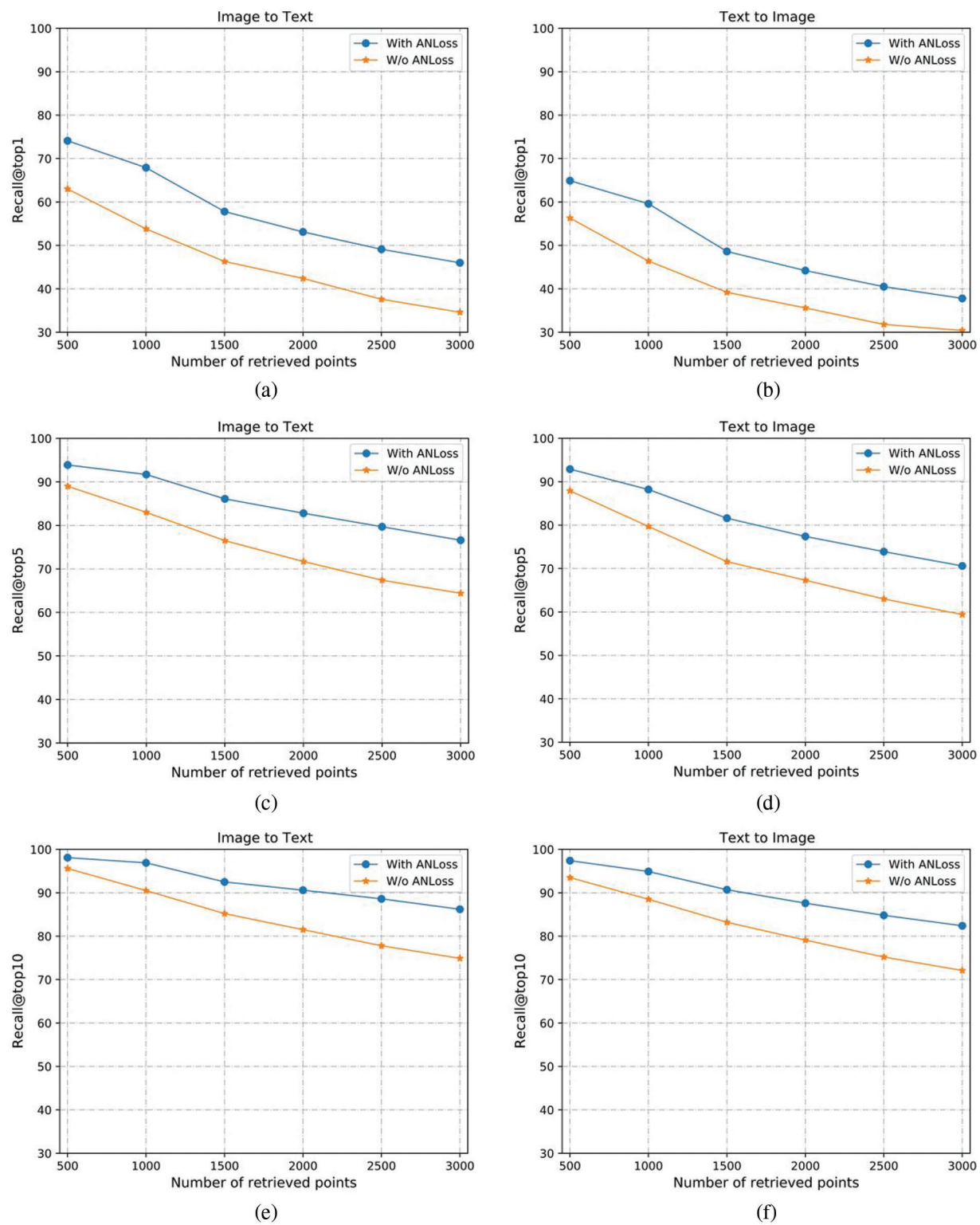
Our loss function consists of triplet loss and angular loss. In order to study the impact of angular loss on the model, we compared the performance of the model on MS-COCO with different values of  $\eta$ . The experimental results are shown in Table 5.

