Deep Hashing with Large Batch Training for Cross-modal Retrieval

Xuewang Zhang[1,2] and Yin Zhou[1*]
[1]School of Software Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China
[2]School of Microelectronics and Communication Engineering, Chongqing University, Chongqing, China
zhangxw@cqupt.edu.cn , 646031898@qq.com

***Abstract*—Cross-modal hashing has attracted considerable attention as it can implement rapid cross-modal retrieval through mapping data of different modalities into a common Hamming space. With the development of deep learning, more and more cross-modal hashing methods based on deep learning are proposed. However, most of these methods use a small batch to train a model. Large batch training can get better gradients and can improve training efficiency. In this paper, we propose a deep hashing with large batch training (DHLBT), which uses large batch training and introduces orthogonal regularization to improve the generalization ability of our model. Moreover, we consider the discreteness of hash codes, therefore, we add the distance between hash codes and features to the objective function. Extensive experiments on three benchmarks show that our method achieves better performance than several existing hashing methods.**

Keywords: cross-modal hashing; large batch training; orthogonal regularization; the distance between hash codes and features

I. Introduction

With the rapid growth of multimedia data with different modalities and the increasing demands of users, cross-modal retrieval is becoming increasingly attractive. Modeling the relationship between different modalities is the key of cross-modal retrieval. The key challenge is a "heterogeneous gap" between different modalities, where the similarity among them cannot be measured directly [1]. However, cross-modal hashing methods can effectively bridge the gap [2, 3]. The hashing methods convert the high-dimensional features of data into a fixed-length hash code. Semantically similar data has similar hash codes. By XOR bitwise operation of hash codes, the similarity of data can be quickly obtained. Moreover, the storage space can be effectively reduced by only storing the hash codes of the data, instead of storing the high-dimensional features.

In recent years, deep learning has received good results in image processing and natural language processing. Therefore, more and more scholars have begun to apply deep learning technology to cross-modal hashing methods [2-7]. However, most of these methods use a small batch size to train the model. For example, in [2, 3, 5], the batch size is 64, and the maximum batch size is 128 [6]. However, when training a model in small batch size, the loss function cannot get a good gradient because of the limited number of samples in each batch, which makes the parameter update not good enough and affects the retrieval performance of the final trained model.

Large batch training which means using large batch size to train, e.g. 2048, 4096, or 8192, which is much larger than 64 or 128, can cover more samples each time when update parameters, resulting in better gradients and shorter training time per epoch. Therefore, more and more scholars in different fields are studying large batch training to get better performance [8-12], while no scholar has explored large batch training in the field of cross-modal hashing. So, it makes sense to study large batch training in the field of cross-modal hashing. However, increasing the batch size will cause the training extremely unstable [9], and then will easily lead to a "generalization gap" problem [13]. Orthogonal regularization will keep the norm of a matrix unchanged and lead the gradients to faithful propagation which will prevent the gradient from vanishing [10, 14]. In the field of image generation, Brock et al. [10, 14] introduced orthogonal regularization, which proves that orthogonal regularization achieves better performance. In multimodal retrieval, Wang et al. [15] also introduced orthogonal regularization, which reduces the redundancy of hash codes and improves performance. Moreover, hash codes are discrete. Relaxing the discrete learning problem of hash codes into continuous learning problem is the common practice of most cross-modal hashing methods. However, when continuous real value features of data are converted into hash codes, information loss will occur, which affects the performance so that the hash codes cannot represent the data well [5]. [5] adds the hash code to the objective function and learns the discrete hash code without relaxing. Inspired by these, we propose a method called deep hashing with large batch training (DHLBT). This method includes three major features, which are 1) Large batch size is used to train the model; 2) Orthogonal regularization is used to improve the generalization ability of the model; 3) Distances between hash codes and features are added to the objective function.

The rest of this paper is presented as follows. Section 2 introduces the proposed DHLBT approach. Section 3 shows the experiments. Finally, conclusions are made in Section 4.

DOI reference number: 10.18293/SEKE2020-148

II. DEEP HASHING WITH LARGE BATCH TRAINING

In this section, we will describe the details of our proposed method.

A. *Notations*

In this paper, we only consider image and text modal data. Therefore, there are two kinds of retrieval tasks in this paper: 1) text query image task and 2) image query text task. Assume that we have k training data, image modality is denoted as $I=\{I_i\}_{i=1}^k$, text modality is denoted as $T=\{T_i\}_{i=1}^k$. Then, we use $F=\{F_{I_i},F_{T_i}\}_{i=1}^k$ to denote the low dimensional features of data, $q=\{q_{I_i},q_{T_i}\}_{i=1}^k$ to denote the query data, $H=\{H_{I_i},H_{T_i}\}_{i=1}^k$ to denote the hash codes of data, and $\|\bullet\|_{Frobenius}$ to denote the Frobenius norm of a matrix, respectively.

B. *Network structure*

Many cross-modal hashing methods based on deep learning, e.g. SCH-GAN [2], use convolutional neural networks (CNN) to extract the features of images as input values for training. In this paper, we use VGG-19 [16] to extract the features of the images and encode them as hash codes through two fully-connected layers. While for texts, the texts are represented by the bag-of-words (BoW) features and are also encoded into hash codes through two fully-connected layers. The whole DHLBT model is shown in Fig. 1.

