Weakly Supervised Deep Image Hashing through Tag Embeddings

Vijetha Gattupalli, Yaoxin Zhuo, Baoxin Li
Arizona State University
{vijetha.gattupalli, yzhuo6, baoxin.li}@asu.edu

Abstract

Many approaches to semantic image hashing have been formulated as supervised learning problems that utilize images and label information to learn the binary hash codes. However, large-scale labelled image data is expensive to obtain, thus imposing a restriction on the usage of such algorithms. On the other hand, unlabelled image data is abundant due to the existence of many Web image repositories. Such Web images may often come with image tags that contain useful information, although raw tags in general do not readily lead to semantic labels. In this paper, we formulate the problem of semantic image hashing as a weakly-supervised learning problem, utilizing user-generated tags associated with the images to learn the hash codes. Specifically, we extract the word2vec semantic embeddings of the tags and use the information contained in them for constraining the learning. Accordingly, we name our model Weakly Supervised Deep Hashing using Tag Embeddings (WDHT). WDHT is tested for the task of semantic image retrieval and is compared against several state-of-art models. Results show that our approach sets a new state-of-art in the area of weekly supervised image hashing.

1. Introduction

Semantic Image Hashing has been an active research area for the past few years due to its usefulness in efficient search of massive image databases. Briefly, the task is to map images to binary codes such that some notion of semantic similarity is preserved. Often, similarity is determined by ground-truth class labels, which are expensive to obtain and thus limit the amount of training data available. On the other hand, many Web images today have associated textual meta-data (tags) often readily available without cost. Owing to these facts, in this paper, we attempt the problem of weekly supervised semantic image hashing by leveraging the tag information associated with the Web images.

We employ a weakly-supervised approach mainly due to the following reasons. Tags may contain some information related to the semantics of the images. However, it is non-trivial to extract explicit label information from raw tags. Table 1 illustrates three samples from the NUS-WIDE dataset. It can be noticed that sample 1 has no tag directly associated with the label "dancing". While samples 2 and 3 have some tags conveying label information, there are other shortcomings. For example, they are associated with too many uninformative tags. These uninformative tags may be a consequence of the social-media behaviour of the public like opinion expression, self presentation, attracting attention etc. [1] This results in tags that may be subjective (eg. *#thegoldendreams, #handsome*), purely context oriented (eg. *#india, #conradhotel, #katrina*), photography related (*#wideangle*) etc. Thus these tags contain information which is not as much related to the image content as labels, making label extraction from tags more difficult. There are some prior works [1], [2] that attempted to address the difficulties in extracting information from raw tags.

Even though our work focuses on using tag information in learning the hash space, our algorithm does not fall under the category of cross-modal hashing (CMH). CMH deals with learning hash spaces that are shared for samples from various modalities. Ideally, a space thus learnt

	Sample 1	Sample 2	Sample 3
Images			
Tags	*#india #cinema #movie #star #still #handsome #bollywood #actor #khan #shahrukh #srk #omshantiom*	*#sunset #bali #reflections #indonesia #mirror #asia #boda #mariage #hochzeit #indonesien #heirat #chappel #conradhotel #50faves #justimagine #weddingchappel #perfectangle #infinestyle #megashot #theroadtoheaven #thegoldendreams*	*#wood #trees #fence #track #derbyshire #farming #wideangle #fields #agriculture #grassland #sigma1020 #autums #marlock #holestone #holestonemoor*
Labels	dancing	buildings, clouds, reflection, sky, sunset	grass, sky, tree

Table 1: Illustrating the image-tag-label triplet for some random samples from NUS-WIDE dataset.

should be able to retrieve samples from one modality by using query samples from a different modality (e.g., retrieving images/videos using text queries and vice versa) [3]. Our work only deals with direct image hashing where the query and retrieval samples are images. We only utilize the information from tags to learn better hash spaces for semantic image retrieval. Further, much work in CMH assumes the availability of image-tag-label triplets and use this information to learn the shared hash space, leading to supervised learning, while ours is a weakly supervised approach.

A key component of our method is the utilization of the *word2vec* model [4], a method for embedding English words onto a vector space such that the cosine similarity between the vectors of the words is in accordance with their semantic similarity. In our task, the <image,tag set> pairs are from the Web image datasets, and the tags generally bear some relevance to the semantics of the image (albeit this relevance may be weak, noisy, and incomplete). Hence we employ the *word2vec* representation of the tags in our model, and regularize the learned hash space in such a way that images having similar tag vectors should have similar hash codes. Using the word vectors of the tags may lead to a better semantic hash space as compared to using only the binary tag vectors themselves. For example, if the training data contains images of cats and dogs, and several other non-animal classes, we would want the hash sub-spaces of the cats and the dogs to be close to each other. Further, an animal in a test set (e.g.,horse), whose true class is not defined in the training set would ideally be mapped to a code closer to the combined sub-space of the cat and the dog, than to other non-animal classes. Such desired arrangement of the sub-spaces could be naturally attained through employing the word-vector similarities of the tags in training.

In this work, we propose a deep neural network, complete with a learning algorithm, for weakly supervised learning of a semantic hashing model through using the word embeddings of the image tags. To the best our knowledge, this is the first work to use an end-to-end deep model to learn hash vectors using images and tags alone (without using labels). On the particular task of image hashing, our method appears to be the first work on using word embeddings of tags in a weakly supervised setting. We evaluate our approach and report systematic comparison with relevant state-of-the-art, and our approach is shown to outperform existing unsupervised or weakly supervised hashing methods for semantic image retrieval.

