LABEL CONSISTENT MATRIX FACTORIZATION BASED HASHING FOR CROSS-MODAL RETRIEVAL

Devraj Mandal and Soma Biswas

Department of Electrical Engineering, Indian Institute of Science, India.

ABSTRACT

Matrix factorization-based hashing has been very effective in addressing the cross-modal retrieval task. In this work, we propose a novel supervised hashing approach utilizing the concepts of matrix factorization which can seamlessly incorporate the label information. In the proposed approach, the latent factors for each individual modality are generated and then converted to the more discriminative label space using modality specific linear transformations. In the first stage of the approach, the hash codes are learnt using an alternating minimization algorithm and in the next stage, modality specific hash functions are learned to convert the original features of the cross-modal data into the hash code domain. In addition, we also propose an extension of the approach for handling very large amounts of data during the training stage. Extensive experiments performed on the single label Wiki, and the multi-labeled MirFlickr and NUS-WIDE datasets show the effectiveness of the proposed approach.

***Index Terms*—** Hashing, Matrix Factorization, Cross-modal retrieval, image-text.

1. INTRODUCTION

Cross-modal retrieval problems involving image-text, image-audio, text-audio, etc., have been gaining increasing attention in the computer vision community due to the easy availability of multimedia data. Research is being carried out keeping in mind the retrieval accuracy and also the speed of retrieval. In this regard, hashing techniques [1]-[2] have gained increasing importance where we represent the data in binary bit strings which has the added advantage that fast comparisons can be performed by using simple XOR operations. In addition, as multi-labeled data is more discriminative in nature (Fig. 1), these techniques need to generalize well for both single label datasets like Wiki [3] and multi-label datasets such as MirFlickr [4] and NUS-WIDE [5]. The hashing techniques can be broadly classified into unsupervised [1] [6] [7] [8] [9] and supervised ones [2] [10]. The performance of the supervised methods is better than that of the unsupervised ones for they can leverage extra cues from the provided label information. In addition, it is observed that techniques that consider all possible relations between the data items from the different modalities [2] generally perform much better than those which consider only pairwise correspondences [7] [1].

Matrix Factorization based hashing [7] has been successfully applied for the task of cross-modal data retrieval in an unsupervised setting. In this work, we propose a novel supervised extension of matrix factorization-based hashing, which can utilize the additional label information for improved retrieval performance. The main idea behind cross-modal retrieval is to project all the data coming from different modalities into a common subspace where similarly labeled data are kept close together and differently labeled items are kept as far away as possible. Inspired by the success of the supervised methods [2] [10] which show the discriminative capability of the labels, we consider the label space itself as the common discriminative space. In this work, we convert the latent representation of each of the individual modalities into the latent factors of the more discriminative label space by using modality specific linear transformations. The proposed algorithm consists of two stages. In the first stage, we leverage the label information to construct the optimal hash codes while solving a non-convex objective. The solution to the objective function is obtained by alternatively solving a series of sub-problems, each of which is convex in nature and thus has a closed-form analytical solution. In the second stage, we use kernel logistic regression (KLR) to learn the hash functions [2].

Fig. 1. Examples of multi-labeled data from MirFlickr Dataset [4].

A supervised extension of matrix factorization-based hashing using the label information has also been proposed in [11] which showed significant performance gains over the original unsupervised version [7]. Compared to both the approaches [7] [11], we use a two-stage learning process to learn the hash codes and hash functions separately. The two-stage approaches [12] [13] showed that learning the hash codes and hash functions independently enables us to design a less complicated hash code learning stage. In addition, it also enables us to use more advanced machine learning techniques to suitably design the hash functions. Another significant difference with [11] is that we consider only pairwise correspondence which enables us to design an iterative version of our algorithm. This modification allows us to learn the hash codes (during the first stage) even for very large training datasets.