**Table 5:** The effect of hyperparameter  $\eta$  on model retrieval performance

MS-COCO	Image-to-text			Text-to-image		
	R@1	R@5	R@10	R@1	R@5	R@10
$\eta = 1$	65.3	90.6	96.8	57.5	87.4	93.5
$\eta = 0.5$	67.4	90.9	96.3	56.7	87.4	94.4
$\eta = 0.1$	67.9	91.7	96.9	59.6	88.2	94.9
$\eta = 0$	53.8	83.0	90.5	46.4	79.7	88.5

From the experimental data, we can see that only using triplet loss, the performance of the model is not as good as using triplet loss and angular loss. And the model can perform best when  $\eta$  is set to 0.1.

To verify the improvement of the robustness of the model by angular loss, we conduct experiments on up to 3,000 retrieval samples, and the experimental results are shown in Fig. 4. As can be seen from the figure, as the number of retrieved samples increases, using angular loss will improve the robustness of the model, and the gap between using angular loss and without angular loss will become larger and larger. With the increase of retrieval samples, the performance of the model without AN loss will decrease more. Whether it is image retrieval or text retrieval, angular loss has a positive effect on the improvement of model performance in both tasks.



**Figure 4:** Recall@topK curves on MS-COCO. The effect of ANLoss on the model is relatively obvious

In addition, to verify the improvement of our method over convolutional neural networks, we compare it with the Resnet-101 pretrained model for extracting raw image features. Other than that, all the settings of the two methods are the same. We conduct experiments on MS-COCO to verify the performance comparison of the shorter hash code (64 bits) and the longer hash code (1024 bits). We present the experimental results in Table 6. ViT2CMH-f stands for our fine-grained feature extraction method and ViT2CMH-c stands for the Convolutional Neural Network feature extraction method. As can be seen from the reported results, adopting the ViT to extract the vision feature is very effective. The Resnet101 pretrained model already has excellent performance, but for fine-grained hashing, our model performs better.

**Table 6:** The impact of different feature extraction methods

MS-COCO	Image-to-Text			Text-to-Image		
Method	R@1	R@5	R@10	R@1	R@5	R@10
<b>ViT2CMH-f(64)</b>	<b>45.9</b>	<b>79.7</b>	<b>90.4</b>	<b>38.9</b>	<b>74.7</b>	<b>84.9</b>
ViT2CMH-c(64)	35.4	63.0	73.6	27.1	50.7	59.1
<b>ViT2CMH-f(1024)</b>	<b>67.9</b>	<b>91.7</b>	<b>96.9</b>	<b>59.6</b>	<b>88.2</b>	<b>94.9</b>
ViT2CMH-c(1024)	53.5	81.9	90.8	43.2	76.3	84.2

#### 4.4 Examples of Retrieval

In Fig. 5, we show examples of results of text query images using our model. The displayed results are generated on the Flickr30K dataset. Each image has five sentence descriptions corresponding to it, and each sentence description has only one pair of pictures. Three examples of text query images are shown. We use green to identify the pairs of images.

Query: A group of people standing on the lawn in front of a building.



Query: Three people sit at an outdoor table in front of a building painted like the Union Jack.



Query: A security officer with a tiny face and big glasses leans on a metal gate looking into the camera.



**Figure 5:** Examples of the first four images found by text query on the Flickr30K dataset

Through the query examples, we can see that our method can find the corresponding information effectively and accurately. In Fig. 4, our method can ensure that semantic matching images are retrieved. In the first four locations, except for the ground truth image, the other images retrieved are also images with similar semantics.