2. Related Work

Much effort in the area of semantic image hashing has been directed towards supervised methodologies. While there is some work in the area of unsupervised hashing, very little attempt was made in weakly supervised hashing. Therefore, we compare our model to both weakly supervised and unsupervised methods during evaluation. On similar lines, in this section, we give a brief overview of the related work from both the areas.

One foremost image hashing algorithm called the Locality Sensitive Hashing [5] works on the principle of projecting the data on to random hyperplanes and computing each bit based on which half-space the sample falls into. This algorithm is data-independent and therefore the produced hash codes do not capture the structure in the data. Several variants [6], [7], [8] have been proposed, all producing hash codes irrespective of the distribution of the data.

Another paradigm of image hashing is the data-dependent hashing methods. Traditionally data-dependent methods have been formulated as independent feature learning and hash coding stages. With the advent of deep learning and big data, the literature has moved towards learning hash codes as single stage algorithms, taking in images as inputs and directly learn the hash codes. This can also be interpreted as an inbuilt feature learning technique that does not require human intervention.

Approaches such as [9], [10], [11] are some representative works of non-deep learning based unsupervised learning. [9] tried to minimize the quantization error between the real-valued uncorrelated feature vector and the binary code by finding a rotation of the zero-centered data. [10] showed the analogy between the problem of finding the optimal hash space distribution and graph partitioning algorithm and attempted the problem using spectral ways. [11] attempted the problem of learning hash spaces in a semi-supervised way by back propagating the classification loss over a limited labeled data-set and an entropy based loss over the entire labelled and unlabelled data-set.

Representative deep-learning-based unsupervised hashing algorithms include [12], [13], [14]. The work of [12], though being deep-learning-based, is not an end-to-end framework that can take in raw images and produce the hashes. They used GIST features as inputs to the neural network and learned the hash codes by minimizing the quantization loss, maximum variance loss, and the independent bit loss. The key idea of [13] is to produce rotation invariant binary codes and showed that they achieve state-of-art performance on three different tasks namely, image matching, image retrieval and object recognition. The approach of [14] learns hash codes as the outputs of the hidden layer of a binary auto-encoder, making the learning problem NP-hard and thus an alternate optimization scheme was used.

Another note-worthy mention in the area of uni-modal image hashing is [15]. They utilized the word embedding of labels as the supervision to learn an image hash space. While this appears similar to our work, they used vector representations of labels, rendering the work to fall under the category of supervised image hashing, whereas our work uses vector representations of raw tags.

A common characteristic among most deep learning and non-deep-learning based semantic hashing methods is that they rely only on the information from the images to learn the hash codes, often completely ignoring other associated metadata. Several works [16], [17], [18] in the area of Cross Modal Hashing (CMH) attempted utilizing tag information along with image data to learn the hash space. However, as mentioned previously, they learn a common hash space for various modalities of input (image and tag in this case), which is different from what we intend to do. Among all the CMH methods, [17] is the closest approach to our work. [17] intends to align the visual space of images and the semantic space of sentences using language (*word2vec*) and vision (CNN based) models. The main difference between their work and ours is that we attempt to use tag information which is much noisy than the actual English sentences they used in their work. Practically, such clean English sentences are as hard to obtain as the supervised label information. An extensive discussion on CMH and uni-modal hash learning can be found in [3] and [19] respectively.

Unlike CMH, weakly supervised hashing methods leverage only the image-tag information during training. [20], [21], [22] are some well-known works in this area. The authors of [20] proposed a framework which consist of two stages, weakly supervised pre-training and fine-tuning using supervised labels. [21] used collaborative filtering with hashing in predicting the image-label associations, where the ground-truth labels are used to generate the label matrix. To our best knowledge, [22] is the only prior approach that attempted truly weekly supervised hashing (i.e., without using label information). More specifically, they attempted to explore the discriminative information and the local geometric structure from the tags and images. They then formulated the hashing problem as an eigenvalue problem. Considering these facts, we only compare our approach to [22] among the weekly-supervised methods.

In this work, we build an end-to-end deep learning hashing model that does not require expensive labels in training but can still generate semantically meaningful hash codes. In the experiments section, we compare our model to the following unsupervised and weakly supervised image hashing approaches: [9], [10], [11], [14], [21], [22], [23], [24], [25], [26]. Additionally, to show the significance of the usage of tag embeddings, we develop a deep-learning baseline that learns a semantic hash space using only the binary tag vectors. More details about our approach and the baseline model are presented in the next section.

3. Proposed Approach

3.1. Problem Formulation

In this work we assume that the datasets have triplets of image-tags-labels ($\boldsymbol{x}_i$, T_i, $\boldsymbol{l}_i$). Here, $\boldsymbol{x}_i$ represents the image feature vector for the i^{th} sample, T_i represents the corresponding tags set and $\boldsymbol{l}_i$ represents its binary label vector. In a generic scenario, each sample is associated with more than one tag and more than one semantic label. Therefore, the tags are represented as a set T_i and the labels are represented as a binary vector $\boldsymbol{l}_i$. In the label vector, the value of an element is 1 if the corresponding label is associated with that image and is 0 otherwise. Our task is to find a function $\Psi\big(\cdot\big)$ that takes ($\boldsymbol{x}_i$, T_i) as inputs and produces a hash vector $\boldsymbol{b}_i$ as output. The hash space thus learnt should map semantically similar images, defined by the label vectors, to nearby hash codes and dissimilar ones to farther codes. While the labels assumed to be unavailable during the training phase, they are employed during the testing phase to measure the performance of the learnt model.