The contributions of this work are as follows : (1) We design a supervised hashing technique for the task of cross-modal retrieval based on matrix factorization. The supervised version (Ours_1) shows significant improvement over [7], while giving comparable performance to [11]. (2) We propose an iterative learning scheme by which hash codes can be efficiently computed for very large datasets. We observe that this enables us to use the whole training data (and not a small subset of it as in [2] [7]). Thus, we can use the learned hash codes from the first stage itself for the retrieval task (Ours_2) which shows impressive performance gains over Ours_1. (3) Com-

Correspondence: devraj89@ee.iisc.ernet.in.

parisons against state-of-the-art hashing techniques on the single label Wiki [3] dataset and multi-label NUS-WIDE [5] and MirFlickr [4] datasets show the efficacy of the approach.

2. PROPOSED WORK

Let the cross modal data be represented as $\mathbf{X}_t \in \mathbb{R}^{d_t \times N}$ ($t \in \{1, 2, .., M\}$), where M is the number of modalities, N is the number of training samples and d_t the dimensionality of the feature. Let the labels be denoted as $\mathbf{X}_L \in \mathbb{R}^{d_c \times N}$, where d_c is the number of classes. The label vector of a data has one's corresponding to the classes to which it belongs, and zeros in the remaining entries. For example, let the total number of classes be 6, then a data point which belongs to classes 1 and 3 will have a label vector $[1\ 0\ 1\ 0\ 0\ 0]^T$. Thus the data can have single labels, like Wiki [3] or multiple labels like MirFlickr [4] and NUS-WIDE [5]. We use data centering as the preprocessing step. The data from each modality can be factorized as

$$\mathbf{X}_t = \mathbf{U}_t \mathbf{V}_t, \quad \forall t \tag{1}$$

where, $\mathbf{U}_t \in \mathbb{R}^{d_t \times k}$, $\mathbf{V}_t \in \mathbb{R}^{k \times N}$ and k is the length of latent factors. Then the hash codes can be generated from the latent factors by using the sign(.) operation. The goal is to generate the latent factors i.e., $\mathbf{V}_t$ in such a way that they are linked to every other modality via a discriminative space which utilizes the label information. In this work, we consider the label space itself as the common discriminative space. To facilitate coupling between the latent factors of the different modalities via this space, the latent factors for the label data are computed in a similar manner as $\mathbf{X}_L = \mathbf{U}_L \mathbf{V}_L$. We assume that the latent factors for the different modalities are coupled to that of the label space via linear transformations, i.e.

$$\mathbf{W}_t \mathbf{V}_t = \mathbf{V}_L, \quad \forall t \tag{2}$$

where, $\mathbf{W}_t$ are the modality specific transformations. Thus, the overall objective function is given as

$$\begin{aligned} \min_{\mathbf{U}_t, \mathbf{V}_t, \mathbf{W}_t} \quad & \sum_{t=1}^{M,L} \lambda_t ||\mathbf{X}_t - \mathbf{U}_t \mathbf{V}_t||_F^2 + \sum_{t=1}^{M} \alpha_t ||\mathbf{V}_L - \mathbf{W}_t \mathbf{V}_t||_F^2 \\ & + \gamma \text{reg}\left(\mathbf{U}_t, \mathbf{U}_L, \mathbf{V}_t, \mathbf{V}_L, \mathbf{W}_t\right) \end{aligned} \tag{3}$$

where, reg is the regularization function defined as $\text{reg}(.) = ||.||_F^2$ with $||.||_F^2$ being the Frobenius norm and α_t, λ_t and γ are the combining coefficients. Now, we describe how to solve the objective function to obtain all the different unknowns.

2.1. Learning the Hash Code

The optimization in (3) is non-convex in the different variables $\mathbf{U}_t, \mathbf{V}_t$, and $\mathbf{W}_t$. Here, we solve the problem using an alternating minimization scheme by fixing all the variables while updating only one of them. Then, the optimization problem can be solved by iteratively solving the following sub-problems until convergence. Each sub-problem is convex and has a closed form solution as described next.