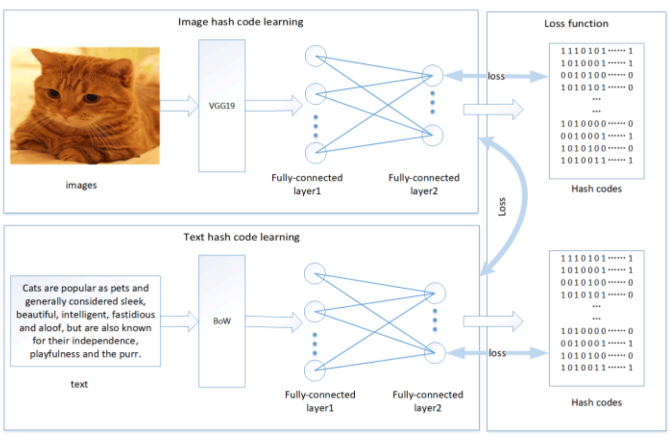

Figure 1. The framework of our DHLBT model.

C. *Feature Learning Part*

We firstly map the extracted image or text features to a common space through the fully-connected layer1 in the Fig. 1, then obtain the low-dimensional features through the fully-connected layer2 in the Fig. 1. The activation functions for the fully-connected layer1 and the fully-connected layer2 are tanh function and sigmoid function, respectively. The process can be represented as:

$$F = sigmoid(W_{c_2}(\tanh(W_{c_1} f + B_{c_1}) + B_{c_2})) \tag{1}$$

where W are the weights, B is the bias, c_1 denotes the fully-connected layer1, c_2 denotes the fully-connected layer2. f denote the input value of VGG-19 [16] features of images or BoW features of texts. The low-dimensional features of images F_I and the low-dimensional features of texts F_T have the same shape, which allows us to measure the similarity between them. The hash code length is also the same as the dimension of the low-dimensional features so that the low-dimensional features F can be directly mapped to the hash codes H by the threshold function:

$$H = \begin{cases} 1, & if\ F \geq 0.5 \\ 0, & if\ F < 0.5 \end{cases} \tag{2}$$

D. *Hashing Objectives*

Our objective function is mainly divided into three parts, which are: 1) the distance between the features of the images F_I and the features of the texts F_T, 2) the distance between the features F and the hash code H, and 3) the regularization items of W and B. The image query text task and the text query image task are symmetric. Therefore, we take the text query image task as an example to show the objective function in the following parts.

The distance between F_I *and* F_T*:*

$$D_{q_{Ti}I^+{}_i} = \left\| F_{q_{Ti}} - F_{I_i^+} \right\|_2^2 \tag{3}$$

$$D_{q_{Ti}I_i^-} = \left\| F_{q_{Ti}} - F_{I_i^+} \right\|_2^2 \tag{4}$$

where D denotes distance, I_i^+ denotes semantically similar image and I_i^- denotes semantically dissimilar image with text query q_{Ti}. $D_{q_{Ti}I_i^+}$ are the distance between I_i^+ and q_{Ti}. $D_{q_{Ti}I_i^-}$ are the distance between I_i^- and q_{Ti}. We use a margin-based hinge loss function to measure the loss, which is shown below:

$$L_1 = \frac{1}{n}\sum_i^n \max(0, \beta + D_{q_{Ti}I_i^+} - D_{q_{Ti}I_i^-}) \tag{5}$$

where β is a margin parameter between $D_{q_{Ti}I_i^+}$ and $D_{q_{Ti}I_i^-}$, and β is an adjustable hyper-parameter. n is the number of triplet (q_{Ti}, I_i^+, I_i^-). While reducing the loss L_1, $D_{q_{Ti}I_i^+}$ will be reduced and $D_{q_{Ti}I_i^-}$ will be increased simultaneously. This also conforms to the principle that small distance between semantically similar data and the large distance between semantically dissimilar data. In the process of training optimization, we intend to decrease the value of $D_{q_{Ti}I_i^+}$ and increase the value of $D_{q_{Ti}I_i^-}$ simultaneously. Therefore, the optimization process can be transformed into a binary classification problem, and then, we apply sigmoid cross-entropy as the loss function on it. The sigmoid cross-entropy formula for binary classification problem is shown below:

$$loss = -[z\ln(sigmoid(x) + (1-z)\ln(1-sigmoid(x)))] \quad s.t. \quad z \in \{0,1\} \tag{6}$$

where x represent a input value, and it can be assigned by either $D_{q_{Ti}I_i^+}$ or $D_{q_{Ti}I_i^-}$. z denotes a target value. For $D_{q_{Ti}I_i^+}$, we want $D_{q_{Ti}I_i^+}$ as small as possible, that is, let $z=0$, bring it into (6), as shown in equation (7):

$$loss_1 = -\ln(1-sigmoid(D_{q_{Ti}I_i^+})) = \ln(1+e^{D_{q_{Ti}I_i^+}}) \tag{7}$$

For $D_{q_{Ti}I_i^-}$, we want $D_{q_{Ti}I_i^-}$ as large as possible, that is, let $z=1$, bring it into (6), as shown in equation (8):

$$loss_2 = -\ln(sigmoid(D_{q_{Ti}I_i^-})) = \ln(1+e^{-D_{q_{Ti}I_i^-}}) \tag{8}$$