3.2. Tag Processing

Let τ_i^j represent a tag in the tag set T_i,. where j is the index of the tag in the set, i.e., $j \in [1, m]$ where m is the total number of tags associated with the i^{th} sample. We convert each tag τ_i^j into a d-dimensional vector using the *word2vec* language model [4]. Thus for each tag τ_i^j, we obtain a vector representation $\boldsymbol{v}_i^j$ which is the *word2vec* representation of the tag word τ_i^j. Since each image has multiple tags, we aggregate all the tag vectors into a single d-dimensional vector. In this work, we adopted basic functions like *tf* (tag frequency), *itf* (inverse tag frequency) and *mean* to compute the aggregated vector $\boldsymbol{w}_i$. In experiments, we will compare these aggregation techniques by their performance.

The formulae used to compute $\boldsymbol{w}_i$ are given below.

$$\begin{aligned} &\textit{mean} : \boldsymbol{w}_i = \frac{1}{m}\sum_{j=1}^{m} \boldsymbol{v}_i^j \quad \textit{tf} : \boldsymbol{w}_i = \frac{1}{m}\sum_{j=1}^{m} \frac{n(\tau_i^j)}{N}\boldsymbol{v}_i^j \\ &\textit{itf} : \boldsymbol{w}_i = \frac{1}{m}\sum_{j=1}^{m} \log \frac{N}{n(\tau_i^j)}\boldsymbol{v}_i^j \end{aligned} \tag{1}$$

where N is the total number of tags in the database, and $n(\tau_i^j)$ is the number of images associated with tag τ_i^j.

Thus we arrive at the *image - tag vector* ($\boldsymbol{x}_i$, $\boldsymbol{w}_i$) pairs from the initial *image - tag set* ($\boldsymbol{x}_i$, T_i) pairs.

3.3. Designing a Network for Hashing

We use the pre-trained AlexNet model as a key building block for our hashing model. The network takes 227X227X3 dimensional images as input and passes them through five convolutional layers and two fully connected layers, labelled as CONVi (i=1,...,5), FC1 and FC2. Until the FC2 layer, the architecture is identical to the AlexNet [27] architecture and the weights are initialized to the pre-trained ImageNet [28] weights. The FC2 layer produces a 4096 dimensional vector, which is given as input to another fully connected layer FC3. FC3 outputs 256 dimensional

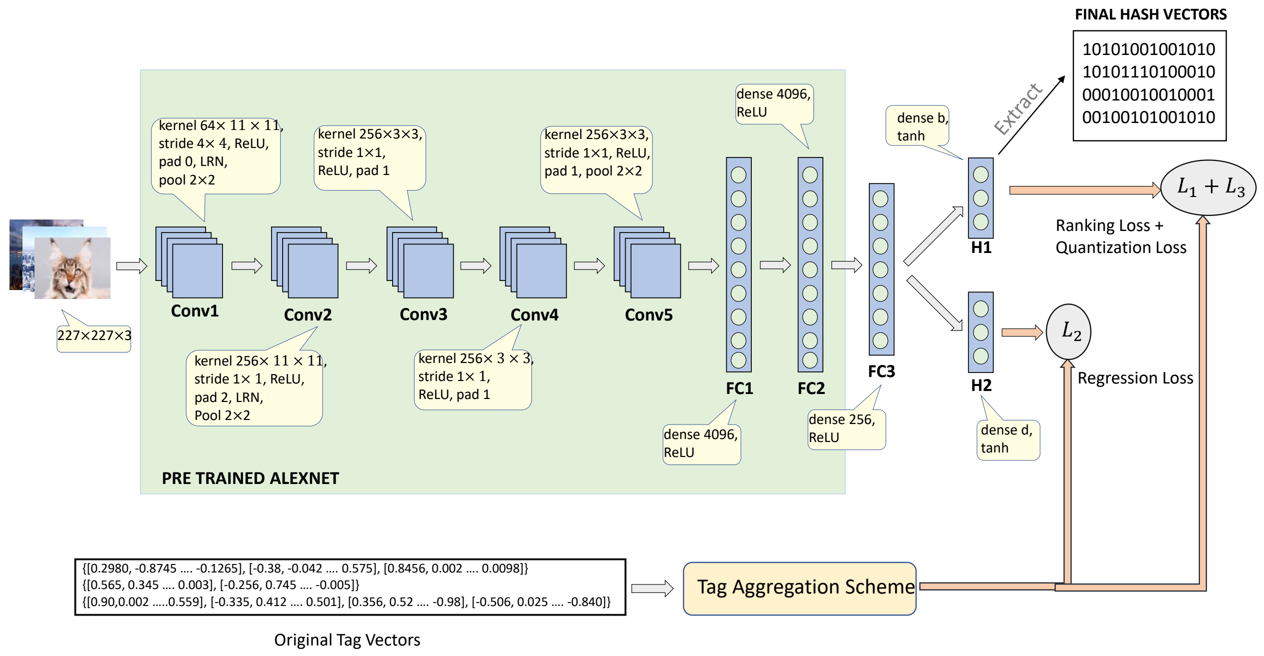

Figure 1: Proposed architecture. The green box represents the pre-trained AlexNet model; The FC3, H1 and H2 layers are the newly appended layers. The final hash codes are extracted from the H1 layer.

vector which is further fully connected to two layers H1 and H2 in a lateral fashion. The acronyms H1 and H2 represent the *Head1* and *Head2* respectively. The outputs of H1 and H2 are b (number of bits in the hash code) and d (dimensionality of the aggregated tag vector) dimensional vectors, which are then topped by *sigmoid* and *tanh* activations respectively. The overall model is shown in Figure 1. The new layers beyond the AlexNet layers are initialized with *glorot normal* [29] weights. VGG-19 was also attempted, giving results similar to those of the AlexNet model but with much-increased training time. Hence we study and compare using only the Alexnet-based model.