## 5 Conclusion

In this paper, we first discuss the limitations of multi-label matching in most current hashing methods when faced with complex semantic queries. We propose a method for extracting global fine-grained features of images and texts based on a transformer encoder, which is a cross-modal hashing network that completely abandons CNN. We embed fine-grained image-text features into a low-dimensional space by minimizing a hard-negative-based triplet loss. The advantages of this method are shown by comparing with hashing and non-hashing methods on two large datasets. We have made early attempts in the field of fine-grained hash retrieval, but there are still some shortcomings that need to be improved, such as the retrieval accuracy of shorter hash codes. In future work, we will learn more advanced techniques, focusing on using shorter hash codes for more efficient and accurate retrieval.

**Funding Statement:** This work was partially supported by Science and Technology Project of Chongqing Education Commission of China (KJZD-K202200513), National Natural Science Foundation of China (61370205), Chongqing Normal University Fund (22XLB003), and Chongqing Education Science Planning Project (2021-GX-320).

**Conflicts of Interest:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## References

- [1] F. Faghri, D. J. Fleet, J. R. Kiros and S. Fidler, “Vse++: Improving visual-semantic embeddings with hard negatives,” arXiv preprint, arXiv: 170705612, 2017.
- [2] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean *et al.*, “Devise: A deep visual-semantic embedding model,” in *Proc. NIPS*, Nevada, USA, pp. 2121–2129, 2013.
- [3] R. Kiros, R. Salakhutdinov and R. S. Zemel, “Unifying visual-semantic embeddings with multimodal neural language models,” arXiv preprint, arXiv: 14112539, 2014.
- [4] R. Girshick, “Fast r-cnn,” in *Proc. ICCV*, Santiago, Chile, pp. 1440–1448, 2015.
- [5] A. Krizhevsky, I. Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [6] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *Proc. CVPR*, Columbus, OH, USA, pp. 1701–1708, 2014.
- [7] Q. Y. Jiang and W. J. Li, “Deep cross-modal hashing,” in *Proc. CVPR*, Honolulu, HI, USA, pp. 3232–3240, 2017.
- [8] Y. Wang, M. Huang, X. Zhu and L. Zhao, “Attention-based lstm for aspect-level sentiment classification,” in *Proc. EMNLP*, Austin, TX, USA, pp. 606–615, 2016.
- [9] M. T. Luong, H. Pham and C. D. Manning, “Effective approaches to attention-based neural machine translation,” arXiv preprint, arXiv: 150804025, 2015.
- [10] A. Karpathy and F. F. Li, “Deep visual-semantic alignments for generating image descriptions,” in *Proc. CVPR*, Boston, MA, USA, pp. 3128–3137, 2015.
- [11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones *et al.*, “Attention is all you need,” *Advances in Neural Information Processing Systems*, vol. 2, pp. 5999–6009, 2017.
- [12] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint, arXiv: 181004805, 2018.