By combining equation (7) and (8), we have our second loss item:

$$L_2 = \frac{1}{n}\sum_i^n (loss_1 + loss_2) \tag{9}$$

The distance between F and H :

Hash codes are discrete, and information loss will occur in the process while converting real value features F to hash codes H :

$$D_{H_{q_{Ti}}F_{q_{Ti}}} = \left\| H_{q_{Ti}} - F_{q_{Ti}} \right\| \tag{10}$$

$$D_{H_{I_i}F_{I_i}} = \left\| H_{I_i^+} - F_{I_i^+} \right\| + \left\| H_{I_i^-} - F_{I_i^-} \right\| \tag{11}$$

where $D_{H_{q_{Ti}}F_{q_{Ti}}}$ denotes the distance between the low-dimensional features $F_{q_{Ti}}$ of text query q_{Ti} and hash codes $H_{q_{Ti}}$ of text query q_{Ti}. $D_{H_{I_i}F_{I_i}}$ denotes the distance between the low-dimensional features F_I of images I and hash codes H_I of images I. The following loss function can be obtained:

$$L_3 = \frac{1}{n}\sum_i^n (D_{H_{q_{Ti}}F_{q_{Ti}}} + D_{H_{I_i}F_{I_i}}) \tag{12}$$

In the optimization process, the loss function will make hash codes more and more close to the features and will reduce the information loss caused by the conversion process from the features to hash codes.

The regularization items of W and B :

Large batch training has low stability while training. To minimize the negative effect of the problem, we introduce the orthogonal regularization as the penalty term of W. For B, we still use L2 regularization as a penalty term. The loss item is as follows:

$$L_4 = \theta \left\| W^{transpose}W - I_{identity} \right\|_{Frobenius}^2 + \omega \left\| B \right\|_{Frobenius}^2 \tag{13}$$

where $W^{transpose}$ is the transpose of the weight matrix W and $I_{identity}$ is an identity matrix. B denotes the bias. θ and ω denotes the hyper-parameters.

By combining L_1, L_2, L_3 and L_4 together, we can get the full objective:

$$\min L = L_1 + \lambda L_2 + \gamma L_3 + L_4 \tag{14}$$

where λ and γ denote adjustable hyper-parameters.

We also take the text query image task as an example to show the training process of our method in Algorithm 1.

Algorithm 1 Training Process of DHLBT

Input: training data I, T
Output: weights W and bias B

```
initialize: Randomly initialize W and B, the batch size is b and the number of training epochs is e;
for epoch = 0, 1, 2, ..., e−1 do
    if epoch%30 == 0 then
        for q_T = T_1, T_2, T_3, ..., T_k do
            Randomly sample m points from I+ and m points from I− to make up a triplet set (q_T, I+, I−) as training data.
        end for
    end if
    for step = 1, 2, ..., ⌈k∗m/b⌉ do
        Train network and update parameters W and B by equation (14);
    end for
end for
```

III. EXPERIMENTS

In this section, we evaluate the performance of DHLBT on two datasets, and compare the result with several current state-of-the-art methods.

A. DATASETS

We use 2 datasets for experiments: Wikipedia [17] and MIRFlickr [18], which are widely used public datasets in cross-modal hashing. And to evaluate this method more fully, we added a larger data set NUS-WIDE [19] for experiments.

Wikipedia dataset [17] is a popular dataset which consists of 2866 text/image pairs divided into 10 categories. Following [2], Wikipedia dataset is separated into two parts: 1) a training data of 2173 pairs which are also used as the retrieval database and 2) a query set of 693 pairs. Each image is represented by 4096 deep features extracted by the fc2 layer of 19-layer VGGNet [16] from Keras applications, and each text is represented as a 1000-dimensional BoW vector.

MIRFlickr dataset [18] contains 25000 images that are collected from the Flickr website and they are annotated with some of 24 provided labels. Each image is described with some textual tags. Therefore, each instance is a text-image pair. Following [2, 20], firstly, we preprocess raw tags of these images by removing punctuations and stop words. Then, we count the number of times for each word appeared in these tags. We only keep words that appeared at least 20 times and add them to the vocabulary of BoW. Furthermore, we remove instances that do not contain the word of the vocabulary and that do not have textual tags or labels. We take 5% of instances in each category as the query set and the rest of the instances as the retrieval database. In addition, we sample 5000 data pairs from the retrieval database as the training data. Each image is represented by 4096 deep features extracted by the fc2 layer of 19-layer VGGNet from Keras applications, and each text is represented as a 1386-dimensional BoW vector.

NUS-WIDE dataset [19] contains 269648 images that are collected from the Flickr website and they are annotated with some of 81 provided labels. Each image is described with some textual tags. Therefore, each instance is a text-image pair. We select the 10 most common labels and the corresponding 186577 text-image pairs. We take 2000 of pairs in each category as the query set and the rest of the pairs as the retrieval database. In addition, we sample 5000 data pairs from the retrieval database as the training data. Each image is represented by 4096 deep features extracted by the fc2 layer of 19-layer VGGNet from Keras applications, and each text is represented as a 1386-dimensional BoW vector. Table 1 shows the number of samples in each set intuitively.