(1) Solve for $\mathbf{U}_t$ and $\mathbf{U}_L$: To solve for variables $\mathbf{U}_t$, for $t = \{1, ..., M, L\}$, all the other variables are kept fixed at the values in the previous iteration. Then the problem to be solved becomes,

$$\lambda_t ||\mathbf{X}_t - \mathbf{U}_t \mathbf{V}_t||_F^2 + \gamma ||\mathbf{U}_t||_F^2 \tag{4}$$

This has a closed form solution given by

$$\mathbf{U}_t = \mathbf{X}_t \mathbf{V}_t^T \left(\mathbf{V}_t \mathbf{V}_t^T + (\gamma/\lambda_t)\, \mathbf{I}_k\right)^{-1} \tag{5}$$

Algorithm 1 Proposed Algorithm

1: **Input** : $\mathbf{X}_t, \mathbf{X}_L$ & k.
2: **Output** : Learned hash codes in the common domain $\mathbf{B}_{t,c}$.
3: **Initialize** : Initialize $(\mathbf{U}_t, \mathbf{V}_t \ \forall t)$, $\mathbf{U}_L, \mathbf{V}_L$, as random matrices, $(\mathbf{W}_t \ \forall t)$ as $\mathbf{I}_k$.
4: Continue until convergence
5: Fix $\mathbf{W}_t, \mathbf{V}_t, \mathbf{V}_L$ and update $\mathbf{U}_t$ and $\mathbf{U}_L$ by using (5).
6: Fix $\mathbf{U}_t, \mathbf{V}_t, \mathbf{U}_L, \mathbf{V}_L$ and update $\mathbf{W}_t$ by using (7).
7: Fix $\mathbf{U}_t, \mathbf{W}_t, \mathbf{U}_L$ and update $\mathbf{V}_t$ and $\mathbf{V}_L$ by using (9) and (10).
8: Set $\mathbf{B}_{t,c} = \text{sign}(\mathbf{W}_t \mathbf{V}_t)$.

where $\mathbf{I}_k$ is the identity matrix of size k.

(2) Solve for $\mathbf{W}_t$: To solve for $\mathbf{W}_t, t = \{1, 2, .., M\}$, all the other variables are kept fixed, leading to the following sub-problem

$$\alpha_t ||\mathbf{V}_L - \mathbf{W}_t \mathbf{V}_t||_F^2 + \gamma ||\mathbf{W}_t||_F^2 \tag{6}$$

whose closed form solution is given by

$$\mathbf{W}_t = \mathbf{V}_L \mathbf{V}_t^T \left(\mathbf{V}_t \mathbf{V}_t^T + (\gamma/\alpha_t)\, \mathbf{I}_k\right)^{-1} \tag{7}$$

(3) Solve for $\mathbf{V}_t$ and $\mathbf{V}_L$: To solve for $\mathbf{V}_t, t = \{1, 2, ..., M\}$ and $\mathbf{V}_L$, all the other variables are fixed. Then the sub-problem to be solved becomes,

$$\lambda_t ||\mathbf{X}_t - \mathbf{U}_t \mathbf{V}_t||_F^2 + \alpha_t ||\mathbf{V}_L - \mathbf{W}_t \mathbf{V}_t||_F^2 + \gamma ||\mathbf{V}_t||_F^2 + \gamma ||\mathbf{V}_L||_F^2 \tag{8}$$

This too has a closed form solution given by

$$\begin{aligned} \mathbf{V}_t &= \mathbf{A}^{-1} \left(\lambda_t \mathbf{U}_t^T \mathbf{X}_t + \alpha_t \mathbf{W}_t^T \mathbf{V}_L\right), \quad \text{where} \\ \mathbf{A} &= \left(\lambda_t \mathbf{U}_t^T \mathbf{U}_t + \alpha_t \mathbf{W}_t^T \mathbf{W}_t + \gamma \mathbf{I}_k\right)^{-1} \\ \mathbf{V}_L &= \mathbf{B}^{-1} \left(\lambda_L \mathbf{U}_L^T \mathbf{X}_L + \left(\sum_{t=1}^{M} \alpha_t \mathbf{W}_t \mathbf{V}_t\right)\right) \quad \text{where} \\ \mathbf{B} &= \left(\lambda_L \mathbf{U}_L^T \mathbf{U}_L + \left(\sum_{t=1}^{M} \alpha_t + \gamma\right) \mathbf{I}_k\right) \end{aligned} \tag{9, 10}$$

The complete algorithm is summarized in Algorithm 1. On completion of the training stage, we can generate the hash codes in the original $\mathbf{X}_t$ domain as $\mathbf{B}_t = \text{sign}(\mathbf{V}_t)$. However, since the label space is more discriminative, we transform the hash codes using the learned transformations $\mathbf{W}_t$ and generate the hash code in the common discriminative space as $\mathbf{B}_{t,c} = \text{sign}(\mathbf{W}_t \mathbf{V}_t)$.