The model is trained on three loss components back propagating from the two heads H1 and H2 into the network. More specifically, we back propagate pair-wise similarity loss and quantization loss from H1, and mini-batch wise hinge loss from H2. Thus we presume that the loss on H2 (hinge loss) forces the network to form feature spaces (especially at the later layers, H2 and FC3) that are in accordance with the semantic information contained in the aggregated tag vectors, $\boldsymbol{w}_i$. On the other hand, the pair-wise loss on H1 aligns the hash space such that semantically similar image pairs are close by and dissimilar pairs are farther. Thus the two main loss components augment each other and guide the network towards learning a semantically meaningful hash space. The third loss component, the quantization loss, forces the output of H1 to be close to 0 or 1.

Pairwise Euclidean loss on H1 was first used for hashing in [30] while the quantization loss was first used in [9]. The hinge loss on the head H2 is a ranking loss first used by [31] to learn a semantically meaningful real-valued image representation using word embeddings of classification labels. While the hinge loss component does not seem to serve a clear purpose in this network architecture, empirical results show that this component contributes significantly to the performance boost of our model. Also, [31] mentioned that using such loss boosts the performance of their model instead of using a $L2$ component. They presume that this could be due to the fact that the problem of forming a semantically meaningful image representation is a ranking problem in general and therefore such a ranking loss could be more relevant. On similar lines, we can argue that the current problem of learning image hashes is a ranking problem as well, and thus, such a hinge loss component could boost the performance of a retrieval system significantly.

During inference, only H1 is used to extract the features, which are then quantized to obtain the hash code according to the following scheme: $\boldsymbol{b}_i = \frac{1}{2}(sgn(\boldsymbol{h}_i^{(1)} - 0.5\boldsymbol{1}) + 1)$. Here, $\boldsymbol{h}_i^{(1)}$ represents the real-valued feature vector obtained at the output of H1, sgn represents a sign function that outputs $1/{-1}$ based on if the input to the sign function is positive or negative and lastly, $\boldsymbol{1}$ represents a vector of ones of length b. Thus, we obtain binary codes which have a value of 1/0 from a raw train/test images.

3.4. Designing the Loss Functions

Pair-wise Similarity Loss: Most state-of-art supervised learning methods assume binary similarity between two images: two images can be either similar (1) or dissimilar (0) depending on if they share a common label. However, in the current weakly supervised learning context, we intend to use cosine similarity between the aggregated tag vectors as the ground truth similarity. Since cosine similarity is real-valued, taking values between -1 and 1, the ground-truth similarity in our case is not binary valued, i.e., we can deem an image pair to be less similar or more similar, instead of absolutely declaring it to be similar or dissimilar. We consider only this notion of ground truth similarity during training and stick with the 0/1 similarity during evaluation.

We formulate the pair-wise similarity loss function as follows. For any image pair ($\boldsymbol{x}_i$, $\boldsymbol{x}_j$), the loss function should push the corresponding hashes closer if the cosine distance between them is smaller and vice-versa. The equation of this loss function is given below,

$$\begin{aligned} L_1 = \sum_{i=1}^{k}\sum_{j=1}^{k}[\frac{1}{b}(\boldsymbol{h}_i^{(1)} - \boldsymbol{h}_j^{(1)})^T \cdot (\boldsymbol{h}_i^{(1)} - \boldsymbol{h}_j^{(1)}) \\ - \frac{1}{2}(1.0 - \frac{\boldsymbol{w}_i^T \cdot \boldsymbol{w}_j}{\|\boldsymbol{w}_i\|\|\boldsymbol{w}_j\|})]^2 \end{aligned} \tag{2}$$

where k is the mini batch size and the two summations signify computing pairwise losses across all possible pairs. The vectors $\boldsymbol{h}_i^{(1)}$ and $\boldsymbol{h}_j^{(1)}$ represent the output vectors of H1 for sample $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ respectively. A lower value of L_1 is obtained when a high value of $1.0 - \frac{\boldsymbol{w}_i^T \cdot \boldsymbol{w}_j}{\|\boldsymbol{w}_i\|\|\boldsymbol{w}_j\|}$ results in a high value of $(\boldsymbol{h}_i^{(1)} - \boldsymbol{h}_j^{(1)})^T \cdot (\boldsymbol{h}_i^{(1)} - \boldsymbol{h}_j^{(1)})$ and vice-versa. Higher value of $1.0 - \frac{\boldsymbol{w}_i^T \cdot \boldsymbol{w}_j}{\|\boldsymbol{w}_i\|\|\boldsymbol{w}_j\|}$ is obtained when the samples are dissimilar, thus the hash codes should be pushed apart. Similarly, lower value of this term is obtained when the samples are similar and therefore the hash codes should be pushed closer.