TABLE I. STATISTICS OF TWO BENCHMARK DATASETS

	Wikipedia	MIRFlickr	NUS-WIDE
Dataset Size	2866	20819	186577
Training Set	2173	5000	5000
Query Set	693	1041	2000
Retrieval Set	2173	19778	186577
labels	10	24	10

B. EVALUATION PROTOCOL

We perform two kinds of retrieval tasks for each dataset: 1) retrieving text by image query, termed image→text; and 2) retrieving image by text query, termed text → image. Following [2], we utilize Hamming ranking to evaluate DHLBT and compared the result with the other state-of-the-art methods. Specifically, we first obtain the hash codes of images and texts, and then compute the Hamming distance between query with all the retrieval database. After ranking the Hamming distance list, we use 2 widely used assessment standards to evaluate the retrieval performance, which are shown below:

1) Mean Average Precision (MAP): The mean of all queries' average precisions (AP) called MAP. $AP = \frac{1}{R}\sum_{k=1}^{n}\frac{R_k}{k}\times rel_k$ is the definition of AP where R is the amount of the related data in the retrieval database, n is the amount of retrieval database, R_k is the amount of the related data in the top k ranks of the Hamming distance ranking list, and rel_k is an indicator of relevance of the Hamming distance ranking list which is set to 1 if the data at k -th position is related and 0 elsewise.

2)Precision Recall curve (PR-curve): The precision at the certain recall of the Hamming distance ranking list, that often evaluates the performance of retrieval.

C. BASELINES AND IMPLEMENT DETAILS

We compare two non-deep learning methods: SePH [20] and GSPH [21], which are both supervised methods. For SePH and GSPH, they are kernel-based methods and both of them achieved best results by using KLR which respectively created in two ways: 1) k-means algorithm and 2) random sampling. So, for these two hashing methods, we use KLR to learn hash function and create kernel by using k-means algorithm (klr+k) and random sampling (klr+r). In addition, we also compare our methods to three state-of-the-art deep learning-based methods, including SCH-GAN [2], UGACH [3] and DCMH [5]. SCH-GAN is a semi-supervised method. UGACH is an unsupervised method and DCMH is a supervised method. In all experiments, two modal data of image and text are used. When the data of one modal is used as the query set, the data of the other modal is used as the retrieval set. Source codes of all methods are kindly provided by the corresponding authors. For the parameters mentioned in all methods, we directly adopt the original parameter settings used in their codes. For an objective comparison between different methods, we use the same image and text features as input data features for all compared methods. Specifically, for Wikipedia and NUS-WIDE datasets, we use the 4096 deep features extracted by the fc2 layer of 19-layer VGGNet from Keras applications for images, and 1000-dimensional bag-of-words features for texts; For MIRFlickr dataset, we use the 4096 deep features extracted by the fc2 layer of 19-layer VGGNet from Keras applications for images, and 1386-dimensional bag-of-words features for texts. For DCMH, which is an end-to-end method, we add an experiment which directly uses original image features as the input value of the image network. DCMH_{vgg19} and $\mathrm{DCMH}_{original}$ denote these two versions of DCMH, respectively. For our method, we set $\lambda = 0.01$, $\gamma = 0.01$, $\theta = 0.0001$ and $\omega = 0.01$. Similar to [2, 3], for each data, the corresponding data is selected to form 4 triplets for training, that is, m in Algorithm 1 is set to 4. So, there are $2173 * 4 = 8692$ triplets for Wikipedia dataset, $5000 * 4 = 20000$ triplets for MIRFlickr and NUS-WIDE datasets. The batch size is set to 8192. And the learning rate for Wikipedia dataset is 0.08, the learning rate for MIRFlickr dataset is 0.016 and the learning rate for NUS-WIDE dataset is 0.016. The hash code bits are 16, 32, and 64, and β is 6, 8, and 10, respectively. We implement the proposed DHLBT by Tensorflow applications. All the experiments conducted on a server with hardware of NVIDIA GTX 1080Ti graphic card, Intel(R) Xeon(R) E5-2620 v4 2.10GHz CPU, 128 GB memory. The model was built by Python3.5.2 and Tensorflow 1.11.0.

D. EXPERIMENTAL RESULTS

We demonstrate the MAP scores of all methods on MIRFlickr, Wikipedia and NUS-WIDE datasets in Table 2, Table 3 and Table 4. From the result, it can be observed that our method achieves the best retrieval accuracy at 32-bit and 64-bit hash code length over all datasets. In general, compared with the second-best method SCH-GAN, in the task of image query text, our method is about 1.8%, 8.2% and 1.1% higher on MIRFlickr, Wikipedia and NUS-WIDE datasets, respectively. And in the task of text query image, our method is about 1.3%, 2% and 0.2% higher on MIRFlickr, Wikipedia and NUS-WIDE datasets, respectively. This is mainly because we use large batch size to train the model which can get better gradients and use orthogonal regularization to improve the generalization ability of our model. And it is also because the distance between the hash codes and the features of data is added to the loss function which makes the hash codes are more realistic to represent the features of data. From the results, we can see that the hash code length has a remarkable impact on the MAP scores. For 16-bit hash code, the length is not enough to get sufficient information. Although our method has achieved the best results on Wikipedia dataset, it is only the second-best on MIRFlickr and NUS-WIDE datasets, indicating that the hash code length has a certain impact on MAP scores.