2.2. Learning the Hash Functions and Out-of-Sample Extension

We use kernel logistic regression (KLR) [2] to learn the hash functions. In effect, we design a set of k binary classifiers for each of the modality $\mathbf{X}_t$ and KLR is used to learn the functions independently for all M modalities. We consider a radial basis function as the kernel of choice in our implementation. Each feature $\mathbf{X}_t^i$ is mapped to the RKHS (Reproducing Kernel Hilbert Space) as $\phi(\mathbf{X}_t^i)$ which can be used to construct the kernel feature matrix Φ. We now need to learn linear hyperplanes in the kernel domain so as to generate the hash codes. Considering the l^{th} bit ($1 \leq l \leq k$), we need to solve for the hash functions $\mathbf{P}_t^l$ by minimizing the following objective: $\sum_{i=1}^{N} log(1 + e^{-\mathbf{B}_{t,cl} \cdot \phi(\mathbf{X}_t^i) \cdot \mathbf{P}_t^l}) + \mu ||\mathbf{P}_t^l||_2^2$ where, μ is the regularization parameter. For features coming from $\mathbf{X}_t$, we need to learn

the set of hash functions $\mathbf{P}_t = \{\mathbf{P}_t^1, \mathbf{P}_t^2, ..., \mathbf{P}_t^k\}$. Similar learning procedure is also followed for all the modalities. The above objective can be solved by using the minFunc solver [14]. Following the same protocol as in [2], we unify the learned hash codes at the end of our algorithm (this version is named as Ours$_1$). We can also directly use the learned hash codes from Stage 1 itself for cross-modal retrieval (this version is named as Ours$_2$).

To generate the hash codes for the query data coming from $\mathbf{Q}_t$ modality, we compute the hash codes in the more discriminative label space domain directly by using the learned hash functions $\mathbf{B}_{q,c} = \text{sign}\left(\mathbf{P}_t(\mathbf{Q}_t)\right)$.

2.3. Training with very large datasets

Supervised approaches such as SePH [2] and SMFH [11] show significant performance improvement over the traditional unsupervised ones. Many of these approaches typically construct the affinity/ Laplacian matrix from the training examples to account for the relationships between the data. Though this results in a boost in performance, because of the high computational cost in creating these matrices for large amounts of data, only a subset of the training set can be used during training. Thus during testing, even though the retrieval set remains the same as the training set, their hash codes need to be re-generated. Also, for two stage algorithms, any losses incurred in stage 2 [2] impacts the retrieval performance. However, if it is possible to utilize the whole of the training set in stage 1 itself, then the hash codes for the training set need not be re-generated. Based on this observation, we propose an iterative scheme to generate the hash codes for the whole training data (retrieval data) in stage 1 itself. This has another important advantage - since the whole training data can be used for generating the hash functions and transformation matrices, the chances of over-fitting is significantly reduced as we show in the analysis section.

Now, we show how the proposed approach can utilize the whole training set for construction of the hash codes by considering the learning in mini-batches. For each mini-batch i, the variables $\mathbf{V}_t^{(i)}, \mathbf{U}_t^{(i)}, \mathbf{V}_L^{(i)}, \mathbf{U}_L^{(i)}, \mathbf{W}_t^{(i)}$ can be learned independently by solving (3) to fit that particular data batch $\mathbf{X}_t^{(i)}$ and $\mathbf{X}_L^{(i)}$. However, learning the hash codes independently may over-fit the variables. To avoid this, we update the variables $\mathbf{U}_t^{(i+1)}, \mathbf{U}_L^{(i+1)}$ and $\mathbf{W}_t^{(i+1)}$ for mini-batch $(i+1)$, by utilizing the previously learned latent factors and transformations of mini-batch i as follows