Mini-batch-wise Hinge Loss: In addition to the pair-wise similarity loss, we also intend to back-propagate a loss that forms a semantic embedding space at the output of H2. Such a loss function adjusts the feature spaces of not only the H2 layer but also some of the previous layers (FC3, FC2), thus transmitting the semantic information from the tags back into the network. As H1 is connected to the output of FC3, the semantic information contained in FC3 will aid in learning the hashes at the output of H2, thus enhancing the model's performance. To this end, we define the following loss,

$$L_2 = \sum_{n}\sum_{j \neq n} \max[0, margin + \boldsymbol{w}_j \cdot \boldsymbol{h}_n^{(2)} - \boldsymbol{w}_n \cdot \boldsymbol{h}_n^{(2)}] \tag{3}$$

where $\boldsymbol{h}_n^{(2)}$ represents the output of the head H2 for the n^{th} sample in the mini-batch. The loss L_2 is 0 only when the quantity $\boldsymbol{w}_n \cdot \boldsymbol{h}_n^{(2)}$ is more than $margin + \boldsymbol{w}_j \cdot \boldsymbol{h}_n^{(2)}$. That is, the value of the loss is zero only when the prediction of head $H2$ for the n^{th} sample is closer to the ground truth aggregated tag vector $\boldsymbol{w}_n$ than to any other ground truth tag vector $\boldsymbol{w}_j$ by a margin $margin$. A similar idea was previously considered in [32] [33], where the goal was to semantically embed videos onto a space using the *word2vec* representation of the video labels. As such, their approach is supervised (i.e., assuming the label information).

Quantization Loss: We further impose the quantization loss on the H1 output to force the outputs to be close to 0 or 1, as follows, which penalizes the network if the output of a neuron is close to 0.5:

$$L_3 = -\sum_{i=1}^{k}\frac{1}{b}(\boldsymbol{h}_n^{(1)} - 0.5\boldsymbol{I})^T \cdot (\boldsymbol{h}_n^{(1)} - 0.5\boldsymbol{I}) \tag{4}$$

During training, we weigh the three loss components L_1, L_2 and L_3 by factors λ_1, λ_2 and λ_3 respectively. Therefore the resultant loss that will be back-propagated is: $L = \lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3$.

3.5. The binary tag-vector model

In addition to comparing our method with several state-of-art models, we built another deep-learning baseline that uses the binary tag vectors for supervision, unlike the *word2vec* tag embeddings we used in WDHT. We call this model the *binary tag-vector model* in the rest of the text. The reason for this baseline is that we want to evaluate how the network performs if it is trained similar to the state-of-art supervised models where label-similarity is used to compute the loss. This model has no aggregated *word2vec* to regress the outputs. To accommodate this, we make slight modifications to our model. First, we suppose that two images are similar if both share at least one tag. This is certainly not a perfect metric, but it is similar to most state-of-art supervised models, where sharing at least 1 common label is used to define similarity. Also, we use L_4 loss rather than L_2 loss in this model since we only have a binary notion of similarity rather than the original case with cosine similarity. Since our problem setting is weakly supervised, we use tag vectors instead of label vectors. Tag vectors are binary vectors whose length is equal to the total number of tags in the data-set and will have a value of 1 if the tag is associated with the image and 0 otherwise.

Regarding the network architecture, only the head H1 is kept and H2 is completely removed. We do this owing to the fact that the real-valued vectors (like aggregated tag vectors in the above scenario) are not available in this case, to regress the outputs to. Additionally, in the previous case, the loss applied on H1, i.e., the $L1$ component has a real-valued ground truth similarity, unlike the current scenario.

Method	12 bits	24 bits	32 bits	48 bits
itf	0.6124	0.6323	0.6531	0.6644
tf	0.6394	**0.6836**	0.6881	**0.6835**
mean	**0.6709**	0.6805	**0.6955**	0.6621

Table 2: Comparing the mAP of the model with the *itf*, *tf* or *mean* aggregation functions for the NUS-WIDE dataset.

Therefore, we use a different loss component (contrastive loss) to accommodate the binary valued ground truth similarity labels. The equation of the loss is as follows,

$$L_4 = \sum_{i=1}^{k}\sum_{j=1}^{k} S(1-\beta)D + (1-S)\beta(\max(0, margin - D))^2$$
$$\text{where} \quad D = \frac{1}{b}(\boldsymbol{h}_i^{(1)} - \boldsymbol{h}_j^{(1)})^T \cdot (\boldsymbol{h}_i^{(1)} - \boldsymbol{h}_j^{(1)}) \quad (5)$$

Here, $margin$ represents the margin associated with the hinge loss component of the contrastive loss, S represents the ground truth similarity label, and β represents the fraction of similar sample pairs present in the mini batch. Weighing the loss sub-components by β and $1-\beta$ respectively are important due to the fact that in any mini-batch only a small fraction of the image pairs will have at least one tag in common, thus making the dataset highly imbalanced. We therefore incorporate β weight factor in the loss.

Thus the final loss for the binary tag-vector model becomes: $L = \lambda_3 L_3 + \lambda_4 L_4$

4. Experiments and Results

This section reports experimental evaluation of the proposed method with comparison to the relevant state-of-the-art approaches. The source code implementing the proposed method is available from the last author's Website.

4.1. Datasets

NUS-WIDE This is a Web image dataset with 269,648 images collected from Flickr. Each image is associated with a set of tags. [34] presents that there is a total of 425,059 tags associated with the 269k images. Further, the authors of [34] conducted manual annotation of these images to a predefined set of 81 labels. For our experiments, we used only the images that are associated with at least one of the 21 most frequent labels. Thus we formed a training set of 100,000 images and a testing set of 2,000 images. We used the whole training set as the database and the testing set as the query set during evaluation.

MIR-FLICKR25K This is a comparatively smaller dataset with 25,000 images collected from Flickr and contains 1386 tags associated with them. [35] manually associated the images with 38 semantic categories. For our experiments, we used the images which are associated with at least one of the 38 categories. Thus we used a total of 16,000 images for training and 2,000 for testing. For both the data-sets, we randomly picked the testing set without considering the labels of the images.