TABLE II. THE MAP SCORES ON MIRFLICKR DATASET

Methods	image→text			text→image		
	16	32	64	16	32	64
$SePH_{klr+r}$ [20]	0.7364	0.7367	0.7451	0.7486	0.7514	0.7573
$SePH_{klr+k}$ [20]	0.7377	0.7459	0.7467	0.7522	0.7595	0.7599
$GSPH_{klr+r}$ [21]	0.7279	0.7425	0.7541	0.7579	0.7693	0.7760
$GSPH_{klr+k}$ [21]	0.7374	0.7485	0.7584	0.7614	0.7729	0.7798
UGACH [3]	0.6100	0.6045	0.5848	0.6278	0.6029	0.6101
$DCMH_{original}$ [5]	0.7296	0.7363	0.7386	0.7639	0.7650	0.7703
$DCMH_{vgg19}$ [5]	**0.7433**	0.7527	0.7592	0.7669	0.7792	0.7837
SCH-GAN [2]	0.7203	0.7481	0.7609	0.7661	0.7851	0.7884
Ours	0.7410	**0.7571**	**0.7718**	**0.7822**	**0.7915**	**0.7953**

TABLE III. THE MAP SCORES ON WIKIPEDIA DATASET

Methods	image→text			text→image		
	16	32	64	16	32	64
$SePH_{klr+r}$ [20]	0.5009	0.5287	0.5393	0.5508	0.5955	0.6190
$SePH_{klr+k}$ [20]	0.4997	0.5252	0.5413	0.5584	0.6009	0.6122
$GSPH_{klr+r}$ [21]	0.5064	0.5289	0.5320	0.5701	0.6001	0.6237
$GSPH_{klr+k}$ [21]	0.5117	0.5318	0.5390	0.5801	0.6036	0.6207
UGACH [3]	0.3332	0.3605	0.3688	0.3222	0.3323	0.3471
$DCMH_{original}$ [5]	0.4503	0.4506	0.4120	0.7419	0.7238	0.6940
$DCMH_{vgg19}$ [5]	0.4387	0.4698	0.4809	0.8279	0.8457	0.7927
SCH-GAN [2]	0.5207	0.5370	0.5076	0.8352	0.8351	0.8288
Ours	**0.5528**	**0.5712**	**0.5688**	**0.8426**	**0.8502**	**0.8572**

TABLE IV. THE MAP SCORES ON NUS-WIDE DATASET

Methods	image→text			text→image		
	16	32	64	16	32	64
$SePH_{klr+r}$ [20]	0.6537	0.676	0.6792	0.6769	0.6857	0.6836
$SePH_{klr+k}$ [20]	0.6625	0.685	0.6848	0.6681	0.6875	0.6886
$GSPH_{klr+r}$ [21]	0.6703	0.6858	0.6961	0.676	0.6838	0.7028
$GSPH_{klr+k}$ [21]	**0.6746**	0.696	0.7049	0.6756	0.6986	0.7052
UGACH [3]	0.6231	0.6296	0.6293	0.6152	0.6116	0.6152
$DCMH_{original}$ [5]	0.6008	0.6419	0.6383	0.6439	0.6739	0.6709
$DCMH_{vgg19}$ [5]	0.6341	0.6552	0.6631	0.6803	0.699	0.7058
SCH-GAN [2]	0.6647	0.6909	0.7027	**0.6862**	0.7086	0.7128
Ours	0.6625	**0.7007**	**0.7179**	0.6848	**0.7091**	**0.7189**

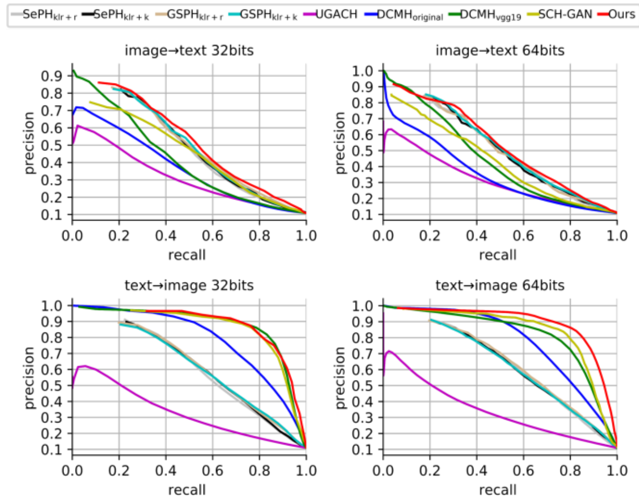

Figure 2. The PR-curves on Wikipedia dataset.