$$\mathbf{U}_t^{(i+1)} = (1-\rho)\mathbf{U}_t^{(i)} + \rho C, \quad \text{where}$$
$$C = \mathbf{X}_t^{(i+1)}\mathbf{V}_t^{(i+1)^T}\left(\mathbf{V}_t^{(i+1)}\mathbf{V}_t^{(i+1)^T} + \frac{\gamma}{\lambda_t}\mathbf{I}_k\right)^{-1} \quad (11)$$
$$\mathbf{W}_t^{(i+1)} = (1-\rho)\mathbf{W}_t^{(i)} + \rho D, \quad \text{where}$$
$$D = \mathbf{V}_L^{(i+1)}\mathbf{V}_t^{(i+1)^T}\left(\mathbf{V}_t^{(i+1)}\mathbf{V}_t^{(i+1)^T} + \frac{\gamma}{\alpha_t}\mathbf{I}_k\right)^{-1} \quad (12)$$

The parameter ρ is the learning rate which provides a link between the learned variables from mini-batch i to mini-batch $(i+1)$. A brief analysis of our algorithm with respect to ρ is provided later. This can be used to learn the hash codes for very large datasets such as MirFlickr [4] and NUS-WIDE [5] in stage 1 itself. In addition, this is also useful when we are dealing with online training data for hashing purposes. In such a scenario, we can first consider a small subset of the data to learn the variables $\mathbf{U}_t$, $\mathbf{U}_L$ and $\mathbf{W}_t$. Then fixing these variables, we can continuously generate the hash codes for the incoming training data.

Table 1. Cross-view retrieval performance (MAP) of the proposed algorithm compared to the state-of-the-art on the Wiki [3] dataset with different hash code lengths k. Best results are marked in bold.

	Image-to-Text				Text-to-Image			
	k=16	k=32	k=64	k=128	k=16	k=32	k=64	k=128
CMSSH [15]	0.187	0.177	0.164	0.155	0.163	0.161	0.153	0.151
CVH [16]	0.125	0.121	0.121	0.117	0.118	0.103	0.102	0.099
IMH [8]	0.157	0.157	0.156	0.165	0.146	0.131	0.129	0.130
LSSH [9]	0.214	0.221	0.221	0.221	0.503	0.522	0.529	0.534
CMFH [7]	0.213	0.225	0.236	0.241	0.488	0.513	0.526	0.537
KSH-CV [17]	0.196	0.183	0.170	0.166	0.171	0.166	0.169	0.157
SCM$_{\text{orth}}$ [10]	0.159	0.146	0.138	0.113	0.155	0.138	0.126	0.109
SCM$_{\text{seq}}$ [10]	0.221	0.233	0.244	0.259	0.213	0.236	0.247	0.257
SePH$_{\text{rnd}}$ [2]	0.276	0.296	0.304	0.313	0.631	0.658	0.663	0.669
SePH$_{\text{knn}}$ [2]	0.278	0.295	0.306	0.313	0.631	0.657	0.664	0.670
SMFH [11]	0.270	0.286	0.295	0.296	0.605	0.626	0.635	0.642
Ours$_1$	0.264	0.284	0.293	0.302	0.619	0.655	0.668	0.674
Ours$_2$	**0.338**	**0.366**	**0.373**	**0.378**	**0.729**	**0.744**	**0.753**	**0.755**

3. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we provide the dataset details, the evaluation criteria and results of the proposed approach along with comparisons with the state-of-the-art. We provide the results of both the versions of the proposed algorithm, Ours$_1$ and Ours$_2$. For Ours$_1$, we re-generate the hash codes for both the query and retrieval set using the learned hashing functions $\mathbf{P}_t$. Similar strategy is also followed in the works of [7] [11] as it is computationally expensive to learn the hash codes for the whole training data. The iterative scheme of our algorithm can utilize the whole training data to generate the hash codes in one-shot which are then used during testing. This version of our algorithm is termed as Ours$_2$. We observe significant performance gains of Ours$_2$ over Ours$_1$ version of our algorithm, which emphasizes the usefulness of using the whole training data during training. For comparison against the state-of-the-art hashing techniques, we report the Mean Average Precision (MAP), i.e., the mean of the average precision of all the queries [2] [1]. The section ends with a brief analysis of the algorithm. The complexity of the algorithm is also provided.