To our best knowledge, these are the only two commonly-used datasets that contain image-tag-label triplets. (E.g., **CIFAR-10** does not have tag information.) So we use these two datasets for experiments.

4.2. Training

We trained our model using mini-batch gradient descent with a learning rate of 0.001 for the last three layers (FC3, H1, and H2) and a learning rate of 0.0001 for the pre-trained layers (from CONV1 to FC2). We also used the momentum term with the rate of momentum equal to 0.9. The weighing factors for the losses, λ_1, λ_2, λ_3 and λ_4, are set to 1.0, 10.0, 1.0 and 1.0 respectively for all the experiments, which were determined by performing a grid search over the hyper-parameter space. The *word2vec* model that we used was pre-trained (using Wikipedia documents) and outputs a 300-dimensional vector for a given word. Therefore the number of output neurons on H2 is set to 300.

4.3. Performance Evaluation

We evaluated the learned hash codes for the task of semantic image retrieval. We used the mean-Average-Precision (mAP) metric for comparing performance. We used the same protocol used by [36], [37], [38] and several others to compute the mAP values. The results are compared against eleven state-of-art approaches ITQ, PCAH, LSH, DSH, SpH, SH, AGH, DH, UH-BDNN, DeepBit and WMH. All the methods, except WMH are run using the code provided by the authors and for the suggested hyper-parameter settings. As most of the works presented here are based on the pre-determined feature vectors, we extracted the 4096-dimensional vectors from the AlexNet model (i.e the output of FC2) and used them as input to these methods. For WMH we directly quote the results from the original paper (the code is not publicly available). For a fair comparison, we run our model with the same experimental setting as WMH and report the results. We first filtered the images and tags in WMH's standard, then performed another round of experiments using only 5,000 training images and 1,000 query images for the two datasets.

Firstly, to finalize the tag aggregation scheme, we compared the performance of our model using the *itf*, *tf* and *mean* functions for aggregation on NUS-WIDE data-set. We noticed that *mean* worked slightly better than the *idf* and *tf* as can be seen from Table 2. Further, we performed a variance analysis on the word vectors of tags associated with each image. More specifically, we computed the variance of the tag vectors for each image and then analyzed the

Algorithm	**NUS-WIDE**				**MIRFLICKR-25K**			
	12bits	24bits	32bits	48bits	12bits	24bits	32bits	48bits
ITQ [9]*(non-deep)*	0.5295	0.5227	0.4932	0.5275	0.6418	0.655	0.6253	0.6504
PCAH [11]*(non-deep)*	0.4566	0.4209	0.4016	0.3971	0.6098	0.6033	0.6085	0.6169
LSH [5]*(non-deep)*	0.3308	0.3682	0.3726	0.3918	0.5708	0.5885	0.5843	0.6015
DSH [23]*(non-deep)*	0.5065	0.5118	0.4902	0.4807	0.6561	0.6593	0.644	0.6422
SpH [24]*(non-deep)*	0.3829	0.3959	0.3907	0.3947	0.586	0.5785	0.5789	0.5789
SH [10]*(non-deep)*	0.4503	0.4029	0.4006	0.3731	0.6251	0.6157	0.6044	0.596
AGH [26]*(non-deep)*	0.535	0.5226	0.497	0.4791	0.6378	0.6484	0.6473	0.6346
DH [12]*(deep)*	0.4036	0.3974	0.3932	0.4014	0.5833	0.5945	0.5932	0.5942
UH-BDNN [14]*(deep)*	0.4982	0.4996	0.4823	0.4853	0.6324	0.6279	0.6274	0.6258
DeepBit [13]*(deep)*	0.4225	0.4247	0.4359	0.431	0.5974	0.6032	0.6077	0.6115
Binary Tag Vector*(deep)*	0.4809	0.475	0.4793	0.4702	0.6064	0.6087	0.6077	0.6098
Proposed(WDHT)*(deep)*	**0.6258**	**0.6397**	**0.6606**	**0.647**	**0.687**	**0.695**	**0.6667**	**0.6621**
WMH**(non-deep)*	0.299	0.306	0.307	0.309	0.585	0.590	0.582	0.573
Proposed(WDHT*)*(deep)*	0.4910	0.4916	0.4835	0.485	0.626	0.6355	0.6326	0.6308

Table 3: MAP values of NUS-WIDE and MIR-FLICKR25k data-sets computed using the top 50,000 retrieved images.

Algorithm	**NUS-WIDE**				**MIRFLICKR-25K**			
	12bits	24bits	32bits	48bits	12bits	24bits	32bits	48bits
ITQ [9]*(non-deep)*	0.6329	0.6299	0.594	0.6478	0.6908	0.7064	0.6684	0.6996
PCAH [11] *(non-deep)*	0.5766	0.5046	0.49	0.4904	0.643	0.6306	0.6372	0.6516
LSH [5]*(non-deep)*	0.3501	0.4093	0.4169	0.4546	0.5736	0.6049	0.5954	0.6239
DSH [23]*(non-deep)*	0.5919	0.5982	0.5713	0.5791	0.6955	0.7071	0.6834	0.6603
SpH [24]*(non-deep)*	0.4645	0.4645	0.4465	0.4472	0.5966	0.5811	0.5828	0.579
SH [10]*(non-deep)*	0.5623	0.5033	0.4896	0.4533	0.6605	0.6405	0.6291	0.6213
AGH [26]*(non-deep)*	0.6551	0.6459	0.6274	0.6225	0.6862	0.7005	0.6998	0.6853
DH [12]*(deep)*	0.4733	0.4601	0.462	0.4763	0.6033	0.6195	0.6135	0.618
UH-BDNN [14]*(deep)*	0.5923	0.5915	0.5902	0.6097	0.6654	0.6684	0.6672	0.6699
DeepBit [13]*(deep)*	0.5463	0.5548	0.5624	0.561	0.589	0.6027	0.609	0.6086
Binary Tag Vector*(deep)*	0.6202	0.627	0.6247	0.6249	0.6365	0.6326	0.6373	0.6352
Proposed(WDHT)*(deep)*	**0.6709**	**0.6805**	**0.6955**	**0.676**	**0.7346**	**0.743**	**0.7034**	**0.7054**