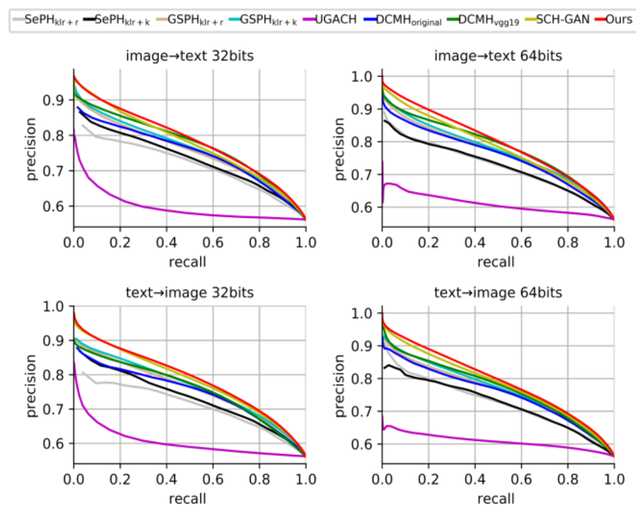

Figure 3. The PR-curves on MIRFlickr dataset.

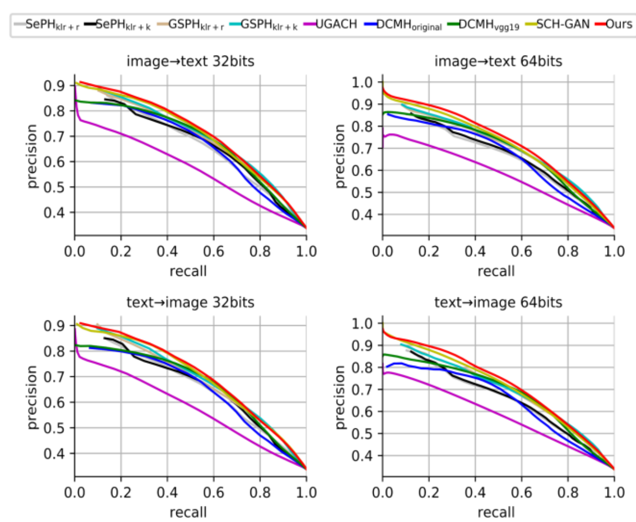

Figure 4. The PR-curves on NUS-WIDE dataset.

TABLE V. THE DIFFERENT BATCH SIZE TRAINING OF DHLBT ON WIKIPEDIA DATASET

Batch Size	Learning rate	Ortho.	MAP (image → text)			MAP (text → image)			The sum of MAP (image→text and text → image)			Time of each epoch		
			16	32	64	16	32	64	16	32	64	16	32	64
512	0.02	N	0.5406	0.5506	0.5612	0.8280	0.8524	0.8432	1.3686	1.4030	1.4044	9.3s	9.6s	10.2s
512	0.02	Y	**0.5588**	0.5629	0.5691	0.8312	**0.8576**	0.8479	1.3900	1.4205	1.4170	16.0s	16.3s	16.8s
2048	0.04	N	0.5509	0.5594	0.5699	0.8231	0.8462	0.8326	1.3740	1.4056	1.4025	8.1s	8.2s	8.5s
2048	0.04	Y	0.5559	0.5668	**0.5702**	0.8380	0.8486	0.8545	1.3939	1.4154	1.4247	10.9s	11.0s	11.3s
8192	0.08	N	0.5487	0.5563	0.5649	0.8217	0.8431	0.8327	1.3704	1.3994	1.3976	7.5s	7.5s	7.9s
8192	0.08	Y	0.5528	**0.5712**	0.5688	**0.8426**	0.8502	**0.8572**	**1.3954**	**1.4214**	**1.4260**	8.2s	8.3s	8.6s

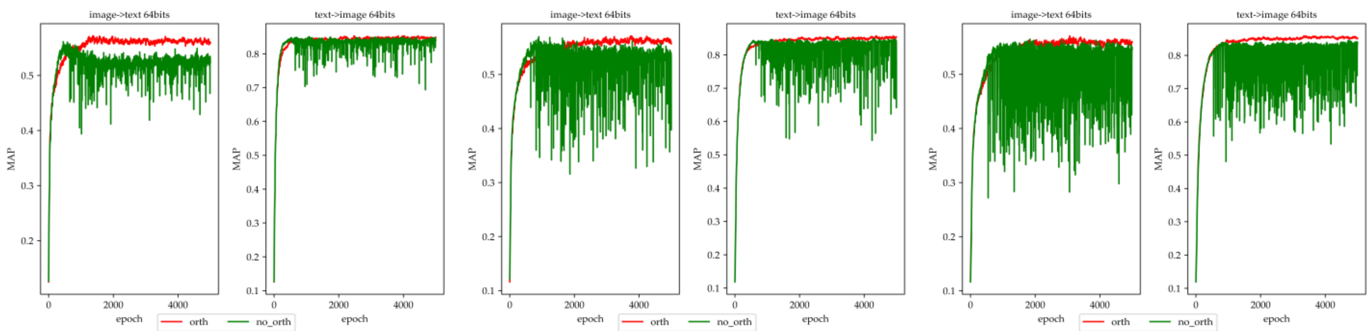

Figure 5. The changes of MAP during the training process with the batch size of 512, 2048 and 8192, respectively.