3.1. Results on the Wiki Dataset

The Wiki Dataset [3] consists of $2,866$ image-text pairs from 10 different categories, with images encoded with 128-d SIFT descriptors and texts represented as 10-d topic vectors. The train:test split considered is 2173:693 as in [2].

The results for the Wiki dataset are shown in Table 1. The results for the other approaches are taken from [2]. We observe that all the supervised approaches [2] [11], including ours, performs much better as compared to the unsupervised ones [7], since they utilize the additional label information. The performance of the proposed Ours$_1$ is comparable to the supervised ones, but slightly less than them, the reason being that the others [2] [11] utilize all the relationships between the data, while we only use the pairwise correspondence. But this allows us to utilize the whole training data unlike the other approaches, which results in significant improvement of the proposed algorithm (Ours$_2$).

3.2. Results on the MirFlickr Dataset

MirFlickr Dataset [4] contains 25,000 images along with their user assigned tags. Each pair is given multiple labels out of 38 classes.

Table 2. Cross-view retrieval performance (MAP@50) of the proposed algorithm compared to the state-of-the-art on the MirFlickr [4] dataset with different hash code lengths k. Best results are marked in bold.

	Image-to-Text				Text-to-Image			
	k=16	k=32	k=64	k=128	k=16	k=32	k=64	k=128
CMSSH [15]	0.646	0.661	0.668	0.662	0.612	0.640	0.638	0.624
CVH [16]	0.645	0.636	0.627	0.620	0.659	0.650	0.646	0.658
MLBE [18]	0.608	0.586	0.584	0.588	0.593	0.618	0.655	0.639
QCH [6]	0.572	0.578	0.561	0.556	0.575	0.600	0.575	0.572
LSSH [9]	0.632	0.640	0.645	0.651	0.650	0.672	0.696	0.701
CMFH [7]	0.588	0.606	0.634	0.655	0.587	0.601	0.647	0.662
CMCQ [1]	0.670	0.671	0.678	0.682	0.724	0.733	0.739	0.755
SePH [2]	0.716	0.734	0.750	0.751	0.715	0.730	0.741	0.736
Ours$_1$	0.666	0.687	0.695	0.710	0.699	0.748	0.736	0.750
Ours$_2$	**0.670**	**0.689**	**0.857**	**0.869**	**0.830**	**0.861**	**0.863**	**0.896**

Following the same protocol as in [1], 10% of the data is used as the query with the remaining as the training set. The features considered are the same as in [1].

The results of the proposed approach and comparisons with the state-of-the-art on the MirFlickr dataset are shown in Table 2. For this dataset also, we observe similar pattern in the results as in the Wiki dataset, where Ours$_1$ performs similarly and Ours$_2$ performs significantly better as compared to SePH algorithm [2]. Since this is a large dataset, SePH algorithm [2] used only about 5000 training samples to construct the affinity matrix and learn the hash codes and hash function. But using the procedure in 2.3, we could use the full training data for generating the hash codes, transformation matrices and the hash functions.

3.3. Results on the NUS-WIDE Dataset

NUS-WIDE Dataset [5] contains $269,648$ images with each image marked with relevant tags. Following the protocol in [2], $1,86,577$ pairs are considered to create the train : test split as $1,82,577$:4000. The images are represented by 500-d bag-of-words features and texts by 1000-d vectors of the most frequent tags.

The results on the NUS-WIDE dataset are shown in Table 3. We observe that for most of the cases, Ours$_1$ gives better results than the supervised approach in [2]. This is also a large dataset, and so [2] uses only a small subset (5000 samples) during the training stage, since it would require an enormous amount of memory to compute the affinity matrix for the whole data. In contrast, the proposed algorithm could utilize all of the $1,82,577$ data samples during training. Thus for this data also, we observe that Ours$_2$ significantly outperforms all the other state-of-the-art approaches.