Table 4: MAP values of NUS-WIDE and MIR-FLICKR25k data-sets computed using the top 5,000 retrieved images

histogram of the variances for all images. It was found that a majority of the variances falls below 8. Note that the maximum distance between any two word vectors in this space can be $2\sqrt{300}$ (the range of each dimension of the tag vector is $[-1, 1]$ and the space is 300-dimensional). This appears to suggest that for most of the images, their tag vectors do not spread out too much, which might explain that the simple *mean* aggregation function is working reasonably well.

Further, we computed the mAP for two different settings, one using the top 50,000 retrieved images and another using the top 5,000 retrieved images for the unsupervised approaches and report the results in Table 3 and Table 4 respectively. The first seven methods presented here are non-deep-learning methods while the last three are deep-learning-based. Additionally, DH [12] and UH-BDNN [14], even though being deep-learning-based, depend on the hand-crafted features. DeepBit [13] is the only work that takes raw images as input, but its performance is inferior to most other methods. In contrast, our approach (WDHT) is an end-to-end framework and performed superior than all the state-of-art methods on both datasets.

The non-deep-learning approaches ITQ [9] and AGH [26] seem to stand in the second and the third places in terms of the mAP values in the experiments. These methods performed superior to the deep-learning-based methods ([12], [14], [13]) as well. On the other hand, the weakly supervised approach WMH seemed to perform quite inferior as compared to WDHT with the new experiment setting. The results are presented as the bottom 2 rows of Table 3.

For further analysis, we plotted the precision-recall curves in Figure 2. These curves are computed taking into consideration all the retrieved samples from the database for a given query image. More specifically, we computed the average precision for various values of recall (1000 discrete values of recall) for all query images. The big performance gain of our approach on the NUS-WIDE data-set can be no-

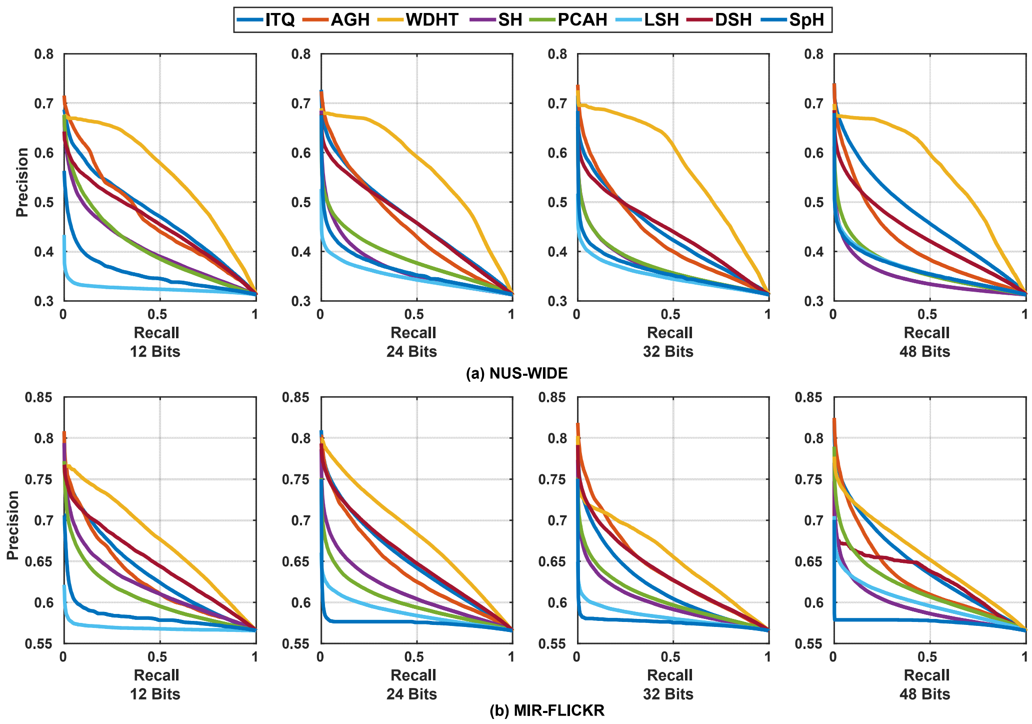

Figure 2: Precision Recall curves for NUS-WIDE and MIR-FLICKR datasets.

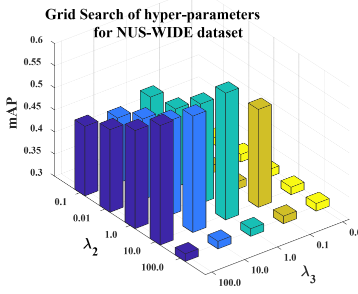

Figure 3: MAP values obtained for various hyper-parameter settings for the NUS-WIDE dataset

ticed from these graphs as well.