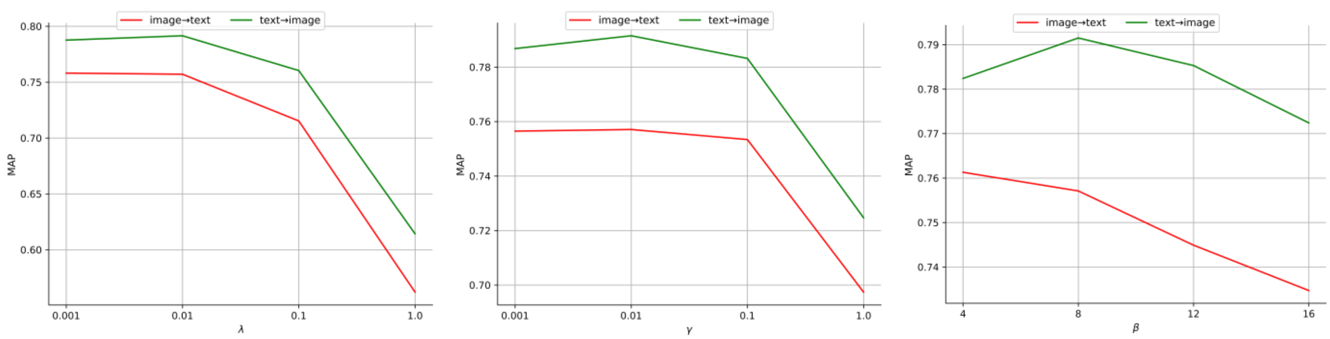

Figure 6. The changes of MAP in different value of the hyperparameter λ , γ and β , respectively.

The PR-curves at 32- and 64-bit hash code length on Wikipedia, MIRFlickr and NUS-WIDE datasets are shown in Fig. 2, Fig. 3 and Fig. 4, respectively. The result shows that our method performs better than other state-of-the-art methods.

In addition, we also compare the effects of different batch sizes and orthogonal regularization on our model training on the Wikipedia dataset. The batch size is set to 512, 2048, and 8192, respectively. When increasing the batch size, it is necessary to increase the learning rate to ensure the convergence speed, so the learning rates are 0.02, 0.04, and 0.08, respectively. N means that orthogonal regularization is not used, and Y means orthogonal regularization is used. When orthogonal regularization is not used, we replace it with L2 regularization, that is, equation (13) is replaced by:

$$L_4 = \omega(\|w\|_{Frobenius}^2 + \|B\|_{Frobenius}^2) \tag{15}$$

We keep configuration for all parameter except the learning rate. The result of the comparison is shown in Table 5. When evaluating the performance, we need to consider not only the map of image→text, but also the map of text→image. Therefore, the map sum of text→image and image→text is given in Table 5. It can be seen that using orthogonal regularization can obviously achieve better performance when using same batch size, but the training time of each epoch will be increased. Extend the batch size will significantly increase training speed. Simply increasing the batch size without using orthogonal regularization does not achieve good performance. When the batch size is 8192 and the orthogonal regularization is used, we achieve the best performance.

We also analyze the training stability of our model. The changes of MAP at 64-bit hash code length during the training process with the batch size of 512, 2048 and 8192 on Wikipedia dataset are shown in Fig. 5. “orth” means that orthogonal regularization is used, and “no_orth” means orthogonal regularization is not used. The result shows that the changes of MAP are more volatile with the increase of batch size. And the changes of MAP become stable and the model can get better performance when used the orthogonal regularization.

At last, we conduct sensitivity experiments for hyperparameter λ, γ and β. λ is set to 0.001, 0.01, 0.1 and 1.0, respectively. γ is set to 0.001, 0.01, 0.1 and 1.0, respectively. β is set to 4, 8, 12 and 16, respectively. When performing sensitivity experiments on one hyperparameter, other hyperparameters remain fixed. That is, we set λ=0.01 when performing sensitivity experiments on γ or β. We set γ=0.01 when performing sensitivity experiments on λ or β. We set β=8 when performing sensitivity experiments on λ or γ. Fig. 6 shows the changes of MAP at 32-bit hash code length in different value of the hyperparameter λ, γ and β on MIRFlickr dataset. The result shows that the changes of MAP are not volatile when the value of λ or γ is set to between 0.001 and 0.01. The changes of MAP are volatile in different value of the hyperparameter β. When the value of β is set to 8, we get the best MAP scores. For hyperparameter $\theta = 0.0001$ and $\omega = 0.01$, we follow [10] and [2], respectively, which are the best parameters selected by the corresponding authors through sufficient experiments. In order to reduce hyperparameters, we set λ and γ to the same value of ω, which is 0.01.

IV. CONCLUSIONS

In this paper, we propose a deep hashing with large batch training (DHLBT) for the cross-modal hashing retrieval. DHLBT is the cross-modal hashing which uses large batch training and uses orthogonal regularization to improve the generalization ability of our model. Moreover, the distance between hash codes and features is added to the objective function which makes hash codes to represent data more realistically. The effectiveness of DHLBT is demonstrated through the experiments on three widely-used datasets: Wikipedia, MIRFlickr and NUS-WIDE.

V. ACKNOWLEDGMENT

This work is supported by the Major Science and Technology Project for “Innovation of Common Technology in Key Industries” of Chongqing (cstc2017zdcy-zdzxX0013), Technology Innovation and Application Development Project of CSTC (cstc2020jscx-fyzx0212), and Basic and Advanced Research Project of CSTC (cstc2019jcyj-zdxmX0008).