3.4. Analysis of the algorithm

Complexity Analysis: In our algorithm, $d_t > k, N > k$. The cost of updating each of the variables $\mathbf{U}_t$, $\mathbf{W}_t$ and $\mathbf{V}_t$ are $O(d_t N k)$, $O(k^2 N)$ and $O(d_t N k)$ respectively. Thus the total complexity of the algorithm is found to be $O(d_t N k)$.

Analysis of the ρ parameter: Here, we conduct a simple experiment to understand the effect of the parameter ρ on the hash code learning for the NUS-WIDE dataset. We have considered $10,000$ samples from the dataset and set mini-batch size to 1000. The hash code length of $k = 32$ and $k = 128$ are considered. Using this data, we analyze how well the learnt hash codes are able to capture the semantic relationship between the data. The average MAP for all the mini-batches and the total MAP for the whole data are reported in Fig. 2. We observe that as ρ increases from 0 to 1 (no link with the previous batch), we are basically over-fitting the learned $\mathbf{U}_t^{(i)}, \mathbf{U}_L^{(i)}$ and $\mathbf{W}_t^{(i)}$ to that particular batch i. Thus though the mean MAP remains consistent for all bit code size, the total MAP for the whole data degrades substantially. We also observe that as k increases, the MAP for the mini-batch as well as the total MAP increases. Thus we see that being able to train on the whole data can considerably reduce the chances of over-fitting, which will result in better performance on the test data.

Table 3. Cross-view retrieval performance (MAP@50) of the proposed algorithm compared to the state-of-the-art on the NUS-WIDE [5] dataset with different hash code lengths k. The best results are marked in bold.

	Image-to-Text				Text-to-Image			
	k=16	k=32	k=64	k=128	k=16	k=32	k=64	k=128
CMSSH [15]	0.524	0.521	0.521	0.481	0.417	0.425	0.418	0.420
CVH [16]	0.535	0.525	0.501	0.470	0.560	0.543	0.516	0.482
MLBE [18]	0.447	0.454	0.470	0.402	0.435	0.488	0.502	0.442
QCH [6]	0.509	0.527	0.520	0.513	0.509	0.517	0.509	0.508
LSSH [9]	0.536	0.552	0.567	0.572	0.635	0.663	0.682	0.692
CMFH [7]	0.474	0.482	0.513	0.506	0.510	0.564	0.589	0.594
CMCQ [1]	0.563	0.590	0.599	0.609	0.689	0.708	0.719	0.725
SePH [2]	-	0.586	0.601	0.607	-	0.726	0.746	0.746
Ours$_1$	0.584	0.599	0.595	0.605	0.738	0.750	0.767	0.767
Ours$_2$	**0.718**	**0.743**	**0.765**	**0.782**	**0.842**	**0.853**	**0.845**	**0.855**

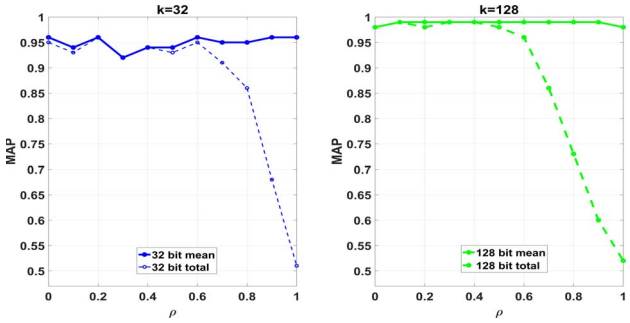

Fig. 2. The effect of ρ on the hash code learning framework.