The presence of three loss components in the objective functions triggers the obvious question of combining them in the right proportions. To analyze this, we fix the value of λ_1 to 1.0 and change the values of λ_2 and λ_3 between 0.01 and 100.0. We performed a grid search over this range and chose the best hyper-parameters for our final model. Specifically, we set three values to the ones that gave maximum mAP value over a validation set during the grid search. For each setting of the hyper-parameter values, we only used 10,000 training sample due to the high training time of these experiments. A bar plot of the validation mAPs of NUS-WIDE dataset for various values of λ_2 and λ_3 is given in Figure 3. It can be noticed that higher values of λ_2 and lower values of λ_3 gave significantly better mAP as compared to other combinations. A similar behaviour was noticed on the MIR-FLICKR dataset as well. This is in accordance with the rationale presented in Section 3.3 that a ranking loss is better at forming semantically meaningful spaces as compared to Euclidean loss components [31]. While this rationale is yet to be validated mathematically, our results suggest this seems to be the case empirically.

5. Conclusion

We attempted the problem of weakly supervised deep image hashing using tag embedding. Our method is an end-to-end framework that takes raw images and tags as inputs and produces hash codes. The model is applicable to Web images where tag information is abundant. Through extensive experiments with comparison with existing state-of-the-art, we demonstrated that the proposed approach was able to deliver significant performance boost when evaluated on two well-known and widely-tested datasets. Future work includes possible better aggregation schemes in the *word2vec* space that may lead to improved performance.

Acknowledgment The work is supported in part by a grant from ONR (N00014-19-1-2119). Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of ONR.

References

[1] Manish Gupta, Rui Li, Zhijun Yin, and Jiawei Han. Survey on social tagging techniques. *ACM Sigkdd Explorations Newsletter*, 12(1):58–72, 2010.

[2] Shilad Sen, F Maxwell Harper, Adam LaPitz, and John Riedl. The quest for quality tags. In *Proceedings of the 2007 international ACM conference on Supporting group work*, pages 361–370. ACM, 2007.

[3] Kaiye Wang, Qiyue Yin, Wei Wang, Shu Wu, and Liang Wang. A comprehensive survey on cross-modal retrieval. *arXiv preprint arXiv:1607.06215*, 2016.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[5] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[6] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. Fast locality-sensitive hashing. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1073–1081. ACM, 2011.

[7] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1092–1104, 2012.

[8] Aniket Chakrabarti, Venu Satuluri, Atreya Srivathsan, and Srinivasan Parthasarathy. A bayesian perspective on locality sensitive hashing with extensions for kernel methods. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(2):19, 2015.

[9] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, 2013.

[10] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

[11] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2393–2406, 2012.

[12] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2475–2483, 2015.

[13] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1183–1192, 2016.

[14] Thanh-Toan Do, Anh-Dzung Doan, and Ngai-Man Cheung. Learning to hash with binary deep neural network. In *European Conference on Computer Vision*, pages 219–234. Springer, 2016.

[15] Yue Cao, Mingsheng Long, Jianmin Wang, and Shichen Liu. Deep visual-semantic quantization for efficient image retrieval. In *CVPR*, volume 2, page 6, 2017.

[16] Qing-Yuan Jiang and Wu-Jun Li. Deep cross-modal hashing. *CoRR*, 2016.

[17] Yue Cao, Mingsheng Long, Jianmin Wang, Qiang Yang, and Philip S Yu. Deep visual-semantic hashing for cross-modal retrieval. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1445–1454. ACM, 2016.

[18] Xing Xu, Fumin Shen, Yang Yang, Heng Tao Shen, and Xuelong Li. Learning discriminative binary codes for large-scale cross-modal retrieval. *IEEE Transactions on Image Processing*, 26(5):2494–2507, 2017.

[19] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[20] Ziyu Guan, Fei Xie, Wanqing Zhao, Xiaopeng Wang, Long Chen, Wei Zhao, and Jinye Peng. Tag-based weakly-supervised hashing for image retrieval.

[21] Hanwang Zhang, Na Zhao, Xindi Shang, Huan-Bo Luan, and Tat-seng Chua. Discrete image hashing using large weakly annotated photo collections. In *AAAI*, pages 3669–3675, 2016.

[22] Jinhui Tang and Zechao Li. Weakly-supervised multimodal hashing for scalable social image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[23] Zhongming Jin, Cheng Li, Yue Lin, and Deng Cai. Density sensitive hashing. *IEEE transactions on cybernetics*, 44(8):1362–1371, 2014.

[24] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2957–2964. IEEE, 2012.

[25] Xiaofeng Zhu, Lei Zhang, and Zi Huang. A sparse embedding and least variance encoding approach to hashing. *IEEE transactions on image processing*, 23(9):3737–3750, 2014.

[26] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *Advances in Neural Information Processing Systems*, pages 3419–3427, 2014.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[29] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[30] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in neural information processing systems*, pages 1042–1050, 2009.

[31] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.

[32] Sheng-Hung Hu, Yikang Li, and Baoxin Li. Video2vec: Learning semantic spatio-temporal embeddings for video representation. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 811–816. IEEE, 2016.

[33] Y. Li, S. Hu, and B. Li. Recognizing unseen actions in a domain-adapted embedding space. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 4195–4199, Sep. 2016.

[34] 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 *Proceedings of the ACM international conference on image and video retrieval*, page 48. ACM, 2009.

[35] Mark J Huiskes and Michael S Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43. ACM, 2008.

[36] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855*, 2015.

[37] Xiaofang Wang, Yi Shi, and Kris M Kitani. Deep supervised hashing with triplet labels. In *Asian Conference on Computer Vision*, pages 70–84. Springer, 2016.

[38] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3270–3278. IEEE, 2015.