REFERENCES

[1] Y. Peng, X. Huang, and Y. Zhao, “An overview of cross-media retrieval: Concepts, methodologies, benchmarks, and challenges,” IEEE Trans. Circuits and Syst. Video Technol., vol. 28, no. 9, pp. 2372-2385, Sep. 2018.

[2] J. Zhang, Y. Peng, and M. Yuan, “SCH-GAN: Semi-supervised cross-modal hashing by generative adversarial network,” IEEE Trans. Cybern., vol. 50, no. 2, pp. 489-502. 2020.

[3] J. Zhang, Y. Peng, and M. Yuan, “Unsupervised generative adversarial cross-modal hashing,” in Proc. 32nd AAAI Conf. Artif. Intell., New Orleans, Lousiana, USA, 2-7 February 2018, pp. 539-546.

[4] L. Wu, Y. Wang, and L. Shao, “Cycle-consistent deep generative hashing for cross-modal retrieval,” IEEE Trans. Image Process., vol. 28, no. 4, pp. 1602 - 1612, Apr. 2019.

[5] Q. Y. Jiang, and W. J. Li, “Deep cross-modal hashing,” in Proc. 30th IEEE Conf. Comput. Vis. Pattern Recognit., Honolulu, HI, USA, 21-26 July 2017, pp. 3232–3240.

[6] Z. Ji, W. Yao, W. Wei, H. Song, and H. Pi, “Deep multi-level semantic hashing for cross-modal retrieval,” IEEE Access, vol. 7, pp. 23667-23674. 2019.

[7] G. Wu, Z. Lin, J. Han, L. Liu, G. Ding, B. Zhang, and J. Shen, “Unsupervised deep hashing via binary latent factor models for large-scale cross-modal retrieval,” in Proc. 27th Int. Joint Conf. Artif. Intell., Stockholm, Sweden, 13-19 July 2018, pp. 2854-2860.

[8] P. Goyal, P. Dollar, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He (2017). “Accurate large minibatch SGD: Training imagenet in 1 hour.” [Online]. Available: http://arxiv.org/abs/1706.02677

[9] Y. You, I. Gitman, and B. Ginsburg (2017). “Large batch training of convolutional networks.” [Online]. Available: https://arxiv.org/abs/1708.03888

[10] A. Brock, J. Donahue, and K. Simonyan (2018). “Large scale GAN training for high fidelity natural image synthesis.” [Online]. Available: https://arxiv.org/abs/1809.11096

[11] X. Jia, S. Song, W. He, Y. Wang, H. Rong, F. Zhou, L. Xie, Z. Guo, Y. Yang, L. Yu, T. Chen, G. Hu, S. Shi, and X. Chu (2018). “Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes.” [Online]. Available: https://arxiv.org/abs/1807.11205

[12] Y. You, J. Hseu, C. Ying, J. Demmel, and C. J. Hsieh (2019). “Large-batch training for LSTM and beyond.” [Online]. Available: https://arxiv.org/abs/1901.08256

[13] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and T. P. T. Ping (2016). “On large-batch training for deep learning: Generalization gap and sharp minima.” [Online]. Available: https://arxiv.org/abs/1609.04836

[14] A. Brock, T. Lim, J. M. Ritchie, and N. Weston (2016). “Neural photo editing with introspective adversarial networks.” [Online]. Available: https://arxiv.org/abs/1609.07093

[15] D. Wang, C. Peng, M. Ou, and W. Zhu, “Deep multimodal hashing with orthogonal regularization,” in Proc. 24th AAAI Conf. Artif. Intell., Buenos Aires, Argentina, 26-27 July 2015, pp. 2291-2297.

[16] K. Simonyan, and A. Zisserman (2014). “Very deep convolutional networks for large-scale image recognition.” [Online]. Available: http://arxiv.org/abs/1409.1556

[17] J. C. Pereira, E. Coviello, G. Doyle, N. Rasiwasia, G. R. G. Lanckriet, R. Levy, and N. Vasconcelos, “On the role of correlation and abstraction in cross-modal multimedia retrieval,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 3, pp. 521-35, Mar. 2014.

[18] M. J. Huiskes, and M. S. Lew, “The MIR flickr retrieval evaluation,” in Proc. 1st ACM Int. Conf. Multimedia Inf. Retr., Vancouver, British Columbia, Canada, 30-31 October 2008, pp. 39-43.

[19] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. "NUS-WIDE: A real-world web image database from national university of singapore" in Proc. of the 2009 ACM Int. Conf. on Image and Video Retr., Santorini, Fira, Greece, 8-10 July 2009, pp. 48-56.

[20] Z. Lin, G. Ding, M. Hu, and J. Wang, “Semantics-Preserving Hashing for Cross-View Retrieval,” in Proc. 28th IEEE Conf. Comput. Vis. Pattern Recognit., Boston, MA, USA, 7-12 June 2015, pp. 3864-3872.

[21] D. Mandal, K. N. Chaudhury, and S. Biswas, “Generalized semantic preserving hashing for cross-modal retrieval,” IEEE Trans. Image Process., vol. 28, no. 1, pp. 102-112, Jan. 2019.