Implementation details: The values of the different parameters used in the proposed algorithm are $\{\lambda_1, \lambda_2, \lambda_L, \alpha_1, \alpha_2, \gamma\}$ = $\{1, 1, 1, 0.1, 0.1, 0.1\}$ for Wiki, $\{0.1, 1, 100, 0.1, 1, 1\}$ for MirFlickr and $\{0.1, 1, 100, 0.1, 1, 1\}$ for NUS-WIDE dataset respectively. These values have been set by cross validation (for $M = 2$ for all cases.) For the second stage of our algorithm we followed the same protocol as in [2]. For the iterative scheme, we have used batch size $= 1000$ and $\rho = 0.1$ for our experiments.

4. CONCLUSIONS

In this paper, we proposed a supervised hashing scheme based on matrix factorization. We also proposed an iterative scheme to train very large datasets. Both the versions of the proposed algorithm are evaluated on three datasets and comparisons with the state-of-the-art show the effectiveness of the proposed approach.

5. REFERENCES

[1] Ting Zhang and Jingdong Wang, "Collaborative quantization for cross-modal similarity search," in *CVPR*, 2016, pp. 2036–2045.

[2] Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang, "Semantics-preserving hashing for cross-view retrieval," in *CVPR*, 2015, pp. 3864–3872.

[3] Nikhil Rasiwasia, Jose Costa Pereira, Emanuele Coviello, Gabriel Doyle, Gert RG Lanckriet, Roger Levy, and Nuno Vasconcelos, "A new approach to cross-modal multimedia retrieval," in *ACM-MM*, 2010, pp. 251–260.

[4] Mark J. Huiskes and Michael S. Lew, "The mir flickr retrieval evaluation," in *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA, 2008, ACM.

[5] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng, "Nus-wide: a real-world web image database from national university of singapore," in *ACM-CIVR*, 2009, pp. 48–56.

[6] Botong Wu, Qiang Yang, Wei-Shi Zheng, Yizhou Wang, and Jingdong Wang, "Quantized correlation hashing for fast cross-modal search," in *IJCAI*, 2015, pp. 25–31.

[7] Guiguang Ding, Yuchen Guo, and Jile Zhou, "Collective matrix factorization hashing for multimodal data," in *CVPR*, 2014, pp. 2083–2090.

[8] Jingkuan Song, Yang Yang, Yi Yang, Zi Huang, and Heng Tao Shen, "Inter-media hashing for large-scale retrieval from heterogeneous data sources," in *SIGMOD*, 2013, pp. 785–796.

[9] Jile Zhou, Guiguang Ding, and Yuchen Guo, "Latent semantic sparse hashing for cross-modal similarity search," in *SIGIR*, 2014, pp. 415–424.

[10] Dongqing Zhang and W. Jun Li, "Large-scale supervised multimodal hashing with semantic correlation maximization.," in *AAAI*, 2014, vol. 1, pp. 2177–2183.

[11] Jun Tang, Ke Wang, and Ling Shao, "Supervised matrix factorization hashing for cross-modal retrieval," *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3157–3166, 2016.

[12] Guosheng Lin, Chunhua Shen, David Suter, and Anton van den Hengel, "A general two-step approach to learning-based hashing," in *ICCV*, 2013, pp. 2552–2559.

[13] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan, "Supervised hashing for image retrieval via image representation learning.," in *AAAI*, 2014, vol. 1, pp. 2156–2162.

[14] Mark Schmidt, "minfunc: unconstrained differentiable multivariate optimization in matlab," 2005.

[15] Michael M Bronstein, Alexander M Bronstein, Fabrice Michel, and Nikos Paragios, "Data fusion through cross-modality metric learning using similarity-sensitive hashing.," in *CVPR*, 2010, vol. 1, pp. 3594–3601.

[16] Shaishav Kumar and Raghavendra Udupa, "Learning hash functions for cross-view similarity search," in *IJCAI*, 2011, vol. 22, pp. 1360–1365.

[17] Jile Zhou, Guiguang Ding, Yuchen Guo, Qiang Liu, and Xin-Peng Dong, "Kernel-based supervised hashing for cross-view similarity search," in *ICME*, 2014, pp. 1–6.

[18] Yi Zhen and Dit-Yan Yeung, "A probabilistic model for multimodal hash function learning," in *SIGKDD*, 2012, pp. 940–948.