- [13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai *et al.*, “An image is worth  $16 \times 16$  words: Transformers for image recognition at scale,” arXiv preprint, arXiv: 2010.11929, 2020.
- [14] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint, arXiv: 1409.1556, 2014.
- [15] Y. Chen, S. Zhang, F. Liu, Z. Chang, M. Ye *et al.*, “Transhash: Transformer-based hamming hashing for efficient image retrieval,” in *Proc. ICMR, Association for Computing Machinery*, New York, NY, USA, pp. 127–136, 2022.
- [16] S. R. Dubey, S. K. Singh and W. T. Chu, “Vision transformer hashing for image retrieval,” arXiv preprint, arXiv: 2109.12564, 2021.
- [17] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona *et al.*, “Microsoft coco: Common objects in context,” in *Proc. ECCV*, Zurich, Switzerland, pp. 740–755, 2014.
- [18] G. Ding, Y. Guo and J. Zhou, “Collective matrix factorization hashing for multimodal data,” in *Proc. CVPR*, Columbus, OH, USA, pp. 2083–2090, 2014.
- [19] Z. Lin, G. Ding, M. Hu and J. Wang, “Semantics-preserving hashing for cross-view retrieval,” in *Proc. CVPR*, Boston, MA, USA, pp. 3864–3872, 2015.
- [20] H. Liu, R. Wang, S. Shan and X. Chen, “Deep supervised hashing for fast image retrieval,” in *Proc. CVPR*, Las Vegas, NV, USA, pp. 2064–2072, 2016.
- [21] C. Li, C. Deng, N. Li, W. Liu, X. Gao *et al.*, “Self-supervised adversarial hashing networks for cross-modal retrieval,” in *Proc. CVPR*, Salt Lake City, UT, USA, pp. 4242–4251, 2018.
- [22] X. Wang, X. Zou, E. M. Bakker and S. Wu, “Self-constraining and attention-based hashing network for bit-scalable cross-modal retrieval,” *Neurocomputing*, vol. 400, pp. 255–271, 2020.
- [23] S. Su, Z. Zhong and C. Zhang, “Deep joint-semantics reconstructing hashing for large-scale unsupervised cross-modal retrieval,” in *Proc. ICCV*, Seoul, Korea (South), pp. 3027–3035, 2019.
- [24] J. Yu, H. Zhou, Y. Zhan and D. Tao, “Deep graph-neighbor coherence preserving network for unsupervised cross-modal hashing,” in *Proc. AAAI Conf. on Artificial Intelligence*, Vancouver, Canada (Online), vol. 35, pp. 4626–4634, 2021.
- [25] T. Li, Z. Zhang, L. Pei and Y. Gan, “Hashformer: Vision transformer based deep hashing for image retrieval,” *IEEE Signal Processing Letters*, vol. 29, pp. 827–831, 2022.
- [26] Y. Wu, S. Wang, G. Song and Q. Huang, “Learning fragment self-attention embeddings for image-text matching,” in *Proc. of the 27th ACM Int. Conf. on Multimedia*, Nice, France, pp. 2088–2096, 2019.
- [27] N. Messina, F. Falchi, A. Esuli and G. Amato, “Transformer reasoning network for image-text matching and retrieval,” in *Proc. ICPR*, Taichung, Taiwan, China, pp. 5222–5229, 2021.
- [28] J. Wang, F. Zhou, S. Wen, X. Liu and Y. Lin, “Deep metric learning with angular loss,” in *Proc. CVPR*, Honolulu, HI, USA, pp. 2593–2601, 2017.
- [29] K. Sohn, “Improved deep metric learning with multi-class n-pair loss objective,” in *Proc. NIPS*, Barcelona, ESP, pp. 1857–1865, 2016.
- [30] P. Young, A. Lai, M. Hodosh and J. Hockenmaier, “From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions,” *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 67–78, 2014.
- [31] Y. Li, Y. Mu, N. Zhuang and X. Liu, “Efficient fine-grained visual-text search using adversarially-learned hash codes,” in *Proc. ICME*, Shenzhen, Guangdong, China, pp. 1–6, 2021.
- [32] Y. Cao, B. Liu, M. Long and J. Wang, “Cross-modal hamming hashing,” in *Proc. ECCV*, Munich, Germany, pp. 202–218, 2018.
- [33] N. Messina, G. Amato, A. Esuli, F. Falchi, C. Gennaro *et al.*, “Fine-grained visual textual alignment for cross-modal retrieval using transformer encoders,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 4, pp. 1–23, 2021.
- [34] Z., Ji, H., Wang, J., Han and Y., Pang, “Sman: Stacked multimodal attention network for cross-modal image-text retrieval,” *IEEE Transactions on Cybernetics*, vol. 52, no. 2, pp. 1086–1097, 2022.
- [35] Z. Ji, Z. Lin, H. Wang and Y. He, “Multi-modal memory enhancement attention network for image-text matching,” *IEEE Access*, vol. 8, pp. 38438–38447, 2020.